T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE MELEZ METASEZGİSEL BİR ALGORİTMANIN TASARIMI

GANİMET NİLAY YÜCENUR

DOKTORA TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI
ENDÜSTRİ MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. NİHAN ÇETİN DEMİREL

İSTANBUL, 2011

ÖNSÖZ

Günümüz toplumlarında teknolojide özellikle de bilişim teknolojisinde yaşanan hızlı gelişmelerle birlikte işletmelerin bu yeni oluşan gelişim ve gelişimin beraberinde getirdiği değişime kendi yapılarını adapte edebilmeleri ve ayakta kalabilmeleri oldukça zorlaşmaktadır. Bu açıdan işletmelerin yaşamlarının devamı niteliğindeki rekabet üstünlüğünün müşteri taleplerinin zamanında, yüksek kalitede ve minimum maliyetle karşılanması ile sağlanabileceği açıktır. Literatürde oldukça geniş bir yer edinen optimizasyon işte bu tür rekabetçi ortamlarda şirketlere bir adım öne çıkmayı sağlamada kullanılan bir kavramdır. Karmaşık yapıların matematiksel modeller haline çevrildiği gerçek yaşam optimizasyon problemlerinin çeşitli optimizasyon teknikleriyle çözülmesi işletmeler için önemli bir kaynaktır.

Optimizasyon tekniklerinin en önemlilerinden olan metasezgisel yöntemler bu tez çalışmasının konusu olma özelliğini taşıyarak, iki metasezgisel yöntemden oluşturulan melez yapı araç rotalama problemlerinden araçların birden fazla depodan hizmet sunma özellikleri ile klasik araç rotalama problemlerinin genişletilmiş bir türü olan çok depolu araç rotalama problemine uygulanmıştır. Çok depolu araç rotalama problemleri gerçek dünyada oldukça sık karşılaşılan bir araç rotalama problem türü olmasına rağmen literatürde aynı yoğunlukta araştırılmamıştır. Bu nedenle bu doktora tez çalışmasının konusu çok depolu araç rotalama problemleri olarak belirlenmiştir. Çalışmada, çok depolu araç rotalama problemlerinin çözümü için metasezgisel yöntemlerden karınca kolonisi optimizasyonu ve genetik algoritma ilk kez bir arada kullanılarak problemin çözümü melez bir yapı önerilmiştir.

Literatürde çok geniş bir araştırma alanına sahip olan optimizasyon konusunda ve lojistik sektöründe önemli bir yer tutan araç rotalama problemleri çeşitlerinden hakkında en az çalışma olan çok depolu araç rotalama problemlerinin çözümü için daha önce yapılmamış bir yöntemle hem Karınca Kolonisi Optimizasyonu hem de Genetik Algoritma metasezgisellerini bir arada melez bir yapı içerisinde kullanmamı amaçlayan böyle bir doktora tezi çalışması yapmamı sağladığı ve tüm çalışmam boyunca hiçbir yardımı benden esirgemediği için öncelikle değerli hocam Sayın Yrd.Doç.Dr. Nihan ÇETİN DEMİREL'e, konuyla ilgili bilgiler arasında doğruları bulabilmemde önemli yardımlarıyla yol gösterici olan Sayın Prof.Dr. Cengiz KAHRAMAN ve Sayın Yrd.Doç.Dr. Hayri BARAÇLI'ya, tüm çalışmalarımızda fikirleriyle bize destek veren Sayın Yrd.Doç.Dr.

Tufan DEMİREL'e, tezimin oluşmasında hayatımdaki önemli varlıklarıyla en büyük katkıyı sağlayan sevgili eşim B.Ertuğrul YÜCENUR'a, biricik kızım Nil YÜCENUR'a ve çalışmalarımda destek ve güvenleriyle her zaman yanımda olan başta annem Meral SERBEST ve babam Nihat SERBEST olmak üzere tüm sevgili aileme teşekkürlerimi sunarım…

Mart, 2011

Ganimet Nilay YÜCENUR

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

A_1	Düğümleri birleştiren yayların kümesini
A,B	Depolar
A_i	*i* noktasının karınca karar tablosu
$A_{i,j}$	*i* ve *j* noktaları arasındaki öklid mesafe
a_i	Zaman penceresinde en erken hizmete başlama zamanı
b_i	Zaman penceresinde en geç hizmete başlama zamanı
$b_i(t)$	*t* anında *i* şehrinde bulunan karınca sayısı
best_s	Tabu arama ile elde edilen iyi sonuçların sayısı
C	Araç kapasitesi
C_i	Seyahat günlerinin olası kombinasyonlarının seti
c	Direkt maliyet
c_{ik}	En yakın nokta maliyeti
cf	Yerel arama ile bulunan en iyi permutasyonun amaç değeri
c_1,c_2	Öğrenme faktörleri
D	Düğümler arasındaki seyahat zamanları veya uzaklıklar
D_1	Maliyet matrisi
D_0	Başlangıç uzaklık matrisi
d	Atama tabanlı algoritmada maliyet
d_i	Stokastik talep
d_{ij}	*i* noktası ile *j* noktası arasındaki mesafe
d_{ijk}	*k* türündeki araç için seyahat maliyeti
E	Kenar kümesi
e_i	Hizmet sıklığı
e	Rastgele bir sayı
F_h	Rotalanmamış müşteri kümesi
f(x)	Fonksiyon
Graf (N,E)	Gezgin satıcı probleminde graf kümesi
G=(V,E)	Graf kümesi
G=(V,A,D)	Stokastik talepli araç rotalama problemi graf kümesi
h	Daha önce rotaya atanmamış noktalar
i,j	Şehirler
I_{pop}	Başlangıç popülasyonu
K	Zaman pencereli araç rotalama probleminde oldukça büyük bir sayı
k	Depo sayısı

$keep$	Eşleştirilen kromozomların sayısı
k_u	Ulaşılan en iyi amaç fonksiyonu (F_B) geliştirilmedenki iterasyon sayacı
k_t	Deneme komşuluk çözümü vektör sayacı
L	Mesafe kısıtı
L_{best}	Geçerli iterasyonda bulunan en iyi turun uzunluğu
L_{ij}	*i, j* düğümleri arasındaki en kısa mesafedeki gezgin satıcı problemi turu
$L^k(t)$	*t.* iterasyonda *k* karıncası tarafından üretilen turun uzunluğu
L_{nn}	En yakın komşu sezgiseli ile üretilen turun uzunluğu
$L(v_k)$	Aracın v_k müşterisini terk ettikten sonraki yükü
M	Toplam araç sayısı
M_1	Periyodik yüklemeli araç rotalama probleminde gün periyodu
M_a	Ana problem
M^k	Karıncanın özel hafızası
m	Kolonideki toplam karınca sayısı
m_b	Kolonideki en iyi karınca
$maxgen$	Genetik algoritmadaki maksimum üretilen gen sayısı
mut	Mutasyon oranı
N	Toplam müşteri sayısı
NC_{max}	Maksimum çevrim sayısı
N_i	*i* düğümünün komşular seti
N_i^k	*k* karıncasının henüz ziyaret etmediği *i* şehrinin komşu düğümlerinin seti
n_1	Alt küme sayısı
n_2	Popülasyondaki kromozom sayısı
n_3	Şehir sayısı
n	Değişken sayısı
o_i	Arz
Q	Toplam kapasite kısıtı
Q_1	Karınca-yoğunluk modelinde feromon miktarı
Q_2	Karınca-miktar modelinde feromon miktarı
Q_3	Karınca-çevrim modelinde feromon miktarı
q_t	Toplam talep vektörü
q	Sözde-rastlantısal-orantılı kuraldaki normal dağılımlı rassal değişken
q_i	*i* müşterisinin talebi
q_a	Müşteri ağırlıkları
q_0	Tercih yapma olasılığı
P	Alt problem
P_c	Geliştirilmiş çaprazlama operatörü
P_m	Geliştirilmiş mutasyon operatörü
P_{ts}	Tabu arama optimizasyonundaki bireylerin olasılığı
$P_{ij}^k(t)$	*t.* iterasyonda *k* karıncasının *i* şehrinden *j* şehrine gitme olasılığı
P_1, P_2	Diziler
p_s	Aracın seyir süresini hesaplamakta kullanılan parametre
p	Eşitlik kısıt sayısı
p_1, p_2, p_3	Müşteriler
p^i	Vaat edilen alan
p^j	*n* değişkenli maliyet fonksiyonu

pop	İlk iterasyon sonunda üretilen popülasyon
pop_size	Popülasyon büyüklüğü
R	Rastgele bir sayı
R_i	Rota
r_1	Başlangıç yarıçapı
$rand_1$, $rand_2$	[0,1] aralığında bulunan iki rassal fonksiyon
S_c	Çözüm uzayı
S	Popülasyondaki tüm kromozomların uygunluk toplamı
S'	Rassal komşu
S_B	Ulaşılan en iyi çözüm vektörü
S_k	*k* rotasına ait nokta kümesi
S_r	*r* rotasında bulunan noktalar kümesi
S_0	Düğüm sırası matrisi
s	Sözde-rastlantısal-orantılı durum geçiş kuralı
s_i	Hizmet süresi
s_{ij}	Mesafe tasarrufu
s_{ik}	*k* aracının *i* müşterisine hizmete başladığı zaman
s_{pq}	Eşleme tabanlı sezgisel algoritmada tasarruf değeri
$\hat{s}$	Optimal çözüm
T	Mevcut sıcaklık
TL	Tabu listesi
$T^k(t)$	*t.* iterasyonda *k* karıncası tarafından üretilen tur
t^j	Komşu tabu listesi
t_{ij}	Stokastik seyahat zamanı
ts_len	Tabu listesi uzunluğu
ts_loop	Tabu iterasyon sayısı
t_{max}	Maksimum iterasyon sayısı
$t(S_k)$	Gezgin satıcı probleminin optimum çözümü
$tabu_k$	*k.* karıncanın tabu listesini içeren vektör
$tabu_k(s)$	*k.* karınca tarafından ziyaret edilmiş *s.* şehir
U	En yakın komşu algoritmasında depoya atanmamış müşteriler seti
u	Rotalanmamış müşteriler için en kısa arama ağacı uzunluğu
u_i	1. düğümden *i.* düğüme en kısa uzaklık
V	Nokta kümesi
V_{id}	Hız
v_i	Müşteri
v_o	Kapasite kısıtlı araç rotalama probleminde merkez depo
$viol(x)$	Fizibil olmayan çözümün kısıt ihlal değeri
X	Çaprazlama noktası
X_{ijk}	Karar değişkeni
x^j	Çözüm uzayındaki herhangi bir nokta
X,Y	Köşelerin seti
W	Eylemsizlik ağırlığı
w	Eşit taleplere sahip kapasiteli rotalama probleminde birim ürün talebi
Z	Amaç fonksiyonu
$[a_i,b_i]$	Zaman penceresi

$[\tau_{min},\tau_{max}]$	Feromon güncelleme aralığı
(v_i,v_j)	Zaman düzlemi
$(0,1)$	İkili bit dizisi
α	Feromon izinin ilgili ağırlığını kontrol eden parametre
α_q	Ant-Q algoritmasında öğrenme adımı parametresi
α^*	Ekleme maliyeti
β	Sezgisel değeri kontrol eden parametre
ε	İşbirliği maliyeti
ξ_i	Stokastik müşteri talep değişkeni
ψ	Heterojen araç filosuna sahip araç rotalama probleminde araç türleri
λ	Müşteriler arasındaki mesafeyi ağırlıklandıran parametre
γ	Ant-Q algoritmasında hesaplama faktörü parametresi
μ	Müşteri ile depo arasındaki mesafeyi ağırlıklandıran parametre
ρ	Feromon buharlaşma katsayısı
ρ_i	Eksen üzerinde dönen ışının uzunluğu
τ_0	Başlangıç feromon değeri
$\tau_{ij}(t)$	t anında (i,j) yayı üzerindeki feromon miktarı
$\Delta\tau_{ij}^{k}(t)$	k karıncasının t. iterasyonda geçtiği (i,j) yayına bıraktığı feromon miktarı
η_{ij}	i düğümünden j düğümüne hareketin sezgisel değeri (seçilebilirlik)
η	Araştırma alanının büyüklüğü
η_1	Ödül değeri
ϕ	Çaprazlama operasyonu sayısı

KISALTMA LİSTESİ

ACS	Ant Colony System (karınca kolonisi sistemi)
ANTS	Approximated Non-deterministic Tree Search (yaklaşık deterministik olmayan ağaç arama)
ARP	Araç Rotalama Problemi
AUARP	Açık Uçlu Araç Rotalama Problemi
AYARP	Asimetrik Yollu Araç Rotalama Problemi
CCP	Chance Constraint Stochastic Programming (şans kısıtlı stokastik programlama)
ÇDARP	Çok Depolu Araç Rotalama Problemi
DARP	Dinamik Araç Rotalama Problemi
DG	Diferansiyel Gelişim Algoritması
FIFO	İlk Giren İlk Çıkar
GA	Genetik Algoritma
GDARP	Geri Dönüşlü Araç Rotalama Problemi
GKKO	Global Karınca Kolonisi Optimizasyonu
IACS	Improved Ant Colony System (iyileştirilmiş karınca kolonisi sistemi)
KKARP	Kapasite Kısıtlı Araç Rotalama Problemi
KKO	Karınca Kolonisi Optimizasyonu
KKS	Karınca Kolonisi Sistemi
KS	Karınca Sistemi
KUARP	Kapalı Uçlu Araç Rotalama Problemi
KYARP	Karma Yüklemeli Araç Rotalama Problemi
LSP	Lot Sizing Problem (sipariş büyüklüğü problemi)
MKARP	Mesafe Kısıtlı Araç Rotalama Problemi
MKKO	Melez Karınca Kolonisi Optimizasyonu
MMKS	Maks-Min Karınca Sistemi
MTFARP	Müşteri Tipi Farklı Araç Rotalama Problemi
PDARP	Parçalı Dağıtımlı Araç Rotalama Problemi
PSO	Parçacık Sürü Optimizasyonu
PYARP	Periyodik Yüklemeli Araç Rotalama Problemi
RM	Stochastic Programming with Recourse (yardımcı eylemli stokastik programlama)
RTKS	Rank Temelli Karınca Sistemi

SARP	Statik Araç Rotalama Problemi
SMTARP	Stokastik Müşteri ve Talepli Araç Rotalama Problemi
SSHZARP	Stokastik Seyahat ve Hizmet Zamanlı Araç Rotalama Problemi
STARP	Stokastik Araç Rotalama Problemi
SYARP	Simetrik Yollu Araç Rotalama Problemi
TA	Tabu Arama Algoritması
TB	Tavlama Benzetimi Algoritması
TDARP	Toplama ve Dağıtım İşlemlerini Kapsayan Araç Rotalama Problemi
YBS	Yapay Bağışıklık Sistemi Algoritması
YSA	Yapay Sinir Ağları
ZPARP	Zaman Pencereli Araç Rotalama Problemi

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE MELEZ METASEZGİSEL BİR ALGORİTMANIN TASARIMI

Ganimet Nilay YÜCENUR

Endüstri Mühendisliği Anabilim Dalı
Doktora Tezi

Tez Danışmanı: Yrd. Doç. Dr. Nihan ÇETİN DEMİREL

Optimizasyon gerçek yaşam problemlerinin matematiksel modeller haline dönüştürülmesi ile karşımıza çıkar ve günlük hayatımızdaki birçok sorun için en iyi çözümü bulmayı amaçlar. Optimizasyon konusunun en önemli dallarından biri de lojistik sektöründe karşımıza çıkar. Lojistik sektöründeki araç rotalama problemlerinde de amaç toplam seyahat mesafesini ve çözümde kullanılan toplam araç sayısını minimize ederek problemde var olan kısıtları da göz önüne alarak optimum rotaları tasarlamaktır. Bu çalışmada, çok depolu araç rotalama probleminin çözümü için karınca kolonisi optimizasyonu ve genetik algoritmanın bir arada kullanılmasıyla oluşturulan melez metasezgisel bir yapı önerilmiştir. Problemde amaç, tüm araçlar tarafından kat edilen toplam seyahat mesafesinin minimize edilmesidir. Problemde araç filosu homojendir, araç kapasiteleri ve müşteri talepleri bilinmektedir. Çok depolu araç rotalama probleminin çözümü için önerilen bu metasezgisel yapı iki aşamadan oluşmaktadır. Problemin çözümü için önerilen ilk aşamada müşterilerin hangi depolardan hizmet alacağının belirlendiği gruplama, ikinci aşamasında ise müşterilerin hangi sıra ile depolardan hizmet alacağının belirlendiği rotalama işlemleri gerçekleştirilir. Birinci aşamadaki gruplama işlemi için genetik algoritma, Thangiah ve Salhi'nin 2001 yılında ortaya koydukları genetik kümeleme yönteminin geliştirilmiş hali ile kullanılmıştır. İkinci aşamadaki rotalama işlemi ise karınca kolonisi optimizasyonu algoritmalarından Gambardella ve Dorigo tarafından 1997 yılında önerilen karınca kolonisi sistemi yaklaşımı ile gerçekleştirilmiştir. Ortaya konan yeni melez metasezgisel yöntem literatürde kabul gören Cordeau vd.'nin (1997) önerdikleri problem setleri ile test edilmiş ve elde edilen sonuçlar var olan diğer yöntem çözümleriyle

karşılaştırılmıştır. Sonuç olarak, klasik araç rotalama problemlerinin önemli kollarından biri olan çok depolu araç rotalama problemlerinin çözümü için önerilen karınca kolonisi optimizasyonunun ve genetik algoritmanın bir arada kullanıldığı bu tez çalışması bu konu ile ilgili literatüre kazandırılan ilk çalışmadır.

Anahtar Kelimeler: Optimizasyon ve optimizasyon teknikleri, metasezgiseller, çok depolu araç rotalama problemi, karınca kolonisi optimizasyonu, genetik algoritma, genetik kümeleme

ABSTRACT

DESIGNING A HYBRID META HEURISTIC ALGORITHM FOR OPTIMIZATION PROBLEMS SOLUTIONS

Ganimet Nilay YÜCENUR

Department of Industrial Engineering
PhD Thesis

Advisor: Assist. Prof. Dr. Nihan ÇETİN DEMİREL

Optimization is the simulation of the real world problems as mathematical models. Vehicle routing problems are very important issue for logistics sector. The objectives of the vehicle routing problems are to design optimal routes minimizing total traveled distance, minimizing number of vehicles which are used for the solution that satisfy corresponding constraints. In this study, for the solution of the multi-depot vehicle routing problem, a new hybrid metaheuristic structure is proposed with ant colony optimization and genetic algorithm. The aim of the problem is to minimize the total traveled distance by the all vehicles. In problem, vehicle fleet is homogeneous, capacity of vehicles and customer demands are known. The metaheuristic structure of the multi-depot vehicle routing problem solution consists of two phases. In the first phase is about the grouping that decides which customer is served by which depot and in the second phase is about the routing that decides the customers sequencing for the serving. In the first phase for grouping Thangiah and Salhi's (2001) genetic clustering method is developed and in the second phase for routing Gambardella and Dorigo's (1997) ant colony system approach is used. The proposed metaheuristic method is tested with the Cordeau et al.'s (1997) problem sets and the results are compared with the other solution techniques in the literature. In conclusion, the multi-depot vehicle routing problems are the important types of the classical vehicle routing problems, to our knowledge this is the first study that investigates the solution with ant colony optimization and genetic algorithm.

Key words: Optimization and optimization techniques, metaheuristics, multi-depot vehicle routing problems, ant colony optimization, ant colony system, genetic algorithm, genetic clustering

BÖLÜM 1

GİRİŞ

"En iyileme" anlamına gelen optimizasyon aynı sektörde faaliyet gösteren benzer hizmetleri, benzer kalitede sunabilen işletmelerin içinde bulundukları rekabet ortamında bir adım öne geçebilmelerini sağlayacak önemli bir kavramdır. Optimizasyon teknikleri de yapılmış veya yapılmakta olan işin en iyi çözümünü ortaya koymak için kullanıldığından globalleşen dünya pazarı ve buna bağlı olarak artan rekabet ortamında işletmelere problemlerini çözmede yardımcı olacaktır.

1.1 Literatür Özeti

Bir problemde belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçmeye çalışan optimizasyon teknikleri kullanılarak ortaya konan "Optimum Çözüm" olarak adlandırılır ve optimizasyonda hedef her zaman için bu optimum çözümü yakalayabilmektir.

Optimizasyon, anlamından da anlaşılacağı gibi, her alanda kullanılabilmektedir. İnşaattan, ekonomiye, tasarım optimizasyonundan, lojistik sektörüne birçok alanda oluşabilecek olan probleme optimizasyon teknikleri yardımı ile bir en iyi çözüm aranır.

Metasezgiseller optimizasyon problemlerine çözüm arayan ve arama sürecine rehberlik eden stratejilerdir. En iyi ya da en iyiye yakın çözümleri bulmak için arama uzayını hızlı bir şekilde araştıran bu optimizasyon teknikleri basit yerel arama algoritmalarından karmaşık öğrenme proseslerine kadar geniş bir yelpazeyi içermektedir.

1.2 Tezin Amacı

Bu tez çalışmasının amacı, literatürde var olan metasezgisel yöntemlerin kullanım yerlerini, özelliklerini ve algoritma yapılarını inceleyerek, literatürde var olmayan yeni bir melez algoritma yapısının ortaya konabilmesini sağlamaktır. Ortaya konan bu melez yapının çok depolu araç rotalama problemine uygulanması ile de bulunan algoritmanın test edilmesi ve gerçekliğinin kanıtlanabilmesi amaçlanmıştır.

Genellikle depodan başlayarak tüm müşterilere en kısa zamanda ve minimum maliyetle hizmet sunabilmesini sağlayan optimum rotanın çizilmesi araç rotalama problemlerinin temelidir. Çok uzun yıllardır araştırılmasına rağmen araç rotalama problemleri için çizilmesi gereken bu optimum rotayı veren matematiksel kesin bir formül bulunamamıştır. Bu nedenle problem çeşidinin kısıt ve özelliklerine göre sezgisel ve metasezgisel yöntemler araç rotalama problemlerinin çözümünde kullanılmaya başlanmış ve elde edilen sonuçlar da bu alanda kullanılmaya devam etmiştir.

1.3 Hipotez

Bu tez çalışması, optimizasyon problemlerine ışık tutan bu iki metasezgisel yöntemi, Karınca Kolonisi Optimizasyonu ve Genetik Algoritmaları göz önüne olarak yeni bir melez algoritma tabanlı yapının oluşturulmasını sağlayacak ve oluşturulan bu yeni melez yapının lojistik sektöründe önemli bir yeri olan ve daha önce uygulanmadığı bir alan olan araç rotalama problemi türlerinden biri olan çok depolu araç rotalama probleminin çözümünde kullanılarak oluşturulan yeni melez yapının çözüm kalitesinin ve performansının iyileştirilebilmesi hedeflenecektir.

Bu amaçla başladığım doktora tez çalışmamda ilk olarak (ikinci bölümde) optimizasyonun ne olduğu, optimizasyon probleminin temel yapısı, çeşitleri ve çözüm teknikleri incelenmiştir. Üçüncü bölümde, optimizasyon problemlerinin çözümü için oluşturulan ve bu tez kapsamında önerilen melez algoritma yapısı açıklanmış ve bu yapıyı oluşturan genetik algoritma ve karınca kolonisi optimizasyonu metasezgiselleri kapsamlı bir şekilde incelenmiştir. Dördüncü bölümde, lojistik sektöründe önemli bir yer tutan araç rotalama problemlerinden bahsedilmiş ve araç rotalama problem çeşitleri ile bu problemleri çözebilen teknikler araştırılmıştır. Beşinci bölümde, çok

depolu araç rotalama problemi için önerilen genetik algoritma ve karınca kolonisi optimizasyonundan oluşan melez metasezgisel yöntem uygulama adımları ve akış diyagramları ile açıklanmış, literatürde yer alan kabul görmüş problem setlerine önerilen yöntemin uygulanması sağlanmış, bulunan sonuçlar diğer yöntemlerle elde edilen sonuçlarla karşılaştırılmış ve son olarak önerilen yapı üzerinde gerçekleştirilen parametre değişimleri ile sonuç değerlerin iyileştirilmesi sağlanmıştır. Son bölümde ise tez çalışması özetlenerek sonuç ve değerlendirme yapılmıştır.

BÖLÜM 2

OPTİMİZASYON VE OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMLERİ

İnsanlar yaşamlarını sürdürmek için karşılaştıkları problemleri birer model olarak algılayarak bunları çözme yolunu seçmişlerdir. Günlük yaşamda kullanılan bu teknik bilgisayar teknolojisinin gelişmesi ve matematik bilimi ile entegre olması sonucu bilim dünyasına yansımıştır. Bilimin gelişmesiyle kurulan matematiksel modeller ilk başta doğrusal olup az sayıda değişkene sahipken, gerçek yaşam problemlerinin daha karmaşık yapıda olmasından dolayı doğrusal olmayan modeller kurulmuş ve bu modellerin çözümlenebilmesi için optimizasyon kavramı geliştirilmiştir.

Bir problem için belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçmeyi amaçlayan optimizasyon günümüzde rekabetin artması, teknolojinin hızla gelişmesi ve kullanılan ortak kaynakların kısıtlı hale gelmesi ile ortaya çıkan karmaşık sistemlerin çözümünde klasik yöntemlerin kullanılma zorluğu nedeniyle oldukça önemli bir hale getirmiştir.

Optimizasyon, en basit tanımı ile eldeki kısıtlı kaynakların en iyi şekilde kullanılmasıdır. Matematiksel olarak ise optimizasyon, bir fonksiyonunun amacına uygun olarak maksimize veya minimize edilmesi yani amaç fonksiyonunun en iyi değerini veren kısıtlardaki değişkenlerin değerinin bulunmasıdır (Kahaner vd. [1]).

En iyileme anlamına gelen optimizasyonda varılmak istenen sonuca optimizasyon teknikleri ile ulaşılmaya çalışılır. Optimizasyon teknikleri ile elde edilecek çözümün en

iyi çözüm olması amaçlanır ve elde edilen bu en iyi çözüm "Optimum Çözüm" olarak adlandırılır.

Optimizasyonda genel olarak belirlenen amaç maksimum karın veya minimum maliyetin elde edilmesi için üretim miktarını kısıtlara bağlı olarak tespit etmekken, günümüzde ise optimizasyon çok çeşitli endüstri kesimlerinde inşaattan, tekstile, mimarlıktan, otomotive birçok alanda kendine uygulama sahası bulmaktadır.

2.1 Optimizasyon Problemi Standart Yapısı ve Formülasyonu

Optimizasyon işlemlerinde ilk iş olarak karar parametreleri veya karar değişkenleri ya da başka bir ifadeyle tasarım parametreleri olarak da isimlendirilen parametreler setinin tanımlanması gerekmektedir. Bu tanımlamanın ardından parametrelere bağlı olarak amaç fonksiyonu (maksimizasyon/minimizasyon) ve fonksiyonu sınırlayan kısıtlar belirlenir. Kısıtlar eşitlik ve eşitsizlik şeklinde formülasyon içerisinde yer alabileceği gibi, parametrelerin alamayacağı değerleri ifade ederler.

Maliyet fonksiyonu ve kısıtların matematiksel formülasyonu aşağıda verildiği gibidir (Bhatti, [2]):

n değişkenli $x = (x_1, x_2, ...x_i, \ldots x_n)$ vektörü tanımlanırsa, burada x_i, *i.* parametrenin değerini gösterir. Maliyet fonksiyonu *f(x)*,

$$f(x) = f(x_1, x_2, \ldots x_n) \tag{2.1}$$

$$h_j(x) = h_j(x_1, x_2, \ldots x_n) = 0, \qquad 1 \le j \le p \tag{2.2}$$

şeklinde tanımlanan *p* tane eşitlik kısıtına ve

$$g_i(x) = g_i(x_1, x_2, ...x_n) \le 0, \qquad 1 \le i \le m \tag{2.3}$$

şeklinde tanımlanan *m* tane eşitsizlik kısıtına sahip olabilir.

Optimizasyon problemlerinde kısıtları sağlayan olası tüm çözümlerin oluşturduğu bölge çözüm için araştırma yapılabilecek olan uygun çözüm alanı olarak düşünülür. Bu bölge içerisinde yer alan optimum çözüm de amaç fonksiyonunu sağlayan en iyi değerdir.

2.2 Optimizasyon Problemlerinin Sınıflandırması

Collette ve Siarry [3] te, çeşitli optimizasyon problemlerini sahip oldukları karakteristik özelliklere göre aşağıdaki gibi sınıflandırmıştır:

- Karar değişkeni sayısına göre
    - Tek değişkenli optimizasyon problemleri
    - Çok değişkenli optimizasyon problemleri
- Karar değişkeni çeşidine göre
    - Tasarım değişkenlerinin ve parametrelerinin alacağı değerlerin sürekli olduğu problem: Sürekli gerçek sayılı (sürekli) optimizasyon problemleri
    - Birbirinden ayrık nesnelerin optimal olarak düzenlenmesi, gruplanması, sıralanması ve seçilmesi problemi: Tam sayılı (tamsayı veya ayrık) optimizasyon problemleri
    - Sonlu büyüklük sayılarının setinin permütasyonları (kombinatoryal) optimizasyon problemleri
- Amaç fonksiyonu çeşidine göre
    - Eğer optimizasyon problemi lineer amaç ve kısıt fonksiyonlarına sahip ise problem: Karar değişkenlerine bağlı lineer programlama problemi
    - Karar değişkenlerine bağlı ikinci dereceden denklem fonksiyonu
    - Eğer optimizasyon problemi lineer amaç ve kısıt fonksiyonlarına sahip iken bu kısıtlardan herhangi bir nonlineer ise problem: Karar değişkenlerine bağlı doğrusal olmayan programlama problemi
- Problem yapısına göre
    - Kısıtlı optimizasyon problemleri
    - Kısıtsız optimizasyon problemleri

2.3 Optimizasyon Problemlerinin Çözümü

Optimizasyon problemi modelinin fiziksel yapısı eğer bir tane amaç fonksiyonu içeriyor ve sonuçta tek bir optimum çözüm elde ediyorsa problem tek-amaçlı optimizasyon problemi olarak tanımlanırken birden fazla sayıda amaç fonksiyonu içeren ve sonuçta birden fazla sayıda optimum çözüm üreten problem ise çok-amaçlı optimizasyon problemi olarak adlandırılır. Çoğu gerçek yaşam problemi, yapısında bulundurduğu birden fazla sayıdaki amaç fonksiyonu nedeniyle çok-amaçlı optimizasyon problemi şeklinde karşımıza çıkar (Deb [4]).

Optimizasyon problemlerinde olduğu gibi bu problemlere sonuç arayan çözüm yöntemleri de çeşitli kriterlere göre sınıflandırılırlar. Bu sınıflandırmalardan en genel olanına göre optimizasyon problemlerine tek-amaçlı ve çok-amaçlı optimizasyon problemleri için geliştirilmiş çözüm metotları ile sonuç aranır.

2.3.1 Tek-Amaçlı Optimizasyon Problemi Çözüm Metotları

Optimizasyon problemleri için geliştirilen çözüm yöntemlerinden tek-amaçlı optimizasyon problemi çözüm metotları Collette ve Siarry tarafından 2003 yılında [3] te Şekil 2.1'de verildiği gibi kendi içinde sınıflandırılmıştır. Bu sınıflandırmaya göre tek-amaçlı optimizasyon problemi çözüm metotları aşağıda verilen gruplara ayrılabilir:

Kesin metotlar: Global optimumu arayan bu metotlar çözümleri araştırma uzayı içinde belirtirler. Global optimumun kesinlikle bulunabileceği birçok problem için çözüm üreten bu metotlar araştırma uzayının oldukça büyük olduğu problem yapıları için kullanılabilir nitelikte değildir.

Yaklaşık metotlar: Tek-amaçlı optimizasyon problemleri için çözüm arayan yaklaşık metotlar sezgisel metotlar ve metasezgisel metotlar olmak üzere ikiye ayrılırlar:

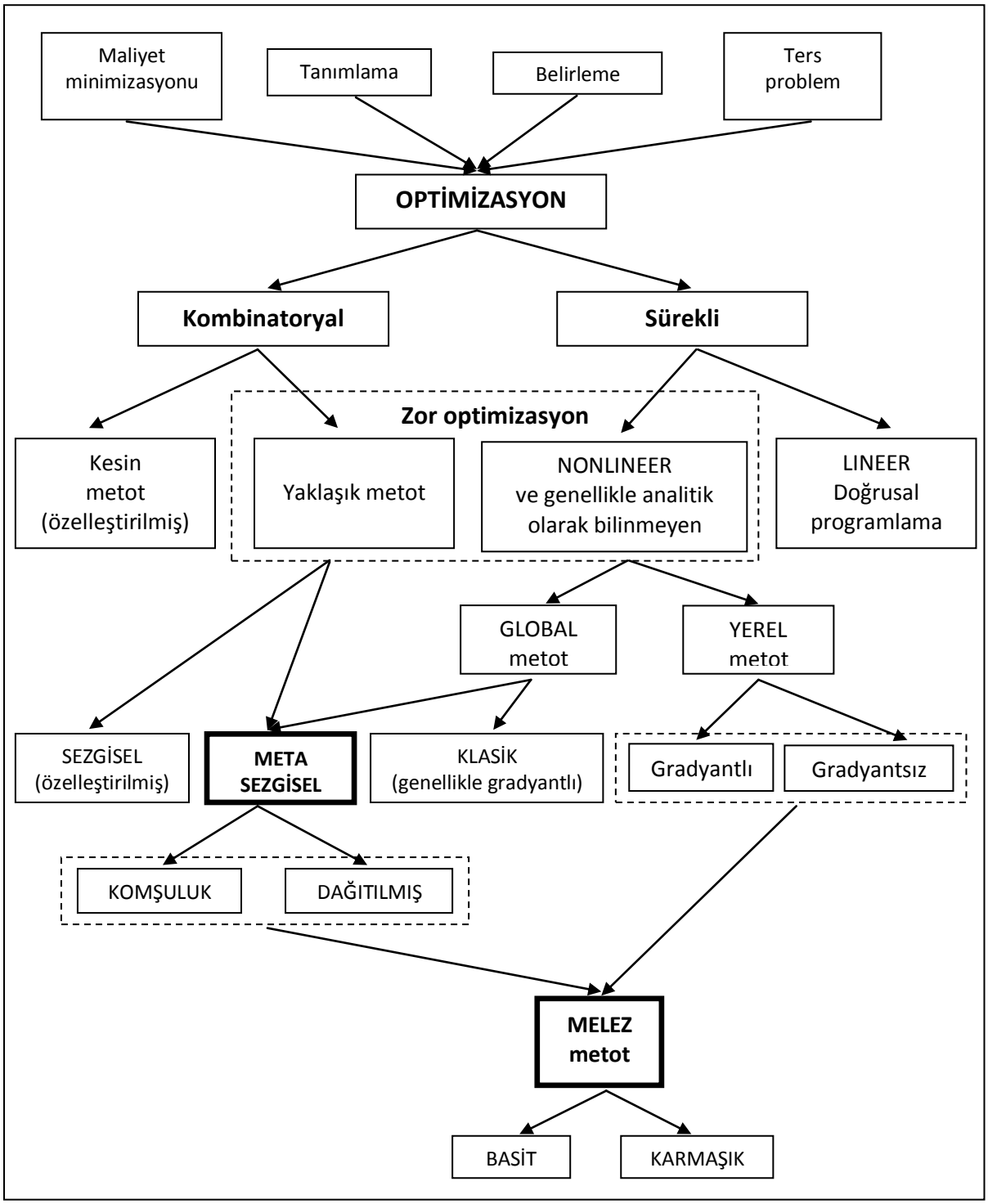
- Sezgisel metotlar: Bu metotlar problemlerin yapısına göre özel bir çözümü bulmak için geliştirilmişlerdir. Sezgisel metotlar, hangi problem için geliştirilmişler ise o probleme çözüm bulmayı amaçlarlar.
- Metasezgisel metotlar: Bu metotlar benzetilmiş tavlama algoritmasındaki metalin tavlama davranışına benzer bir şekilde doğal yaşamın bir benzetimi gibi

hareket edebilecek olan genel bir düşünceyi temel alırlar. Metasezgisel algoritmalar adapte edildikleri her tür problem için oldukça iyi sonuç verirler.

Doğrusal olmayan metotlar: Tek-amaçlı optimizasyon problemlerini çözen bir diğer grup çözüm metodu da global ve yerel olmak üzere ikiye ayrılabilen doğrusal olmayan metotlardır.

- Global: Metasezgisel ve klasik olarak ikiye ayrılır:
    - Metasezgisel: Etkili olacağı düşünülen metasezgisel yapı üzerinden çözüm kurulur.
    - Klasik: Optimizasyon içerisindeki kurulum algoritmaları olan bu metotlar konveks fonksiyon uygulandığında global optimumun bulunmasına izin verirler.
- Yerel: Amaç fonksiyonu türevli ve amaç fonksiyonu türevsiz olarak ikiye ayrılır:
    - Amaç fonksiyonu türevli: Bu metotlarda, optimumun bulunması için türevler kullanılır. Genel olarak sürekli optimizasyon problemlerine uygulanan bu metotlar, problem için her zaman global optimumu sağlamayı başaramayıp bazen de bizleri yerel optimum ile buluşturur.
    - Amaç fonksiyonu türevsiz: Bu metotlar, çeşitli metotların bir araya getirilmesi ile oluşturulur.
        - Yapıcı metotlar: Bu metotlarda çözüm, bir değişkenin ardından bir başka değişkene çözüm bulunması ile yapılandırılır ve bu yapılandırma esnasında kullanıcının değişkenleri değiştirmesi yasaklanır.
        - Stokastik metotlar: Bu metotlar rassal bir proses temellidir.

Şekil 2.1'de tek-amaçlı optimizasyon problemlerinin genel sınıflandırması gösterilmiştir.



Şekil 2.1 Tek-amaçlı optimizasyon problemlerinin genel sınıflandırması (Collette ve Siarry [3])

Şekil 2.1'e göre tek-amaçlı optimizasyon problemleri öncelikle kombinatoryal optimizasyon ve sürekli optimizasyon olarak ikiye ayrılmıştır. Kombinatoryal optimizasyonun çözümünde zor problemlerle karşılaşıldığında probleme çözüm sağlamaya uygun bir şekilde özelleştirilmiş yaklaşık metotlar kullanılırken sürekli

optimizasyonda ise problem yapıları doğrusal ve doğrusal olmayan olmak üzere ikiye ayrılır. Doğrusal olmayan optimizasyonda pragmatik çözüm sıklıkla yerel metodu kullansa da amaç fonksiyonunun türevleri ile çözüm bulunur. Çok fazla yerel minimumun bulunduğu bir çözüm prosedüründe global metot kullanılır ve klasik global optimizasyon metotlarına en iyi alternatif olan metasezgisel yöntem bulunur.

Metasezgiseller kendi aralarında birim zamanda tek bir çözüm üreten komşuluk temelli (benzetimli tavlama, tabu arama, …vb) ve tüm popülasyon içerisinde ustaca idare edilen paralel çözümler üreten dağıtılmış temelli metotlar olarak ikiye ayrılır. Son olarak, herhangi bir metasezgisel ile bir araya getirilen bir yerel arama metodu ile melez yapılar oluşturulabilinir. Bu melez yapılar mevcut çözümün iyileştirilmesini amaçlar.

2.3.2 Çok-Amaçlı Optimizasyon Problemi Çözüm Metotları

Çok-amaçlı optimizasyon problemlerinin çözüm sürecinde kullanılabilecek olan metotlar karar vericiden alınacak olan tercih bilgisine ve bu tercih bilgisinin alınma zamanına dayalı olarak sınıflandırılır. Bu sınıflandırmaya bir de tercih bilgisinin istenmediği durumlar eklendiğinde çok-amaçlı optimizasyon problemleri için geliştirilen çözüm yöntemleri dört gruba ayrılabilir (Atlas, [5]):

Tercih bilgisinin kullanılmadığı yöntemler: Karar vericiden tercih bilgisinin istenmediği bu tekniklerde tercih bilgisi istenmese de gerçekte kullanıcı veya problemi modelleyen tarafından açıkça belirtilmeyen örtülü bir tercih dikte edilmektedir. Tercih bilgisinin istenmediği yöntemler ideal uzaklık minimizasyonu tekniği, maksimum etkinlik ilkesi ve min-maks formülasyonu ve küresel ölçüt tekniğidir.

Sonsal tercih bilgisini kullanan yöntemler: Tercih bilgisinin çözüm sürecinden sonra kullanıldığı bu yaklaşımlarda, bir probleme yönelik tüm etkin çözümler bulunmaya çalışılır. Karar vericiye tüm çözümlerin sunulduğu ve karar vericiden etkin çözümlerden birisini seçmesinin istendiği bu teknikler çözüm alternatiflerinin türetilmesi ile ilgilenir. En büyük avantajı, çözümlerin karar vericinin tercihlerinden bağımsız olması olan bu yöntem algoritmalarının karmaşık olması, çoğu gerçek problemin bu yaklaşımla çözülmek için çok büyük olması ve karar vericinin çok fazla sayıda çözüm arasından

seçim yapmak zorunda kalıyor olması gibi de dezavantajları mevcuttur. Çoklu ortaya çıkan yaklaşımlar, çok amaçlı genetik algoritmalar, dinamik çok amaçlı programlama, ulaşılabilir küme tekniği ve benzer doğrular yaklaşımı çözüm sürecinin sonunda tercih bilgisini kullanan yöntemlerdir.

Önsel tercih bilgisini kullanan yöntemler: Çok amaçlı optimizasyonu gerçekleştirmede en yaygın olarak kullanılan yol, karar vericinin tercihlerini önceden belirtilmesine dayalı olan bu tekniklerdir. Bu tür tercih bilgisinin çözüm sürecinin başında verildiği yaklaşımlar ile optimum çözümlere ulaşmak veya yaklaşmak daha kısa zamanda ve daha kolay olmaktadır. Önsel tercih bilgisini kullanan yöntemlerin en büyük dezavantajı bir anlamda belirsizlik altında tercih yapmak olarak kabul edilen karar vericinin tercih bilgisini belirlemede yaşadığı güçlüktür. Önsel tercih bilgisini kullanan yöntemler: çok amaçlı ayrışım tekniği, ağırlıklandırma tekniği, doğrusal olmayan yaklaşım, bulanık mantık yaklaşımı, değer fonksiyonu, kabul edilebilirlik fonksiyonu, hedef programlama, ardışık sıralama tekniği ve sınırlandırılmış amaçlar tekniğidir.

Adımsal geliştirimli tercih bilgisini kullanan yöntemler: Tercih bilgisinin çözüm süreciyle birlikte kullanıldığı yaklaşımlarda, karar verici çözümün her aşamasında tercih yapabilir yani çözüm prosesinde karar verici ile çözümün aşamaları sürekli etkileşim halindedir. Bilgisayarın ürettiği aday çözümü karar vericiye sunan bu yöntemlerde eğer karar verici sonucu kabul ederse durulur, aksi taktirde karar verici sonucu kabul etmezse bilgisayar karar vericiye bir önceki çözümden daha iyi bir çözüm üretir. Karar verici ulaşılan çözümü kabul edinceye kadar veya daha iyi bir çözüm bulunamayana kadar sürdürülen bu yöntemler hiyerarşik ayrıştırma tekniği, STEM tekniği, çok amaçlı grafik teorisi, kısıtlama tekniği, parametre uzayı araştırma tekniği, rassal arama tekniği, vektör-gevşeme tekniği, etkileşimli □-şebeke tekniği, yerel geliştirme tekniği, pareto sınırlı haritalar ve STEUER tekniğinden meydana gelir.

2.4 Optimizasyon Problemlerinin Çözümü için Kullanılan Sezgisel Teknikler

Bu tez konusunun kapsamı dahilinde incelenen optimizasyon problemleri kombinatoryal optimizasyon problemleri ve bu problemleri çözen teknik ise metasezgisel tekniklerdir. Literatürde var olan ve optimizasyon problemlerinin

çözümünde sıklıkla kullanılan metasezgisel teknikler genetik algoritma, tabu arama, karınca koloni optimizasyonu, yapay sinir ağları ve parçacık sürü optimizasyonudur. Bu teknikler çözüm buldukları optimizasyon probleminin çeşidine göre performans açısından farklılık göstermektedirler.

Sürekli Optimizasyon ve Sezgisel Teknikler: Ayrık optimizasyon problemlerine nazaran daha kolay çözüm bulunabilen sürekli optimizasyon problemlerinin olumlu olan tarafı çözümün komşularının daha kolay aranabilmesi ile sağlanan hareket serbestliğidir. Ayrıca sürekli optimizasyonda iyi bir çözüme yakınsama ayrık optimizasyona göre daha kolay sağlanır. Yine de bu olumlu özelliklerine rağmen sürekli optimizasyon problemleri literatürde oldukça az sayıda çalışma ile araştırılmıştır. Sürekli optimizasyon problemlerinde tabu arama ve karınca kolonisi optimizasyonu yöntemlerinin genel olarak kullanılmadığı bilinmektedir.

Ayrık Optimizasyon ve Sezgisel Teknikler: Sıralama, çizelgeleme, rotalama, programlama gibi örneklendirilebilen problemlerin hepsi ayrık optimizasyon problemleridir. Bu problem türleri genel olarak literatürde var olan genetik algoritmalar, tabu arama, parçacık sürü optimizasyonu, karınca kolonisi optimizasyonu gibi sezgisel tekniklerle çözülebilmektedir. Ayrık optimizasyon problemlerinde çözümün nasıl gösterilerek temsil edeceği ve çözümün komşu çözümünün seçilmesi sıkıntıları yaşansa da bu problem türü ile yapılan çalışmalara sıklıkla rastlanılabilmektedir.

Bu çalışmasının konusu olan metasezgisel yöntemlerle elde edilen melez yapılar ayrık optimizasyon yöntemi içerisinde kullanım çeşitlerine ve yerlerine göre incelenmesi gerekmektedir.

2.5 Optimizasyon Problemleri için Oluşturulan Melez Yapılar

Literatüre bakıldığında tek-amaçlı optimizasyon problemlerinin çözümde kullanılan metasezgisellerin bir araya getirilmesi ile oluşturulmuş olan ve melez olarak adlandırılan yapılar söz konusudur. Oluşturulan bu yapılar ile problemlere çözüm aranmış ve genel olarak tek başına kullanılan metasezgisellere oranla melez yapıların daha iyi sonuçlar elde ettiği anlaşılarak bu konudaki çalışmalar gerek bir araya getirilen

metasezgisellerin değiştirilmesi ve gerekse uygulandıkları problemlerin farklı seçilmesi ile hızla devam etmiştir.

Literatürde var olan başlıca melez yapılar genetik algoritmalar, benzetimli tavlama algoritması, karınca kolonisi optimizasyonu, tabu arama algoritması ve parçacık sürü optimizasyonu ile bazı yerel arama sezgisellerin bir araya getirilmesi ile oluşturulmuştur.

Genetik Algoritma – Tabu Arama: Ramkumar ve diğerleri [6] da geliştirilmiş sürekli tabu arama algoritması ve seçkinlik temelli genetik algoritmadan oluşan önerdikleri melez yapı ile akıllı optimizasyon problemlerinde parametre tahminini gerçekleştirmişlerdir. Tabu arama algoritması çözümün ilk adımı olan küçük araştırma uzayının belirlenmesinde genetik algoritma ise çözümün ikinci aşaması olan yerel minimumun bulunmasında kullanılmış ve elde edilen simülasyon sonuçları kurulan yapının uygunluğunu kanıtlamıştır. Böylelikle matematiksel modelli sistemlerin girdi ve çıktı değerlerinin tanımlanabilmesi için mühendislik alanına önemli bir çözüm önerisi getirilmiştir.

Önerilen algoritmanın ilk aşamasında tabu arama sezgiseli aşağıdaki adımlar eşliğinde gerçekleştirilmiştir:

Adım 1. Homojen komşuların üretilmesi: x^j, çözüm uzayındaki herhangi bir nokta ise s^j tarafından üretilen N komşu, r_1 ise komşunun başlangıç yarıçapı olduğu yerde

$$\left\| x^i - s^j \right\| = \sqrt{\left(x_1^i - s_1^j \right)^2 + \left(x_2^i - s_2^j \right)^2 + \ldots + \left(x_n^i - s_n^j \right)^2} \tag{2.4}$$

$$\left(j-1 \right) r_1 \le \left\| x^i - s^j \right\| \le j r_1 \tag{2.5}$$

’dir.

Adım 2. Tabu listesi ile karşılaştırma: Önceki adımda üretilen her bir s^j $(j = 1, 2, \ldots, N)$ komşu tabu listesindeki $t^i (i = 1, 2, \ldots, N_1)$ elemanları ile karşılaştırılır ve eğer $\left\| t^i - s^j \right\| \le r_2$ $(i = 1, 2, \ldots, N_1)$ ise

$$\left\| t^i - s^j \right\| = \sqrt{\left(t_1^i - s_1^j \right)^2 + \left(t_2^i - s_2^j \right)^2 + \ldots + \left(t_n^i - s_n^j \right)^2} \tag{2.6}$$

hesaplanır ve ardından sorumlu s^j reddedilir.

Adım 3. Umut verilen liste ile karşılaştırma: N' tabu listesinde olmayan ve h^i *(i = 1, 2, …,* N'*)* şeklinde belirtilen komşuların sayısını gösterir. Bu elemanların her biri umut verilen listesi olarak tanımlanan ve p^j *(j = 1, 2, …,* N_2*)* şeklinde gösterilen değerler ile karşılaştırılır. Eğer *i = 1, 2, …,* N_2 için $\left\| p^i - h \right\|^i \le r_3$ ise

$$\left\| p^i - h^j \right\| = \sqrt{\left(p_1^i - h_1^j \right)^2 + \left(p_2^i - h_2^j \right)^2 + ... + \left(p_n^i - h_n^j \right)^2} \qquad (2.7)$$

hesaplanır ve h^j reddedilir.

Adım 4. En iyi komşunun bulunması: Tabu listesi içerisinde olmayan komşulardan minimum maliyet fonksiyonunu sağlayan belirlenir.

Adım 5. Tabu listesinin güncellenmesi: Tabu listesi içerisindeki en iyi komşu ilk giren ilk çıkar (FIFO) sırasına göre güncellenir.

Adım 6. Umut verilen listesinin güncellenmesi: Eğer en iyi komşu toplamdaki en iyi komşu ise yine FIFO sırasına göre liste güncellenir.

Adım 7. Dönüştürme sınırlaması: Eğer yapılan hareketlerde herhangi bir iyileşme sağlanamıyorsa farklılaştırma durdurulur.

Önerilen algoritmanın ikinci aşamasında ise genetik algoritma aşağıdaki adımlar eşliğinde gerçekleştirilmiştir:

Adım 1. Başlangıç parametreleri: Başlangıç popülasyonu *(Ipop)*, ilk iterasyon sonunda üretilen popülasyon *(pop)*, mutasyon oranı *(mut)* ve eşleştirilen kromozomların sayısı *(keep)* ile gösterilir.

Adım 2. Homojen popülasyonun üretimi: Başlangıç popülasyonunu oluşturmak için araştırma alanında *N* tane eleman üretilir. Burada vaat edilen alan p^j *(j = 1, 2, …, N)* ile *n* değişkenli maliyet fonksiyonu ise $p^j = \left(p_1^j, p_2^j, ..., p_n^j \right)$ ile gösterilir. Homojen başlangıç popülasyonunda, η araştırma alanının büyüklüğüne göre değeri belirlenmesiyle $\left\| p^j - p^i \right\| \ge \eta$ şeklinde bir eleman seçilir.

Adım 3. Kromozomların değerlendirilmesi: Araştırma alanından seçilen her bir p^i elemanı kromozom olarak adlandırılır ve tüm kromozomlar maliyet fonksiyonuna bağlı olarak değerlendirilir.

Adım 4. Genetik operasyon seçimi: Popülasyon içinden kromozomlar çiftleşmek için olasılıksal olarak seçilirler. Maliyet fonksiyonu içerisinde en düşük dağılım payına sahip olan kromozom seçilme eğilimi gösterir ve çaprazlama operasyonu sayısı olan ϕ aşağıdaki formül ile bulunur:

$$\phi = round\left(\frac{pop - keep}{2}\right) \tag{2.8}$$

Olasılık yoğunluk fonksiyonu ise

$$P_N(n) = \frac{n}{\phi^2} \tag{2.9}$$

ile bulunur.

Adım 5. Son: Çaprazlama ve mutasyon operatörlerinin tekrarlamalı olarak çalışması ile maksimum iterasyon sayısı sınırına ulaşılır ve durulur.

Önerilen melez yapı bilinen çoklu biçimli karşılaştırma problemleri ile test edilmiş ve parametre tahmin problemlerine uygulanmıştır. Simülasyon sonuçlarına göre parametre tahmininde önerilen melez yapı ile daha kesin sonuçlar elde edildiği bulunmuş ve yöntemin en küçük kareler algoritmasından daha tutarlı sonuçlar elde ettiği sonucuna varılmıştır.

Yu ve diğerleri de [7] de, açık araç rotalama problemlerinin çözümü için genetik algoritmanın paralel hesaplama ve global optimizasyon özelliği ile tabu arama algoritmasının araştırma yeteneğini ve hızlı yerel aramasını bir araya getirerek melez bir yapı oluşturmuşlardır. Bu melez yapıdaki temel düşünceye göre, müşteri taleplerine ve araçların kapasitelerine göre global optimizasyon problemi kodlanır ve oluşturulan popülasyondaki tüm bireyler tabu arama algoritmasının yerel arama sezgiselini gerçekleştirerek tüm müşterilerin araçlara rotalanması sağlanır. Yazarların ortaya koydukları melez yapının işleyiş adımları aşağıda verildiği gibidir:

Adım 1. Başlangıç parametreleri: Popülasyon büyüklüğü *pop_size*, genetik algoritmadaki maksimum üretilen gen sayısı *maxgen*, geliştirilmiş P_c (çaprazlama operatörü) ve P_m (mutasyon operatörü) parametreleri, tabu arama optimizasyonundaki bireylerin olasılığı P_{ts}, tabu arama ile elde edilen iyi sonuçların sayısı *best_s*, tabu listesi uzunluğu *ts_len* ve tabu iterasyon sayısı *ts_loop*'dır.

Adım 2. Kodlama metodu ile başlangıç popülasyonu oluşturulur.

Adım 3. Kod çözülür ve uygunluk değeri hesaplanır.

Adım 4. Genetik operatörler uygulanır.

Seçim ve uygunluk değerine göre yeniden üretim yapılır.

P_c uyabilen olasılığı hesaplanır ve çaprazlama operatörü işletilir.

P_m uyabilen olasılığı hesaplanır ve mutasyon operatörü işletilir.

Adım 5. $p = 1$ yapılır.

Adım 6. Eğer $p \le pop\_size$ (6.1) ile devam edilir, aksi halde Adım 7'ye gidilir.

6.1 Eğer $rand \le P_{ts}$ tabu arama için (6.2)'ye gidilir, aksi halde (6.9)'a gidilir.

6.2 Rastlantısal olarak geliştirilen başlangıç çözümü ile tabu arama listesi başlangıçta boşken doldurulmaya başlanır.

6.3 $k = 1$ yapılır.

6.4 Eğer $k \le ts\_loop$ ise (6.5)'e gidilir, aksi halde (6.8)'e gidilir.

6.5 2-opt komşu arama sezgiseli ile mevcut çözüm kullanılarak komşuluk çözümleri oluşturulur ve çözümler *best_s* aday çözümleri ile sınırlandırılır.

6.6 Aday çözümler istenen seviyede ise mevcut çözüm güncellenir ve (6.7)'ye gidilir, aksi halde tabu listesi güncellenir ve Adım 7'ye gidilir.

6.7 $k = k + 1$ yapılır ve (6.4)'e gidilir.

6.8 Tabu arama metodu ile optimize edilmiş yeni bir kromozom yapılandırılır.

6.9 *p = p + 1* yapılır ve Adım 6'ya gidilir.

Adım 7. *gen = gen + 1* yapılır.

Adım 8. Eğer *gen<=maxgen* ise bir sonraki jenerasyona gidilir ve yeniden genetik algoritma-tabu arama operasyonları iterasyon durana kadar araştırılır.

Algoritma adımlarından da anlaşılacağı üzere yazarlar açık araç rotalama probleminin içerisinde tabu arama metodunu bir yerel arama sezgiseli olarak kullanmış ve sadece gidilen müşterilerin tekrar ziyaretini önlemek için tabu listesini bir hafıza şeklinde yürütmüşlerdir.

Genetik Algoritma – Karınca Kolonisi Optimizasyonu: Tseng ve Chen [8] de yaptıkları çalışmada kaynak kısıtlı proje çizelgeleme probleminin çözümü için genetik algoritma ve karınca kolonisi optimizasyonundan oluşan ANGEL isimli melez bir yapı ortaya koymuşlardır. ANGEL melez yapısı içinde ilk olarak karınca kolonisi optimizasyonu çözüm alanını araştırır ve genetik algoritma için başlangıç popülasyon listesini oluşturur. Bu işlemin ardından genetik algoritma çalıştırılarak daha iyi bir çözüm elde ettiğinde feromon güncellemesi için değerler karınca kolonisi optimizasyonuna verilir ve karınca kolonisi optimizasyonu bu yeni feromon değerleri ile yeniden yürütülür. Yöntemde iki meta sezgiselin yanı sıra bir yerel arama prosedürü kullanılmıştır. Kendisini oluşturan genetik algoritma ve karınca kolonisi optimizasyonu yöntemlerinden hiçbirinin algoritma yapısının değiştirilmediği bu melez yapı içinde karınca kolonisi ile bulunan sonucun genetik algoritma ile iyileştirilmesine çalışılmıştır.

Lee ve diğerlerinin [9] da bir çalışmasında ise çoklu sıralama probleminin çözümü için yine genetik algoritma ve karınca kolonisi optimizasyonu yöntemleri önerilmiştir. Bu yönteme göre çözümün ilk aşamasında genetik algoritma yürütüldükten sonra karınca kolonisi optimizasyonu elde edilen sonucun iyileştirilmesi için bir yerel arama sezgiseli gibi çalıştırılmıştır.

Genetik Algoritma – Parçacık Sürü Optimizasyonu: Kuo ve Lin [10] da sipariş kümeleme problemi için genetik algoritma ve parçacık sürü optimizasyonundan oluşan

melez bir yapı ortaya koymuşlardır. Bu melez yapıyı oluşturan her iki teknik de popülasyon temelli stokastik arama prosesleridir. Melez yapıda rassal olarak *2N* büyüklüğünde bir başlangıç popülasyonu oluşturulur. Bireyler genetik algoritma için gen iken parçacık sürü optimizasyonu için de parçacıklardır. *2N* bireyler kodlanarak genetik algoritma operatörlerinden olan yeniden üretim, çaprazlama ve mutasyona tabii tutulduktan sonra *2N* bireyler uygunluklarına göre ayrılır ve en iyi *N* bireylere geliştirilmiş parçacık sürü optimizasyonu yöntemi uygulanır. Bu yöntemde optimum sonucu bulmak için başlangıç popülasyonu ile daha iyi araştırma yapılabilir. Bu melez yapıda *k*-ortalama algoritması da hızlı yakınsama amacını gerçekleştirmede kullanılır. Önerilen melez algoritma adımları aşağıdaki gibidir:

Adım 1. *2N* popülasyon büyüklüğü (parçacıkların sayısı), eylemsizlik ağırlığı *W*, maksimum hız V_{max} ve iki öğrenme faktörü c_1 ve c_2 parametreleri belirlenir.

Adım 2. Her bir parçacığa rassal olarak başlangıç pozisyonu X_{id} ve başlangıç hızı V_{id} atanır. Parçacık sürü optimizasyonu kümelemede *k* küme sayısının bilinmesi gerekmektedir. Her bir parçacık *k* küme vektörünü içerdiğinde kümelerin pozisyonu X_{id}

$$X_{id} = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ik}) \tag{2.10}$$

ile bulunur. Burada x_{ij} *i.* parçacık için *j.* kümenin ağırlık merkezidir.

Adım 3. Her bir parçacığın uygunluk değeri hesaplanır.

$$\text{Uygunluk değeri} = \sum_{j=1}^{k} \left| \sum_{\forall x \in c_{ij}} \left\| x - z_{ij} \right\| \right| \tag{2.11}$$

Burada *k* küme sayısı, *x* kümelenen veri vektörü, c_{ij} *j* kümesi içindeki *i* parçacığının veri vektörleri sayısı ve $\left\| x - z_{ij} \right\|$ veri vektörleri ve küme ağırlık merkezleri arasındaki öklid mesafedir.

Tüm küme ağırlık merkezlerinden $d(x,z_{ij})$ öklid mesafesi hesaplanır.

$$d(x,z_{ij}) = \left\| x - z_{ij} \right\|$$

x, z_{ij} kümesine $d(x,z_{ij}) = Min\forall c_{1,\ldots,Nc}\left\{d\left(x,z_{ij}\right)\right\}$ şeklinde atanır.

Uygunluk (2.11) numaralı denklem ile hesaplanır.

Adım 4. *2N* popülasyonuna kodlanmış genetik algoritma operatörleri (yeniden üretme, çaprazlama, mutasyon) işletilir ve başka *2N* popülasyon yaratılır.

Adım 5. Her bir *2N* bireyleri için uygunluk hesaplanır ve bireyler uygunluk değerleri temel alınarak sıralanır.

Adım 6. Global en iyi (P_{gd}) ve yerel en iyi pozisyonlar (P_{id}) güncellenir.

Adım 7. Melez parçacık sürü optimizasyonuna göre her bir P_{gd} ve P_{id}'ye çaprazlama operatörü uygulanır ve elde edilen yeni çocuk parçacıklar karşılaştırılarak en küçük uygunluk değerini sağlayan çocuklarla işleme devam edilir.

Adım 8. (a) V_{id} hızı güncellenir.

$$V_{eski}^{yeni} = W \times V_{id}^{eski} + c_1 \times rand_1 \times \left(P_{id} - X_{id}^{eski}\right) + c_2 \times rand_2 \times \left(P_{gd} - X_{id}^{eski}\right) \quad (2.12)$$

Burada c_1 ve c_2 iki pozitif sabit, $rand_1$ ile $rand_2$ [0,1] aralığında bulunan iki rassal fonksiyon ve W eylemsizlik ağırlığıdır.

(b) Mutasyon: V_{id} verimliliği güncellenir.

$$V_{eski}^{yeni'} = V_{eski}^{yeni} + rand \times N\left(0,1\right) \quad (2.13)$$

Adım 9. (a) En iyi *N* bireylerinin pozisyon vektörü X_{id} güncellenir.

$$X_{eski}^{yeni} = X_{id}^{eski} + V_{id}^{yeni} \quad (2.14)$$

(b) Mutasyon: En iyi *N* bireylerinin kullanımı X_{id} güncellenir.

$$X_{eski}^{yeni'} = X_{id}^{eski} + rand \times N\left(0,1\right) \quad (2.15)$$

Adım 10. *k*-ortalama operatörü *2N* popülasyonuna uygulanır ve farklı *2N* popülasyon oluşturulur.

$$z_{i}{}_{j} = \frac{1}{c_{ij}} \sum_{\forall x \in c_{ij}} x \quad (2.16)$$

Adım 11. Durma kriterlerinden biri sağlanıyorsa Adım 12'ye gidilir, aksi halde Adım 3'e geri dönülür.

Adım 12. Minimum uygunluk değeri ile elde edilen çıktı parçacık son jenerasyondur.

Önerilen melez yapı üretim performansını arttırmak için sipariş kümelemelerine uygulanmış ve hem üretim zamanı hem de makine bekleme zamanlarında ciddi düşüşler elde edilmiştir. Kullanılabilecek olunan farklı güncelleme kuralları ile daha iyi üretim sonuçları elde edilebilinecek ve çalışma müşteri ilişkileri yönetimi, pazar sınıflandırması gibi farklı alanlara da uygulanabilecek niteliktedir.

Abd-El-Wahed ve diğerlerinin [11] de yaptıkları çalışmada ise genetik algoritma ve parçacık sürü optimizasyonundan oluşan melez yapı ile doğrusal olmayan optimizasyon problemlerine çözüm aranmıştır. Çalışmada öncelikle araştırma alanı içinde seyahat edecek olan rassal parçacıklar seti oluşturulmuş ve bu parçacıkların performansı seyahatleri boyunca genetik algoritma ve parçacık sürü optimizasyonu tarafından değerlendirilmiştir. Ayrıca parçacıkların hızı da bu sayede kontrol edilmiş ve sınırlama faktörleri geliştirilmiştir. Oluşturulan melez yapı aşağıdaki işlem adımlarını içerir:

Adım 1. Başlangıç: Başlangıç parçacık popülasyonu rassal pozisyonları ile belirlenir ve hızları *n*-boyutlu problem alanı içinde yer alır.

Adım 2. Değerlendirme: Her bir parçacık için *n* değişkenli uygunluk fonksiyonu değerlendirilir.

Adım 3. P_{best} ve G_{best} ayarlanır. P_{best} her bir parçacık için ayarlanır ve bunun amaç değeri şimdiki pozisyonuna ve amaç değerine eşit olur. G_{best} ayarlanır ve bunun amaç değeri de en iyi başlangıç parçacığının pozisyonuna ve amaç değerine eşit olur.

Adım 4. Hızlar ve pozisyonlar güncellenir.

Hız için: $$v_i^{k+1} = w \times v_i^k + c_1 \times r_1 \times \left(p_i - x_i^k \right) + c_2 \times r_2 \times \left(p_g - x_i^k \right) \quad (2.17)$$

Pozisyonlar için: $$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2.18)$$

Burada *i = 1, 2, …, N* ve *N* popülasyon büyüklüğü, *w* eylemsizlik ağırlığı, c_1 ve c_2 iki pozitif katsayı, r_1 ve r_2 [0,1] aralığında düzgün dağılmış rassal iki sayıdır. v_i^{k+1} iterasyonda parçacığın yeni hızı, x_i^{k+1} *i.* parçacığın pozisyonudur.

Adım 5. Parçacıkların değerlendirilmesi: Hız χ faktörü ile kontrol edilir. Burada *i* parçacığı fizibil alanda x_i^k pozisyonunda ve v_i^{k+1} hızı ile bulunurken yeni pozisyonu x_i^{k+1} hızına bağlıdır.

$$x_i^{k+1} = x_i^k + v_i^{k+1}$$

v_i^{k+1} parçacığın fizibilitesi kaybolur ve yeni χ faktörü

$$\chi = \frac{2}{\left|-2-\tau-\sqrt{\tau^2+\tau}\right|} \tag{2.19}$$

ile bulunur. Burada τ fizibil olmayan parçacığın fizibil olmama süresidir. Parçacığın yeni iyileştirilmiş pozisyonu aşağıdaki formülle bulunur:

$$x_i^{k+1} = x_i^k + \chi v_i^{k+1} \tag{2.20}$$

Adım 6. Değerlendirme: Her bir parçacık için *n* değişkenli uygunluk fonksiyonu değerlendirilir.

Adım 7. P_{best} ve G_{best} güncellenir.

Adım 8. Sıralama: Bireyler (parçacıklar) amaç değerlerine göre sıralanırlar ve kolon vektörü bireysel uygunluk değerlerine çevrilir.

Adım 9. Seçim: Rassal olarak iyi birey seçilerek yeni çocuklar oluşturulur.

Adım 10. Çaprazlama: Yeni birey oluşumu sağlanır.

Adım 11. Mutasyon operatörü çalıştırılır.

Adım 12. Yeniden yerleştirme yapılır: Mevcut popülasyondan rastgele seçilen bireyler çıkartılır ve yerine önceki popülasyondan daha iyi bireyler eklenir.

Adım 13. Düzeltme: Fizibil çözümde yer alan fizibil olmayan bireyler düzeltilir. Böylelikle fizibil çözümü sağlayan fizibil bireyler elde edilir.

Global optimumun bulunmasında önemli bir katkıda bulunan bu melez yapıda başlangıç çözüm yaklaşımı parçacık sürü optimizasyonu ile verilmiş ve problem çözümü genetik algoritmanın kullanılmasıyla bulunmuştur.

Genetik Algoritma – Benzetimli Tavlama Algoritması: Chen ve Shahandashti [12] de çok kaynak kısıtlı proje çizelgeleme probleminin çözümü için genetik algoritma ve benzetimli tavlama yöntemlerinden oluşan bir yapı ortaya koymuşlardır. Ortaya konan bu yapı genetik algoritma ve geliştirilmiş benzetimli tavlama yöntemi ile kıyaslanmıştır. Bu melez yapı içerisinde, başlangıç çözümünün üretildiği genetik algoritma adımlarının gerçekleştirilmesinin ardından popülasyon içindeki kötü bireylerin operatörler sonucu elde edilen iyi bireylerle yer değiştirmesinin ardından elde edilen yeni popülasyona sıcaklığın azaltılması ile benzetimli tavlama yöntemi uygulanır ve maksimum iterasyon sayısına ulaşıldığında da algoritma durdurulur. Bu melez yapı içerisinde, *t* iterasyon sayısını gösterdiğinde uygunluk değerlendirme fonksiyonu

$$f_i = (D_{max} - D_i)^{\alpha x t} \tag{2.21}$$

olur. Bu melez yapı içinde jenerasyon büyüklüğü sayısı gibi mutasyon oranı da azalacaktır:

$$C_m = 1 - t/t_{max} \tag{2.22}$$

Burada *t* iterasyon veya jenerasyon sayısı, T_{max} ise maksimum jenerasyon sayısını gösterir.

Bu melez yapı içerisinde benzetimli tavlama yönteminin en büyük katkısı olarak döller popülasyon içinde en kötü çözümden daha az uyumlu ise, aşağıda verilen olasılık formülüne göre yeni döller en kötü çözüm içerisinde yer değiştirirler.

$$\delta \leq e^{(-\Delta E/T)} \tag{2.23}$$

Önerilen melez yapının test edildiği problem çözümünde genetik algoritma, benzetimli tavlama algoritması ve geliştirilmiş benzetimli tavlama yöntemleri ile kıyaslanan melez yapı sonuçları, uygulandıkları problem setleri için ikinci en iyi çözümü sunmuştur.

Tabu Arama – Parçacık Sürü Optimizasyonu: Shen ve diğerleri [13] de tümörlerin sınıflandırılması için parçacık sürü optimizasyonundan ve tabu arama metodundan oluşan melez bir yapı önermişlerdir. Tabu arama algoritması yerel arama prosedürü olarak hem yerel optimumu araştırır hem de sistem performansını iyileştirmeye çalışır. Üç farklı mikro cihaz veri setlerine uygulanan melez yöntemin (HPSOTS) yapısı aşağıda verildiği gibidir:

Adım 1. HPSOTS için ikili kodlama ile IND başlangıç zinciri rassal olarak oluşturulur ve IND bireyinin uygunluk değeri hesaplanır.

Adım 2. Parçacık sürü optimizasyonunun bilgi paylaşım mekanizması ile IND bireyinin komşularının %90'ı üretilir ve değerlendirilir.

Eğer $(0 < v_{id} < a)$ ise $x_{id}(yeni) = x_{id}(eski)$

Eğer $a < v_{id} \leq \frac{(1+a)}{2}$ ise $x_{id}(yeni) = p_{id}$

Eğer $\left(\frac{(1+a)}{2} < v_{id} \leq 1 \right)$ ise $x_{id}(yeni) = p_{gd}$

Burada, x_i *i.* parçacığın pozisyonu, p_g en iyi parçacığın pozisyonu, v_i *i.* parçacığın hızı, p_i *i.* parçacığın uygunluk değeri ve a ise rassal bir değerdir.

Adım 3. Amaç kriterine ve tabu şartlarına göre komşulardan yeni bireyler toplanır ve IND popülasyonu güncellenir.

Adım 4. Yerel optimumu tamamlayan HPSOTS yeteneği geliştirilir ve IND içindeki %10'dan rassal olarak en iyi iki parçacık uçar ve bu %10 parçacık için uygunluk fonksiyonu hesaplanır.

Adım 5. Prosesin sonunda en iyi amaç fonksiyonu bulunmuşsa öğrenme durdurulur, aksi halde Adım 2'ye gidilir.

Tümörlerin sınıflandırılmasında gen seçimi için geliştirilen bu melez yapı ile parçacık sürü optimizasyonu problemin çözümünü sağlamış tabu arama algoritması için bir yerel

arama sezgisel olarak kullanılarak gen seçimi ve yüksek boyutlu verilerin incelenmesinde yardımcı bir yapı olmuştur.

Zhang ve diğerleri de [14] de yaptıkları çalışmada çok amaçlı akış tipi çizelgeleme probleminin çözümü için parçacık sürü optimizasyonu ve tabu arama algoritmasından oluşan melez bir yapı ortaya koymuşlardır. Yazarlar bu yapıda parçacık sürü optimizasyonunu yerel arama ile birleştirilirken, tabu arama sezgiseli ile de kombinatoryal optimizasyon problemi için yakın optimum çözümleri araştırmışlardır. Önerilen algoritma yapısı aşağıda verildiği gibidir:

Adım 1. Parametre değerleri ayarlanır: Popülasyon büyüklüğü P_s, maksimum iterasyon sayısı *Gen*, *w* (eylemsizlik ağırlığı), c_1 ve c_2 (iki pozitif sabit sayı) ayarlanır.

$$v_{td} = w \times v_{td} + c_1 \times rand(\ ) \times (p_{ld} - x_{td}) + c_2 \times rand(\ ) \times (p_{gd} - x_{td}) \quad (2.24)$$

Burada *x* parçacığın kendisini, p_l ve p_g *t.* parçacığının iyi değerlerini ve *rand( )* ise [0, 1] aralığında bulunan iki rassal fonksiyonu ifade eder.

MaxIterNum (iterasyon sayısı) değeri, *CurInterNum = 0* (mevcut iterasyon sayısı) ve $t = \varnothing$ ayarlanır.

Adım 2. Stokastik olarak popülasyon başlatılır.

Adım 3. Sürü içindeki her bir parçacığın amaç değeri hesaplanır ve kıyaslanır. Ardından parçacığın kopyası olan *pbest* pozisyonu ve sürü içindeki en düşük uygunluklu *gbest* pozisyonu ayarlanır.

Adım 4. Belirlenen iterasyon sayısına ulaşılmadıysa Adım 5 ile devam edilir, aksi halde Adım 9'a gidilir.

Adım 5. *CurIterNum = CurIterNum + 1* yapılır.

Adım 6. Bilgi değiştirilir.

Adım 7. Mevcut parçacık akış tipi çizelgeleme problemine dönüştürülür ve tabu arama metodu başlatılarak tabu arama ile bulunan s^* yeni değeri bu mevcut parçacık ile yer değiştirilir.

Adım 8. Sürü içindeki parçacıklar değerlendirilir ve mevcut parçacık kendi deneyimine göre veya tüm popülasyonun deneyimine göre güncellenir ve Adım 4'e geri dönülür.

Adım 9. Çıktı değerleri elde edilir.

Önerilen melez yapı çok amaçlı akış tipi çizelgeleme problemleri için etkili ve verimli sonuçlar elde etmiştir. Özellikle büyük boyutlu problem tipleri için önce parçacık sürü optimizasyonunun çalıştırılması, ardından da tabu arama algoritması ile yerel arama prosedürünün gerçekleştirilmesi ile sonuçlar bulunmuştur.

Tabu Arama – Benzetimli Tavlama Algoritması: Ongsakul ve Bhasaputra [15] de esnek aktarma sistemleri ile optimum güç akışının kontrolünde benzin maliyetinin minimize edilmesi probleminin çözümü için tabu arama ve benzetimli tavlama algoritmalarından oluşan mezel bir yapı önermişlerdir. Problem parametreleri bu melez yapı ile belirlenirken, optimum güç akışı parametreleri ise kuadratik programlama ile çözülmüştür. Önerilen bu melez yapıda tabu arama genel aramayı oluştururken, benzetimli tavlama ile deneme komşu çözümü üretilir. Benzetimli tavlama yöntemi, tabu arama yönteminin aspirasyon aşamasında kullanılır. Önerilen melez yapının işlem adımları aşağıda verildiği gibidir:

k_u: Ulaşılan en iyi amaç fonksiyonu (F_B) geliştirilmedenki iterasyon sayacı

k_t : Deneme komşuluk çözümü vektör sayacı

S_B: Ulaşılan en iyi çözüm vektörü

$S_T^{(k+1,0)}$: *k + 1* iterasyonundaki başlangıç çözüm vektörü

$\left\{S_T^{(k,m)}\right\}$: *k* iterasyonundaki *M* deneme komşuluk çözüm vektörlerinin seti, *m =1,2,…,M*.

$\left\{SS_T^{(k,m)}\right\}$: Amaç fonksiyonlarına göre sıralanmış *k* iterasyonundaki *M* deneme komşuluk çözüm vektörlerinin seti, *m = 1, 2, …, M*.

S_C^k: $\left\{SS_T^{(k,m)}\right\}$ seti ile ayrılmış *k* iterasyonundaki mevcut komşuluk çözüm vektörü

F(S): Çözüm vektörünün amaç fonksiyonu

F_B: Ulaşılan en iyi amaç fonksiyonu

TL: Tabu listesi

$k_{u,max}$: F_B'nin gelişimi dışında belirlenmiş maksimum izin verilen iterasyon sayısı

k_{max}: İzin verilen iterasyonların maksimum sayısı

M: Belirlenmiş deneme komşuluk çözüm vektörlerinin sayısı

Adım 1. Sistem verileri okunur ve yüklenir.

Adım 2. T_1 ve *r* belirlenir.

Adım 3. k_{max}, $k_{u,max}$, *M* ve *TL*'nin büyüklüğü belirlenir.

Adım 4. *TL* boştur. $k = 1$ ve $k_u = 1$ olarak ayarlanır.

Adım 5. $S_T^{(k,0)}$ rassal olarak oluşturulur. $S_B = S_T^{(k,0)}$ ve $F_B = F(S_T^{(k,0)})$ olarak ayarlanır.

Adım 6. T_k,

$T_k = r^{(k-1)}.T_1$ ile belirlenir. (2.25)

Adım 7. $S_T^{(k,m)}$'yi oluşturmada

$$S_T^{(k,m)} = S_T^{(k,0)} + \left(T_k.[U].F_u\right) \quad (2.26)$$

ile *M* deneme çözüm vektörleri rassal olarak oluşturulur.

Adım 8. $\left\{SS_T^{(k,m)}\right\}$ içinde ilk vektör S_C^k'ya ve $k_T = 1$'e ayarlanır.

Adım 9. Eğer $F(S_C^k) < F_B$ ise $F_B = F(S_C^k)$, $S_B = S_C^k$ ve $k_u = 1$ olarak ayarlanır, aksi halde $k_u = k_u + 1$ yapılır.

Adım 10. *TL* kontrol edilir. Eğer S_C^k *TL* içinde değilse *TL* içinde S_C^k güncellenir, $S_T^{(k+1,0)} = S_C^k$ ayarlanır ve Adım 13'e gidilir.

Adım 11. Kabul kriteri kontrol edilir. Eğer $p^k > u$ ise $S_T^{(k+1,0)} = S_C^k$ ayarlanır ve Adım 13'e gidilir.

Adım 12. Eğer $k_T < M$ ise $\left\{SS_T^{(k,m)}\right\}$ içinde sıradaki vektör S_C^k şeklinde ayarlanır, $k_T = k_T + 1$'e ayarlanır ve Adım 10'a gidilir, aksi halde $S_T^{(k+1,0)} = S_C^{(k,0)}$'a ayarlanır.

Adım 13. Eğer $k < k_{max}$ ve $k_u < k_{u,max}$ ise $k = k + 1$ olarak ayarlanır ve Adım 6'ya gidilir, aksi halde proses sınırlandırılır ve S_B problemin çözümü olur.

30 adet otobüs sisteminin çözümüne uygulanan melez yapı ile daha iyi sonuçlar elde edilmiş ve bu sonuçlara ulaşırken melez yapı genetik algoritma, benzetimli tavlama ve tabu arama algoritmalarından daha az işlem süresi gerektirmiştir.

Swarnkar ve Tiwari [16] da ise esnek imalat sistemleri içindeki makine yükleme probleminin çözümü için sistem dengesizliklerinin minimizasyonu ve makine zamanı ve parça dilimleri gibi teknolojik kısıtların varlığında üretilen işin maksimizasyonunu sağlamak için tabu arama ile benzetimli tavlama yöntemlerini bir araya getirmişlerdir. Kurulan melez yapıda tabu listesi ile kısa dönemli hafıza sağlanırken çözümün elde edilmesinde benzetimli tavlama metodunun stokastik doğasından faydalanılır.

Yinelemeli arama metotlarından tabu aramada çözümün kabulü için şartların kontrol edildiği istek ve tüm yolların araştırılmasını sağlayan çeşitlilik faktörleri vardır. Buna göre tabu arama algoritması aşağıda verilen adımları içerir:

Adım 1. Başlangıç çözümü *S* oluşturulur.

Adım 2. Başlangıçta tabu listesi *TL* boştur.

Adım 3. *N* düğümü için aşağıdaki adımlar gerçekleştirilir:

S bozulur ve S'' elde edilir.

Eğer $f(S') > f(S)$ veya $f(S') > istek$ ise $S = S'$; $TL \longleftarrow S$

Eğer $f(S) \geq f(S_{Best})$ ise $S_{Best} = S$ olur.

Adım 4. S_{Best} en iyi çözüm olarak elde edilir.

Melez yapıdaki benzetimli tavlama algoritmasında ise

Adım 1. Başlangıç çözümü *S* oluşturulur.

Adım 2. Başlangıç sıcaklığı $T > 0$ belirlenir.

Adım 3. Donmadan

Sıradaki *n* düğümü için

S'nin rassal komşusu S' bulunur.

$\Delta = f(S') - f(S)$ belirlenir.

Eğer $\Delta < 0$ ise $S = S'$ olur.

Eğer $\Delta \geq 0$ ise $P(\Delta, T)$ olasılığı ile $S = S'$ olur.

Eğer $f(S) \geq f(S_{Best})$ ise $S_{Best} = S$ olur.

T = yeni sıcaklık derecesi olarak belirlenir.

Adım 4. S_{Best} en iyi çözüm olarak bulunur.

Önerilen melez yapı konuyla ilgili 10 problem seti üzerinde test edilmiş ve sonuç karşılaştırmaları yapılmıştır. Büyük yapıdaki problemlerin çözümünde karmaşıklık yaratabilen bu melez yapı amaç fonksiyonlarının değiştirilmesi ve makine esnekliklerinin ölçümü, geri dönüşlerin belirlenmesi gibi faktörlerle daha da verimli hale getirilebilinir.

Karınca Kolonisi Optimizasyonu – Benzetimli Tavlama: Ji ve diğerleri [17] de karınca kolonisi optimizasyonu temelli Bayesian öğrenme ağları için melez bir yapı sunmuşlardır. Bu melez yapı içerinde bağımlılık analizleri, karınca kolonisi optimizasyonu ve benzetimli tavlama stratejisi bir araya getirilmiştir. İlk olarak, araştırma alanının boyutunu küçültmek için bağımsızlık testleri yapılmış ve en yakın optimum çözüme ulaşım zamanı kısaltılmaya çalışılmıştır. İkinci olarak, Bayesian ağları karınca kolonisi optimizasyonu ile üretilmiş ve son olarak da benzetimli tavlama yöntemi ile karıncaların bulduğu çözüm çerçevesinde geliştirmeler araştırılmıştır.

Karınca Kolonisi Optimizasyonu – Parçacık Sürü Optimizasyonu: Shelokar ve diğerleri [18] de yaptıkları çalışmada geliştirilmiş sürekli optimizasyon için karınca kolonisi optimizasyonu ve parçacık sürü optimizasyonundan oluşan melez bir yapı önermişlerdir. Bu melez yapıyı oluşturan her iki yöntemde işbirlikçi, popülasyon temelli

ve global arama yapabilen algoritmalardır. Yazarlar tarafından oluşturulan melez yapıda parçacık sürü optimizasyonu metodunun performansının arttırılması için feromon temelli bir mekanizma geliştirilmiştir.

İki aşamalı olarak yürütülen bu yapıda ilk aşama da parçacık sürü optimizasyonu gerçekleştirildikten sonra ikinci aşamada da basit feromon temelli karınca kolonisi optimizasyonu yerel aramaya uygulanmıştır.

Algoritmada, her bir *i* karıncası *t* iterasyonunda üretilen tüm parçacıklar arasından en iyi global pozisyona sahip p_t^g çevresinden z_t^i çözümünü üretir.

$$z_t^i = \mathrm{N}\left(p_t^g, \sigma\right) \tag{2.27}$$

Bu denkleme göre Gaussian dağılımına uygun olarak p_t^g ortalamalı ve σ standart sapmalı çözüm vektörü z_t^i elde edilir. Başlangıçta *t = 1* ve σ *= 1*'dir ve σ her iterasyon sonunda $\sigma = \sigma$ *x d* ile güncellenir. Burada *d* bir parametredir ve eğer $\sigma < \sigma_{min}$ ise $\sigma = \sigma_{min}$'dur. Amaç fonksiyonu olan f(z_t^i) z_t^i kullanılır ve *f(* z_t^i *)* < *f(* x_t^i *)* $x_t^i = z_t^i$, $f(x_t^i) = f(z_t^i)$ ise sürü içindeki *i* parçacığının mevcut x_t^i pozisyonu ile yer değiştirir. Basit feromon temelli mekanizma, yüksek iz yoğunluğuna göre herhangi bir *t + 1* iterasyonunda yapılır ve tüm karıncalar global en iyi çözümün komşulukları arasından daha iyi bir çözüm araştırır.

Önerilen melez yapının test edilmesi ile elde edilen verilere göre de fizibil çözüm alanının araştırılması ve optimum ya da optimuma yakın çözümlerin bulunmasında karınca kolonisi optimizasyonu parçacık sürü optimizasyonuna yardımcı olmuştur.

Benzetimli Tavlama – Parçacık Sürü Optimizasyonu: He ve Wang [19] da kısıtlı optimizasyon probleminin çözümü için oluşturdukları melez yapıda ceza fonksiyonu metoduna karşılık, ileri parametreler gerektirmeyen kurallar ve hızlı fizibil çözüm üretimi nedeniyle parçacık sürü optimizasyonunu kullanmış ve zamanından önce oluşabilecek olan bir yakınsamayı engellemek için parçacık sürü optimizasyonu ile bulunan çözüme benzetimli tavlama algoritmasını uygulamışlardır.

Çalışmada fizibil olmayan çözümün kısıt ihlal değeri

$$viol(x) = \sum_{j=1}^{N} \left[\max\left(g_j(x), 0 \right) \right] \tag{2.28}$$

formülüyle hesaplanır. Burada $P_i(k)$, *k.* jenerasyondaki *i* parçacığının *pbest* değerini ve $X_i(k+1)$ ise *(k+1)*. jenerasyondaki *i* parçacığının üretilen yeni pozisyonunu ifade eder. Strandart bir parçacık sürü optimizasyonunda eğer $f(X_i(k+1)) < f(P_i(k))$ ise $P_i(k+1) = X_i(k+1)$'dir. Oluşturulan melez yapıda ise fizibilite temelli kural çalıştırılır ve $P_i(k)$ değeri aşağıdaki senaryoya göre $X_i(k+1)$ ile yer değiştirir:

(1) $P_i(k)$ fizibil değil, fakat $X_i(k+1)$ fizibildir.

(2) $P_i(k)$ ve $X_i(k+1)$ fizibildir, fakat $f(X_i(k+1)) < f(P_i(k))$'dir.

(3) $P_i(k)$ ve $X_i(k+1)$ fizibil değildir, fakat $viol(X_i(k+1)) < viol(P_i(k))$'dir.

Benzer olarak *gbest* değeri de her iterasyonda güncellenir. Melez algoritmanın ikinci aşamasında ise yerel aramayı ifade eden *gbest*'in geliştirilmesi için benzetimli tavlama algoritması temelli bir yerel arama prosedürü gerçekleştirilir. *k* jenerasyonunda, $P_g(k)$ popülasyonun *gbest* değerini ve p_a ise üretilen yeni çözümün kabul edilme olasılığını ifade ederken benzetimli tavlama algoritma adımları aşağıda verildiği gibi işletilir:

Adım 1. $m = 1,\ P'_g = P_g(k)$

Adım 2. Yeni çözüm aşağıda verilen denkleme göre bulunur.

$$x' = P'_g + \eta \times \left(X_{\max} - X_{\min} \right) \times N(0,1) \tag{2.29}$$

Adım 3. p_a değeri aşağıda verilen kriterlere göre hesaplanır

(1) Eğer x' fizibil ve P'_g fizibil değil ise, $p_a = 1$

(2) Eğer P'_g fizibil ve x' fizibil değil ise, $p_a = 0$

(3) *Eğer* x' ve P'_g fizibil ise p_a,

$$p_a = min\left\{ 1 \backslash \exp\left[\left(f\left(P'_g\right) = f\left(x'\right) \right) / t(k) \right] \right\} \tag{2.30}$$

(4) *Eğer* x' ve P'_g fizibil değil ise p_a,

$$p_a = min\left\{ 1 \backslash \exp\left[\left(viol\left(P'_g\right) = viol\left(x'\right) \right) / t(k) \right] \right\} \tag{2.31}$$

Adım 4. Eğer $p_a \geq U[0,1]$ ise $P_g' = x'$

Adım 5. *m = m+1* yapılır. Eğer *m > L* ise durulur ve P_g' çıktı değeri yeni *gbest* değeri olarak belirlenir, aksi halde Adım 2'ye geri dönülür.

Melez yapı ile ilgili simülasyon sonuçları ve karşılaştırmalara göre sonuçlar diğer tekniklerle karşılaştırıldığında iyi olsa da esneklik temelli kurallarla daha iyi uyum sağlaması açısından diferansiyel değerlendirme gibi sezgiseller kısıtlı optimizasyon problemlerine uygulanabilir.

Liu ve diğerleri ise [20] de oluşturdukları melez yapıda paralel birimli ve farklı envanter depolama politikalarına sahip çok basamaklı akış tipi çizelgeleme problemini parçacık sürü optimizasyonu ile çözmüş ve etkili yerel aramayı ise benzetimli tavlama sezgiseli ve uyarlamalı Meta-Lamarckian öğrenme stratejisi temelli Nawaz-Enscore-Ham sezgiseli ile gerçekleştirmiştir.

Çözüm aşamasında standart parçacık sürü optimizasyonunun kullanıldığı bu çalışmada parçacığın mevcut pozisyonu ile komşularının en iyi pozisyonlarına göre önceki hızına bağlı olan yeni hızının hesaplanmasında aşağıdaki formül kullanırken,

$$V_{i,j}(t+1) = wv_{i,j}(t) + c_1r_1(pb_{i,j} - x_{i,j}(t)) + c_2r_2(pb_{g,j} - x_{i,j}(t)) \qquad j = 1, 2, \ldots, d \qquad (2.32)$$

parçacıkların yeni pozisyonlara uçması içinde aşağıda verilen formül kullanılır:

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \qquad j = 1, 2, \ldots, d \qquad (2.33)$$

Standart parçacık sürü optimizasyonu metodunun durma kuralı sağlandığında çözümün ikinci aşaması olan tabu arama meta-Lamarckian öğrenme stratejisi çalıştırılır. Geliştirilmiş meta-Lamarckian öğrenme stratejisinde benzetimli tavlama temelli yerel arama prosedürü iki aşamada gerçekleştirilir:

Öğrenme aşaması

Adım 1. Benzetimli tavlama temelli yerel arama sezgiselinin uygulanması ile farklı komşulara *n(n-1)* adımda uygulanabilmesi için tek bir şansı vardır.

Adım 2. Her bir komşunun η_1 ödül değeri hesaplanır.

$$\eta = \frac{|pf - cf|}{n(n-1)} \tag{2.34}$$

Öğrenmeme aşaması

Adım 1. Her bir komşunun $prob_{ut}$ kullanım olasılığı belirlenir.

$$prob_{ut,i} = \frac{\eta_i}{\sum_{j=1}^{K} \eta_j} \tag{2.35}$$

Adım 2. Hangi komşuya benzetimli tavlama temelli yerel arama sezgiselinin uygulanacağına rulet tekeri seçim yöntemi ile karar verilir.

Adım 3. Eğer *i.* komşu kullanılmışsa ödül $\eta_i = \eta_i + \Delta\eta_i$ ile güncellenir. *n* yığınları, *pf* eski permütasyonun amaç değerini, *cf* yerel arama ile bulunan en iyi permutasyonun amaç değerini, *i.* komşunun ödül değerini ve *K* toplam komşuluk sayılarını gösterdiği bu güncelleme de $\Delta\eta_i$ (2.34) numaralı denkleme göre hesaplanır.

Sonuç olarak, yapılan literatür araştırması ışığında birçok farklı optimizasyon probleminin çözümü için birçok yöntemin geliştirildiği, farklı meta sezgisellerin bir araya getirildiği ve kurulan yeni yapıların farklı problem türlerine uyarlandığı görülmüştür. Bu çözüm yöntemleri ile bazı optimizasyon problemleri için daha iyi sonuçlar elde edilirken, bazı problem türleri için ise yükselen hesaplama zamanları ve azalan esneklik ile problemlere çözüm arayışında ulaşılan başarı sınırlanmıştır.

Genel olarak literatürde var olan melez yapıların çoğuna bakıldığında optimizasyon problemleri için önerilen çözümlerin genel olarak iki aşamadan oluşturulduğu yapıları kuran metasezgisellerden birinin kullanılması ile bulunan çözümün bir başka metasezgiselle geliştirildiği görülmektedir. Birbiri içerisinde çalışan sistemlerden ziyade birbirini takip eden çözüm aşamalarını oluşturan bu melez yapılar ile ilgili çalışmalar günümüzde de devam etmektedir. Bu tez çalışması kapsamında ortaya konan genetik algoritma ve karınca kolonisi optimizasyonundan oluşan melez yapı Bölüm 3'de açıklanacaktır.

BÖLÜM 3

GENETİK ALGORİTMA VE KARINCA KOLONİSİ OPTİMİZASYONUNUN BİR ARAYA GETİRİLMESİ İLE OLUŞMUŞ ÖNERİLEN MELEZ YAPI

Bir önceki bölümde verilen bilgiler ve örnekler ışığında bu tez çalışması kapsamında ortaya konacak olup bir optimizasyon probleminin çözümünde kullanılmak üzere önerilen melez yapı genetik algoritma ve karınca kolonisi optimizasyonundan oluşacaktır.

Yapılan literatür taraması ışığında esnek olması, yapısı içerisinde gerçekleştirilebilecek yeni kanal ekleme veya çıkarma gibi değişikliklerin kolayca adapte edilebilmesi ve optimizasyon problemlerinin çözümünde yaygın kullanım alanına sahip olması avantajları göz önüne alınarak bu tez çalışmasında kullanılacak meta sezgisellerden ilki karınca kolonisi optimizasyonu olarak belirlendi. Daha sonra bu metasezgisel ile bir araya getirilecek olan diğer yapılar incelendiğinde parçacık sürü optimizasyonunun zaten karınca kolonisi ile benzer yapıda popülasyon temelli bir algoritma olmasından ve tabu arama algoritmasının da zaten tabu listesi ile karınca kolonisi optimizasyonun içerisinde yer almasından dolayı karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arayan genetik algoritma olarak karar verildi.

Önerilen melez yapı birbiri ile uyumlu olarak çalışan iki metasezgisel yöntemden oluşur. Bu yapıya göre genetik algoritma, bir optimizasyon probleminin çözümü için üreteceği rassal başlangıç çözümünün ardından tüm operatörleri ile yürütülerek genetik algoritmanın durma kuralı ile ulaşılan sonuç popülasyonu karınca kolonisi optimizasyonuna vererek probleme nihai bir sonuç aranmasını sağlayacaktır. Bu

noktada kurulacak olan melez algoritma, genetik algoritma temelinde başlatılarak elde ettiği sonucu işlemesi, geliştirmesi ve sonlandırması için karınca kolonisi optimizasyonuna verecektir.

Canlılarda bulunan genetik gelişimi simüle eden bir teknik olan genetik algoritma, popülasyon temelli bir algoritma olan karıncalar ile uyum içinde çalışacak ve problemlere birlikte çözüm arayacağı melez yapı içerisinde genel hatları ile aşağıda verildiği gibi uygulanacaktır:

Adım 1. Başlangıç popülasyonu belirlenir.

Adım 2. Her bir kromozomun uygunluk değeri hesaplanır.

Adım 3. Araştırma prosesi işletilir:

Seçim: Tekrar üretim için en iyi sıralamalı kromozomlar seçilir.

Çaprazlama: Rassal olarak iki kromozom ailesi seçilerek bunlardan yeni bireyler oluşturulur.

Mutasyon: Belirli sayıda kromozom mutasyon işlemi için rassal olarak seçilir.

Adım 4. Üretilen maksimum popülasyon sayısına ulaşıldığında en iyi uygunluk değerine sahip olan çözüm en iyi çözüm ise Adım 5'e gidilir, aksi halde Adım 3'e geri dönülür.

Adım 5. Maksimum iterasyon sayısı, karınca sayısı *A*, geçiş kuralını belirlenmesinde kullanılan parametre q_0, α, β parametreleri belirlenir.

Adım 6. Tüm kenarlara başlangıç feromon değeri atanır.

$$\tau_0 = \left(n * L_{nn}\right)^{-1} \tag{3.1}$$

Adım 7. *k = 1*'den *A*'ya kadar

Adım 8. *i = 1* için aşağıdaki işlemler gerçekleştirilir.

Tüm olası hareketler seçilir.

Tüm amaç fonksiyonları için tüm olası hareketlerin çekiciliği hesaplanır.

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3.2}$$

Burada η_{ij}, seçilebilirlik parametresi ve d_{ij} karıncanın gideceği *i* ve *j* noktaları arasındaki öklid mesafesidir.

$$d_{ij} = \left[\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2\right]^{1/2} \tag{3.3}$$

Tüm olası hareketlerin olasılığı hesaplanır.

$$p_{ij}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}, & \text{eğer } j \in izin \text{ verilen değerler} \\ 0, & \text{aksi halde} \end{cases} \tag{3.4}$$

Burada $\tau_{ij}(t)$, *t* anında *i* ve *j* noktaları arasındaki (*i,j*) hattında depolanan feromon maddesi miktarı ve α ile β parametreleri de kullanıcıya, olasılık değerini hesaplarken yapay feromon maddesi ile seçilebilirlik arasında nispi önemi belirleme imkanı veren parametrelerdir.

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t, t+1) \tag{3.5}$$

ρ, buharlaşma katsayısı olarak tanımlanan bir parametredir.

Atama yapılır.

Adım 9. Yapılandırılan çözüme göre amaç fonksiyonu hesaplanır.

Adım 10. Feromon güncellemesi yapılır.

Yerel feromon güncelleme:

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} 1/L^{k}(t+1) & k. \text{ karınca } (i,j) \text{ yolunu kullanmışsa} \\ 0 & \text{aksi halde} \end{cases} \tag{3.6}$$

$L^k(t + 1)$, *k* karıncasının toplam tur uzunluğudur.

Global feromon güncelleme:

$$\Delta\tau_{ij}(t+) = \begin{cases} 1/L_{best}(t+1) & (i,j) \text{ en iyi tura ait ise} \\ 0 & \text{aksi halde} \end{cases} \quad (3.7)$$

$L_{best}(t + 1)$ geçerli iterasyonda bulunan en iyi turun uzunluğudur.

Adım 11. Eğer ulaşılan iterasyon sayısı < maksimum iterasyon sayısı ise Adım 7'ye geri dönülür, aksi halde durulur.

Şimdi sırasıyla önerilen melez algoritma yapısını oluşturan metasezgiseller gerek yapıları ve gerekse problemlere ürettikleri çözüm yolları ile incelenecektir.

3.1 Genetik Algoritma

Genetik Algoritmalar (GA) uyarlanabilir sezgisel arama algoritmalarıdır. Bu algoritmalar, genetik ve doğal seleksiyonu temel alan evrimsel algoritmaların başını çekmektedirler. Genetik algoritmalar temel olarak evrimsel sistemin doğal işleyişini canlandırabilecek şekilde biçimlendirilmiştir (Cura [21]). Genetik algoritmalar genel olarak problem çözümlerinin genetik sunumu, çözümlerin başlangıç popülasyonunun yaratılma yolu, sahip oldukları uygunluklara göre çözümleri sıralayan değerlendirme fonksiyonu, tekrar üreme boyunca yavruların genetik bileşimlerinin değiştirilmesinde kullanılan genetik operatörler ve genetik algoritmanın değişkenleri için değerler şeklinde verilebilecek olan verilen beş temel bileşeni içerirler (Gen ve Cheng [22]).

Genetik algoritma prensibi, esinlendiği doğal seçilim ilkesine uygun olarak, problemin çözümü olabilecek mümkün alternatifler kümesini, var olan tasarım sınırlayıcıları ve koşulları altında hayatta kalmak için mücadele eden canlı topluluklarına benzetir. En iyinin hayatta kalmasının hedeflendiği bu benzetimde, genetik algoritma en uygun çözüme ulaşan stokastik ve global bir optimizasyon yöntemidir (Toğan ve Daloğlu [23]). Genetik algoritmalar, depolanmış bilgi, hedefe giden yollar ve çözümler için arama yapan bir tekniktir (Mitchell [24]). Genetik algoritmanın sahip olduğu özellikler aşağıda verildiği gibi sıralanabilir (Wang ve Lu [25]):

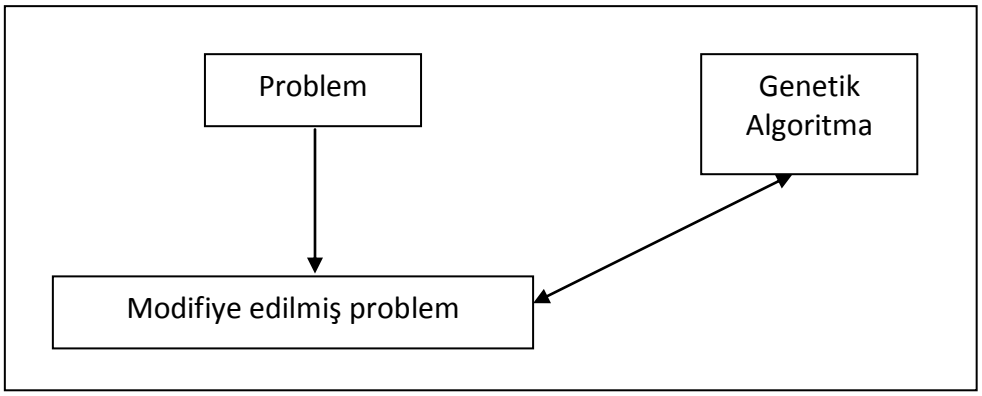
- Genetik algoritma hesapları parametre değerlerinden çok kodlanmış parametreler temellidir.
- Genetik algoritmalar, yerel optimuma yakalanmaktan kaçınarak yüksek paralel arama yeteneğini ortaya çıkarırlar.
- Genetik algoritmalar, hesaplanmak zorunda olan uygunluk fonksiyonu dışında karmaşık matematiksel formüller içermezler.
- Genetik algoritmalar, olasılık kuralını kullanan rassal aramaya göre optimum için arama yönlerine yol göstermede özel kurallara sahip değildirler.

Genetik algoritmalar, çözümün kendisi yerine çözüm setleriyle çalışması, tekil çözümler yerine çözümler topluluğunda arama yapması, yardımcı ek bilgiler yerine uygunluk fonksiyonunu kullanması ve deterministik kurallar yerine stokastik kurallarla çalışması nedeniyle diğer geleneksel optimizasyon yöntemlerinden ve arama prosedürlerinden ayrılır (Gen ve Cheng [22]).

3.1.1 Genetik Algoritma Tanımı ve Tarihçesi

Rassal arama tekniklerini kullanarak çözüm bulmaya çalışan genetik algoritma, parametre kodlama esasına dayalı bir arama tekniğidir. Algoritma doğadaki canlıların geçirdiği evrim sürecini dikkate alır.

Genetik algoritmalar, ilk defa 1960'larda Michigan Üniversitesi'nde John Holland ve çalışma arkadaşları tarafından geliştirilmiştir. Holland, araştırmalarını, arama ve optimumu bulma için, doğal seçme ve genetik evrimden yola çıkarak yapmıştır. İşlem boyunca, biyolojik sistemde bireyin bulunduğu çevreye uyum sağlayıp daha uygun hale gelmesi örnek alınmıştır (Er vd. [26]). Araştırma ekibi tarafından yapılan ilk çalışmalar daha çok bilgisayar yazılımları, numune tanıma üzerine olsa da kısa süre sonra makine öğrenme problemleri için de bilgisayar yazılımları geliştirilmeye başlanmıştır. Şekil 3.1'de genetik algoritma yaklaşımı şematik olarak gösterilmiştir.



Şekil 3.1 Genetik algoritma yaklaşımı (Michalewicz [27])

3.1.2 Genetik Algoritma Terimleri

Genetik algoritma ile ilgili genel olarak kabul görmüş terimler aşağıda verildiği gibi sıralanabilir (Cura [21]):

Gen: Bir kromozom içinde yer alan ve kalıtsal özelliği taşıyan birimdir.

Kromozom: Birden fazla genin bir araya gelerek oluşturdukları dizidir. Bu dizi, genlerle gelen özellikleri taşır ve birey olarak da adlandırılır.

Allele: Bir genin alabileceği değerlerdir.

Locus: Kromozom içinde genin bulunduğu yerdir.

Popülasyon (Topluluk): Çözüm kümesini oluşturan kromozomların oluşturduğu topluluktur.

Genotip: Bir kromozomun genetik yapısıdır ve genlerin temsil ediliş biçimini yansıtır.

Jenerasyon: Yeni bir topluluktur.

Fenotip: Genotipin fiziksel açıklamasıdır. Örneğin genotipi “0100101” olan bir kromozomun fenotipi 37 olabilir (ikilik düzenden ondalık düzene çevrilmiştir) ve amaç fonksiyonu hesaplanmasında fenotip kullanılır.

Uygunluk: Belirli bir kromozom veya kromozom grubunun amaç fonksiyonundaki performansıdır.

3.1.3 Genetik Algoritmanın Aşamaları

Genetik algoritmanın aşamalarını oluşturan, birbirini takip eden jenerasyonların geliştirilmesi ve değerlendirilmesi çevrimini gerçekleştiren ve algoritmanın performansını etkileyen elemanlar bu bölümde açıklanacaktır.

3.1.3.1 Kodlama Türleri

Genetik algoritma ile bir problem için çözüm geliştirilmesinin ilk adımı, tüm çözümlerin aynı boyutlara sahip bitler dizisi biçiminde gösterilmesidir. Parametrelerin kodlanması, probleme özgü bilgilerin genetik algoritmanın kullanacağı şekile çevrilmesine olanak tanır (Emel ve Taşkın [28]). Kromozomlar temsil ettikleri problem türlerine göre çeşitli şekillerde gösterilebilirler:

İkili (Binary) kodlama: Genetik algoritmalarda karmaşık yapıların temsilinde ilk ve en çok kullanılan mekanizma (*0,1*) alfabesini kullanan ikili bit dizisidir (Karaboğa, 2004). İkili kodlamada, her kromozom *0* veya *1* karakter dizilerinden oluşmaktadır. Şekil 3.2'de ikili kodlama örneği gösterilmiştir.

Kromozom A: 0100011111010001
Kromozom B: 0111101110001111

Şekil 3.2 İkili (binary) kodlama gösterimi

Permütasyon kodlama: Özellikle gezgin satıcı probleminin ve sıralama problemlerinin çözümü için uygun olan permütasyon kodlamada her kromozom, ilgili karakterin sıralamadaki pozisyonunu belirten sayılardan oluşan bir dizi ile ifade edilir. Şekil 3.3'de permütasyon kodlama örneği gösterilmiştir.

Kromozom A: 845967213
Kromozom B: 584274136

Şekil 3.3 Permütasyon kodlama gösterimi

Değer kodlama: Gerçek sayılar gibi karmaşık değerlerin kullanıldığı problemlerde doğrudan değer kodlama kullanılabilir [29]. Değer kodlamada, her kromozom gerçek

sayılar, karakterler gibi değerler veya problemle ilgili olabilecek herhangi nesneler olabilir. Şekil 3.4'de değer kodlama örneği gösterilmiştir.

Kromozom A: 1.2324 5.3243 0.4556 2.3293 2.4545
Kromozom B: ABDJEIFJDHDIERJFDLDFLFEGT
Kromozom C: (geri), (geri), (sağ), (ileri), (sol)

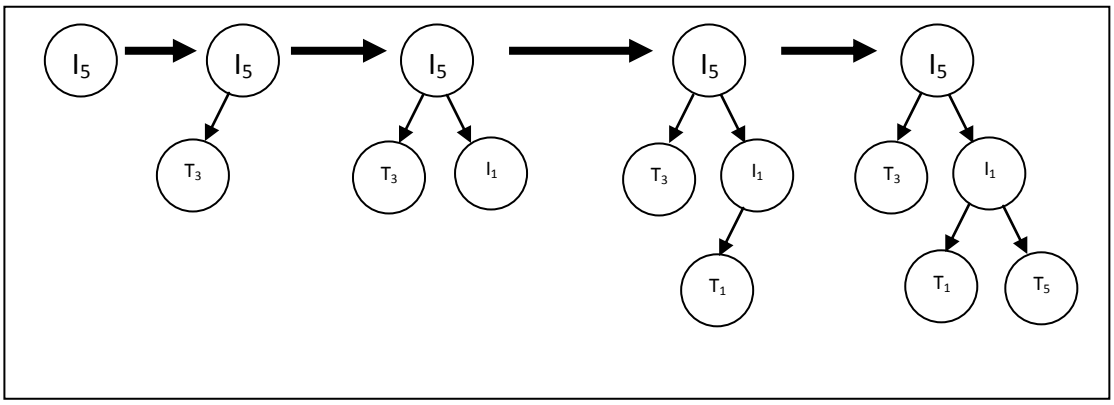
Şekil 3.4 Değer kodlama gösterimi [29]

Ağaç kodlama: Genellikle evrimleşen program veya ifadeler için kullanılan ağaç kodlamada her kromozom bazı nesnelerin ağacıdır. Ağaç kodlama evrimleşen programlar veya ağaç şeklinde kodlanabilecek yapılar için uygundur [29].

3.1.3.2 Birey Havuzunun Oluşturulması

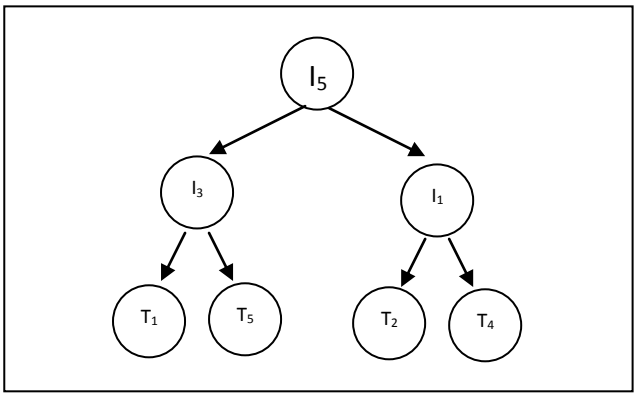
Yapılan çalışmalarda kullanılan genetik algoritmaların başlangıç popülasyonu genellikle rastgele sayı üreticisinin kullanılmasıyla oluşturulur ya da problemle ilgili kabaca bilinen bazı çözümler başlangıç popülasyonu olarak algoritma içine katılabilir. Literatürde başlangıç popülasyonunun oluşturulmasında üç farklı ağaç metodu kullanılmaktadır.

Grow metodu: Bu metotta, bireyler ağaçtır. Metodu yürüten, tekrarlanan seçim fonksiyonunun elemanları olan yapraklar (node) ve bu yaprakların derinliğidir. Başlangıçta rastgele seçilen fonksiyonda yaprak derinliği sıfırdır ve bu değer maksimum ağaç derinliğine kadar değişebilir. Aynı ağaç kökünden çıkan yapraklar farklı derinliklerde olabilirler (Eggermont [30]). Şekil 3.5'de Grow metodu şematik olarak gösterilmiştir.



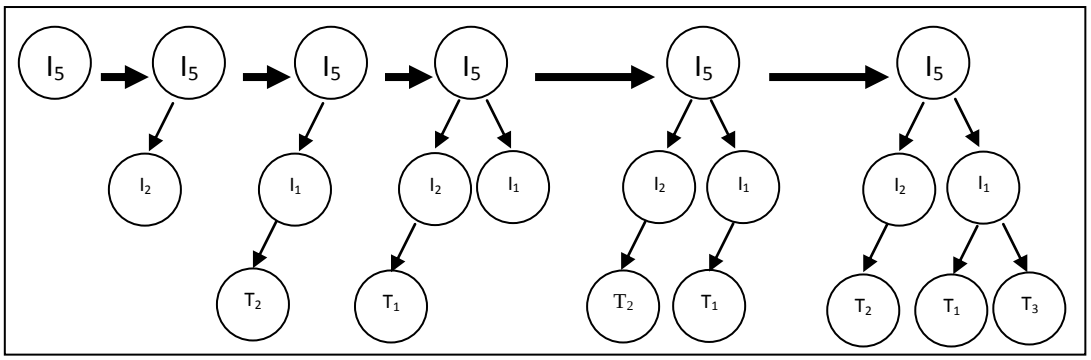
Şekil 3.5 Grow metodu ile oluşturulmuş ağaç yapısı (Eggermont [30])

Full metodu: Grow metoduna benzeyen bu metotta ise aynı ağaç kökünden çıkan yaprakların aynı derinlikte olmalı ve bu derinlikler maksimum ağaç derinliğine eşit olarak belirlenmelidir (Eggermont [30]). Şekil 3.6'da Full metodu şematik olarak gösterilmiştir.



Şekil 3.6 Full metodu ile oluşturulmuş ağaç yapısı (Eggermont [30])

Ramped half-and-half metodu: Grow ve Full başlangıç metotları bu yeni metotta birleştirilmiştir. Bu yönteme göre ağaç kökünün bir yaprağı Grow metodu ile büyürken diğer yaprak Full metodu ile büyür. Böylelikle Ramped half-and-half yöntemi ortaya çıkar (Eggermont [30]). Şekil 3.7'de Ramped half-and-half metodu şematik olarak gösterilmiştir.



Şekil 3.7 Ramped half-and-half metodu ile oluşturulmuş ağaç yapısı (Eggermont [30])

3.1.3.3 Genetik Algoritmada Kullanılan Operatörler

Bir problemin çözümünde kullanılan genetik algoritma için birey üzerinde işlem yapmaya yarayan birçok operatör geliştirilmiştir. Bu operatörlerin çoğu seçim, çaprazlama ve mutasyon operatörlerinin birer türleri olarak ortaya çıkmıştır. Problemin

çözümünde, çözüm performansı üzerinde oldukça büyük etkisi olan genetik operatörler bu bölümde açıklanacaktır.

Tekrar Üreme Operatörü: Tekrar üreme operatörü, genetik algoritmada tabii seçme işlemi olarak adlandırılan kalitesi yüksek bireylerin hayatta kalmalarını ve bu bireylerin sayılarının artmasını, kalitesi düşük bireylerin ise sayılarının azalarak kaybolması amaçlar. Bireyler arasındaki bu seçme işlemi tabiatta çevre tarafından yapılırken, yapay sistemlerde ise amaç fonksiyonu ve diğer kalite değerlendirme işlemleri tarafından yapılmaktadır (Kuo ve Lin [31]).

Seçim Operatörü ve Türleri: Seçim operatörü ile yeni popülasyonun oluşturulması için seçilecek olan birey sayısı ve hangi bireylerin eşleme için seçileceği, bireylerin uygunluk değerleri göz önüne alınarak belirlenir. Uygunluk değeri yüksek olan bireylerin seçilme ve bu bireylerin özelliklerinin bir sonraki nesile aktarılma şansı yüksektir.

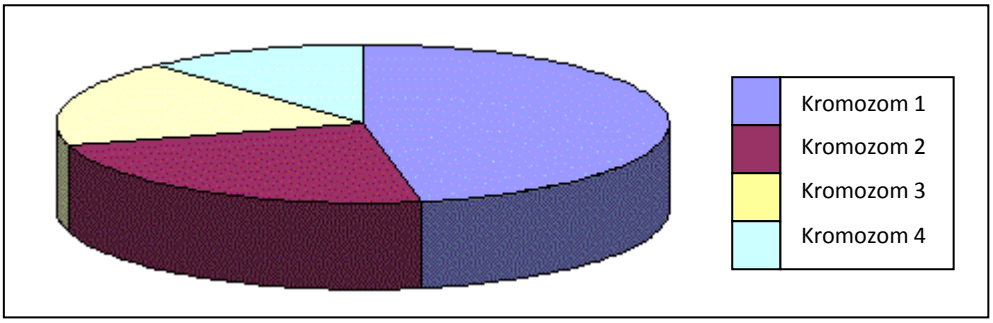
Rulet tekeri seçimi: Bu seçim metodunda, tüm kromozomların yerleştirildiği bir rulet tekeri mevcuttur. Kromozomların uygunluk değerlerine göre bu rulet tekeri üzerinde kaplayacakları alanlar belirlenir. Bu metodun çalışma prensibinde, bir bilye rulet tekerine atılmakta ve bilyenin durduğu yerdeki kromozom seçilmektedir. Uygunluk değeri yüksek olan ve dolayısıyla teker üzerinde daha büyük yer kaplayan kromozomlar daha fazla sayıda seçilmektedir. Süreç aşağıda verilen algoritma adımları ile açıklanabilir [32]:

Adım 1. [Toplam] Popülasyondaki tüm kromozomların uygunluk toplamı hesaplanır – S

Adım 2. [Seçim] $(0,S)$ aralığından rastgele bir sayı üretilir – r

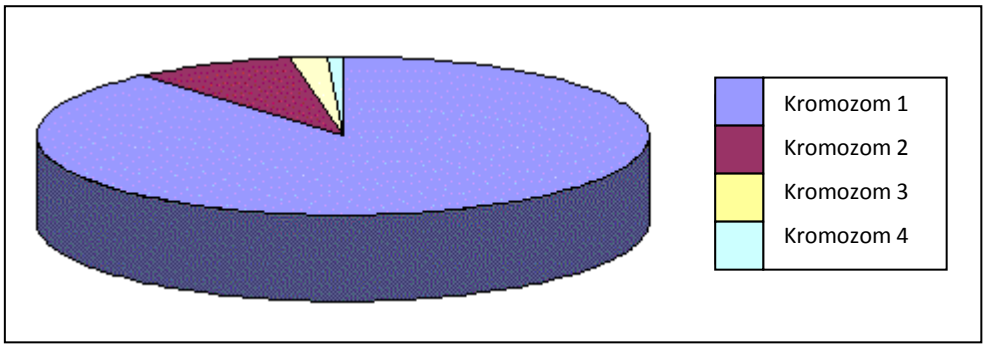
Adım 3. [Döngü] Popülasyon üzerinden gidip 0'dan itibaren uygunlukların toplamı alınır – S (S, r'den büyük olduğu zaman durulur ve bulunulan yerdeki kromozoma dönülür).

Burada Adım 1, her bir topluluk için bir kez yapılır. Şekil 3.8'de rulet tekeri üzerinde kromozom dağılımı örneği gösterilmiştir.

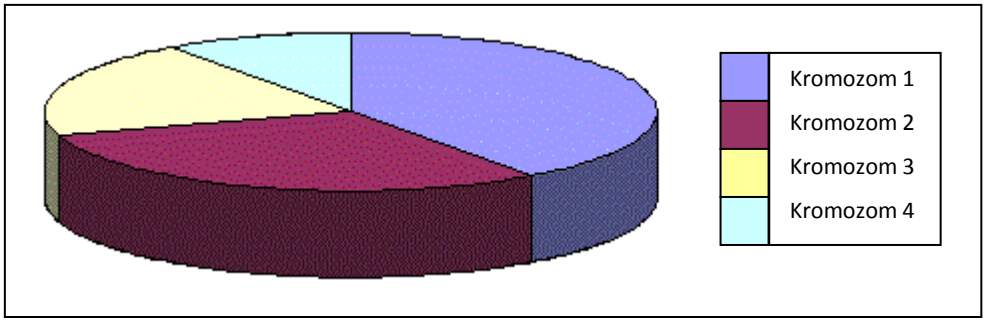


Şekil 3.8 Rulet tekeri üzerinde kromozom dağılımı [32]

Sıralama seçimi: Rulet tekeri seçim yönteminde kromozomların uygunluk değerleri arasında büyük farklar bulunduğu zaman ortaya bir problem çıkmakta ve çok yüksek olan en iyi kromozom uygunluğu diğer kromozomların seçilme şansını oldukça azaltmaktadır. Sıralama seçimi ile uygunluk değeri düşük olan kromozomların seçilme şansı artırılır. Sıralama seçimi ilk olarak popülasyonu sıralar ve her kromozom bu sıralamadan uygunluk değerini alır. En kötü 1 uygunluğunu, en kötü ikinci 2 uygunluğunu,…, en iyi n_2 (popülasyondaki kromozom sayısı) uygunluğunu alır. Seçim bu değerlere göre yapılır. Şekil 3.9 ve Şekil 3.10 sırasıyla sıralamadan önceki uygunluk değeri grafiğine ve sıralamadan sonraki sıra numaraları grafiğine örnek olarak gösterilmiştir.



Şekil 3.9 Sıralamadan önceki durum (uygunlukların grafiği) [32]



Şekil 3.10 Sıralamadan sonraki durum (sıra numaralarının grafiği) [32]

Sabit durum seçimi: Kromozomların büyük kısımının bir sonraki nesilde hayatta kalmak zorunda olması temeline dayanan bu metotta yeni çocuklar oluşturmak için her yeni nesilde yüksek uygunluk değerine sahip kromozomlar yeni çocukları oluşturmak için seçilir ve düşük uygunluk değerine sahip yavrular kaldırılarak yerlerine bu yeni oluşturulan yavrular yerleştirilir. Toplumun geri kalan kısımı aynen yeni nesile aktarılır [29]. Bu yöntemde alt popülasyon oluşturulduktan sonra uygunluklar hesaplanır, en kötü kromozomlar yerlerini başlangıç popülasyonundaki en iyi kromozomlara bırakır.

Turnuva seçimi: Bu metotta popülasyon içerisinden rastgele *k* adet birey alınır ve bu bireylerin içerisinden uygunluk değeri en iyi olan birey seçilir. İşlem popülasyondaki kromozom sayısı kadar tekrar edilir.

Seçkinlik: Genetik algoritma aşamalarından olan üreme, çaprazlama, mutasyon işlemleri ile yeni bir nesil oluşturulurken, en iyi kromozomları kaybetme olasılığı vardır. Seçkinlik, en iyi kromozomların (ya da bir kısımının) ilk önce kopyalanıp yeni nesile aktarıldığı yöntemdir. Geri kalan kromozomlar klasik yöntemlerle üretilir [32]. Seçkinliğin, bulunan en iyi çözümün kaybolmasını önleyebilme özelliği ile genetik algoritmanın başarısı hızlı bir şekilde arttırılabilinir.

Çaprazlama Operatörü ve Türleri: Tabii sistemlerde meydana gelen veya genetik çaprazlama olayı ile ortaya çıkan melez yapıların üretilmesine eşdeğer bir özelliği genetik algoritmaya kazandıran çaprazlama operatörü ile eşleştirme havuzunda bulunan yapıların birer çifti rastgele seçilerek, operatör seçilen bu iki yapıdan yeni iki yapı meydana getirmek için kullanılır. Çaprazlama operatörünün çalıştırılmasının ardında eski yapılar ve çaprazlama işlemi sonucunda elde edilen yeni yapılar mevcut jenerasyonda tutulur veya popülasyon büyüklüğünün sabit kalmasını sağlayan ve daha

iyi yapıların eldesini amaçlayan eski ile yeni yapıların yer değiştirmesi sağlanarak genetik algoritma prosedürünün diğer adımları işletilmeye devam ettirilir. Bu ikinci durumda, kötü yapılar atılır ve popülasyon büyüklüğü sabit olarak korunur.

İyi çözümlerin farklı bölümlerini birleştirerek daha iyi çözümler elde etmeyi amaçlayan çaprazlama operatörü çeşitleri aşağıda verildiği gibi sıralanabilir:

Tek nokta çaprazlama: Holland'ın orjinal çaprazlama tekniği olan bu metotta, tek bir çaprazlama noktası belirlenir. İlk bireyden çaprazlama noktasına kadar olan özellikler, diğer bireyden de çaprazlama noktası sonrasındaki özellikler alınarak, bu genler birleştirilir. İşlem ikinci bireyin çaprazlama noktasına kadar olan özelliklerinin, ilk bireyin de çaprazlama noktası sonrasındaki özelliklerinin alınması ve bu genlerin birleştirilmesi ile tamamlanır. Böylelikle yeni yavru bireyler elde edilir. Şekil 3.11'de tek nokta çaprazlama örneği gösterilmiştir.

Kromozom A:	010001	1111010001
Kromozom B:	**011110**	**1110001111**
Yavru 1:	010001	**1110001111**
Yavru 2:	**011110**	1111010001

Şekil 3.11 Tek nokta çaprazlama örneği

İki noktalı çaprazlama: Bu metotta ise iki çaprazlama noktası seçilir. Kromozomun başından ilk kesme noktasına kadar olan ikili karakter dizisi ilk bireyden, iki kesme noktası arasındaki kısım ikinci bireyden ve ikinci kesme noktasından sonraki kısım tekrar ilk bireyden alınır [29]. İşlem kromozomun başından ilk kesme noktasına kadar olan ikili karakter dizisinin ikinci bireyden, iki kesme noktası arasındaki kısımın birinci bireyden ve ikinci kesme noktasından sonraki kısımın tekrar ikinci bireyden alınması ile sonlanır. Böylelikle yeni yavru bireyler elde edilir. Şekil 3.12'de iki noktalı çaprazlama örneği gösterilmiştir.

Kromozom A:	0100	011111	010001
Kromozom B:	**0111**	**101110**	**001111**
Yavru 1:	0100	**101110**	010001
Yavru 2:	**0111**	011111	**001111**

Şekil 3.12 İki noktalı çaprazlama örneği

Tekdüze (uniform) çaprazlama: Bu metotta bir çaprazlama noktası bulunmaz. Bu çaprazlama noktası yerine bireylerden gelen genler sırayla yavruya kopyalanır. Bu kopyalamada her bir gen belli bir X_i [*0,1*] (*i = 1,2,…,n*) olasılığına göre bireylerden gelir. $X_i < 0.5$ ise *i* geni birinci bireyden, $X_i > 0.5$ ise *i* geni ikinci bireyden kopyalanır. Bu metotta tek bir yavru oluşur (Cura [21]). Şekil 3.13'de tekdüze çaprazlama örneği gösterilmiştir.

Kromozom A:	**1**	0	**1**	0	0	**1**	**0**	1
X_i	0.3	1.0	0.4	0.8	0.5	0.4	0.2	0.8
Kromozom B:	1	**1**	0	**1**	**1**	0	0	**1**
Yavru :	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**

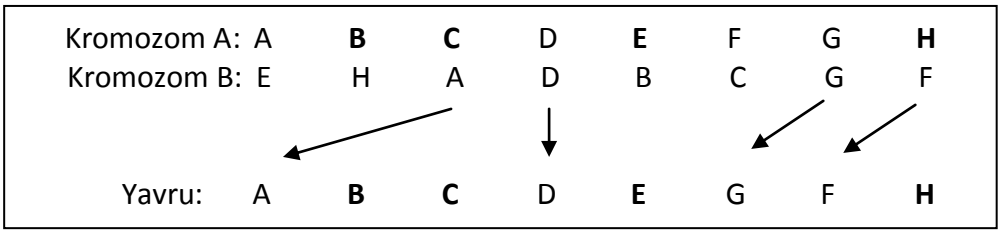
Şekil 3.13 Tekdüze (uniform) çaprazlama örneği

Dairesel çaprazlama: Davis, Goldberg ve Lingle tarafından geliştirilmiş bir metottur. Bu metotta ilk bireyden en baştaki gen seçilir ve bu gen yeni diziye yerleştirilir. Bu gene karşılık gelen ikinci bireydeki gen belirlenir bu değer de yeni kromozom üzerine yerleştirilerek dairesel bir şekilde bütün genler belirlenir (Engin ve Fığlalı [33]). İşlem ikinci bireyden başlanarak tüm dairesel adımların tekrarlanmasıyla sonlandırılır. Böylelikle yeni yavru bireyler elde edilir. Şekil 3.14'de dairesel çaprazlama örneği gösterilmiştir.

Kromozom A:	9	8	2	1	7	4	5	10	6	3
Kromozom B:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Yavru 1:	9	**2**	**3**	1	**5**	4	**7**	**8**	6	**10**
Yavru 2:	**1**	8	2	**4**	7	**6**	5	10	**9**	3

Şekil 3.14 Dairesel çaprazlama örneği

Pozisyona dayalı çaprazlama: Bu metotta rastlantısal olarak seçilmiş pozisyondaki genler, bir bireyden yavruya kalıtsallaştırılır. Diğer genler ise diğer bireyden bulundukları sıra ile alınır (Engin ve Fığlalı [33]). Şekil 3.15'de pozisyona dayalı çaprazlama örneği gösterilmiştir.



Şekil 3.15 Pozisyona dayalı çaprazlama örneği

Sıraya dayalı çaprazlama: Sıralı kromozomların çaprazlaması için rastlantısal olarak bir çaprazlama noktası (*X*) seçilir ve birinci bireyin çaprazlama noktasına kadar olan tüm genleri aynen yavruya aktarılırken kalan kısım için ise sırasıyla diğer bireyden yavruya henüz kopyalanmamış genler kopyalanır. İşlem ikinci bireyin çaprazlama noktasına kadar tüm genlerinin aynen yavruya aktarılması, kalan kısım için ise sırasıyla ilk bireyden yavruya henüz kopyalanmamış genlerin kopyalanması ve geriye kalan boş pozisyonlara ilk bireyden aktarılan yeni karakterler de göz önüne alınarak ikinci kromozomun kullanılmayan karakterlerinin soldan sağa sıra ile yerleştirilmesiyle sonlanır. Böylece iki yavru oluşur. Şekil 3.16'da sıraya dayalı çaprazlama örneği gösterilmiştir.

Kromozom A: A	C	D	B	X	E	G	F	H
Kromozom B: **G**	**A**	**E**	**C**		**D**	**H**	**B**	**F**
Yavru 1:	A	C	**D**	**B**	G	E	**H**	**F**
Yavru 2:	**G**	**A**	**E**	**C**	D	B	F	H

Şekil 3.16 Sıraya dayalı çaprazlama örneği

Kısmi planlı çaprazlama: Goldberg tarafından geliştirilen bu çaprazlama ilk olarak gezgin satıcı probleminde kullanılmıştır. Bu metotta iki ayrı bireyin genleri arasında rassal olarak aralıklar belirlenir ve bu aralıkta yer alan genlerin yeri karşılıklı olarak değiştirilir (Engin ve Fığlalı [33]). Sonuçta iki yavru elde edilir. Şekil 3.17'de kısmi planlı çaprazlama örneği gösterilmiştir.

Kromozom A:	2	8	6	4	5	7	1	3
Kromozom B:	**8**	**7**	**2**	**1**	**3**	**4**	**6**	**5**
	2	8	**2**	**1**	**3**	7	1	3
	8	**7**	6	4	5	**4**	**6**	**5**
Yavru 1:	6	8	2	1	3	7	4	5
Yavru 2:	8	7	6	4	5	1	2	3

Şekil 3.17 Kısmi planlı çaprazlama örneği

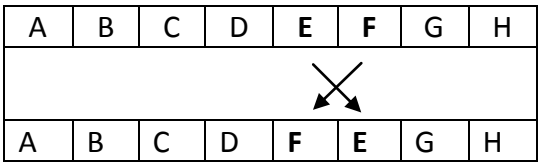
Doğrusal sıralı çaprazlama: Falkenauer ve Bouffouix tarafından geliştirilmiştir. Bu metotta mevcut popülasyon içerisinden rassal olarak iki birey seçilir, seçilen bu iki dizi (kromozom) üzerinde rassal olarak iki alt dizi seçilir. P_1 dizisinden seçilen alt dizi kromozomdan koparılır ve boş kalan yerler belirlenir, benzer şekilde P_2 dizisinde de aynı işlemler gerçekleştirilir. Son olarak birinci alt dizi P_1'e ve ikinci alt dizi de P_2'ye yerleştirilir (Engin ve Fığlalı [33]).

Mutasyon Operatörü ve Türleri: Doğal genetik mutasyon olayına benzeyen ve genetik algoritmanın performansında çaprazlama operatörü gibi önemli ve temel bir rol oynayan mutasyon operatörü tekniği genellikle kromozomların kodlanmasına bağlıdır. İkili kodlamada her bir bit tek tek kontrol edilerek mutasyon oranına göre bitler 1 ise 0'a, 0 ise 1'e çevrilirken, permütasyon kodlamada ise mutasyon rastgele seçilen genlerin yer değiştirilmesi ile gerçekleştirilebilir.

Yeni, görülmemiş ve araştırılmamış çözüm elemanlarının bulunmasını sağlayan mutasyon operatörü çeşitleri aşağıda verildiği gibi sıralanabilir:

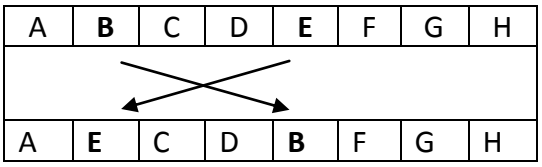
Ters mutasyon: Çaprazlama yapıldıktan sonra oluşan yeni bireylerin, eskilerle aynı olmasını engellemek için yapılan mutasyon işleminin bu çeşidinde, bir kromozomda rassal olarak iki pozisyon seçilir ve bu iki pozisyondaki alt diziler ters çevrilir (Kaya [34]).

Komşu iki geni değiştirme: Bu metotta, rassal olarak seçilen iki komşu gen değiştirilir (Kaya [34]). Şekil 3.18'de komşu iki geni değiştirme mutasyon örneği gösterilmiştir.



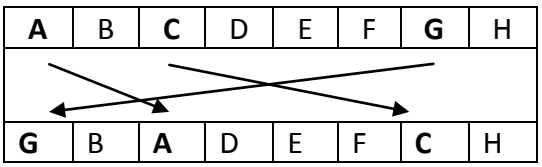
Şekil 3.18 Komşu iki geni değiştirme mutasyon örneği

Keyfi iki geni değiştirme: Bu metotta, rassal seçilen iki gen değiştirilir (Kaya [34]). Şekil 3.19'da keyfi iki geni değiştirme mutasyon örneği gösterilmiştir.



Şekil 3.19 Keyfi iki geni değiştirme mutasyon örneği

Keyfi üç geni değiştirme: Bu metotta, rassal olarak seçilen üç gen keyfi olarak değiştirilir (Kaya [34]). Şekil 3.20'de keyfi üç geni değiştirme mutasyon örneği gösterilmiştir.



Şekil 3.20 Keyfi üç geni değiştirme mutasyon örneği

Araya yerleştirme: Bu metotta, kromozomun yapısında bulunan bir gen bulunduğu konumdan kaydırılır ve diğer bir konuma getirilir. Kaydırmanın yapıldığı bu nokta rassal olarak seçilir.

Kontrol Parametreleri: Kullanıcı tarafından seçilen operatörler gibi, algoritmanın kontrol parametre değerlerinin seçimi de algoritmanın performansı üzerinde oldukça etkilidir. Basit bir genetik algoritmanın temel kontrol parametreleri aşağıda verildiği gibi sıralanabilir:

Popülasyon büyüklüğü: Popülasyon büyüklüğü için seçilen değer, algoritmanın performansını iki şekilde etkilemektedir. Bu değer çok küçük olduğunda, genetik algoritma yerel bir optimuma takılabilirken, popülasyon büyüklük değerinin çok yüksek olması durumunda ise çözüme ulaşma zamanı artabilmektedir (Emel ve Taşkın [28]). Genetik algoritmanın yürütülmesi aşamasında popülasyon büyüklüğü için uygun bir değer belirlenmelidir.

Çaprazlama oranı: Popülasyon bireylerine, yeni birey üretiminde uygulanacak çaprazlama operatörünün frekansını belirlemek amacıyla kullanılan parametredir. Düşük çaprazlama oranı yeni kuşağa çok az sayıda yeni yapının girmesine sebep olmaktadır. Dolayısıyla tekrar üreme operatörü algoritmada aşırı etkili operatör haline gelmekte ve araştırmanın yakınsama hızı düşmektedir. Yüksek çaprazlama oranı ise araştırma uzayının çok hızlı bir şekilde araştırılmasına sebep olmaktadır. Ama oran aşırı yüksek ise çaprazlama operatörü benzer veya daha iyi yapıları üretemeden kuvvetli olan yapılar çok hızlı olarak bozulduğundan algoritmanın performansı düşmektedir (Karaboğa [35]).

Mutasyon oranı: Mutasyon operasyonunun frekansı, etkili bir genetik algoritma tasarlamak için iyi bir şekilde kontrol edilmelidir. Mutasyon operasyonu, araştırma sahasına yeni bölgelerin girmesini sağlar. Yüksek mutasyon oranı, araştırmaya aşırı bir rastgelelik kazandıracak ve araştırmayı hızlı bir şekilde optimal çözümden uzaklaştıracaktır. Başka bir deyişle, popülasyonun gelişmesine değil tahribatına sebep olacaktır. Bu durumun tersine, çok düşük mutasyon oranının kullanılması ise ıraksamayı aşırı düşürecek ve araştırma uzayının tamamen araştırılmasını engelleyecektir. Dolayısıyla, algoritmanın alt optimal çözüm bulmasına sebep olacaktır (Karaboğa [35]).

Kuşak Aralığı: Her kuşaktaki yeni kromozom oranına verilen bir isim olan kuşak aralığı, genetik operatörler için kaç tane kromozomun seçildiğini gösterirken bu aralık için elde edilebilecek olan yüksek bir değer birçok kromozomun yer değiştirdiği anlamına gelmektedir.

Seçim Stratejisi: Eski kuşağı yenilemenin çeşitli yöntemleri mevcuttur (Emel ve Taşkın [28]):

- *Kuşaksal strateji:* Kuşaksal stratejide, mevcut popülasyondaki kromozomlar tamamen yavrular ile yer değiştirir. Popülasyonun en iyi kromozomu da yenilendiğinden dolayı bir sonraki kuşağa aktarılamaz ve bu yüzden bu strateji en uygun stratejisi ile beraber kullanılmaktadır.

- *En uygun stratejisi:* En uygun stratejisinde, popülasyondaki en iyi kromozomlar hiçbir zaman yenilenmemektedir, bundan dolayı çoğalma için en iyi çözüm her zaman elverişlidir.
- *Denge durumu stratejisi:* Denge durumu stratejisinde ise, her kuşakta yalnızca birkaç kromozom yenilenmektedir. Genellikle, yeni kromozomlar popülasyona katıldığında en kötü kromozomlar yenilenir.

Fonksiyon Ölçeklemesi: Doğrusal ölçekleme ve üstsel ölçekleme gibi yöntemleri mevcut olan fonksiyon ölçeklemesi çeşidinin probleme göre en uygun olacak şekilde seçilmesi genetik algoritmanın etkin ve verimli bir şekilde çalışabilmesi açısından önemlidir.

3.1.4 Genetik Algoritmanın Çalışma Prensibi

Bir problemin genetik algoritma ile çözümünde arama uzayındaki bütün muhtemel çözümler, dizi olarak kodlanır. Her bir dizi, arama uzayında belirli bir bölgeye karşılık gelir. Genellikle rassal bir çözüm seti seçilir ve bu set başlangıç popülasyonu olarak kabul edilir. Her bir dizi için bir uygunluk değeri hesaplanır. Bir grup dizi belirli bir olasılık değerine göre rassal olarak seçilip genetik operatörler yardımıyla üreme işlemi gerçekleştirilir. Bu işlemler ile bulunan yeni bireylerin uygunluk değerleri hesaplanır ve bireyler çaprazlama ve mutasyon işlemlerine tabi tutulurlar. Önceden belirlenen ve algoritmanın durma kriteri olarak adlandırılan nesil sayısı boyunca bu işlemler devam ettirilir. İterasyon, nesil sayısına ulaşınca işlem bitirilir ve uygunluk değeri en yüksek olan dizi seçilir.

Bu işlemler sırasında unutulmamalıdır ki, genetik algoritmalar bir çözüm uzayındaki her noktayı, kromozom adı verilen ikili bit dizisi ile kodlar. Her noktanın bir uygunluk değeri vardır. Her kuşakta genetik algoritma, çaprazlama ve mutasyon gibi genetik operatörleri kullanarak yeni bir popülasyon oluşturur. Birkaç kuşak sonunda, popülasyon daha iyi uygunluk değerine sahip üyeleri içerir. Genetik algoritmalar, çözümlerin kodlanmasını, uygunlukların hesaplanmasını, çoğalma, çaprazlama ve mutasyon operatörlerinin uygulanmasını içerirler (Emel ve Taşkın [28]).

3.1.5 Genetik Algoritmanın Performansını Etkileyen Nedenler

Genetik algoritma hesaplamaları parametre değerlerinden çok kodlanmış parametreler temellidir ve genetik algoritmalar yerel optimum içinde ayırıcı olmaktan kaçınarak yüksek oranda paralel arama yeteneği sağlarlar. Genetik algoritmalar uygunluk fonksiyonunun hesaplanması dışında hiçbir karmaşık matematiksel formül içermez ve genetik algoritmalar optimum için arama yönünde özel kurallara sahip değildirler (Wang ve Lu [25]).

Akcayol'a göre [36] da genetik algoritmanın performansını etkileyen faktörler aşağıda verildiği gibi sıralanabilir:

Kromozom sayısı: Kromozom sayısını artırmak çalışma süresini artırır, kromozom sayısını azaltmak ise kromozom çeşitliliğini yok eder.

Mutasyon oranı: Kromozomlar birbirine benzemeye başladığında hala uygun çözüm bulunamadıysa mutasyon işlemi genetik algoritmanın sıkıştığı yerden kurtulması için tek yoldur. Ancak mutasyon oranına yüksek bir değer vermek genetik algoritmanın kararlı bir noktaya ulaşmasını engeller.

Kaç noktalı çaprazlama yapılacağı: Normal olarak çaprazlama tek noktada gerçekleştirilmekle beraber, yapılan araştırmalar bazı problemlerde çok noktalı çaprazlamanın faydalı olduğunu göstermiştir.

Çaprazlamanın sonucu elde edilen bireylerin nasıl değerlendirileceği: Elde edilen iki bireyin birden kullanılıp kullanılamayacağı bazen önemli olmaktadır.

Uygunluk değerlendirmesinin yapılışı: Probleme tam olarak uygun oluşturulmamış bir değerlendirme fonksiyonu, çalışma süresini uzatmakta hatta çözüme hiçbir zaman ulaşılamamasına neden olabilmektedir.

3.1.6 Genetik Algoritmanın Uygulandığı Problemler

Literatür çalışmalarına bakıldığında genetik algoritmaların birçok farklı problem türünün çözümüne uygulandığı görülmektedir. Bu problem türleri ve son 5 yılda bu problem türleri için geliştirilmiş bazı çözüm teknikleri aşağıda verildiği gibi sıralanabilir:

Gezgin satıcı problemi: Samanlıoğlu ve diğerleri [37] de simetrik çok amaçlı gezgin satıcı probleminin çözümü için rastlantısal anahtarlı genetik algoritmayı önerirken, Yang ve diğerleri [38] de yine aynı problem türünün çözümü için genelleştirilmiş kromozomlar kullanan genetik algoritma yapısını kullanmışlardır. Xing ve diğerleri ise [39] da asimetrik gezgin satıcı probleminin çözümü için genetik algoritma ve optimizasyon stratejileri ile geliştirilmiş bir yaklaşımdan yararlanmışlardır.

Araç rotalama problemi: Jeon ve diğerleri [40] da çok depolu araç rotalama probleminin çözümü için melez bir genetik algoritma yapısı önerirken, Bae ve diğerleri [41] de yine aynı problem türünün çözümü için çok depolu sistem için birleştirilmiş genetik algoritma ile araç rotalama problemi yaklaşımını kullanmışlardır. Wang ve Lu [25] te kapasiteli araç rotalama probleminin çözümü için melez bir genetik algoritma yaklaşımını önerirken, Alba ve Dorronsoro [42] de araç rotalama problemine hücresel genetik algoritma yapısı ile çözüm bulmaya çalışmışlardır. Cheng ve Wang ise [43] te zaman pencereli araç rotalama probleminin çözümü için ayrıştırma tekniğini ve genetik algoritmayı kullanmışlardır.

Kuadratik atama problemi: Liu ve Li [44] te cezalı bulanık kuadratik atama probleminin çözümü için genetik algoritma temelli yeni modeller geliştirirken, Drezner [45] de kuadratik atama probleminin çözümü için tecrübeye dayalı melez genetik algoritma yapısını önermiştir.

Çizelgeleme problemi: Pezzella ve diğerleri [46] da esnek iş çizelgeleme probleminin çözümü için genetik algoritmayı kullanırken, Gao ve diğerleri [47] de yine aynı problem türünün çözümü için genetik algoritmadan ve komşuluk değişkeninden oluşan melez bir yapı ortaya koymuşlardır. Essafi ve diğerleri [48] de iş çizelgeleme problemindeki toplam gecikme ağırlığının minimizasyonu için genetik yerel arama algoritmasını önerirken, Vilcot ve Billaut [49] da tek kriterli genel iş çizelgeleme probleminin çözümü için genetik algoritma ve tabu aramadan oluşan bir yapı kullanmışlardır. Li ve diğerleri [50] de ise iş çizelgeleme probleminin çözümü için tahmin edilebilir ve gruplanabilir genetik algoritma tekniği ile probleme çözüm aramışlardır.

Atölye tipi çizelgeleme problemi: Chou [51] de tek makine toplam ağırlıklı atölye çizelgeleme probleminin çözümü için tecrübeyle öğrenen genetik algoritma metodunu

önerirken, Chang ve diğerleri [52] de serbest kalma zamanlı tek makine çizelgeleme problemi için olay içeren genetik algoritmayı kullanmışlardır. Chang ve diğerleri ise [53] de cezalara bağlı tek makineli çizelgeleme probleminin çözümü için dominant özellikleriyle melez genetik algoritma yapısını önermişlerdir.

Proje çizelgeleme problemi: Kılıç ve diğerleri [54] de risk altında proje çizelgeleme probleminin çözümü için tek amaçlı genetik algoritma yaklaşımını kullanırken, Shadrokh ve Kianfar [55] de kaynak yatırımlı proje çizelgeleme probleminin çözümü için genetik algoritmayı önermişlerdir. Gonçalves ve diğerleri [56] da kaynak kısıtlı çoklu proje çizelgeleme probleminin çözümü için genetik algoritmadan yararlanırken, Valls ve diğerleri [57] de yine aynı problem türünün çözümü için melez bir genetik algoritma yapısı önermişlerdir.

Akış tipi çizelgeleme problemi: Nagano ve diğerleri [58] de akış tipi çizelgeleme probleminin çözümü için yapısal genetik algoritma tekniğini önerirken, Chen ve diğerleri [59] da tekrar başlamalı akış tipi çizelgeleme probleminin çözümü için melez bir genetik algoritma yapısını ortaya koymuşlardır. Cheng ve Chang ise [60] da akış tipi çizelgeleme probleminin çözümü için genetik algoritma ile Taguchi deneysel tasarımından oluşan bir yapı üzerinde çalışmışlardır.

Çizge boyama problemi: Mabrouk ve diğerleri [61] de çizge boyama probleminin çözümü için paralel genetik algoritma ve tabu arama temelli bir algoritma önerirken, Lu ve Hao [62] de yine aynı problem türünün çözümü için havuz güncellemesi ve çaprazlama operatörü ile entegre olmuş genetik algoritma tekniğinden yararlanmışlardır.

Kaynak atama problemi: Lin ve Gen [63] te çok basamaklı kombinasyonel optimizasyon problemlerinden çok kriterli insan kaynak atama probleminin çözümü için çok amaçlı melez bir genetik algoritma tekniği kullanmışlardır.

Tesis yerleştirme problemi: Lee ve diğerleri [64] de çok katlı tesis yerleştirme probleminin çözümü için geliştirilmiş bir genetik algoritma yapısı önerirken, Wang ve diğerleri [65] de eşit olmayan alanlara sahip tesis yerleştirme probleminin çözümü için yine genetik algoritma tekniğini önermişlerdir. Aiello ve diğerleri [66] da ise tesis

yerleştirme probleminin çözümü için genetik arama algoritması ve electre metodundan oluşan çok amaçlı bir yaklaşım kullanmışlardır.

Montaj hattı dengeleme problemi: Levitin ve diğerleri [67] de montaj hattı dengeleme probleminin çözümü için genetik algoritmayı önerirken, Kim ve diğerleri [68] de iki taraflı montaj hattı dengeleme probleminin çözümü için matematik model ve genetik algoritmadan oluşan bir yaklaşım ileri sürmüşlerdir. Mansouri [69] da ise tam zamanlı üretimde montaj hatlarının dengelenmesi probleminin çözümü için çok amaçlı genetik algoritma yaklaşımını kullanmışlardır.

Sipariş büyüklüğü problemi: Defersha ve Chen [70] te hücre belirleme ve ürün kalitesi temelindeki sipariş büyüklüğü problemi ile entegre olmuş genetik algoritmaya gömülmüş lineer programlama tekniğini önerirken, Chang ve diğerleri [71] de ise bulanık ekonomik sipariş büyüklüğü probleminin çözümü için genetik algoritmayı kullanmışlardır.

3.2 Karınca Kolonisi Optimizasyonu

Doğada topluluk halinde yaşayan, aralarında belirli bir iş paylaşımının olduğu ve karşılaştıkları problemleri yardımlaşarak birlikte çözen hayvanlar "Sosyal Hayvanlar" olarak adlandırılmaktadırlar. Bu tür hayvanların problem çözme özellikleri örnek alınarak gerçek dünya problemlerinin çözümü için başarılı yeni yaklaşımlar gerçekleştirilebilinir. Karıncalar koloni halinde yaşayan ve problem çözmede örnek alınabilecek önemli davranış özellikleri gösteren canlılardır (Karaboğa [35]).

Gerçek karıncaların beslenme davranışlarından ilham alan bir yöntem olan karınca kolonisi optimizasyonu yöntemi, gerçek yaşamdaki karınca koloni davranışlarının matematiksel modelleri üzerine dayalı bir algoritmadır. Bu noktada, algoritmanın temelini oluşturan düşünce karıncaların yuvalarından bir gıda kaynağına giden en kısa yolu, herhangi bir görsel ipucu kullanmadan bulma yeteneğine sahip olmalarıdır.

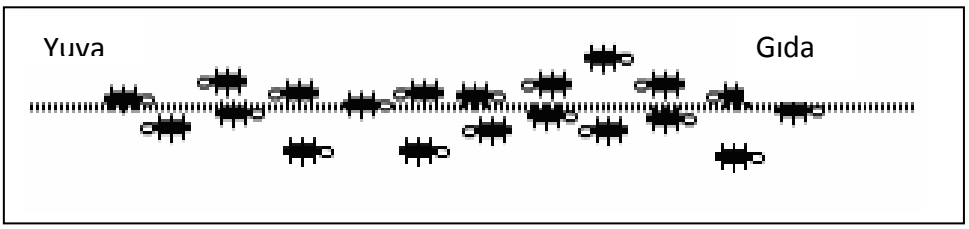
Karınca kolonisi optimizasyonu, hem statik hem de dinamik kombinasyonel optimizasyon problemlerinin çözümünde kullanılabilir. Statik problemlerde, problemin karakteristik özellikleri bir kez verilir ve bu özellikler tüm problem tanımlanıp çözümlenene kadar değiştirilmez. Buna karşın, dinamik problemlerde ise sistemin

temelinde bulunan dinamikler ile bazı büyüklüklerin değerleri fonksiyonel olarak tanımlanır (Dorigo ve Stützle [72]).

3.2.1 Karınca Kolonisi Optimizasyonu Tanımı ve Tarihçesi

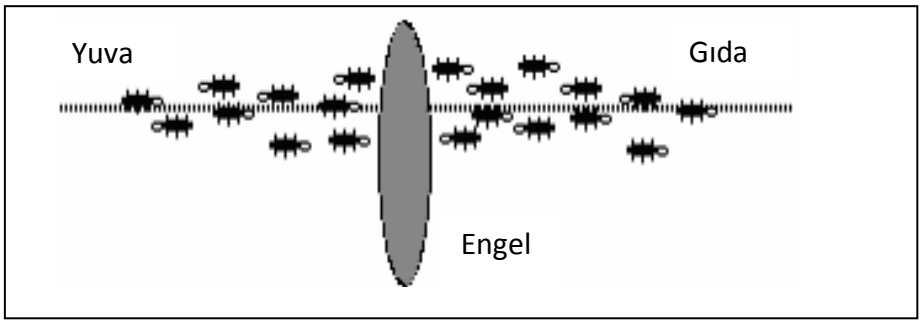
Gerçek karıncaların beslenme davranışları gibi biyolojik davranışları inceleyen Fransız entomolojist Pierre Paul Grass 1940'lı ve 1950'li yıllarda bazı termit türlerinin "Anlamlı Dürtü" olarak tanımladığı dürtüye tepki gösterdiklerini gözlemlemiştir. Ayrıca bu tepkilerin, hem dürtüyü üreten böcek hem de kolonideki diğer böcekler için yeni bir anlamlı dürtü gibi davranabileceğini de ortaya koymuştur. Grass, çalışanların eriştikleri performansa göre uyarıldığı bu haberleşme türünü tanımlamak için "Stigmergy" terimini kullanmıştır. Stigmergy örnekleri karınca kolonilerinde gözlenebilir (Cura [21]).

Karınca Kolonisi Optimizasyonu (KKO) metasezgiselinin temel mekanizması, yapay karınca kolonisi içerisindeki yapay karıncaların, optimizasyon problemine iyi sonuçlar verebilmek için birbirleriyle işbirliği yapması üzerine kuruludur. Karınca kolonilerinin önemli ve ilgi çekici olan davranış biçimi yiyecek arama sürecinde ortaya çıkmaktadır. Karıncalar, yuvaları ile yiyecek arasındaki en kısa yolu görsel işaretleri kullanmadan bulabilmektedirler. Karıncalar, yiyecek kaynağından yuvalarına gidip gelirken kullandıkları yol üzerine "Feromon" adlı bir madde bırakırlar. Karıncalar feromon maddesinin kokusunu alabilirler ve ayrım noktalarında yollarını seçerken kokunun, başka bir deyişle iz miktarının yoğun olduğu tarafı daha yüksek bir olasılıkla seçme eğilimi gösterirler. Yapılan deneylerde, karınca kolonisinin feromon izini takip etme davranışı neticesinde en kısa yolun bulunabildiği görülmüştür (Yağmahan ve Yenisey [73]). Şekil 3.21'de yuva ile yiyecek arasında karıncaların bulduğu en kısa yol gösterilmiştir.



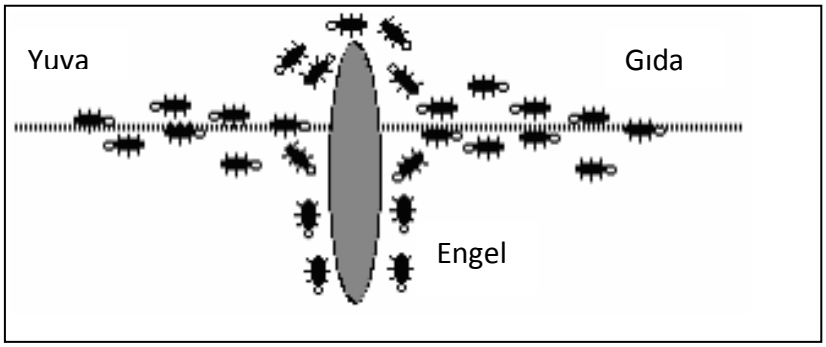
Şekil 3.21 Karıncaların izlediği yol (Dalkılıç ve Türkmen [74])

Hareket halinde bulundukları yola belirli miktarda feromon maddesi bırakan karıncalar, gitmek için seçecekleri yönün belirlenmesinde bu maddenin miktarını dikkate alırlar. Karıncaların, feromon maddesinin daha yoğun olduğu yönleri tercih etme olasılığı, daha az feromon maddesine sahip yönleri tercih etme olasılığından daha fazla olmasına rağmen yine de feromon maddesinin daha yoğun olduğu yönün seçilememe veya az yoğun olduğu yönün seçilebilme olasılığı da besinlerin karıncaya olan mesafesine bağlı olarak söz konusudur. Şekil 3.21'de gösterilen yol, önceden keşfedilen en kısa yol olsun ve bu yola bir cisim konularak bu yolun bozulduğu varsayılsın. Yani kullanılmakta olan en kısa yol artık en kısa değildir. Karıncaların önüne cisim konması durumu Şekil 3.22'de gösterilmiştir.



Şekil 3.22 Karıncaların bir engelle karşılaşması (Dalkılıç ve Türkmen [74])

Yiyeceğe ulaşma güzergahları üzerine konan bir cisim ile cismin hemen önündeki ve arkasındaki karıncalar, tercih edilmesi gereken feromonsuz yönlerle karşılaşırlar ve sağa veya sola dönme seçenekleri arasından rastgele birini tercih ederek yollarına devam ederler. Koloni halindeki karıncaların yaklaşık olarak yarısının sol tarafa döndüğünün ve diğer yarısının da sağ tarafa döndüğünün kabul edildiği böylesi bir durumda, kolonideki karınca sayısı arttıkça feromonsuz olarak sağa ve sola dönen karıncaların sayıları arasındaki fark azalacak yani iki grubun sayıları yaklaşık olarak birbirine eşit olacaktır. Engelle karşılaştıktan sonra karıncaların yollarına devam etmeleri Şekil 3.23'de gösterilmiştir.

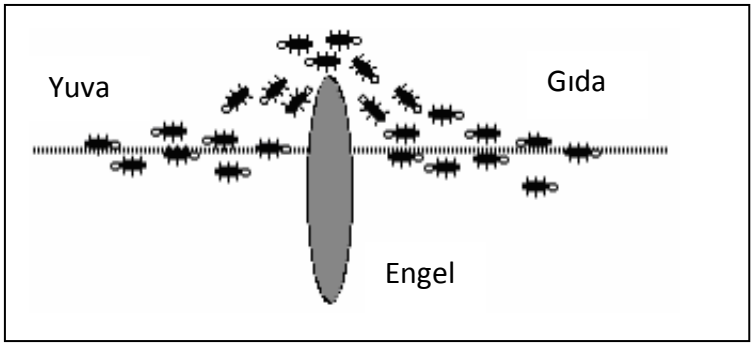


Şekil 3.23 Engelle karşılaşan karıncaların seçimi (Dalkılıç ve Türkmen [74])

Yolları üzerine cisim konan karıncalar, cisimin etrafındaki daha kısa yolu rastgele seçerler ve bu kısa yolu tercih eden karıncalar uzun yolu tercih eden karıncalarla kıyaslandığında, kesilmiş olan feromon maddesi bağlantı yolunu çok daha hızlı şekilde oluştururlar. Bu durumda karıncaların hızlarının hemen hemen aynı olduğu varsayıldığında, birim zamanda kısa yol üzerinden geçen karıncaların sayısının birim zamanda uzun yoldan geçen karıncaların sayısından daha fazla olacağı açıktır.

Kolonideki karıncaların yaklaşık olarak geçtikleri yola eşit seviyede feromon maddesi bıraktıkları kabul edilirse, birim zamanda kısa yola bırakılan koku miktarı çok fazla olacaktır. Yön tercihinin koku maddesi ile bağlantılı olması, yeni gelen karıncaların kısa yolu daha yüksek ihtimalle seçmelerine ve bu yola daha fazla kokunun depolanmasına sebep olacaktır. Böylece, kısa sürede kolonideki çoğu karınca yeni kısa yolu seçerek değişmiş olan çevreye adaptasyon sağlayacaktır. Yani karınca davranışları üzerinde pozitif bir geri besleme söz konusudur. Bir yoldan geçen karınca sayısı arttıkça yola bırakılan koku miktarı artmakta ve koku miktarı arttıkça da yolu tercih eden karınca sayısı artmaktadır. Bu olay "Oto-Katalitik" olarak adlandırılır. Pozitif geri besleme olayından dolayı, karıncaların çoğunluğunun kısa süre içerisinde daha kısa yolu seçmesi gerçekleşir. Bu pozitif besleme işleminin ilginç yönü olarak, cismin etrafındaki en kısa yolun keşfedilmesi, karıncaların dağıtılmış davranışı ile cismin şekli arasındaki etkileşimin ortaya çıkardığı bir özellik olarak görülmesi sayılabilir. Tüm karıncaların yaklaşık olarak aynı hızda hareket etmelerinden ve yola aynı oranda feromon maddesi bırakmalarından dolayı, cisimlerin daha uzun taraflarından dolaşmak kısa taraflarından dolaşmaktan daha uzun süre alacağı ve bu sebeple feromon maddesinin kısa tarafta daha hızlı toplanacağı da bir gerçektir. Bundan dolayı feromon maddesinin kısa yol üzerinde daha hızlı toplanmasını sağlayan neden, karıncaların yön seçerken daha

yüksek feromon maddesine sahip yönleri tercih etmeleridir (Karaboğa [35]). Şekil 3.24'de engelle karşılaştıktan sonra yeniden en kısa yolu bulan karıncalar gösterilmiştir.



Şekil 3.24 Karıncaların kısa yolu bulmaları (Dalkılıç ve Türkmen [74])

Gerçek karınca davranışlarının matematiksel modeller üzerine aktarılmasına dayalı bir metot olan karınca kolonisi optimizasyonunda amaç, kullanılan yapay karıncalar ile problemdeki en kısa yolu bulmayı başarmaktadır. Bu yapay karıncalar bazı özellikleri doğal karıncalardan aynen alırken, problemin performansını arttırmak için doğal karıncalardan bu yapay karıncalara bazı yeni özellikler de eklenmiştir.

Yapay karıncalar, gerçek karıncalar gibi kendi aralarında feromon kimyasalı ile haberleşirken feromon kimyasalının fazla olduğu yollar karıncalar tarafından öncelikle tercih edilir ve böylelikle kısa yollar üzerindeki feromon miktarı bu yolu kullanan karıncalar sayesinde daha hızlı artar.

Doğal karıncalardan farklı olarak ise yapay karıncalara, belirli bir hafızaya sahip olma özelliği, doğal karıncaların tersine tamamen kör olmama özelliği ve ayrık zamanlı bir çevrede yaşayabilme özelliği verilmiştir.

3.2.2 Karınca Kolonisi Optimizasyonu Algoritmaları

Karınca kolonisi optimizasyonu ile ilgili ilk çalışma Dorigo ve arkadaşları tarafından 1991 yılında yapılmış ve yazarların kendi sistemleri "Karınca Sistemi", ortaya çıkardıkları algoritma da "Karınca Algoritması" olarak tanımlanmıştır. Bulunan bu algoritmanın performansını arttırmak ve algoritmanın uygulanabileceği çözüm alanlarını geliştirmek amacıyla birçok değişik karınca kolonisi algoritması ortaya

konmuştur. Bu bölümde sırasıyla literatürde adı geçen karınca kolonisi algoritmaları incelenecektir.

3.2.2.1 Karınca Sistemi

Karınca Sistemi (KS), 1991 yılında Dorigo tarafından ortaya konan ilk karınca kolonisi optimizasyon algoritmasıdır. Bu sistem, ilginç ve başarılı uygulamalarından dolayı birçok karınca kolonisi algoritmasının temel prototipi niteliğindedir (Dorigo vd. [75]).

Karınca sistemi ile ele alınan bir problemin sonucunun, o problemi meydana getiren *n* adet varlığın (gezgin satıcı problemindeki şehirler ya da atölye tipi çizelgeleme problemindeki operasyonlar gibi) permütasyonu olduğu varsayılır (Alaykıran ve Engin [76]).

Karınca sisteminde, bir grafik üzerinde bulunan şehirlerin birinden diğerine hareket eden gezgin satıcı probleminin çözümleri (turları) yapay karıncalar ile inşa edilir. Algoritma adımları durdurma kuralı olan maksimum sayıdaki iterasyona (t_{max}) ulaşana kadar devam ettirilir. Her bir iterasyon boyunca *m* tane karınca, olasılıksal karar kuralı (durum geçiş kuralı) ile *n* adımda tur oluşturur. Uygulamada, bir karınca *i* şehrinden *j* şehrine hareket etmeyi seçerse, (*i,j*) yayı tur yapısına dahil edilir. Bu adım karınca, turunu tamamlayana kadar devam ettirilir. Karınca sisteminde tüm karıncaların turlarını tamamlamasının ardından, her bir karınca ziyaret ettikleri yaylara feromon maddesi bırakırlar. Feromon maddesinin bırakılması işlemi feromonun buharlaşma prosedürünü takiben yapılır (Dorigo vd. [75]).

(*i,j*) yayı üzerindeki feromon izi miktarı, *i* şehrinden sonra *j* şehrinin seçilme kararında etkilidir. Feromon izi bilgisi, karıncaların tecrübelerini yansıtacak şekilde problem çözümü boyunca değiştirilir. Karıncalar, ürettikleri çözümlerin kalitesine göre feromon miktarlarını kısmi olarak bırakırlar. En kısa turu üreten karınca bu kısa tur üzerine büyük miktarda feromon bırakır. Bu seçim en iyi çözümlerin bulunması için yardımcı olur. Feromon buharlaşmasının temel görevi ise, problem çözümünü durağanlıktan uzaklaştırmak ve karıncaların tümünün aynı turu yapmasını engellemektir.

Karınca sisteminde, her bir karınca belli bir hafızaya sahiptir. Bu hafıza karıncanın ziyaret ettiği şehirleri içerir ve "Tabu Listesi" olarak adlandırılır. Bu hafıza, her bir *k*

karıncası için, ziyaret ettiği şehirleri tanımlar ve ziyaret edilmiş şehirlerin tekrar ziyaretini önler.

Yerel sezgisel değerleri ile birlikte yerel feromon izi değerleri *i* düğümünde $A_i = \left[a_j^i(t)\right]_{|\mathrm{N}_i|}$ olarak belirtilen "Karınca Karar Tablosu" ile aşağıda verilen formülle elde edilir (Dorigo vd. [75]):

$$a_j^i \frac{\left[\tau_{ij}(t)\right]^\alpha \left[\eta_{ij}\right]^\beta}{\sum_{l\in \mathrm{N}_i} \left[\tau_{il}(t)\right]^\alpha \left[\eta_{il}\right]^\beta} \qquad \forall j \in \mathrm{N}_i \tag{3.8}$$

Burada $\tau_{ij}(t)$, *t* zamanında (*i,j*) yayı üzerindeki feromon izinin miktarı, $\eta_{ij}=1/d_{ij}$ *i* düğümünden *j* düğümüne hareketin sezgisel değeri, N_i düğümün komşular seti, *α* ile *β* ise sırasıyla feromon izinin ilgili ağırlığını ve sezgisel değeri kontrol eden parametrelerdir.

Algoritmanın *t.* iterasyonda, *k* karıncasının turunu yapılandırırken *i* şehrinden sonra $j \in \mathrm{N}_i^k$ şehrine gitme olasılığı,

$$\mathrm{P}_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{l\in \mathrm{N}_i^k} a_{il}(t)} \tag{3.9}$$

dir. Burada, $\mathrm{N}_i^k \subseteq \mathrm{N}_i$ *i* düğümünün *k* karıncası tarafından henüz ziyaret edilmeyen komşu düğümlerinin setidir. N_i^k içindeki düğümler, N_i'den karıncanın M^k özel hafızası ile seçilir (Dorigo vd. [75]).

α ve *β* parametreleri ise feromon miktarını ve sezgiselliği gösteren parametrelerdir. Eğer *α=0* ise, karınca tarafından en yakın şehrin seçilmesi muhtemeldir. Buna karşın, eğer *β=0* ise sadece feromon yükselmesi çalışır. Bu metot, genellikle durağanlaşma durumunun hızla ortaya çıkmasına olanak sağlayarak güçlü yarı optimal turların üretilmesine neden olur. Ortaya çıkabilecek bu durağanlığın engellenmesi için sezgisel değer ile feromon iz değerinin değiş-tokuş yapılması gerekir.

Tüm karıncalar turlarını tamamladıktan sonra, tüm kenarlarda feromon buharlaşması gerçekleşir ve her bir *k* karıncası geçtiği her kenara $\Delta\tau_{ij}^k(t)$ miktarında feromon bırakır:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}1/L^{k}(t) & \text{eğer } (i,j)\in T^{k}(t)\\ 0 & \text{eğer } (i,j)\notin T^{k}(t)\end{cases} \tag{3.10}$$

Burada, $T^k(t)$, t iterasyonunda k karıncası tarafından yapılan tur ve $L^k(t)$ de yapılan bu turun uzunluğudur. $\Delta\tau_{ij}^{k}(t)$ değeri ise, karıncanın daha kısa bir tur yapmasına bağlı olarak daha yüksek miktarda feromon bırakılmasına bağlıdır.

Uygulamada, karıncalar tarafından yeni feromon ilavesi ve feromon buharlaşması aşağıda verilen kuralın tüm kenarlara uygulanması ile gerçekleştirilir (Rizzoli vd. [77]):

$$\tau_{ij}(t)\leftarrow(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{3.11}$$

Burada, $\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t)$'dir. m her bir iterasyondaki karıncaların sayısı ve $\rho\in(0,1]$ feromon ayrışma katsayısıdır. Feromonun başlangıç değeri $\tau_{ij}(0)$, tüm kenarlar üzerinde τ_0 gibi küçük pozitif bir sabite ayarlanır ve karıncaların toplam sayısı $m=n$ olarak belirlenir (Dorigo vd. [75]).

Genel olarak, karınca sistemi, optimizasyon probleminin çözümünde feromon modelinin kullanılmasıyla aday çözümlerin yapılandırılması ve feromon değerlerinin modifiye edilmesinde aday çözümlerin kullanılması adımlarının tekrarlanmasından yararlanır.

3.2.2.2 Karınca-Yoğunluk, Karınca-Miktar ve Karınca-Çevrim Algoritmaları

Karınca sistemi yapısı temelinde üç farklı yaklaşım şeklinde Colorni, Dorigo ve Maniezzo tarafından 1991 yılında önerilen algoritmalar karınca-yoğunluk, karınca-miktar ve karınca-çevrim algoritmalarıdır.

Karınca-yoğunluk modelinde, k karıncası i şehrinden j şehrine gidişinde bu şehirlerarasındaki (i,j) hattına, birim uzunluk başına Q_1 miktarı kadar feromon maddesi bırakmaktadır. Karınca-miktar modelinde ise i şehrinden j şehrine giden k karıncası, Q_2/d_{ij} miktarı kadar feromon maddesini birim uzunluk başına bırakmaktadır. Bu nedenle, karınca-yoğunluk modelinde, (i,j) hattına k. karınca tarafından bırakılan feromon miktarı aşağıda verilen bağıntı ile tanımlanırken

$$\Delta\tau_{ij}^{k}(t,t+1)\begin{cases} Q_1 & k. \text{ karınca } (i,j) \text{ hattını } t \text{ ve } t+1 \text{ zaman aralığında kullanırsa} \\ 0 & \text{aksi halde} \end{cases} \quad (3.12)$$

karınca-miktar modelinde ise,

$$\Delta\tau_{ij}^{k}(t,t+1)=\begin{cases} Q_2 / d_{ij} & k. \text{ karınca } (i,j) \text{ hattını } t \text{ ve } t+1 \text{ aralığında kullanırsa} \\ 0 & \text{aksi halde} \end{cases} \quad (3.13)$$

ifadesi ile tanımlanır (Karaboğa [35]).

Bu tanımlamalardan, *k* karıncasının *i* şehrinden *j* şehrine gidişinde, bu hatta bulunan feromon maddesinin yoğunluk miktarında oluşacak artışın karınca-yoğunluk modelinde d_{ij} mesafesinden bağımsız olduğu görülmektedir. Bunun tersi olarak, karınca-miktar modelinde ise hatta bulunan feromon maddesinin yoğunluk miktarında oluşacak artışın, d_{ij} mesafesi ile ters orantılı olduğu açıktır. Yani, karınca-miktar modeli daha kısa hatları karıncalar tarafından çok daha arzu edilir hale getirmekte ve böylece seçilebilirlik faktörü önemli derecede kuvvetlenmektedir.

Karınca-yoğunluk ve karınca-miktar algoritmalarında, başlangıçta karıncalar farklı şehirlerde konumlandırılırlar ve hatlara feromon miktarlarının başlangıç değerleri atanır. Her karıncanın tabu listesinin ilk elemanı başlangıç şehrine ayarlanır. Bundan sonra her karınca *i* şehrinden *j* şehrine α ve β parametrelerinin bir fonksiyonu olarak hesaplanan olasılık değerine göre hareket eder. İlk parametre olan feromon maddesi τ_{ij}, aynı (*i,j*) hattını geçmişte ne kadar çok karıncanın seçtiği hakkında bilgi verirken, ikinci parametre olan seçilebilirlik η_{ij}, daha yakın şehrin daha çok tercih edilmesi gerektiğini vurgulamaktadır. $\alpha=0$ yapılarak çoklu başlama noktasına sahip stokastik bir aç gözlü algoritma üretilebilir (Karaboğa [35]).

Bir karınca, hareketi gerçekleştirdiğinde (*i,j*) hattına bırakmış olduğu feromon miktarı, aynı hatta önceden varolan feromon miktarı ile toplanır. Her karınca hareketinden sonra geçiş olasılıkları yeni feromon miktarları kullanılarak hesaplanır.

Karıncaların toplam sefer sayısı, n_3-1 (toplam şehir sayısından bir eksik) olduğunda karıncaların tabu listesi dolar. Kolonideki *m* karınca tarafından bulunan en kısa tur hesaplanır ve hafızaya alınır. Bu işlemin ardından, tüm tabu listeleri boşaltılır ve

işlemler ardışık olarak tur sayıcısı maksimum çevrim sayısına NC_{max} eşit oluncaya veya tüm karıncalar aynı turu oluşturana kadar tekrar ettirilir.

Karınca-çevrim algoritması, karınca-yoğunluk ve karınca-miktar algoritmalarından önemli bir farkla ayrılır. Karınca-çevrim algoritmasında, (*i,j*) hattından geçen karıncanın buraya bırakacağı feromon miktarı $\Delta\tau_{ij}^{k}$, her adımda hesaplanmamakta ancak tur tamamlandıktan sonra yani *n* adımdan sonra hesaplanmaktadır.

(*t,t+1*) zaman aralığında (*i,j*) hattından geçen karıncanın bırakacağı feromon miktarı $\Delta\tau_{ij}^{k}(t,t+n)$ aşağıda verilen formülle belirlenmektedir (Dorigo vd. [78]):

$$\Delta\tau_{ij}^{k}(t,t+n)=\begin{cases} Q_3 / L^k & \text{eğer } k. \text{ karınca } 1. \text{ turda } (i,j) \text{ hattını kullanırsa} \\ 0 & \text{aksi halde} \end{cases} \tag{3.14}$$

Burada, Q_3 bir sabit ve L^k ise *k* karıncasının (*i,j*) hattını da kullanarak tamamlamış olduğu yolun uzunluğudur. Bu algoritma, karınca-miktar yaklaşımının adaptasyonuna karşılık gelmektedir. Feromon miktarları her hareket sonrası değil, bir kapalı tur bittiğinde yenilenmektedir. Karınca-çevrim algoritmasının, karınca-yoğunluk ve karınca-miktar algoritmalarından daha iyi performans göstereceği beklenmektedir. Bu beklentinin sebebi, üretilen çözümün problem için tam bir çözüm olmasıdır. Yani değerlendirme yapmak amacıyla küresel bilginin kullanılmasıdır.

Feromon miktarının değeri, daha önceki formülde belirtildiği gibi *n* adımdan sonra

$$\tau_{ij}(t+n)=\rho_1\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t,t+n) \tag{3.15}$$

formülü ile yenilenir. Bu ifade de

$$\Delta\tau_{ij}(t,t+n)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t,t+n) \tag{3.16}$$

dir. Bu formülde kullanılan ρ_1 daha önce tanımlanan ρ'den farklıdır. Çünkü, denklem artık her adımdan sonra değil sadece tam bir tur üretildikten sonra değişmektedir (Karaboğa [35]).

3.2.2.3 Ant-Q

Gambardella ve Dorigo tarafından 1995 yılında ortaya konan bu algoritma simetrik ve asimetrik mesafeli gezgin satıcı probleminin çözümünde Q-öğrenme ile benzerlik gösterir. Karınca sisteminin temelinde çalışan bu algoritmada kısa turların bulunmasında karıncalar Q öğrenme değerlerini öğrenirler. Algoritmada, α_q ve γ parametreleri "Öğrenme Adımı" ve "Hesaplama Faktörü" olarak kullanılır (Gambardella ve Dorigo [79]).

Ant-Q, karınca kolonisi sisteminden sadece eş zamanlı olarak adım adım feromon güncellemesinde kullanılan τ_0 değeriyle ayrılır. Buradaki düşünce, bir sonraki durumun değer tahminine göre feromon izinin güncellenmesidir. Ant-Q algoritmasında feromon güncellenmesi aşağıda verilen denkleme göre yapılmaktadır (Dorigo vd. [75]):

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \rho\gamma * \max_{l \in N_j^k} \tau_{jl} \tag{3.17}$$

Bu algoritma üzerinde yapılan çalışmalar ile küçük sabit değerler ile yapılan tahminin karmaşık olduğu ve bu algoritmanın karınca kolonisi sistemi ile benzer bir performans gösterdiği ortaya konmuştur.

3.2.2.4 Karınca Kolonisi Sistemi

1996 yılında Dorigo ve Gambardella tarafından mevcut karınca sisteminin performansının geliştirilmesi için ortaya konan Karınca Kolonisi Sistemi (KKS) yapısı literatürde oldukça geniş bir yer bulan gezgin satıcı probleminin çözümü üzerinden aşağıda verildiği gibi tanımlanabilir (Karaboğa [35]):

Gezgin satıcı problemi, *n* tane şehir verildiğinde, yapay gezgin satıcının her bir şehri bir kez ziyaret etmek şartıyla minimum uzunluklu kapalı bir turu oluşturma problemi olarak tanımlanabilir. *i.* ve *j.* şehirlerarasındaki yolun uzunluğu d_{ij} olarak tanımlansın. Öklid gezgin satıcı problemi durumunda d_{ij}, *i* ve *j* şehirleri arasındaki öklid mesafesidir. Yani, $d_{ij} = \left[\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2\right]^{1/2}$ dir. Gezgin satıcı probleminin herhangi bir durumu ağırlıklandırılmış bir G*raf*(*N,E*) ile tanımlanabilir. Burada, *N* şehirler seti ve *E* ise mesafelerle ağırlıklandırılmış şehirlerarasındaki bağlantılar setidir. *m*, kolonideki toplam karınca sayısını gösterir ve *t* anında *i* şehrinde bulunan karıncaların sayısı $b_i(t)$ (*i*

= 1, 2, …., n) ile verilirse bu durumda bu iki büyüklük arasındaki ilişki için $m = \sum_{i=1}^{n} b_i(t)$ bağıntısı yazılabilir. Buna ek olarak her karıncanın aşağıda verilen özelliklere sahip bir bireyi temsil ettiği kabul edilir.

Karınca,

- *i* şehrinden *j* şehrine giderken (*i,j*) hattına bir miktar feromon maddesi bıraksın.
- Gideceği şehri, o hatta mevcut olan feromon maddesi miktarı ile iki şehir arasındaki mesafenin fonksiyonu olan bir olasılıkla seçsin.
- Ziyaret edilmiş şehirlerin tekrar ziyaret edilmesi yasaklanarak tur tamamlansın.

$\tau_{ij}(t)$, *t* anında *i* ve *j* şehirleri arasındaki (*i,j*) hattında depolanan feromon maddesi miktarını gösterirse, (*t+1*) anındaki feromon maddesi miktarı aşağıda verilen denklem ile tanımlanabilir:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t,t+1) \tag{3.5}$$

Burada *ρ*, buharlaşma katsayısı olarak tanımlanan bir parametredir ve feromon maddesinin sınırsız bir şekilde büyümesini önlemek için *1*'den küçük pozitif bir değere ayarlanmalıdır. (*1–ρ*) faktörü ise feromon maddesinin buharlaşan oranını belirlemektedir. Birim zamanda, (*i,j*) hattına bırakılan feromon maddesi miktarı ise,

$$\Delta\tau_{ij}(t,t+1) = \sum_{k=1}^{n} \Delta\tau_{ij}^{k}(t,t+1) \tag{3.18}$$

olarak tanımlanabilir ve burada $\Delta\tau_{ij}^{k}(t,t+1)$, *t* ve (*t+1*) zaman aralığında *k* karıncası tarafından (*i,j*) şehir hattına bırakılan feromon maddesinin birim uzunluk başına miktarıdır.

Bir karıncanın *n* farklı şehri (*n* şehirli tur) ziyaret etmesi şartını sağlaması amacıyla her karıncaya, bir veri yapısı atanır (tabu listesi). Böylece, karınca *t* anına kadar ziyaret etmiş olduğu şehirleri hafızaya alır ve tur tamamlanıncaya kadar bu şehirlerin tekrar ziyaret edilmesi önlenir. Tur tamamlandığında tabu listesi boşaltılır ve karınca yolunu seçmek için tekrar serbest kalır. *k.* karıncanın tabu listesini içeren bir vektör, $tabu_k$

tanımlanır ve $tabu_k(s)$, k karıncasının tabu listesindeki $s.$ elemanını yani karınca tarafından ziyaret edilmiş $s.$ şehri gösterir.

Seçilebilirlik parametresi olarak adlandırılan η_{ij}, $\eta_{ij} = \frac{1}{d_{ij}}$ şeklinde tanımlandıktan sonra, k karıncasının i şehrinden j şehrine geçiş olasılığı aşağıda verilen bağıntı ile tanımlanır:

$$p_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum [\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}, & \text{eğer } j \in izin \text{ verilen değerler} \\ 0, & \text{aksi halde} \end{cases} \tag{3.4}$$

Burada, izin verilen şehir: $j \notin tabu_k$ anlamındadır ve α ile β parametreleri kullanıcıya, olasılık değerini hesaplarken yapay feromon maddesi ile seçilebilirlik arasında nispi önemi belirleme imkanı vermektedir. Bundan dolayı geçiş olasılığı, seçilebilirlik değeri ve feromon maddesi miktarı arasında bir uzlaşmayı tayin eder.

Karınca kolonisi sistemi, karınca sistemi temelli olsa da, yapısında birçok farklılıklar taşır (Dorigo vd. [75]).

İlk olarak durum geçiş kuralı yeni ve önceki kenarların ortaya çıkarılmasında ve problem hakkında bilgi toplanmasında direkt bir yol sağlar. İkinci olarak, global feromon güncellemesi sadece en iyi karınca turu ile ilgili kenarlara uygulanır. Feromon güncellemesi için algoritmanın iterasyonları sonunda, tüm karıncalar bir kez çözüm yapılandırır ve en iyi turu bulan karıncanın geçtiği kenarlara feromon maddesi bırakılır. Üçüncü olarak ise, karıncalar çözümü yapılandırırken yerel feromon güncelleme kuralı uygulanır. Karınca kolonisi sisteminde karıncalar feromon güncellemelerini sadece adım adım gerçekleştirir ve bu güncellemeler aşağıda belirtilen kurala göre yapılır (Gambardella ve Dorigo [80]):

$$\tau_{ij}(t) \leftarrow (1-\varphi)\tau_{ij}(t) + \varphi\tau_0 \tag{5.19}$$

Burada $0 < \varphi < 1$'dir. Bu kurala göre, k karıncası i şehrinden $j \in \mathrm{N}_i^k$ şehrine hareket ederken (i,j) kenarında feromon güncellemesi yapar. τ_0 değeri, feromon izlerinin

başlangıçtaki değeriyle aynıdır ve $\tau_0 = (nL_{nn})^{-1}$'e ayarlanır. Burada, *n* şehirlerin sayısı ve L_{nn} ise en yakın komşu sezgiseli ile üretilen turun uzunluğudur (Dorigo vd. [75]).

Karınca kolonisi sisteminde geçiş kuralı: Karınca kolonisi sisteminde bir tur esnasında, *i* noktasında bulunan *k* karıncası için, sonraki *j* noktasını seçerken iki alternatif yol söz konusudur. İlk alternatif, gidebileceği yollar içerisinden feromon miktarlarına bağlı olarak hesaplanan seçim değerlerinden maksimum olanını seçmesidir. Genellikle bu yolla tercih yapma olasılığı (q_0) *%90* olarak belirlenmektedir. İkinci alternatifte ise yollardaki feromon miktarları göz önüne alınarak oluşturulan olasılık dağılımına bağlı olarak yollar seçilir (Söyler ve Keskintürk [81]).

Karınca kolonisi sisteminde kullanılan sözde-rastlantısal-orantılı durum geçiş kuralı aşağıda verilen bağıntı ile ifade edilir (Maniezzo vd. [82]):

$$s = \begin{cases} \arg \max\limits_{(ij) \notin tabu_k} \left\{ \tau_{ij}^{\alpha} * \eta_{ij}^{\beta} \right\} & \text{eğer } q \le q_0 \\ P_{ij}^{k} & \text{aksi halde} \end{cases} \tag{3.20}$$

Karınca kolonisi sisteminde, karıncalar bu sözde-rastlantısal-orantılı durum geçiş kuralını kullanırlar. Bu kuralla, *i* şehrindeki *k* karıncası gideceği $j \in \mathrm{N}_i^k$ şehrini seçerken aşağıda belirtilen $\mathrm{A}_i = \left\lfloor a_{ij}(t) \right\rfloor_{|\mathrm{N}_i|}$ karınca-karar tablosunu kullanır (Dorigo vd. [75]):

$$a_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in \mathrm{N}_i} \left[\tau_{il}(t)\right]\left[\eta_{il}\right]^{\beta}} \qquad \forall j \in \mathrm{N}_i \tag{3.21}$$

q, [0,1] aralığında normal dağılıma uyan rassal bir değişkendir ve $q_0 \in [0,1]$ ayarlanabilir bir parametredir. Sözde-rastlantısal-orantılı kural, *i* düğümündeki *k* karıncası tarafından bir sonraki $j \in \mathrm{N}_i^k$ düğümünün seçiminde aşağıda verildiği şekliyle kullanılır:

Eğer $q \le q_0$ ise,

$$P_{ij}^{k}(t) = \begin{cases} 1 & \text{eğer } j = arg\ max\ a_{ij} \\ 0 & \text{aksi halde} \end{cases} \tag{3.22}$$

Tersi olarak, eğer $q > q_0$ ise

$$P_{ij}^{k}(t)=\frac{a_{ij}(t)}{\sum_{l\in N_i^k} a_{il}(t)} \tag{3.23}$$

dir. Bu karar kuralı çift fonksiyona sahiptir. $q \le q_0$ olduğunda kural problem hakkındaki erişilebilir bilgiyi kullanır. Bu bilgi şehirlerarasındaki mesafeler hakkındaki sezgisel bilgi ve feromon izleri şeklinde hafızaya alınan öğrenilmiş bilgidir. Diğer taraftan $q > q_0$ olduğunda ise, taraflı araştırma yapılır. q_0, araştırma derecesini ayarlamaya, en iyi çözümlerde sistemin aktivitesini yoğunlaştırmayı seçmeye veya çözüm uzayının araştırılmasına izin verir.

Karınca kolonisi sisteminde feromon güncelleme: Tüm karıncalar turlarını tamamladıktan sonra feromon miktarları güncellenmektedir. İlk olarak tüm yollardaki feromonlar, belirlenen oranda (buharlaşma oranı) buharlaştırılmaktadır. Daha sonra karıncaların geçiş yapmış oldukları yollardaki feromon miktarları, o yolu kullanan karıncanın toplam yol uzunluğuyla ters orantılı olarak arttırılmaktadır. Böylelikle daha kısa yola sahip karıncaların kullandıkları yollardaki feromon miktarları daha fazla artış göstermektedir (Söyler ve Keskintürk [81]).

Karınca koloni sistemi içerisinde feromon güncellemesi iki şekilde yapılmaktadır. Bunlar "Yerel Feromon Güncellemesi" ve "Global Feromon Güncellemesi" olarak adlandırılır ve aşağıda verildiği gibi açıklanırlar:

Yerel feromon güncellemesi: Tüm karıncalar turlarını tamamladıktan sonra, eski feromon miktarları belli bir oranda buharlaştırılır, her bir karıncanın turu boyunca geçiş yapmış olduğu yollarda belirli bir miktarda feromon artışı sağlanır. Bu işlemler aşağıda verilen formülle ifade edilir (Söyler ve Keskintürk [81]):

$$t_{ij}(t+1)=(1-\rho)t_{ij}(t)+\sum_{k=1}^{m}\Delta t_{ij}^{k}(t+1) \tag{3.24}$$

Burada $\tau_{ij}(t)$ başlangıç feromon düzeyidir ve ρ, $(0\le\rho\le1)$ feromon buharlaşma parametresidir. Yerel güncelleme kuralı, turları dinamik olarak değiştirerek geçiş yapılan yolları cazip hale getirir. Karıncalar farklı yollardan giderken, yüksek bir olasılıkla bunlardan biri önceki çözümlerden daha kısa bir yoldan giderek çözümü iyileştirir. $\Delta\tau_{ij}^{k}(t+1)$, değişik karınca kolonisi algoritmalarında farklı yöntemlerle

hesaplanmaktadır. En çok kullanılan yerel feromon güncelleme formülü aşağıda verildiği gibidir:

$$\Delta\tau_{ij}^{k}(t+1)=\begin{cases}1/L^{k}(t+1) & k.\text{ karınca }(i,j)\text{ yolunu kullanmışsa}\\ 0 & \text{aksi halde}\end{cases} \quad (3.6)$$

$L^k(t + 1)$, *k* karıncasının toplam tur uzunluğudur.

Global feromon güncellemesi: Global feromon güncellemesi tüm karıncalar turlarını tamamladıktan sonra yapılır. Karıncaların her birinin toplam yol uzunlukları hesaplandıktan sonra en kısa yolu kullanan karınca bulunur. Bu karıncanın geçiş yapmış olduğu yollardaki feromon miktarları aşağıda verilen formüle göre arttırılır (Söyler ve Keskintürk [81]):

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}^{k}(t+1) \quad (3.25)$$

$$\Delta\tau_{ij}(t+)=\begin{cases}1/L_{best}(t+1) & (i,j)\text{ en iyi tura ait ise}\\ 0 & \text{aksi halde}\end{cases} \quad (3.7)$$

Burada, $L_{best}(t + 1)$ geçerli iterasyonda bulunan en iyi turun uzunluğudur.

3.2.2.5 Karıncalar

Karıncalar (ANTS), Maniezzo tarafından 1999 yılında ortaya konmuştur. Bu algoritma, karınca sisteminin genişletilmiş bir halidir. Karıncalar algoritması, kullanılan çekicilik fonksiyonu veya başlangıç feromon dağılımı gibi genel algoritmanın bazı bileşenlerini belirtir. Bu noktada, genel karınca kolonisi optimizasyonu çerçevesinin bir varyasyonu olarak, sonuç algoritma ağaç arama algoritmalarının yapısına benzer olarak ortaya çıkar. Karıncaları, ağaç arama algoritmasından ayıran özellik, hareket edilecek durum seçeneğinin olasılıksal (deterministik olmayan) araştırmasının ve arama ağacının tamamlanmamış (yaklaşık) araştırmasının yerine kullanımında tamamen gerileme mekanizmasının eksikliğidir. Karıncalar, ANTS adını yaklaşık deterministik olmayan ağaç arama (Approximated Non-deterministic Tree Search) algoritmasının baş harflerinden alır (Maniezzo vd. [82]).

Karıncalar algoritmasında, hareketin çekiciliği kısmi çözümün tamamlanma maliyeti üzerindeki en düşük sınırların (maksimizasyon problemleri için en yüksek sınırların) ortalaması tarafından etkili bir şekilde tahmin edilir. Feromon güncelleme mekanizmasında ise durağanlaşmadan kaçınılır ve feromon güncelleme prosedürü her çözüme karşı en son global olarak ANTS tarafından yapılandırılmış çözümlerin değerlendirilmesiyle gerçekleştirilir.

3.2.2.6 Maksimum-Minimum Karınca Sistemi

Maks-Min Karınca Sistemi (MMKS), Stützle ve Hoos tarafından 2000 yılında geliştirilmiştir. Bu karınca sistemi ile karınca kolonisi optimizasyonu üzerine yapılan araştırmalarda en iyi çözüme sahip olan karıncayı çözüm araştırması esnasında daha fazla kullanmanın performansı arttırdığı gözlemlenmiştir. Ayrıca Stützle ve Hoos'un çözüm uzayı üzerine yaptıkları analiz de bu sonucu desteklemektedir. Karınca sistemi yaklaşımıyla daha geniş aramalar, potansiyel olarak çözüm uzayındaki aramanın durağanlaşması riskini fazlasıyla içinde barındırmaktadır. Bu nedenle, Stützle ve Hoos, karınca kolonisi optimizasyonu algoritmalarının en iyi performansına ulaşma yönteminin etkili bir mekanizmayla arama esnasında bulunan en iyi sonuçların daha fazla kullanılması olabileceğini önermişlerdir. Bunun temelinde yazarların geliştirdikleri maks-min karınca sistemi aşağıda verilen üç madde ile karınca sistemi yaklaşımından ayrılmaktadır (Cura [21]):

- Bir çevrimde en iyi sonuçların kullanılabilmesi için her çevrimde tek bir karıncanın feromon izini güncellemesi gerekmektedir. Söz konusu karınca ya mevcut çevrimdeki en iyi sonucu bulan karınca olur (*çevrim_eniyi*) veya algoritmanın başından beri elde edilmiş en iyi sonucunu bulan karınca olur (*global_eniyi*).
- Çözüm uzayındaki aramanın durağanlaşmasından, başka bir ifadeyle tıkanmasından kaçınmak için mümkün feromon izlerinin güncellenmesi [τ_{min}, τ_{max}] aralığıyla kısıtlanır. Böylece güncellenen feromon bu aralığın dışında değer alamaz.

Feromon izlerinin sınırlı aralığı [τ_{min}, τ_{max}], $\tau_{min} \leq \tau \leq \tau_{max}$ ise,

$$\tau_{\max} = \frac{1}{\rho}\frac{1}{L^{best}} \tag{3.26}$$

$$\tau_{\min} = \frac{\tau_{\max}}{2n} \tag{3.27}$$

dir. Burada, L^{best} bulunan en iyi turun uzunluğu, ρ feromon buharlaşma katsayısı, n ise karıncaların hareket ettikleri sistemdeki nokta sayısını gösterir.

- Stützle ve Hoos feromon izlerini başlangıçta kasıtlı olarak τ_{max}'a eşitlemişlerdir. Bu yolla algoritmanın başında daha geniş çözüm araştırmaları sağlamışlardır.

3.2.2.7 Rank Temelli Karınca Sistemi

Rank Temelli Karınca Sistemi (RTKS)'nde, sadece belli sayıdaki karıncaların geçiş yapmış olduğu yolların feromon izlerinin yenilenmesine izin verilir. Karıncalar tur uzunluklarına göre sıralanır ($L1(t) \leq L2(t) \leq \ldots Lm(t)$) ve belirlenen sayıdaki karıncanın geçiş yaptığı yollardaki feromon düzeyi tur uzunluklarına bağlı olarak yenilenir. Dolayısıyla global en iyi çözümün feromon güncelleme düzeyi en yüksektir. Rank temelli karınca kolonisi algoritmasında kullanılan feromon güncelleme formülü aşağıda verildiği gibidir (Söyler ve Keskintürk [81]):

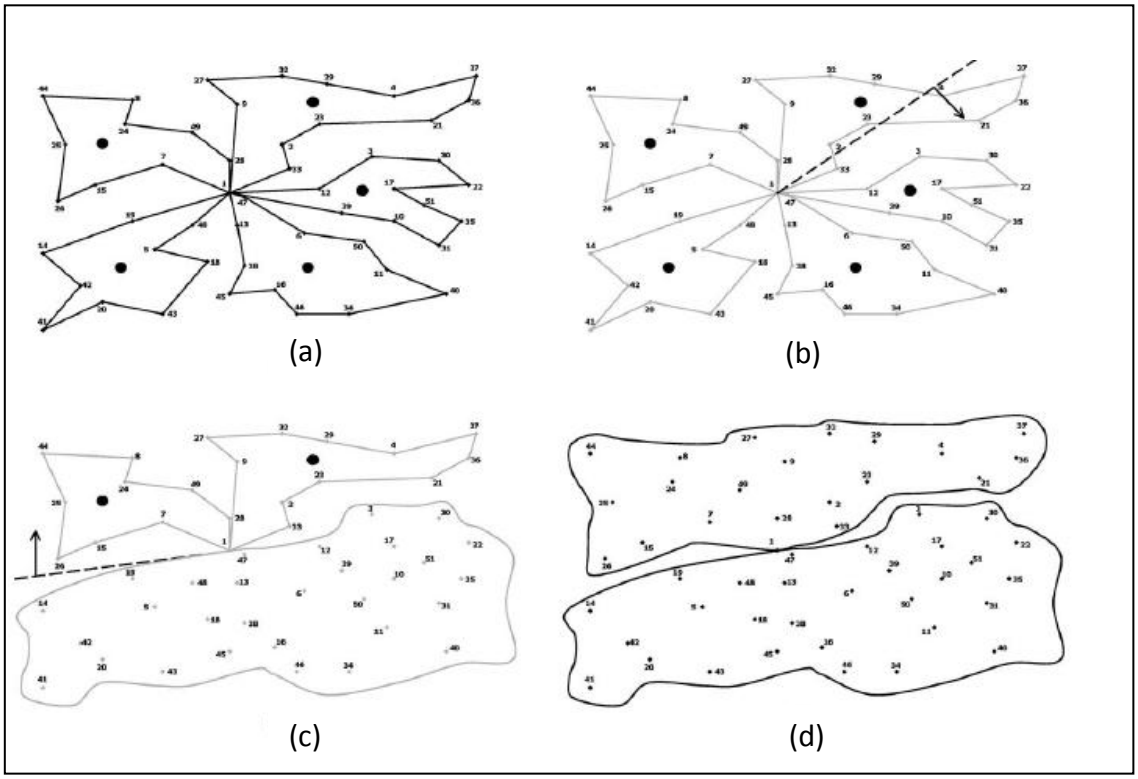
$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{t}(t) + w\Delta\tau_{ij}^{best}(t) \tag{3.28}$$

$$\Delta\tau_{ij}^{t}(t) = \frac{1}{L^{r}(t)}, \Delta\tau_{ij}^{gb}(t) = \frac{1}{L^{best}} \tag{3.29}$$

3.2.2.8 D-Ants

Reimann ve diğerleri tarafından ortaya atılan bu algoritmada ayrıştırılan karıncalar kullanılır. Ayrıştırma yaklaşımı Taillard'ın tabu arama algoritması ile birleştirilmiştir. D-ants algoritması tasarruf temelli karınca sistemi üzerine yapılandırılmıştır. Bu algoritma, araç rotalama probleminin çözüm prosedürü içerisinde yer alır ve problemin tamamını çözmektense süpürme algoritması ile belirlenmiş kümelere uygulanır. Karınca sistemi, mevcut problem için en iyi sonucu bulana kadar iterasyon döngüsü içerisinde sürekli D-

ants uygulanır (Reimann vd. [83]). Şekil 3.25'de D-ants ile ayrıştırma adımları gösterilmiştir.



Şekil 3.25 D-Ants ile ayrıştırma adımları (Reimann vd. [83])

3.2.2.9 Çoklu Karınca Sistemi

Çoklu karınca sistemi yaklaşımı, iki veya daha fazla koloninin çözümünün birleştirilmesi esasına dayanır. Bu konudaki iki temel yaklaşımdan ilki olan heterojen yaklaşım, tüm kolonilerin birbirinden farklı davranış sergilemesi esasına dayanmaktadır ve çok kriterli optimizasyon problemlerinin çözümünde kullanılır. İkinci yaklaşımda ise, tüm koloniler birbirleriyle paralel olarak çalışırlar ve her *i* jenerasyon sonunda koloniler arasında feromonlar vasıtası ile karşılıklı bilgi alış-verişi olur. Sözü edilen ikinci yaklaşım, bilgi alış-verişinin sınırlı olduğu ve çok kısa olmayan aralıklarla yapıldığı takdirde başarılı sonuçlar vermektedir. Bilgi alışverişi için önerilen 4 yöntem aşağıda verildiği gibi sıralanabilir (Keskintürk ve Söyler [84]):

- Her bilgi alış-verişinde global en iyi çözümler hesaplanır ve tüm kolonilere gönderilir. Böylece her koloni yeni bir yerel en iyi çözüme sahip olur.

- Koloniler arasında bir köprü kurulur ve her bilgi alış-verişi adımında, her koloni yerel en iyi çözümleri en yakınındaki bir veya birden fazla koloniye gönderir.
- Bir önceki seçenekteki bilgiler kullanılarak her bilgi alışverişi adımında, bir kolonideki en iyi m_b karınca, en yakınındaki en iyi m_b karıncayla karşılaştırılır ve $2m_b$ karıncanın en iyi m_b tanesi alınarak feromon güncellemesinde kullanılır.
- Bu seçenek ikinci ve üçüncü seçeneklerin birleştirilmesinden oluşur.

3.2.2.10 Melez Karınca Kolonisi Optimizasyonu

Karınca kolonisi optimizasyonu ile herhangi bir yaklaşımın birleştirilmesiyle elde edilen yeni algoritmalar genel olarak "Melez Karınca Kolonisi Optimizasyonu (MKKO)" olarak adlandırılırlar. Örneğin, Sitarz [85] te karınca kolonisi optimizasyonu ile tavlama benzetimi algoritmasını çok kriterli dinamik programla problemine uygularken, McKendall ve Shang [86] da aynı iki tekniği dinamik tesis yerleştirme probleminin çözümünde kullanmışlardır. Huang ve Liao [87] de ise iş çizelgeleme problemi için karınca kolonisi optimizasyonu ile tabu arama algoritmasını önermiştir. Tseng ve Chen [8] de kaynak kısıtlı proje çizelgeleme problemi için genetik algoritma ve karınca kolonisi optimizasyonundan oluşan yeni ANGEL isimli bir algoritma ortaya koyarken, Lee ve diğerleri [9] da çoklu ardışık sıralama problemi için yine genetik algoritma ve karınca kolonisi optimizasyonundan oluşan melez bir yapı önermişlerdir.

3.2.2.11 Global Karınca Kolonisi Optimizasyonu

Global Karınca Kolonisi Optimizasyonu (GKKO), bilinen karınca kolonisi optimizasyonu algoritmalarından farklı bir yapıya ve uygulama alanına sahiptir. En temel farklar, karıncaların tam bir tur yapmıyor olmaları, yerel feromon güncellemesinin olmayışı ve yol seçimlerinin düğümler arası değil yolun tamamı için yapılmasıdır. Ayrıca farklı yolların araştırılmasını sağlamak üzere genetik algoritmadaki mutasyon benzeri bir operatör de global karınca kolonisi optimizasyonuna eklenmiştir. Özellikle sipariş büyüklüğü problemlerine (Lot Sizing Problem-LSP) alternatif bir çözüm önerisi olarak geliştirilen bu algoritmada herhangi bir düğümden başlayarak ve herhangi bir ya da birden fazla düğüme uğrayarak bir çözüm alternatifi geliştirmek mümkündür. Mevcut

karınca kolonisi optimizasyonu algoritmalarından ayrılan bu yeni algoritmada feromon güncellemesi ve karıncaların yol seçimi yolun tamamı dikkate alınarak gerçekleştirilmektedir (Keskintürk ve Söyler [84]).

3.2.2.12 İyileştirilmiş Karınca Kolonisi Sistemi

Karınca kolonisi sisteminin geliştirilmesi ve performansının arttırılması için önerilen algoritmalardan biri de "İyileştirilmiş Karınca Kolonisi Sistemi (Improved Ant Colony System-IACS)"dir. Bu algoritma ilk kez Ting ve Chen tarafından 2004 yılında ortaya konmuştur. Dorigo ve Gambardella'nın [88] de önerdiği karınca kolonisi sistemi temelli olan bu algoritmanın içerdiği dört adım aşağıda verildiği gibidir (Chen ve Ting [89]):

Adım 1. Parametreler ve başlangıç feromon izleri ayarlanır.

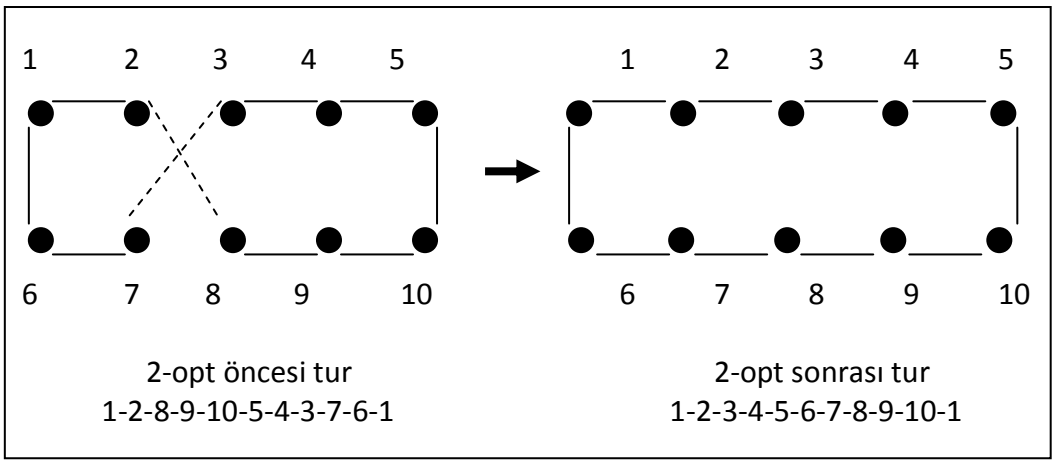
Adım 2. Her bir karınca durum geçiş kuralı ile çözüm inşa eder ve yerel feromon güncellemesi yapılır.

Adım 3. Karıncaların çözümünü geliştirmede yerel arama uygulanır.

Adım 4. Global feromon bilgisi güncellenir.

3.2.2.13 ACS - 3-opt

Karınca kolonisi sistemine eklenmiş olan bir diğer yaklaşımda 3-opt yerel arama sezgiselidir. Yerel arama sezgisellerinden biri olan 2-opt yaklaşımında verilen bir başlangıç turu (τ) için iki nokta (i,j) seçilip, bu iki nokta arasındaki düğümlerin ters çevrilmesiyle oluşacak yeni turda (τ') iyileştirme olup olmadığı araştırılır (Baykoç ve İşleyen [90]). Bu işlemin tamamlanmasıyla optimuma oldukça yakın sonuçlar bulunur (Demirten [91]). Şekil 3.26'da 2-opt yerel arama sezgiselinin uygulama örneği verilmiştir.



Şekil 3.26 2-opt yerel arama sezgiseli örneği

3-opt yerel arama sezgiselinde ise, 2-opt sezgiselinden farklı olarak 3 kenar karşılaştırılır ve bu yöntem 2-opt yönteminden daha iyi sonuçlar verir (Demirten [91]).

Dorigo ve Gambardella [88] de karınca kolonisi sistemi ile bu 3-opt yerel arama sezgiselini birleştirip oluşturdukları bu yöntemi, tahmin edilemeyen tur uzunluğu değişikliklerinden kaçınarak simetrik ve asimetrik gezgin satıcı probleminin çözümü için önermişlerdir. Bu algoritmayı gezgin satıcı probleminin çözümü için diğer yaklaşımlardan ayıran üç temel özellik vardır. İlk olarak, 3-opt prosedürü boyunca aday düğümler sadece aday düğüm listesi içinde aranır. İkinci olarak, prosedür her biri tur düğümleriyle birleşmiş bitlerden oluşan *don't look bit* isimli bir veri yapısı kullanır. Üçüncü ve son olarak ise, sadece gezgin satıcı probleminde geçerli olmak şartıyla 3-opt *r* düğümünden başlangıç hareketlerini araştırırken, prosedür *r* ilk düğümü ile olası 2-opt hareketlerini de göz önüne alır (Dorigo ve Gambardella [88]).

3.2.3 Karınca Kolonisi Optimizasyonu Algoritmalarının Uygulandığı Problemler

Literatür çalışmalarına bakıldığında karınca kolonisi optimizasyon algoritmalarının birçok farklı problem türünün çözümüne uygulandığı görülmektedir. Bu problem türleri ve son 5 yılda bu problem türleri için geliştirilmiş olan tekniklerden bazıları aşağıda verildiği gibi sıralanabilir:

Gezgin satıcı problemi: Garcia-Martinez ve diğerleri [92] de tek kriterli gezgin satıcı probleminin çözümü için çok amaçlı karınca kolonisi optimizasyon algoritmalarının deneysel analizini ve sınıflandırılmasını gerçekleştirmişlerdir. Cheng ve Mao [93] te

zaman pencereli gezgin satıcı probleminin çözümü için modifiye edilmiş karınca kolonisi sistemini önermişlerdir.

Araç rotalama problemi: Donati ve diğerleri [94] de zamana bağımlı araç rotalama probleminin çözümü için çoklu karınca kolonisi sistemini önermişlerdir. Mazzeo ve Loiseau [95] te kapasiteli araç rotalama problemi için karınca kolonisi algoritmasını kullanırken, Chen ve Ting [89] da zaman pencereli araç rotalama probleminin çözümü için yine aynı yöntemi önermişlerdir. Gajpal ve Abad [96] da araç rotalama probleminin çözümü için çoklu karınca sistemini kullanırken, Fuellerer ve diğerleri [97] de iki boyutlu yüklenen araç rotalama, Zhang ve diğerleri [98] de ve Gajpal ve Abad [99] da eş zamanlı dağıtım ve toplamalı araç rotalama problemlerinin çözümü için karınca kolonisi sistemini önermişlerdir.

Kuadratik atama problemi: Saremi ve diğerleri [100] de web sitesi yapısının geliştirilmesi için kuadratik atama problemi yaklaşımından ve karınca kolonisi meta-sezgisel yaklaşımından yararlanmışlardır. Demirel ve Toksarı [101] de ise kuadratik atama probleminin optimizasyonu için karınca kolonisi algoritmasını kullanmışlardır.

Çizelgeleme problemi: Silva ve diğerleri [102] de lojistik proseslerinin optimizasyonu ve tekrar çizelgelenmesi için genetik algoritma ve karınca kolonisi optimizasyonundan oluşan bir yapı önermişlerdir. Seçkiner ve Kurt [103] te iş rotasyonu çizelgeleme probleminin çözümü için karınca kolonisi optimizasyonunu kullanırken, Huang ve Liao [87] de iş çizelgeleme probleminin çözümü için karınca kolonisi optimizasyonunu tabu arama ile birleştirmişlerdir. Silva ve diğerleri [104] de çizelgeleme problemi için optimizasyon metotlarından karınca kolonisi ve genetik algoritmayı kullanırken, Heinonen ve Pettersson [105] te iş çizelgeleme problemine karınca kolonisi optimizasyonu ile görülebilirlik çalışmalarını birlikte uygulamışlardır. Rossi ve Dini [106] da ise rotalama esnekliği ve ayrılabilir ayar zamanları ile esnek iş çizelgeleme problemi için karınca kolonisi optimizasyonu metodunu önermişlerdir.

Atölye tipi çizelgeleme problemi: Liao ve Juan [107] de sıra bağımlı ayarlamalarda tek-makine gecikme çizelgeleme problemi için karınca kolonisi optimizasyonunu kullanırken, Cheng ve diğerleri [108] de tek-makine toplam gecikme problemi için karınca kolonisi temelli melez bir yapı önermişlerdir.

Proje çizelgeleme problemi: Tseng ve Chen [8] de kaynak kısıtlı proje çizelgeleme probleminin çözümü için karınca kolonisi optimizasyonu ile genetik algoritmanın bir arada kullanılmasıyla oluşmuş olan ANGEL isimli bir meta-sezgisel algoritma önermişlerdir.

Akış tipi çizelgeleme problemi: Gajpal ve Rajendran [109] da akış içindeki işlerin tamamlanma-zaman varyanslarının minimizasyonu için karınca kolonisi optimizasyon algoritmasını önerirken, Yağmahan ve Yenisey [110] da çok amaçlı akış tipi çizelgeleme problemi için karınca kolonisi optimizasyonunu kullanmışlardır. Lin ve diğerleri [111] de ise yine aynı problem türünün çözümü için karınca kolonisi optimizasyonunun yeni özelliklerini geliştirmişlerdir.

Çizge boyama problemi: Bui ve diğerleri [112] de çizge boyama probleminin çözümü için karınca temelli bir algoritma önerirken, Dowsland ve Thompson [113] de yine aynı problem türü için Costa ve Herts tarafından ortaya konan ANTCOL isimli algoritmayı geliştirmişlerdir.

Kaynak atama problemi: Chaharsooghi ve Kermani [114] de çok amaçlı kaynak atama probleminin çözümü için etkili bir karınca kolonisi optimizasyon algoritmasını önerirken, Yin ve Wang [115] de lineer olmayan kaynak atama probleminin çözümü için yine karınca kolonisi optimizasyonunu kullanmışlardır.

Ardışık sıralama problemi: Montemanni ve diğerleri [116] da ardışık sıralama probleminin çözümü için karınca kolonisi optimizasyonu ile sezgisel bir teknik önermişlerdir.

Tesis yerleştirme problemi: Baykasoğlu ve diğerleri [117] de bütçe kısıtlı ve kısıtlanmamış dinamik tesis yerleştirme problemleri için karınca kolonisi algoritmasını önerirken, Solimanpur ve diğerleri [118] de esnek imalat sistemleri içerisinde tek sıra tesis yerleştirme problemi için karınca kolonisi algoritmasını kullanmışlardır. Hani ve diğerleri [119] da endüstriyel yerleştirme probleminin çözümü için karınca kolonisi optimizasyonundan yararlanırken, McKendall ve Shang [86] da dinamik tesis yerleştirme problemi için tavlama benzetimi ile entegre edilmiş melez karınca sistemlerini kullanmışlardır.

Montaj hattı dengeleme problemi: Simaria ve Vilarinho [120] de iki taraflı montaj hattı dengeleme probleminin çözümü için karınca kolonisi optimizasyon algoritması olarak 2-ANTBAL'ı önerirken, Bautista ve Pereira [121] de zaman ve yer kısıtlı montaj hattı dengeleme problemi için karınca algoritmasını kullanmışlardır.

Sipariş büyüklüğü problemi: Pitakaso ve diğerleri [122] de kısıtlanmamış çok seviyeli sipariş büyüklüğü probleminin çözümü için maks-min karınca sistemini önerirken, Su ve Wong [123] de kırbaç etkisi altındaki stokastik dinamik üretim/tahmin sipariş büyüklüğü problemi için karınca kolonisi algoritmasından oluşan yenileme tasarımını ortaya koymuşlardır.

BÖLÜM 4

ARAÇ ROTALAMA PROBLEMLERİ

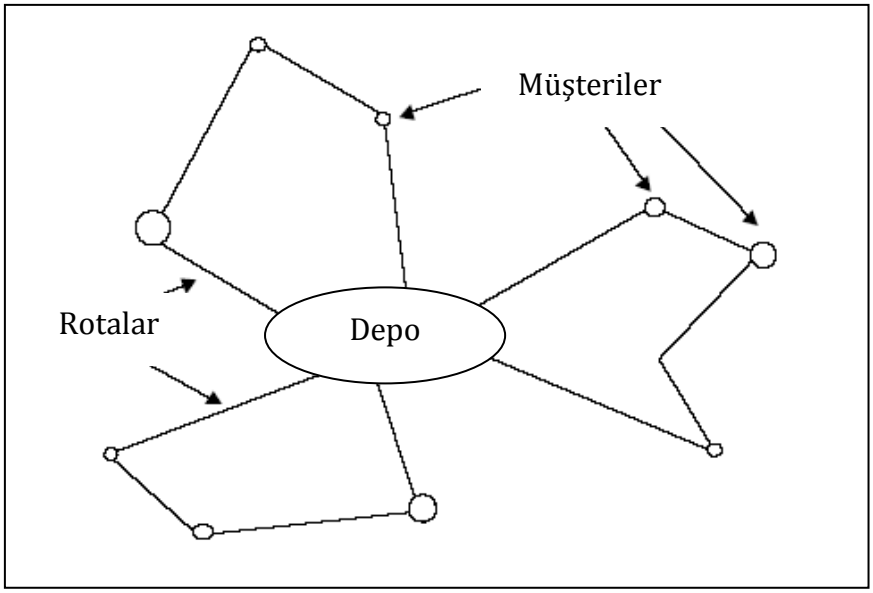
Araç rotalama problemi (ARP) en basit tanımıyla, bir merkezde bulunan araçların, talepleri bilinen müşteri kümesine hizmet edip tekrar merkeze dönmesini sağlayacak en kısa rotaların bulunması problemidir (Gencer ve Yaşa [124]). Araç rotalama problemleri lojistik ve tedarik zinciri konularının önemine bağlı olarak genişçe araştırılmış ve bu araştırmalarda birçok problem çeşidi ortaya konarak, bu problem çeşitleri için birçok çözüm yolları aranmıştır.

İlk kez 1959 yılında Dantzig ve Ramser tarafından tanımlanan araç rotalama problemi günümüze kadar üzerinde en çok durulan problem çeşitlerinden birisi olmuştur. Klasik araç rotalama probleminde, aynı tip ve kapasiteye sahip olan homojen bir araç filosu, merkezi bir depodan (dağıtım merkezinden) hareket ederek talepleri önceden bilinen bir grup müşteriye hizmet vermektedir. Bu problemde, her müşteri sadece bir araçtan hizmet almakta ve her araç sadece bir rota izlemektedir. Araçlar için kapasite kısıtının yanı sıra, araçların depodan hareket edip rotanın sonunda depoya geri dönmesi zorunluluğu vardır (Alabaş ve Dengiz [125]). Tipik bir araç rotalama probleminde, depodan başlayarak belirli sayıdaki müşterileri gezen ve ardından tekrardan depoya dönen belirli koşullar altında toplam yolculuk maliyetini minimize eden rotalar kümesinin bulunması amaçtır (Yu vd. [126]). Ayrıca toplam seyahat zamanının veya mesafesinin minimize edilmesi ile de amaçlandırılabilen klasik araç rotalama problemine çözüm aranırken araçların ilerlediği yol şebekesi, belirli bir miktarda talebi bulunan müşterilerin lokasyonları, işletme biriminin müşterilere göre konumu, yük kapasitesi veya gidebileceği maksimum yol gibi dikkat edilmesi gereken hususlara sahip

olan araç filosu ve mevcut çalışma periyodları ile araç sürücü kısıtları da dikkate alınmalıdır.

Sonuç olarak, klasik bir araç rotalama probleminde amaç, müşterilere hizmet sunan toplam araç sayısının minimize edilmesi veya araçlar tarafından kat edilen toplam seyahat mesafesinin minimize edilmesidir. Bu amaçlardan herhangi birinin sağlanabilmesi için, her bir müşteri yalnızca bir araçtan bir kez hizmet almalı, her bir araç bir depoda rotasına başlamalı ve rotasını aynı depoda sonlandırmalı, her bir rotadaki toplam talep rotaya atanan araçların kapasitesine eşit veya araçların kapasitesinden az olmalı ve önceden ayarlanmış olan seyahat ve hizmet zamanını da içeren toplam rota süreleri belirlenen değeri aşmamalıdır (Ombuki ve Hanshar [127]).

Şekil 4.1'de araç rotalama probleminin yapısı gösterilmiştir.



Şekil 4.1 Araç rotalama probleminin yapısı (Gambardella [128])

4.1 Araç Rotalama Probleminin Karakteristikleri

Araç rotalama probleminin karakteristik özelliklerini temel bileşenleri, optimizasyon kriterleri ve kısıtları başlıkları altında toplayabiliriz.

4.1.1 Araç Rotalama Probleminin Temel Bileşenleri

Araç rotalama probleminin temel bileşenleri aşağıda verildiği gibi sıralanabilir (Eryavuz ve Gencer [129]):

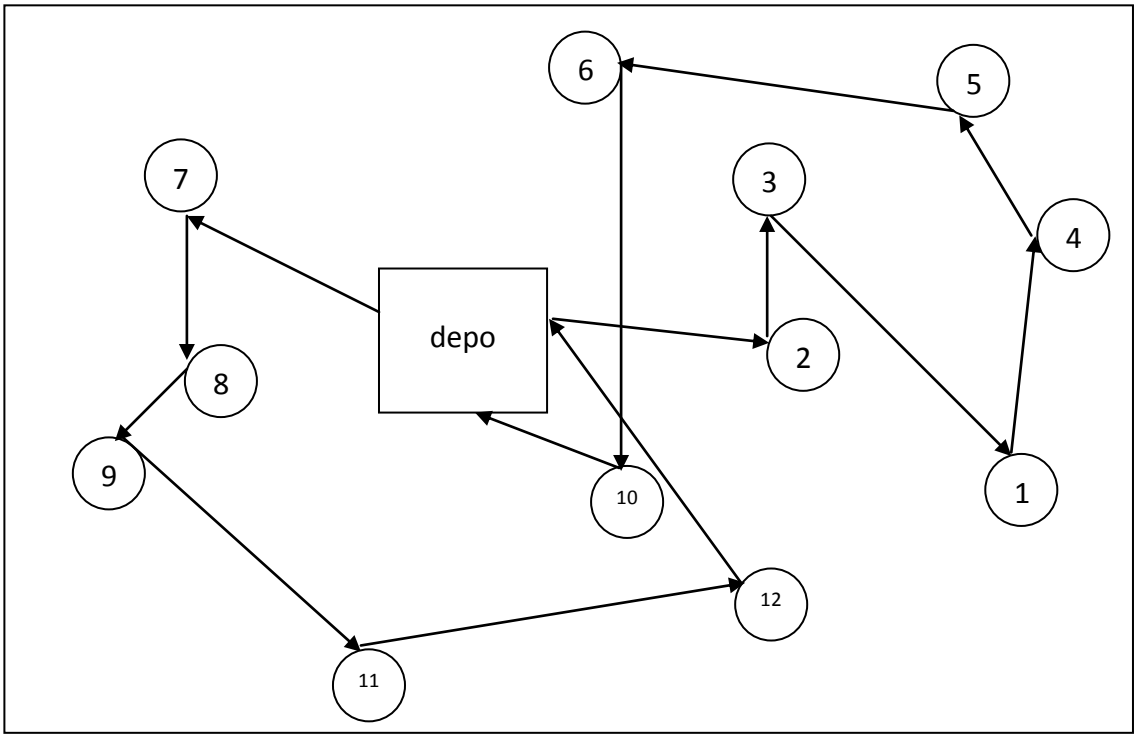
Talep: Araç rotalama problemlerinde talep statik veya dinamik olabilir. Statik talep durumunda talep önceden bilinir. Dinamik durumda ise bazı düğümlerdeki talep bilinmekte bazı düğümlerdeki talepler ise araç rotasına devam ederken belirlenmektedir. Araç rotalama probleminin temel bileşenlerinden biri olan talep yük ya da yolcuların hangi adreslerden alınarak hangi adreslere bırakılacağının belirlendiği talebin yeri, adreslere hangi zamanlarda uğranılacağının belirlendiği adres-zaman ilişkisi, miktarsal yapı, mekansal yapı ve zamansal yapı olmak üzere üç farklı bileşenle ifade edilebilen talebin yapısı ve müşterilerin talep ettikleri hizmet düzeylerinin belirlendiği müşteri tercihleri olmak üzere dört alt alana ayrılabilir [130].

Malzeme tipi: Araçlarla çok çeşitli malzemeler taşınır. Tehlikeli maddeler, gıda maddeleri, gazete dağıtımı, çöp toplama gibi bütün bu malzemeler basit paketler olarak adlandırılır ve probleme ilave bir karmaşıklık getirmezler. Diğer taraftan öğrenci servisleri, güvenlik, etkinlik, eşitlik gibi ilave bazı amaçlardan ötürü daha karmaşık bir yapıya sahiptir. Tehlikeli maddeleri taşıyan araç rotalarının belirlenmesinde ise coğrafi özellikler büyük önem taşır.

Dağıtım/toplama noktaları: Birçok araç rotalama probleminde, dağıtım noktaları müşterilerin bulunduğu yer, toplama noktaları ise depodur. Depo genellikle aracın rotasına başladığı ve rotası bittiğinde geri döndüğü noktadır. Dağıtım noktaları sabit ve önceden biliniyorsa hangi noktalara, hangi araçların hizmet vereceği belirlenmelidir. Diğer durumda dağıtım noktaları potansiyel yerler arasından seçileceği için ilave bir yerleştirme kararı gerekir. Bazı araç rotalama problemlerinde dağıtım ve toplama noktaları aynıdır.

Araç Filosu: Bütün araç rotalama problemlerinde araçların kapasitesinin bilindiği ve çoğunlukla araçların homojen (aynı kapasitede) olduğu varsayılır. Araç filosu heterojen ise filodaki araçların taşıma kapasiteleri farklıdır. Bu durum hangi araç tipinin, hangi rotaya hizmet vereceğinin belirlenmesi için ilave bir karar gerektirir. Araçların diğer özellikleri arasında hız, yakıt tüketimi, taşınacak malzemeye uygunluğu sayılabilir. Bu özelliklerin rotalama kararlarına doğrudan etkisi yoktur.

Şekil 4.2'de tek depolu ve 12 müşterili araç rotalama problemi örneği gösterilmiştir.



Şekil 4.2 Araç rotalama problemi örneği: Tek depolu ve 12 müşterili.
Her bir rota depodan başlıyor ve depoda bitiyor (Tan vd. [131])

4.1.2 Araç Rotalama Probleminin Optimizasyon Kriterleri

Araç rotalama problemini karakterize eden bir diğer unsur da optimizasyon kriterleridir. Araç rotalama problemi literatüründe çok çeşitli optimizasyon kriterleri mevcuttur. Bunlardan en yaygın olanları, rota sayısı, toplam rota uzunluğu (oluşturulan rotaların uzunlukları toplamı), rota süresi (seyahat, yükleme-boşaltma ve dinlenme süreleri toplamı), müşteri memnuniyeti ve yük dengeleme (araçlar arasındaki yük farkının minimize edilmesi)'dir. Bu kriterlerden rota sayısı ve toplam rota uzunluğu araç rotalama probleminin amaç fonksiyonunda en yaygın olarak kullanılanlarıdır.

4.1.3 Araç Rotalama Probleminin Rotalama Kısıtları

Araç rotalama probleminde kapasite kısıtı dışında uygulandığı olayın temeline göre literatürde var olan çeşitli yan kısıtlar mevcuttur. Bu kısıtlar, bir rotadaki dağıtım ve toplama noktaları üst sınırı, rota uzunluğu üst sınırı, rota süresi üst sınırı, toplam zaman kısıtı, zaman aralığı kısıtı, düğümler arası öncelik kısıtı, zaman harcayan (yükleme ve boşaltma) faktörler ve birden fazla depo olarak sıralanabilir. Araç rotalama probleminin tipine ve özelliğine göre bu kısıt miktarı değiştirilebilmektedir.

Gerçek hayatta lojistiğin doğası gereği, kalite düzeyini artırmak ve müşteriye daha iyi hizmet sunabilmek amacıyla araç rotalama problemlerinde aracın gidebileceği yol kapasitesi, sürücünün çalışma saat aralığı, araç yükleme/boşaltma zamanları, müşteriye hizmette uyulması gereken zaman aralıkları ve taşımayı gerçekleştirecek olan araç sayısı yetersizliği gibi çeşitli operasyonel kısıtlar da bulunabilmektedir.

4.1.4 Araç Rotalama Probleminin Prensipleri

Bugüne kadar yapılan konuyla ilgili çalışmalara göre daha başarılı ve uygulanabilir rotaların oluşturulması için birbirine yakın noktalar seçilmeli, taşıma yüksek kapasiteli araçlar ile yapılmalı, farklı günlerdeki dağıtımlar aynı güne birleştirilmeli, rotalar mümkün olan en uzak noktadan başlamalı ve tekrar benzer bir konuma gelerek elips şekli çizebilmeli ve olabiliyorsa dağıtım ve toplama aynı araçla yapılmalıdır.

4.1.5 Araç Rotalama Probleminin Karşılaşıldığı Süreçler

Araç rotalama problemlerine günümüzün organizasyonlarında özellikle ürün dağıtımında, mal ve insan taşımasında sıkça rastlanmaktadır. Bu tip problemler ürünlerin bir veya daha fazla sayıdaki depodan alınarak müşterilere dağıtılması, bar ve lokantalarda içecek dağıtılması, kargo dağıtılması, depolardan bayilere mal sevkiyatı, sütlerin kaynaklarından toplanarak dağıtılması, çöplerin toplanması ve taşınması, okul servis araçlarının güzergahlarının belirlenmesi, istasyonlara benzin ve mazotun dağıtılması ve stok planlaması ve ürünlerin satış yerlerine sevkiyatı gibi süreçlerde görülebilmektedir.

4.1.6 Araç Rotalama Probleminin Notasyonları ve Matematiksel Modeli

Klasik bir araç rotalama problemi modelinde $G = (V, E)$ bir grafı, $V = \{v_0 , v_1, ...,v_n\}$ bir nokta kümesini ve $E = \left\{ \left(v_i, v_j \right) : v_i, v_j \in < V, i \neq j \right\}$ bir kenar kümesini gösterdiği varsayılırsa V kümesinde, v_0 merkez depoyu, n ise müşterileri ifade eder. Klasik araç rotalama probleminde her müşteri q_i talebine sahiptir ve depoda her biri C kapasiteli m araçtan oluşan bir araç filosu vardır. Bu problemlerde temel amaç varolan kısıtlara uyulması ile maliyet minimizasyonunu sağlayarak tüm müşterilere hizmet götürmek

için her aracın gideceği yolu çizmek yani *m* araç sayısı kadar rota belirleyebilmektir. Tek depolu klasik bir araç rotalama probleminin doğrusal modeli aşağıda verildiği gibi formüle edilebilir:

M: Araç sayısı

N: Müşteri sayısı

d_{ij}: *i* noktası ile *j* noktası arasındaki mesafe

q_i: *i* müşterisinin talep miktarı

X_{ijk} = $\left\{ \begin{array}{ll} 1, & \text{eğer } k \text{ aracı } i \text{ noktasından } j \text{ noktasına hareket ederse} \\ 0, & \text{aksi taktirde} \end{array} \right\}$

Amaç fonksiyonu: $\min Z = \sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \sum_{k=1}^{M} d_{ij} X_{ijk}$ (4.1)

Şu kısıtlara göre:

i = 0 için $\sum_{k=1}^{M} \sum_{j=1}^{N} X_{ijk} = M$ (4.2)

$i \in \{1,...N\}$ için $\sum_{k=1}^{M} \sum_{j=0, j\neq i}^{N} X_{ijk} = 1$ (4.3)

$j \in \{1,..., N\}$ için $\sum_{k=1}^{M} \sum_{i=0, i\neq j}^{N} X_{ijk} = 1$ (4.4)

$k \in \{1,...M\}$ için $\sum_{i=1}^{N} X_{i0k} \leq 1$ (4.5)

$k \in \{1,..., M\}$ için $\sum_{i=1}^{N} q_i \sum_{j=0, j\neq i}^{N} X_{ijk} \leq C$ (4.6)

Burada (4.1) numaralı kısıt denkleminde amaç fonksiyonu yer almakta ve bu denklem toplam kat edilecek mesafenin yani maliyetin minimize edilmesi gerektiğini ifade etmektedir. Öte yandan (4.2) numaralı kısıt denklemi işletme biriminden çıkacak araç sayısının *M* adet olduğunu belirtirken, (4.3) numaralı kısıt denklemi bir müşterinin sadece bir araç tarafından ziyaret edilmesi gerektiğini ve (4.4) numaralı kısıt denklemi ise müşteriye gelen ve müşteriden çıkan yollardan sadece bir tanesinin kullanılmasının

zorunlu olduğunu ifade etmektedir. (4.5) numaralı kısıt denklemine göre bir araç yalnızca bir defa işletme biriminden çıkacağından aynı araç sadece bir defa rotalamada kullanılmaktayken, (4.6) numaralı kısıt denklemine göre ise araçlara yapılan yüklemeler araç kapasite değeri olan *C*'yi geçemez.

4.2 Araç Rotalama Probleminin Türleri

Araç rotalama problemleri çeşitli kısıtlara ve özelliklere göre birçok sınıfa ayrılmaktadırlar. Değişen her kısıt veya özellik için araç rotalama problem türü de değişmektedir.

4.2.1 Dinamik ve Statik Çevre Durumuna Göre Araç Rotalama Problemleri

Araç rotalama problemi içindeki olayların dinamik veya statik olması başka bir ifadeyle bilgilerin deterministik veya stokastik olarak ele alınması durumda araç rotalama problemleri, statik araç rotalama problemi ve dinamik araç rotalama problemi olarak ikiye ayrılır.

***Statik Araç Rotalama Problemi*:** Statik araç rotalama problemleri (SARP), talep, kısıt, kapasite gibi problem çözümü ile ilişkili olan tüm bilgilerin problem çözücüsü tarafından önceden bilindiği araç rotalama problem çeşididir. Burada baştan beri bilinen problem bilgileri değişmez yani sabittir. Literatürde deterministik araç rotalama problemi olarak da bilinen bu problem türü ile ilgili günümüze kadar yapılmış olan birçok çalışma ve araştırma mevcut olup günümüzde halen talep ve kapasite gibi kısıtların belirli olduğu ve değişmediği dağıtım problemleri konusunda yapılan çalışmalar sıklıkla karşımıza çıkmaktadır.

***Dinamik Araç Rotalama Problemi*:** Dinamik araç rotalama problemleri (DARP), araçlar rotalarında dağıtım işlemi gerçekleştirilirken yeni talep noktalarının aniden ortaya çıkması, rota yolları üzerinde meydana gelebilecek bozukluk veya ani müşteri talep değişimleri gibi önceden planlanamayan durumlar karşısında hızlı bir şekilde yeni kararların verilmesini gerektirecek olan bir problem çeşididir. Bu problem türünde oluşabilecek değişikliklere bağlı olarak gerçekleştirilmeyen dağıtımlar ertesi güne ertelenecektir (Montemanni vd. [132]).

4.2.2 Rotaların Durumlarına Göre Araç Rotalama Problemleri

Araç rotalama problemleri, bir aracın güzergahıyla ilgili bilgiler ışığında yani aracın rotasının bir depodan başlayıp aynı depoda sona ermesi veya aracın depodan bağımsız olup rotanın müşteride bitirilebilmesi durumlarına göre açık ve kapalı uçlu olmak üzere ikiye ayrılır.

***Kapalı Uçlu Araç Rotalama Problemi*:** Kapalı uçlu araç rotalama problemlerinde (KUARP), klasik araç rotalamanın tersine her rota depo yerine bir işletme biriminde başlar ve aynı işletme biriminde biter. Tek işletmeye sahip problemlerde bu yapının kurulabilmesi için klasik araç rotalama problemi formülasyonuna (4.7) kısıt denklemi eklenmelidir (Brandao [133]).

$$k \in \{1,...,M\} \text{ için } \sum_{j=1}^{N} X_{0jk} = \sum_{i=1}^{N} X_{i0k} \leq 1 \tag{4.7}$$

***Açık Uçlu Araç Rotalama Problemi*:** Açık uçlu araç rotalama problemleri (AUARP), bilinen talep ve müşteri noktaları ile yapılacak olan en iyi rotayı içerir. Bu problem türünde araçlar rotalarına bir merkezi depodan başlamakta ve başladıkları rotalarını talep noktası ile sonlandırmaktadırlar. Rotaların, araçların başladığı merkezi depo yerine müşterilerde sonlanmasını sağlamak için klasik araç rotalama problemi formülasyonuna (4.8) kısıt denklemi eklenmelidir.

$$k \in \{1,...,M\} \text{ için } \sum_{j=1}^{N} X_{0jk} + \sum_{i=1}^{N} X_{i0k} = 1 \tag{4.8}$$

Bu kısıt denklemine göre, bir aracın *0* numaralı işletme birimi ile başlayan veya biten ilgili *X* değişkenlerinden ancak biri *1* değerini alabilmektedir (Brandao [133]).

4.2.3 Kısıtlarına Göre Araç Rotalama Problemleri

Gerçek hayatta karşımıza çıkan lojistik sistemlerin modellenmesinin ve çözülmesinin kolaylaştırılması için bazı önemli kısıtlar göz önüne alınırken, bazı unsurlarda göz ardı edilir ve böylece optimum sonuç bulunmaya çalışılır. Bu kısıtların hepsinin aynı anda göz önüne alınması zor olduğundan araç rotalama problemlerine eklenen bazı kısıtlar aşağıda verildiği gibi sıralanabilir.

4.2.3.1 Kapasite Kısıtlı Araç Rotalama Problemleri

Kapasite kısıtlı araç rotalama problemleri (KKARP), bir veya birden fazla sayıda işletme birimi (depo) bulunan bir işletmenin taleplerinin belli sayıdaki müşterisine ulaşabilmesi için kullanılan araçların belirli birer yükleme kapasitesine sahip oldukları araç rotalama problemi çeşididir. Literatürde yer alan tüm klasik araç rotalama problemlerinin birer kapasite kısıtlı araç rotalama problemi olduğu kabul edilir ve bu kabul araç rotalama problemlerinin matematiksel yapısı içerisinde (4.6) kısıt denklemi ile verilir.

Kapasiteli araç rotalama problemlerinde, araç sayısı ya belirlenir veya bir karar değişkeni olarak bırakılır (Lacomme vd. [134]) ve kapasite kısıtlı araç rotalama problemleri her aracın sahip olduğu sabit maliyeti ve problemlerde kullanılan farklı tip ve yükleme kapasitesine sahip araçlar nedeniyle birbirinden ayrılan tiplere sahiptir.

Kapasite kısıtlı araç rotalama problemi minimum toplam maliyete sahip, yola v_0 deposundan başlayıp yol sonunda aynı depoya geri dönen m tane rotayı içerir. Her bir rotadaki toplam talep q_t'yu aşamaz ve her bir müşteri sadece bir araç tarafından bir kez ziyaret edilebilir (Alba ve Dorronsoro [42]). Kapasite kısıtlı araç rotalama problemi, araçların rotalarına başlayıp bitirdiği bir depodan, müşteri ve bu müşterilerin taleplerinden, maksimum kapasiteye sahip araçlardan ve müşteriler arasında ve müşteriler ile depo arasında meydana gelen maliyetlerden ve mesafelerden oluşur (Mazzeo ve Loiseau [95]).

Kapasite kısıtlı araç rotalama problemleri eşit taleplere sahip ve eşit olmayan taleplere sahip araç rotalama problemleri olmak üzere ikiye ayrılır.

***Eşit Taleplere Sahip Kapasiteli Araç Rotalama Problemi*:** "Ayrılabilir Talepli Kapasite Kısıtlı Araç Rotalama Problemi" ya da "Eşit Talepli Kapasite Kısıtlı Araç Rotalama Problemi" adını alan bu problem türünde müşterinin talebi birden fazla araca bölünerek, aynı lokasyondaki farklı müşterilere birer birim ürün ulaştırılması sağlanır. Böylelikle yeni problemde kapasite sınırı, bir aracın ziyaret edebileceği maksimum müşteri sayısı olarak ifade edilir.

***Eşit Olmayan Taleplere Sahip Kapasiteli Araç Rotalama Problemi*:** Eşit olmayan taleplere sahip kapasite kısıtlı araç rotalama problemlerinde ise müşteri talepleri

bölünemez ve birden fazla araca dağıtılamaz. "Ayrılamaz Talepli Kapasite Kısıtlı Araç Rotalama Problemi" olarak da adlandırılan bu problem türünde her müşteriye yalnızca bir araç hizmet götürür. Özellikle müşteri taleplerinin bölünmesinin mümkün olmadığı ve hem finansal hem de yönetimsel açıdan bu bölünmenin uygun olmadığı durumlarda eşit olmayan taleplere sahip kapasiteli araç rotalama problemi kullanılır.

4.2.3.2 Mesafe Kısıtlı Araç Rotalama Problemi

Mesafe kısıtlı araç rotalama problemleri (MKARP), rotalara atanan her aracın gidebileceği maksimum mesafe kısıtını içermekte ve klasik araç rotalama problemi modeline eklenecek olan (4.9) numaralı kısıt denklemi ile problem yapısı mesafe kısıtlı araç rotalama problemine dönüştürülmektedir.

$$k \in \{1,...,M\} \text{ için } \sum_{i=0}^{N}\sum_{j=0}^{N} d_{ij} \sum_{j=0, j\neq i}^{N} X_{ijk} \leq L \tag{4.9}$$

Burada, *L* ile bir aracın gidebileceği maksimum mesafe değeri gösterilmektedir.

Literatürde yer alan mesafe kısıtlı araç rotalama problemlerinin farklı türlerinden birinde farklı tipteki araçlar için farklı mesafe kısıtı (L_k , $k = 1,...,M$) söz konusu olabilirken bir başka mesafe kısıtlı araç rotalama problem türünde ise mesafe kısıtı yerine mesafeyle orantılı seyir süresi sınırlaması kullanılabilir. Seyir süresinin dahil edildiği problem yapısı içerisinde araç her müşteriye uğradığında s_i hizmet süresi kadar bekleyeceğinden, bu tip problemler klasik araç rotalama problemi modeline eklenecek olan (4.10) numaralı kısıt denklemi ile aşağıda verildiği gibi ifade edilebilecektir.

$$k \in \{1,...,M\} \text{ için } \sum_{i=0}^{N}\sum_{j=0}^{N} d_{ij} \sum_{j=0, j\neq i}^{N} pX_{ijk} + \sum_{i=0}^{N}\sum_{j=0}^{N} \frac{s_i + s_j}{2} \sum_{j=0, j\neq i}^{N} X_{ijk} \leq T \tag{4.10}$$

Burada p_s, aracın seyir süresini kat ettiği toplam mesafe yoluyla hesaplamakta kullanılan parametreyi ifade etmektedir. (4.10) numaralı kısıt denklemine göre ikiye bölünen hizmet süreleri (s_i) ile hem aracın rotasında bulunan yollar sıralanırken, hem de rota içerisindeki bir müşteri bir yolun başlangıcında diğer yolun ise sonunda yer alabilir, böylelikle aynı müşteri ile iki kez karşılaşılır (Toth ve Vigo [135]).

4.2.3.3 Zaman Pencereli Araç Rotalama Problemi

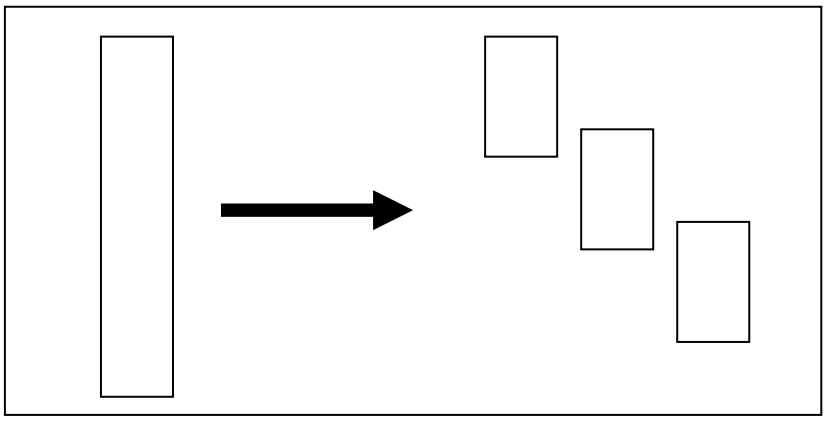
Zaman pencereli araç rotalama problemleri (ZPARP), yöneylem araştırmasının merkezi problemlerinden ve çok sayıdaki kombinasyonel optimizasyon pratik uygulamalarından biridir (Bouthillier ve Crainic [136]).

Zaman pencereli araç rotalama probleminde fizibil çözümün elde edilebilmesi için aşağıda verilen kısıtlar sağlanmalıdır (Dondo ve Cerda [137], Demirel vd. [138], Russell ve Chiang [139]):

- Her bir rota bir depoda başlar ve başladığı aynı depoda son bulur.
- Her bir müşteri sadece bir araçtan hizmet alır.
- Araç kapasitesinden fazla yüklenemez.
- Her bir *i* müşterisi için gerekli bir hizmet zamanı vardır.
- Her bir *i* müşteri [a_i, b_i] zaman penceresi ile hizmet görür.

Zaman pencereli araç rotalama probleminde, her müşteriye "Zaman Penceresi" olarak adlandırılan ve [a_i,b_i] şeklinde ifade edilen belirli bir zaman aralığında ulaşılması gerekmektedir. Zaman pencereli araç rotalama problemlerinde, her araç işletme biriminden *0* zamanında ayrılır ve bir *i* müşterisine uğradığında s_i hizmet süresi kadar müşteriye hizmet eder. Zaman pencereli araç rotalama probleminde aracın talep noktasına gelme zamanı en erken ve en geç olarak tanımlanan belirli bir zaman aralığı içinde açıklanır. Eğer araç müşteriye en erken hizmete başlama zamanı olan a_i'den daha erken ulaşmış ise, araç a_i başlangıç zamanına kadar bekler ve araç hizmet vermek için muhakkak b_i zamanına kadar *i* noktasına varmalıdır. Zaman pencereli araç rotalama probleminde amaç, zaman pencerelerine uyularak minimum maliyetle yapılabilecek bir rotalamayı gerçekleştirmektir.

Şekil 4.3'de zaman pencereli araç rotalama problemi için zaman penceresinin geometrik olarak bölünmesi gösterilmektedir.



Şekil 4.3 Zaman penceresinin geometrik şekil bölünmesi ile gösterimi
(Wang ve Regan [140])

Zaman pencereli araç rotalama probleminde, zamanın uygulanabilirlik kısıtı her bir müşteri tarafından sağlanmak zorunda olduğundan müşterilerin hoş görülebilen dağıtım zamanları araç rotalama problemine bir karmaşıklık eklemektedir. Hizmetlerin başlanacağı zamanı açıkça belirtmede klasik araç rotalama probleminin matematiksel modeline eklenen karar değişkenleri ve kısıtlar takımı aşağıda verildiği gibidir (Kallehauge vd. [141]):

s_{ik}: k aracının i müşterisine hizmete başladığı zaman $\forall i \in N,\ \forall k \in V$

Farz edelim ki, $a_0 = 0$, $s_{0k} = 0$ $\forall k \in V$

$$s_{ik} + t_{ij} - K(1-x_{ijk}) \leq s_{jk} \qquad \forall i, j \in N,\ \forall k \in V \qquad (4.11)$$

$$a_i \leq s_{ik} \leq b_i \qquad \forall i \in N,\ \forall k \in V \qquad (4.12)$$

(4.11) numaralı kısıt denklemi, i'den j'ye giden k aracının $s_{ik}+t_{ij}$'den önce j'ye varamayabileceğini ifade eder. Bu kısıttaki K oldukça büyük bir sayıdır. (4.12) numaralı kısıt denklemi ise zaman penceresinin gözlemlerini kesinleştirir.

Bazı durumlarda, araçlara müşteri alanına tam vardıkları zamanda hizmete başlamaları yönünde izin verilir. Böylece bu tip problemlerde, müşteri alanlarında araçlar için bekleme zamanı olmaz.

Gerçek hayata bakıldığında zaman pencereli araç rotalama problemleri ile günlük hayatta, daha çok mal alınacağı veya malların dağıtılacağı adreslere belirli bir zaman aralığında hizmetin verilmesi gerekli olduğu durumlarda karşılaşılmaktadır.

4.2.3.4 Müşteri Tipi Farklı Araç Rotalama Problemi

Müşteri tipi farklı araç rotalama problemlerinde (MTFARP), ürün teslim edilen ve kendisinden ürün teslim alınan olmak üzere iki farklı grup müşteri bulunmaktadır. Bu problem tipinde *L* grubu müşterilere ürün teslim edilirken, *B* grubu müşterilerden de ürün alımı yapılmakta olup bir araç her iki müşteriye de hizmet sunabilmektedir. Hem dağıtım hem de toplama işlemini yapan araç önce mal teslimini ardından da mal alımını gerçekleştirdiğinde ürün teslimatı yapacak müşterilerin talebi pozitifle, ürün alımı yapılacak olan müşterilerin talepleri de negatif olarak belirtilerek müşteri tipi farklı araç rotalama problemlerine çözüm aranır (Toth ve Vigo [135]).

4.2.3.5 Toplama ve Dağıtım İşlemlerini Kapsayan Araç Rotalama Problemleri

Toplama ve dağıtım işlemlerini kapsayan araç rotalama problemlerinde (TDARP), dağıtım ve toplama işlemleri birlikte yürütülür. Bu araç rotalama problem türünde, tüm dağıtım talepleri depodan başlar ve bütün toplama talepleri depoya getirilir. Müşteriler arasında mal alışverişi yoktur.

Toplama ve dağıtım işlemlerini kapsayan araç rotalama probleminde amaç, toplam seyahat zamanını ve araç akışını minimize etmektir. Bu noktada araçların malları müşterilere dağıtmak ve müşterilerden malları toplamak için yeterli kapasiteye sahip olmaları gerektiği unutulmamalıdır. Eğer her rotaya atanmış olan toplam miktar araç kapasitesini aşmıyorsa ve araçlar müşterilerden malları toplayabilecek yeterli kapasiteye sahipse çözüm fizibildir. Burada rotanın maliyeti klasik araç rotalama problemine benzerdir. Klasik araç rotalama problemindeki maliyet hesabına ek olarak, çözümün fizibil olmasının dağıtım fizibilitesine, toplama fizibilitesine ve yükleme fizibilitesine sahip olmasına bağlı olduğu sınırlaması mevcuttur. Her şeyden önce, *q* değişkeni müşterinin toplama talebinin vektörü olarak tanımlanır [142].

Dağıtım fizibilitesi: Dağıtım fizibilitesine göre, bir rotada dağıtılan malların toplam miktarı araç kapasitesini aşamaz. $R_i = \{v_0, v_1, \ldots v_{m+1}\}$ rotası verildiğinde ve araç bu rotaya *C* kapasitesi ile atandığında bu kısıt $C_d(v_k) \le C$ ve $C_d(v_{k+1}) > C$ şeklinde matematiksel olarak ifade edilebilir. Burada, $C_d(v_k)$, v_0 (depo)'dan başlayan ve v_k'da biten rota yolundaki tüm müşterilere dağıtılacak olan malların toplam miktarıdır. Ayrıca

$P(1,v_k)$, depodan v_k'ya kadar olan yol boyunca ziyaret edilmiş olan müşterileri gösterirse,

$$C_d\left(v_k\right)=\sum_{v_i\in P\left(1,v_k\right)} d_i \tag{4.13}$$

dir.

Toplama fizibilitesi: Toplama fizibilitesine göre, bir rotadaki tüm müşterilerden toplanacak olan malların kapasitesi için yeterli araç kapasitesi mevcuttur. Bu kısıt $C_p\left(v_k\right)\le C$ ve $C_p\left(v_{k+1}\right)>C$ şeklinde matematiksel olarak ifade edilebilir. Burada, $C_p(v_k)$, v_k'ya kadar ve v_k'nın da dahil olduğu rota yolu boyunca tüm müşterilerden toplanacak olan malların toplam miktarıdır.

$$C_p\left(v_k\right)=\sum_{v_i\in P\left(1,v_k\right)} p_i \tag{4.14}$$

Yükleme fizibilitesi: Araç kapasitesi rotanın herhangi bir düğümünde bozulabilir. Bu bozulma müşterilerin sıralanışına bağlıdır. $L(v_k)$, aracın v_k müşterisini terk ettikten sonraki yükü olsun ve aracın depoyu $L\left(1\right)\le C$ başlangıç yükü ile terk ettiği varsayılsın. Böylece rotadaki herhangi bir noktada aracın yükü,

$$L\left(v_k\right)=C_p\left(v_k\right)+L\left(1\right)-C_d\left(i_k\right) \tag{4.15}$$

olur. Bu denklem ile verilen araç yükünün araç kapasitesini aşabileceği görülmektedir. Yani, aracın yol üzerindeki bir sonraki v_{k+1} müşterisine hizmet edemeyecek olmasından dolayı kullanılan yol fizibil değildir. Rotanın fizibil olma şartı $L\left(v_k\right)\le C$ ve $L\left(v_{k+1}\right)>C$' dir.

Toplama ve dağıtım işlemlerini kapsayan araç rotalama probleminde, her müşteride yapılan dağıtım ve toplama işlemleri üç farklı şekilde gerçekleştirilebilir. Bu aynı zamanda toplama ve dağıtım problemlerinin de çeşitlerini ifade etmektedir.

Önce dağıtım sonra toplama problemleri: Müşteriler dağıtım ve toplama hattı olmak üzere ikiye ayrılırlar. Araçlar depolardan çıkarak önce dağıtım hattındaki müşterilere gidip dağıtım işlemini yapar, daha sonra da toplama hattına geçerek müşterilerden toplama işlemini yaparak depoya dönerler. Probleme zaman penceresi, tek ve/veya

çok depolu ve heterojen/homojen araç filosu kısıtları ilave edilebilir (Gencer ve Yaşa [124]).

Karışık dağıtım ve toplama problemleri: Müşteriler dağıtım ve toplama hattı olmak üzere ikiye ayrılırlar. Araçların önce dağıtım sonra toplama yapması gibi bir kısıt yoktur. Bunu karışık olarak da yapabilirler. Bu problem türünde dağıtımlar sadece bazı noktalara yapılırken, toplamalar da sadece bazı noktalardan yapılabilir (Ganesh ve Narendran [143]).

Eşzamanlı dağıtım ve toplama problemleri: Bu modelde, müşteriler eşzamanlı olarak mal alıp gönderebilirler (Nagy ve Salhi [144]). Burada eş zamanlı ifadesinden anlatılmak istenen müşteriye uğrandığında, dağıtılacağı bırakıp toplanacağı almaktır. Dolayısıyla müşteriler herhangi bir ayrıma tabi tutulmazlar. Araçlar her müşteriye bir defa gider ve toplama dağıtım işlemini yaparak müşteriden ayrılırlar. Eşzamanlı dağıtım ve toplama işlemleri özellikle tersine lojistik faaliyetlerinde araçların planlanması konusunda yol göstericidir (Gajpal ve Abad [99]).

4.2.3.6 Geri Dönüşlü Araç Rotalama Problemi

Geri dönüşlü araç rotalama problemleri (GDARP), müşterilerin bazı malları talep edebildiği bazılarını da geri gönderebildiği araç rotalama problemleridir. Bu nedenle toplama ve dağıtım işlemlerini kapsayan araç rotalama problemlerinde müşterilerden geri gönderilebilecek malların araca sığma zorunluluğu hesaba katılmalıdır. Buradaki kritik varsayım, her bir rotada toplama işleminden önce dağıtımların yapılma zorunluluğudur. Bu durum, araçların arkadan yüklemeli olmasından kaynaklanır ve araç yüklerinin dağıtım noktalarında yeniden düzenlenmelerinin ekonomik ve fizibil olmadığı açıktır. Geri dönüşlü araç rotalama probleminde dağıtılacak ve toplanacak miktarlar sabittir ve önceden bilinir. Bu problem türü, her rota için toplama işlemleri yapılmadan önce tüm dağıtımların tamamlanması zorunluluğundan toplama ve dağıtım işlemlerini kapsayan araç rotalama problemine benzer.

Geri dönüşlü araç rotalama probleminde iki tür müşteri vardır. Bunlardan ilki nakliye yapılacak yani ürün dağıtılacak müşteriler ve diğeri ise geri dönüşçü yani ürün toplaması yapılacak olan müşterilerdir (Gajpal ve Abad [96]).

Geri dönüşlü araç rotalama probleminin pratik kullanım alanları oldukça geniştir. Genellikle süpermarketler ve dükkanlar dağıtım noktalarıyken, bakkallar ürünlerin depoya geri gelmesini öneren geri dönüşçülerdir (Fan vd. [145]).

Geri dönüşlü araç rotalama probleminde amaç, toplam kat edilen yolun minimize edildiği rotaların setini bulmaktır. Problemin fizibil çözümü, herhangi bir toplamadan önce dağıtımların tamamlandığı ve araç kapasitesinin rotaya atanan geri dönüş noktalarıyla bozulmadığı rotaların setini içerir. Geri dönüşlü araç rotalama probleminde rotaların maliyeti klasik araç rotalama problemiyle benzerdir. Buna ek olarak, problem çözümünün fizibil olabilmesi için, bir rotanın dağıtım fizibilitesi, toplama fizibilitesi ve yükleme fizibilitesine sahip olması gerektiği sınırlaması getirilir [142].

Dağıtım fizibilitesi: Dağıtım fizibilitesine göre, bir rotada dağıtılan malların toplam miktarı araç kapasitesini aşamaz. Rota $R_i = \{v_0, v_1, ..., v_{m+1}\}$ şeklinde verilir ve araç kapasitesi *C* olarak tanımlanırsa, bu kısıt matematiksel olarak $C_d(v_k) \le C$ ve $C_d(v_{k+1}) > C$ şeklinde ifade edilir. Burada, $C_d(v_k)$, v_0 (depo)'dan başlayan v_k müşterisinde biten rota yolundaki tüm müşterilere dağıtılacak olan malların toplam miktarıdır. Ayrıca, $P(1,v_k)$, depodan v_k'ya kadar olan yol boyunca ziyaret edilmiş olan müşterileri gösterirse geri dönüşlü araç rotalama problemi için (4.13) kısıt denklemi kullanılır.

Toplama fizibilitesi: Toplama fizibilitesine göre, bir rotadaki tüm müşterilerden toplanacak mallar için yeterli araç kapasitesi mevcuttur. Bu kısıt $C_p(v_k) \le C$ ve $C_p(v_{k+1}) > C$ şeklinde matematiksel olarak ifade edilebilir. Burada $C_p(v_k)$, v_k'ya kadar ve v_k'nın da dahil olduğu rota yolu boyunca tüm müşterilerden toplanacak olan malların toplam miktarıdır ve (4.14) kısıt denklemi ile gösterilir.

Yükleme fizibilitesi: Araç kapasitesi rotanın herhangi bir düğümünde bozulabilir. Bu bozulma müşterilerin sıralanışına bağlıdır. $L(v_k)$, aracın v_k müşterisini terk ettikten sonraki yükü olsun ve aracın depoyu $L(1) \le C$ başlangıç yükü ile terk ettiği varsayılsın. Böylece rotadaki herhangi bir noktada aracın yükü (4.15) kısıt denklemi ile bulunur. Bu denklem ile verilen araç yükünün araç kapasitesini aşabileceği görülmektedir. Yani,

aracın yol üzerindeki bir sonraki v_{k+1} müşterisine hizmet edemeyecek olmasından dolayı kullanılan yol fizibil değildir. Rotanın fizibil olma şartı $L(v_k) \leq C$ ve $L(v_{k+1}) > C'$ dir.

4.2.3.7 Parçalı Dağıtımlı Araç Rotalama Problemi

Parçalı dağıtımlı araç rotalama problemleri (PDARP), eğer tüm maliyetlerin düşürülmesi mümkünse bir müşteriye farklı araçların hizmet vermelerine izin veren bir araç rotalama problem türüdür. Burada, müşteri siparişleri araç kapasitesi kadar büyükse klasik araç rotalama probleminin genişletilmesi önem kazanmaktadır. Parçalı dağıtımlı araç rotalama probleminde optimum sonucu bulmak klasik araç rotalama probleminden daha zordur. Bu problem türünde amaç, araç akışını ve tüm müşteri tedariklerinin karşılanabilmesi için gerekli olan toplam seyahat süresinin minimize edilmesidir. Bir müşterinin birden fazla araçtan hizmet görmesi dışındaki araç rotalama kısıtlarının tümü sağlanıyorsa çözüm fizibildir. Tüm rota maliyetlerinin toplamı parçalı dağıtımlı araç rotalama problemi formülasyonunda minimize edilir. Her bir müşteri siparişinin daha küçük olan ve bölünemeyen siparişlere ayrılarak dağıtımların parçalanmasına izin vermekle, araç rotalama problemi kolaylıkla parçalı dağıtımlı araç rotalama problemine dönüştürülebilir.

4.2.3.8 Periyodik Yüklemeli Araç Rotalama Problemi

Klasik araç rotalama probleminde planlama periyodu bir gündür. Periyodik yüklemeli araç rotalama probleminde (PYARP) ise planlama periyodu *M* güne kadar yükseltilerek klasik araç rotalama problemi genelleştirilir.

Periyodik yüklemeli araç rotalama probleminde, planlama periyodu, her bir *i* müşterisi için e_i hizmet sıklığı ve C_i seyahat günlerinin olası kombinasyonlarının setinden oluşur. Problem, her bir müşteri için eş zamanlı olarak seyahat kombinasyonlarının seçilmesini ve araç rotalama probleminin kuralları ışığında planlama periyodunun her bir günü için araç rotalarının kurulmasını içerir (Cordeau vd. [146]).

Periyodik yüklemeli araç rotalama probleminde amaç, araç akışını ve tüm müşterilerin tedarikleri için ihtiyaç duyulan toplam seyahat zamanını minimize etmektir. Araç

rotalama probleminin tüm kısıtları sağlanıyorsa çözüm fizibildir. Ayrıca, araç depoyu terk ettiği aynı gün depoya dönmeyebilir. Bu problem türüne göre, M_1 gün periyodu içinde tüm müşteriler en az bir kez ziyaret edilmiş olmalıdır.

Periyodik yüklemeli araç rotalama probleminde tüm rotaların maliyetlerinin toplamı minimize edilmeye çalışılır. Her müşteri, tek bir araçla sadece bir kez ziyaret edilerek karşılanabilecek önceden bilinen bir talebe sahiptir. Planlama periyodu $M_1=1$ ise periyodik yüklemeli araç rotalama problemi klasik araç rotalama problemi haline gelir. Periyodik yüklemeli araç rotalama problemindeki her bir müşteri k kere ziyaret edilmelidir. $1 \leq k \leq M_1$'dir. Periyodik yüklemeli araç rotalama probleminin klasik modelinde, müşterilerin günlük talepleri her zaman sabittir. Bu problem türü, global maliyetlerin minimizasyonu ve kısıtların karşılanabilmesi için her günkü rotaların üretildiği bir problem olarak görülebilir. Ayrıca bu problem türü, çok aşamalı kombinasyonel optimizasyon problemi olarak da görülebilir. İlk aşamada, amaç her bir müşteri için bir grup fizibil alternatif (kombinasyon) üretmektir. İkinci aşamada, her bir müşteri için günlük kısıtları sağlayan alternatiflerden biri seçilmelidir. Böylece, gün içinde ziyaret edilmiş müşterilerden seçim yapılmak zorundadır. Üçüncü ve son aşamada ise her bir gün için araç rotalama problemi çözülür.

Periyodik araç rotalama probleminin market endüstrisi, alkolsüz içecek endüstrisi (satış otomatı), otomotiv endüstrisi (parça dağıtımı), endüstriyel gaz dağıtımı ve atık toplama gibi birçok pratik uygulamaları bulunmaktadır (Mingozzi [147]).

4.2.3.9 Karma Yüklemeli Araç Rotalama Problemi

Karma yüklemeli araç rotalama problemleri (KYARP), aynı müşteriye hem ürün teslimatının ve hem de müşteriden ürün alımının yapıldığı araç rotalama problemi çeşididir. Bu problem türünde müşteriye ilk olarak q_i miktarında ürün teslimatı yapılırken, daha sonra da aynı müşteriden o_i miktarında ürün alımı yapılmaktadır. Ürün teslimatının ve alımının aksamaması için karma yüklemeli araç rotalama problemlerinde araç müşteriye ulaşmadan önce araçta yeterli miktarda ürün veya yer olup olmadığı kontrol edilmelidir.

4.2.4 Stokastik Araç Rotalama Problemleri

Stokastik araç rotalama problemleri (STARP), problemin bazı elemanlarının rassal olduğu araç rotalama problemi türüdür. Literatürde stokastik araç rotalama problemlerinde oluşabilecek üç farklı stokastik durum vardır [142]:

Stokastik müşteriler: Her bir v_i müşterisi, p_i olasılığı ile vardır ve $1\text{-}p_i$ olasılığı ile de yoktur.

Stokastik talepler: d_i talebi her bir müşteri için rassal bir değişkendir.

Stokastik zamanlar: s_i hizmet zamanları ve t_{ij} seyahat zamanları da rassal değişkenlerdir.

Stokastik araç rotalama probleminde, çözüme ulaşma iki aşamada gerçekleşir. İlk olarak rassal değişkenlerin gerçekleşme değerleri bilinmeden ilk çözüm belirlenir. İkinci aşamada ise, rassal değişken değerlerinin bilinmesinin ardından düzeltme ve doğrulama faaliyetleri gerçekleştirilir. Bu problem türünde amaç, araç akışını ve rassal değerleri bilinerek hizmet alan tüm müşterilere hizmet verebilmek için geçen toplam seyahat zamanının ve hizmet sürelerinin minimize edilmesidir. Bazı veriler rassal olduğunda, rassal değişkenlerin gerçekleşmesi için tüm kısıtların sağlanabilmesi mümkün olmayabilir. Böyle bir durumda, karar verici ya verilen belirli bir olasılıkla bazı kısıtların karşılanmasına ya da herhangi bir kısıt bozulduğunda modele doğrulayıcı faaliyetlerin eklenmesine ihtiyaç duyabilir [142].

$$\min \sum_{i<j} c_{ij} x_{ij} + Q(x) \tag{4.16}$$

Burada, x_{ij} çözümün ilk aşamasında görülen (v_i, v_j) zaman düzleminin sayısına eşit tam sayılı bir değişkendir. Eğer $i,j>1$ ise, x_{ij} sadece 0 veya 1 değerini alır. Eğer $i=1$ ise, araç depo ile v_j arasında bir geri dönüş yolculuğu yaparsa x_{ij} 2' ye eşit olur. $Q(x)$ ikinci aşamada beklenen başvuru fonksiyonudur. Probleme bağımlıdır ve olası başvuru faaliyetlerinin kısmi seçimi ile de ilişkilidir.

Örneğin, kapasite kısıtlı toplamalı stokastik araç rotalama problemi için olası başvuru faaliyetleri aşağıda verildiği gibidir:

- Araç dolu ise boşaltmak için depoya dönülür ve daha sonra toplamalara planlandığı gibi devam edilir.
- Araç dolduğunda bir önceki durumdaki gibi depoya dönülür ve planlanan rotanın geri kalan kısımı yeniden optimize edilir.
- Araç dolu değilse bile depoya önleyici bir geri dönüş planlanır. Bu durumda, bu karar henüz toplanmış miktara ve aracın depodan olan uzaklık mesafesine bağlıdır.

Henüz dolu olmayan bir aracın, gideceği bir sonraki müşteri ile kapasitesinin aşılabileceği biliniyorsa, araç depoya dolmadan da dönebilir.

Stokastik araç rotalama problemleri, müşteri ve taleplerin veya seyahat ve hizmet zamanlarının stokastik olmasına göre ikiye ayrılır.

***Stokastik Müşteri ve Talepli Araç Rotalama Problemi*:** Talebin deterministik olduğu bir durumda müşteriye hizmet vermenin p_i olasılığıyla gerçekleştiği stokastik müşteri ve talepli araç rotalama problemleri (SMTARP) dağıtım noktalarında talebin belirsiz yani rassal değişken olması durumunu kapsamaktadırlar.

Stokastik talepli araç rotalama problemi $G = (V,A_1,D)$ grafı üzerinde tanımlanır. Burada $V=\{0, 1, ..., n\}$ düğümler (müşteriler) kümesini gösterir. 0 düğümü depoyu temsil ederken, $A_1 = \{(i, j) : i \neq j, i, j \in V\}$ düğümleri birleştiren yayların kümesini ve $D = \{d_{ij} : i \neq j, i, j \in V\}$ düğümler arasındaki seyahat zamanlarını veya uzaklıkları gösterir. Maliyet matrisi D_1 simetriktir ve üçgensel eşitsizliği sağlar. Q kapasitesine sahip bir araç müşterilerin taleplerine bağlı olarak teslimatı gerçekleştirirken toplam beklenen seyahat mesafesi de minimize edilir. Bu problem türünde aşağıda verilen kabuller yapılır (Baykoç ve İşleyen [90]):

- Müşterilerin talepleri bilinen olasılık dağılımlarından gelen stokastik değişkenlerdir $(\xi_i, i = 1,...n)$.
- Her bir müşterinin gerçek talebi araç müşteriye ulaştıktan sonra elde edilmektedir.

- Müşteri talepleri ξ_i araç kapasitesi *Q*'yu geçemez ve talepler kesikli veya sürekli olasılık dağılımlarından gelebilirler.

Stokastik talepli araç rotalama problemi için uygun bir çözüm depodan başlayan ve depoda sonlanan tur için müşterilerin *s*=(*s*(*0*), *s*(*1*),....., *s*(*n*), *s*(*0*)) permütasyonudur. Araç müşterilere başlangıç (öncelik) turundaki sıraya göre hizmet verir ve müşterinin gerçek talebine göre bir sonraki düğüme gitmeye ya da depoya geri dönmeye karar verir. Bu problem türü modellenirken genelde iki yöntem kullanılır. Bu yöntemler şans kısıtlı stokastik programlama (chance constraint stochastic programming-CCP) ve yardımcı eylemli stokastik programlamadır (stochastic programming with recourse-RM) (Baykoç ve İşleyen [90]).

***Stokastik Seyahat ve Hizmet Zamanlı Araç Rotalama Problemi*:** Stokastik seyahat ve hizmet zamanlı araç rotalama problemleri (SSHZARP), önceden belirlenemeyen çevre şartlarının ve trafik durumunun etkisiyle oluşan belirsiz seyahat ve hizmet zamanlarını kapsar.

4.2.5 Yolların Durumuna Göre Araç Rotalama Problemleri

Araç rotalama problemleri iki nokta arasındaki yol mesafesinin geliş ve gidiş yönünün göz önüne alınarak bu yolların aynı kalıp kalmamasına göre simetrik yollu araç rotalama problemi ve asimetrik yollu araç rotalama problemi olarak ikiye ayrılırlar (Samanlıoğlu vd. [37]):

***Simetrik Yollu Araç Rotalama Problemi*:** Simetrik yollu araç rotalama problemi (SYARP), bir noktadan bir diğerine olan mümkün gidiş dönüş mesafesinin birbirine eşit olduğunun kabul edildiği (d_{ij}=d_{ji}) araç rotalama problemi çeşididir.

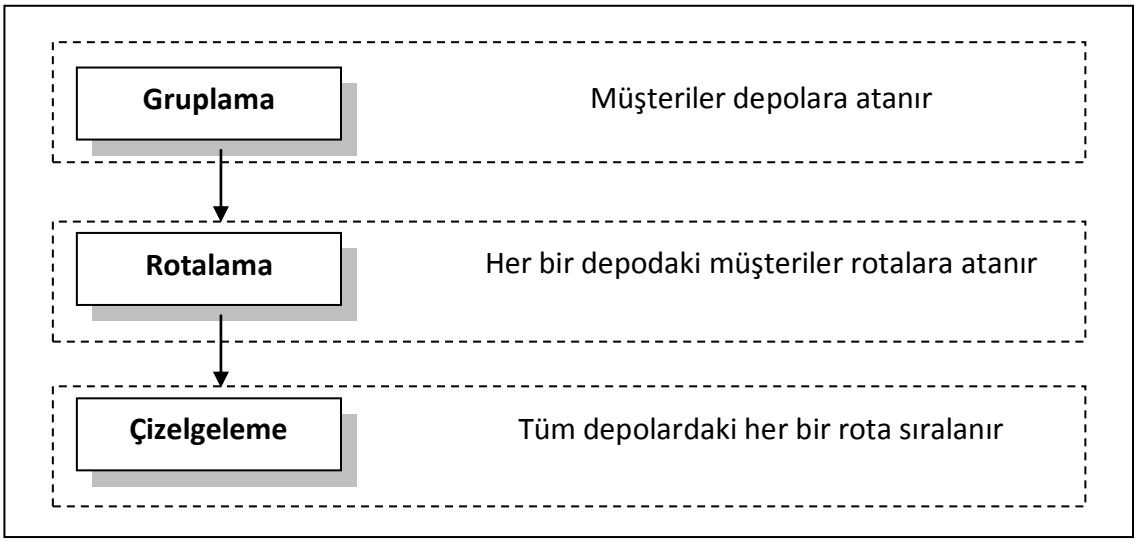
***Asimetrik Yollu Araç Rotalama Problemi*:** Asimetrik yollu araç rotalama problemlerinde (AYARP), iki müşterinin bulunduğu *y* ve *z* noktaları arasında *y* noktasından *z* noktasına gitmek için gerekli olan mesafe, *z* noktasından *y* noktasına gitmek için gerekli olan mesafeye eşit olmayabilir $\left(d_{yz} \neq d_{zy}\right)$. Rotanın dönüş yönünün saptanarak rota mesafesinin hesaplanmasının büyük önem taşığıdı bu tip kapalı uçlu

araç rotalama problemlerinde araçların ilk olarak hangi müşteriye gideceği belirlenmelidir.

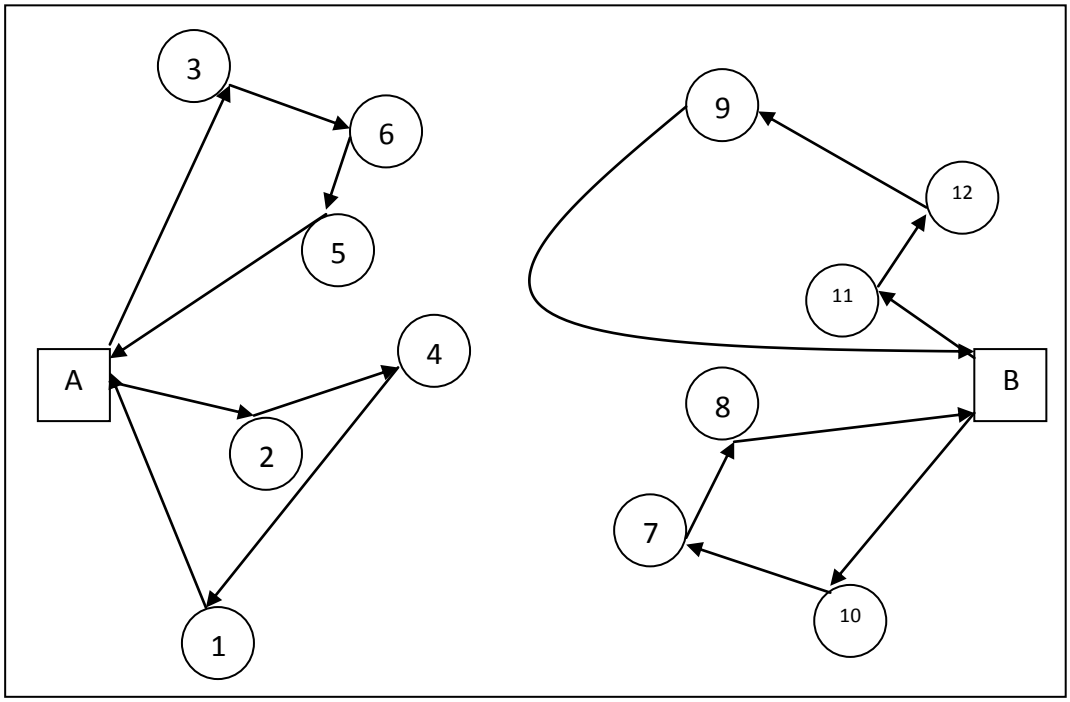
4.2.6 Depo Sayısına Göre Araç Rotalama Problemleri

Klasik araç rotalama problemleri için literatürde var olan sınıflandırma çeşitlerinden biri de depo sayısına göre yapılmaktadır. Depo sayısına göre araç rotalama problemleri çok depolu ve tek depolu olarak iki gruba ayrılabilirler.

***Çok Depo Çok Müşterili Araç Rotalama Problemleri*:** Çok depolu araç rotalama problemleri (ÇDARP), müşterilere hizmet veren birden fazla depoya sahip araç rotalama probleminin bir türüdür. Bu problem türünde eğer müşteriler depolar etrafında kümelendirilirse, dağıtım problemi araç rotalama problemlerine benzer olarak modellendirilebilir. Ancak, eğer müşteriler ve depolar birbiri içine karışmışsa çok depolu araç rotalama problemi çözülebilmelidir. Şekil 4.4'de çok depolu araç rotalama probleminin çözümü için karar hiyerarşisi ve Şekil 4.5'de de çok depolu araç rotalama problemi için çözülmüş bir örnek gösterilmiştir.



Şekil 4.4 Çok depolu araç rotalama probleminde karar hiyerarşisi (Ho vd. [148])



Şekil 4.5 Çok depolu araç rotalama problemi örneği (Ho vd. [148])

Çok depolu araç rotalama probleminde müşteriler depolara atanır. Her depoda bir araç filosu vardır. Her araç depodan yola çıkar, o depoya atanmış olan müşterilere hizmet verir ve aynı depoya geri döner. Çok depolu araç rotalama probleminin amacı, minimum araç sayısı ve seyahat mesafesi ile tüm müşterilere hizmet etmektir. Bu problem türünde araç filosu ve toplam seyahat zamanı minimize edilirken, ticari malların toplamı birçok depodan sunulur. Depodan başlayarak yine aynı depoda son bulan her bir rota için klasik araç rotalama probleminin kısıtları karşılandığında çözüm fizibildir.

Tek Depo Çok Müşterili Araç Rotalama Problemleri: Tek bir başlangıç noktasının (deponun) ve çok sayıda varış noktasının (müşteri) bulunduğu problem türü içinde rotalama tek araçla veya birden fazla araçla olmak üzere iki farklı şekilde yapılmaktadır.

- *Tek Araçlı Rotalama:* Şebeke içerisinde bulunan tüm müşterilerin ihtiyaçlarının sadece bir araç kullanılarak karşılandığı tek araçlı rotalamada, araç kendi kapasitesini dolduracak kadar yüklendikten sonra birinci rotasına başlar ve belirlenen rota üzerindeki tüm müşterilere uğradıktan sonra tekrar depoya döner. Şebeke içerisinde talebi henüz karşılanmayan diğer müşterilerin talepleri

ise yine aynı yöntemle boşaltılan aracın kapasitesine uygun bir şekilde tekrar yüklenmesi ile gerçekleştirilerek araç ikinci rotasına çıkar ve şebeke içindeki müşterilerin ihtiyaçlarının tamamı karşılanıncaya kadar araç rotalamaya devam edilir (Tan [131]).

- *Çok Araçlı Rotalama:* Şebekede bulunan müşteri ihtiyaçlarının çok sayıda araç kullanılarak karşılanmaya çalışıldığı çok araçlı rotalamada, müşterilerin talepleri araç kapasitelerine uygun olarak yüklendikten sonra yüklenen araçlar aynı anda belirlenen rota üzerindeki noktalara uğrayarak talepleri karşılarlar ve tekrar depoya geri dönerler. Şebeke içerisindeki rota sayısı kadar araca ihtiyaç duyulan çok araçlı rotalama yöntemi ile diğer yöntemlere göre talep karşılama hızı daha yüksek olduğundan zaman kısıtının önemli olduğu durumlarda bu tip bir rotalama sıklıkla kullanılmaktadır (Tan [131]).

4.2.7 Araç Filosu Özelliğine Göre Araç Rotalama Problemleri

Araç rotalama problemleri, bulundurdukları araç filosunun özelliğine göre homojen araç filosuna sahip araç rotalama problemleri ve heterojen araç filosuna sahip araç rotalama problemleri olarak iki gruba ayrılır.

***Homojen Araç Filosuna Sahip Araç Rotalama Problemi*:** Aksi belirtilmedikçe bütün araç rotalama problem türlerinde, rotalama yapan araçların kapasitesinin bilindiği ve çoğunlukla bu araçların aynı kapasitede (homojen) olduğu varsayılır. Bir filo içerisindeki araçların hepsinin türünün ve kapasitesinin eşit olması o araç rotalama probleminin homojen araç filosuna sahip araç rotalama problemi olduğunu gösterir.

***Heterojen Araç Filosuna Sahip Araç Rotalama Problemi*:** Bir araç rotalama probleminde, rotalama yapan araçların türleri ve/veya kapasiteleri birbirinden farklı yani heterojen ise bu araç rotalama problemi "Heterojen Araç Filosuna Sahip Araç Rotalama Problemi" olarak adlandırılır. Bir araç filosundaki heterojen araçların çeşitliliği, filo içerisinde rotalama yapan araçların maliyet, yükleme-boşaltma işlemleri için gerekli zaman ve kapasitelerine bağlıdır (Irnich [149]).

Heterojen araç filosuna sahip araç rotalama probleminde farklı araç türleri $\psi = \{1,...,k\}$ seti ile belirtilir. $k \in \psi$ aracı, Q_k taşıma kapasitesine sahiptir. k türlü

araçların sayısı n_k'ya kadar ulaşabilir. *i* müşterisinden *j* müşterisine olan seyahatin maliyeti *k* türündeki araç ile d_{ijk}'dır. f_k maliyeti *k* türündeki araca aitken, diğer araç türleri içinde farklı maliyetler ortaya çıkar (Ochi vd. [150]).

Heterojen araç filosuna sahip araç rotalama problemlerinde her bir farklı türdeki araçların sayıları filoların büyüklüklerinin sınırlandırılması gerektiği durumlarda limitlendirilebilir (Belfiore ve Yoshizaki [151]).

4.3 Araç Rotalama Problemleri için Çözüm Yöntemleri

Araç rotalama problemleri, üzerinde en çok durulan ve en çok çalışılan optimizasyon problemlerinden biridir. Dağıtım (veya toplama) maliyetlerinin toplam lojistik maliyetinden büyük bir pay alması nedeniyle araç rotalama, dağıtım yönetimi ve lojistik alanında önemli bir rol oynamaktadır.

Bir araç rotalama probleminin çözümünün elde edilmesi için optimize edilecek olan problem fonksiyonu, müşteriler arası ve müşteriler ile işletme birim/birimleri arası mesafeler, müşteri noktalarında bulunan talep miktarları ve rotalamada kullanılacak araç sayısı ile bu araçların kapasite bilgileri bilinmelidir. Araç rotalama problemleri için önerilen çözüm yöntemleri kesin yöntemler ve sezgisel yöntemler olmak üzere iki gruba ayrılır. Bu bölümde, araç rotalama probleminin çözümü için önerilen yöntemler incelenecektir.

4.3.1 Kesin Yöntemler

Araç rotalama problemlerinin çözümünde kullanılan kesin yöntem algoritmaları yol formülasyonu temellidir. Bu yöntemlerin araç rotalama problemi için çözümün araştırılmasında önemli ve başarılı etkileri vardır (Kallehauge [152]). Araç rotalama problemleri genel olarak sezgisel yöntemler ile çözümlenmeye çalışılsa da, bu problem türlerini çözmede oldukça etkili kesin yöntemler de vardır.

4.3.1.1 Dal ve Sınır Algoritması

Dal ve sınır algoritması, böl ve ele geçir stratejisini kullanarak çözüm uzayını alt problemlere böler ve her bir alt problemi ayrı ayrı kendi içinde optimize eder. Dal ve

sınır algoritması kullanılarak, S_c çözüm uzayı değerlendirilir. İlerleme ve sınırlama aşamalarında problem esnek hale getirilir. Bu sayede, fizibil *S* seti içinde bulunmayan çözümlerde kabul edilir. Bu gevşemenin çözülmesi optimum çözüme daha düşük bir alt sınır değeri getirir. Bu gevşemenin çözümü *S*'nin elemanıysa veya $\hat{s}$ ($\hat{s} \in S$) ile eşit maliyete sahipse problem bitmiştir, yani $\hat{s}$ çözümü yeni optimal çözümdür. Aksi halde, S_c çözüm uzayında $S_1,\ldots,S_n$ olmak üzere n_1 tane alt küme oluşturulur. Burada,

$$U_{i=1}^{n} S_i = S_c \tag{4.17}$$

dir. n_1 tane alt kümenin her biri bir alt problem olarak tanımlanır. $S_1,\ldots,S_n$ isimli bu alt problemler bazen de *S*'nin çocukları olarak adlandırılabilir. *S*'nin bu çocukları aday alt problemler listesine eklenir. Bu işlem dallandırma olarak adlandırılır. Bu noktadan sonra, algoritmaya devam etmek için aday alt problemlerden biri seçilir ve işlem ilerletilir. Burada dört olası sonuç söz konusudur [142]:

- Eğer $\hat{s}$'den daha iyi bir fizibil çözüm bulunursa, bu yeni çözüm $\hat{s}$ ile yer değiştirilir ve devam edilir.
- Alt problem için hiçbir çözüm bulunamayabilir, böyle bir durumda olay elimine edilir.
- Aksi halde, en iyi fizibil çözüm tarafından verilen global en yüksek sınır ile alt problemin alt sınırı karşılaştırılır. Eğer alt sınır mevcut üst sınırdan büyükse veya ona eşitse yine alt problem elimine edilir.
- Son olarak, eğer alt problem elimine edilemiyorsa, dallanma ve aktif adaylar listesine alt problemin çocuklarını eklemek zorlaşır. Bu noktada, aday alt problemler listesi boşalana yani mevcut en iyi çözüme ulaşana kadar algoritmaya devam edilir.

İşlemlerin alt gruplara ayrılarak çözüm arandığı dal ve sınır algoritmalarındaki en önemli nokta keskin alt sınırların üretilebilmesidir. Keskin alt (duruma bağlı olarak üst) sınırlar dal ve sınır ağacının daha fazla budanmasına neden olarak yöntemin etkinliğini artırır.

Dal ve sınır algoritması yöntemi ile minimizasyon tipli bir problem çözülürken temel mantık aşağıda verildiği gibi açıklanabilir (Çevik [153]):

- Tamsayılı programlama ile uygun çözüm alanı, çok sayıda alt setlere ayrılır. Her bir alt set için sınır değeri hesaplanır. Alt ve üst sınırlardan, her bir alt set içinde çözüm değeri seçilir.
- Bir alt sınır (ilk alt setler arasından en küçüğü) ile alt set diğer bölümler için seçilir. Daha önce olduğu gibi alt ve üst sınırların her ikisi de hesaplanır. Bölünmeye optimal çözüm bulunana kadar devam edilir. Optimal çözüm herhangi bir alt set için alt sınırdan daha büyük olamaz.

Problem maksimizasyon tipli ise, çözüm yöntemi alt ve üst sınır seçimi dışında minimizasyon tipli problem gibidir.

4.3.1.2 Dal ve Kesme Algoritması

Dal ve kesme algoritması, tamsayılı doğrusal programlamanın çözümü için bazılarının veya tümünün bilinmediği tamsayı değerlerinin sınırlandığı kombinasyonel optimizasyon metodudur. Dal ve kesme algoritması, dal ve sınır ve kesme düzlemi algoritmalarının oluşturduğu melez bir yapıdır.

Dal ve kesme algoritmaları, düzenli simpleks algoritmasını kullanarak tamsayı kısıtı olmayan doğrusal programı çözer. Optimum çözüm elde edildiğinde ve bu çözüm tamsayı olarak düşünülen bir değişken için tamsayı olmayan bir değer içerdiğinde, mevcut kesirli çözüm tarafından bozulan tüm fizibil tamsayı noktaları için sonraki doğrusal kısıtların bulunmasında kesme düzlemi algoritması kullanılır. Eğer böyle bir eşitsizlik bulunursa, bu doğrusal programa eklenir ve böylelikle daha az kesirli farklı bir sonuç elde edilebilir. Proses, tamsayılı çözüm bulunana veya daha fazla kesme düzlemi bulunmayana kadar devam eder [154].

4.3.1.3 Kesme Düzlemi Algoritması

Doğrusal programlama problemlerinin tamsayılı çözümlerini sağlayacak hesaplama yöntemi 1959 yılında Gomory tarafından geliştirilmiştir. Gomory'nin geliştirdiği

hesaplama yöntemine “Tamsayılı Algoritma” veya “Kesme Düzlemi Yöntemi” adı verilmiştir.

Dal ve sınır algoritmasında olduğu gibi, kesme düzlemi algoritması da sürekli bir doğrusal programlama probleminin optimum çözümüyle başlar. Ancak bu yöntemde dallanma ve sınırlamadan çok, kesme adı verilen özel kısıtlar ardarda oluşturularak çözüm uzayının düzenlenmesine çalışılır (Taha [155]).

Kesme düzlemi algoritması tüm (saf) tamsayılı programlamayı ve karışık tamsayılı programlamayı içermektedir. Bu yöntemde takip edilecek aşamalar aşağıda verildiği gibi sıralanabilir (Çevik [153]):

- Bir tamsayılı doğrusal programlama probleminde ilk aşama, eğer gerekli ise, orjinal sınırlamaları tamsayılaştırmaktır. Bu, katsayılar tam olsun diye, tüm sınırların değiştirilmesi anlamına gelir.
- İkinci aşamada, doğrusal programlama probleminin optimal çözüm tablosu bulunur. Eğer optimal çözüm değerleri tamsayı ise, tamsayılı doğrusal programlama problemi için çözüm elde edilmiştir. Aksi halde, sonraki aşamaya geçilir.
- Üçüncü ve son aşamada ise kesme bulunur. Bu amaçla optimal çözüm tablosundan tamsayı olmayan değişkenlerden biri seçilir ve yeni bir kısıtlama elde edilir.

4.3.1.4 Sütun Üretme Algoritması

Sütun üretme algoritması Dantzig ve Wolfe tarafından 1960 yılında ayrışabilir yapılı doğrusal programların çözümü için önerilmiştir. Sütun üretme yöntemi, birçok probleme başarıyla uygulanmış ve rotalama ve çizelgeleme problemlerinin çözümünde öncü bir optimizasyon tekniği haline gelmiştir. Simpleks algoritmasının özelliklerini kullanan bu yöntemde bazen, problemlerdeki ağır çakışıklıklara bağlı olarak bazı sütun üretme metotları sık sık kısmi yavaş yakınsama gösterirler. Bu problemlerden bir tanesi, çoklu dual çözümlerin her bir primal çözümle birleştirildiğinde ortaya çıkmaktadır. Dual çözümün seçilmesiyle, sütun üretme algoritmasının can alıcı bölümü alt problem çözümünün genellikle dual değerlere bağlı olmasıdır (Rousseau vd. [156]).

Sütun üretme algoritmasında, M_a ana problemi ve P de alt problemi gösterir. Ana problemin doğrusal programlama gevşemesinden elde edilen bilginin kullanılmasıyla rotaların üretildiği bu yöntemde, her bir iterasyonda amaç fonksiyonuna uygun daha iyi bir sütun aranır. Böyle bir sütun bulunamadığı taktirde algoritma son çözümle bitirilir (Jin vd. [157]).

4.3.1.5 Dal ve Değer Algoritması

Sütun üretme döngüsü doğrusal programlamayı optimize ederken, sütun değerleri için optimum değerlerin *0* veya *1*'e eşit olması gerekmemektedir. Bu sütun değişkenleri için tamsayı değerlerin elde edilmesine bağlı olarak, sütun üretme optimizasyon döngüsü, sayısı belirli dal ve sınır çerçevesine dönüşür. Bu yöntem "Dal ve Değer Algoritması" olarak adlandırılır. Dal ve değer algoritması aşağıda verilen üç aşamadan oluşur (Cardoen vd. [158]):

Dallanma stratejisi: Dallanma için bir sonraki düğümün seçilmesi temeline dayanan iki dallanma stratejisi tanımlanabilir. Bunlar derinlik-önce veya gerileme stratejisi ve en iyi-önce veya çıkarım izleme stratejileridir.

Dallanma şemaları: Dallanma şemalarında, çözüm uzayının bölünmesine bağlı olarak sütun değişkenleri dallanır ve var olan kesirli sütun değişkenleri elimine edilir.

Hızlandırma teknikleri: Dal ve değer algoritmalarının performansını iyileştirmek amacıyla, bazı hızlandırma teknikleri geliştirilip algoritmaya uygulanabilir. Bu teknikler, başlangıç çözümde sütun üretme optimizasyon döngüsü içinde en alt sınır ve dallanma ağacı boyunca sütunların eliminasyonu şeklinde karşımıza çıkabilir.

4.3.1.6 Dinamik Programlama

Dinamik programlama, bir dizi karar verme işlemini optimize eden matematik işlemler bütünüdür. Temelde, dinamik programlama, problem çözümüne, problemin veya problemin bir kısımının parçalara bölünmesi ve bu parçaların çözülerek, bu çözümlerin depolanması şeklinde bir problem çözüm yaklaşımı sunmaktadır. Bu çözümler, ihtiyaç duyulduğunda, yeniden çözülmek yerine, yeniden canlandırılmak suretiyle problemin genel çözümüne eklenerek, dinamik programlama prosedüründe nihai çözüme

ulaşılmaktadır. Dinamik programlama, sistem analizi alanında yaygın olarak kullanılan bir yöntemdir ve çok aşamalı karar verme problemlerinde optimal bir silsileye karar vermede kullanılabilir. Dinamik programlama, özellikle karar aşamasının zaman periyodunda silsile halinde olan problemlere uygundur. Bu problemlerde, periyotlar birbirine öyle bir bağla bağlıdır ki, bir zaman döneminde alınan kararlar sonraki karar verme aşamalarını etkilemektedir. Problem, alt problemlere bölünür ve her bir alt problem için optimal bir çözüm bulunur. *n* sayıda karar verme aşamalarına sahip bir problem, *n* sayıda ve her biri tek bir karar değişkenine sahip, problemlere bölünür. Hesaplama süresi, bir problem içindeki değişkenler sayısınca eksponansiyel olarak büyürken, aynı zamanda alt problemler sayısınca doğrusal olarak büyür. Bir problemin tümü sistem ve alt problemler de basamak olarak düşünülebilir. Dinamik programlamada basamaklar, genellikle, bir zaman aralığını temsil eder. Bir sistemin her bir basamağında, problemin çözüm adımlarına karşılık gelen birden fazla durum vardır. Durumlar, tamamlanmamış çözümleri karakterize eder. Karar verici, her bir basamakta, o basamak için en iyi sonucu veren kararı vermelidir. Bir karar, sistemi bir durumdan diğerine taşır. Bir sistemi bir durumdan diğerine taşıyan her bir aşamaya "Basamak" denir. Dinamik programlama genellikle geriye doğru, yani son durumdan ilk duruma doğru, bir işlemler silsilesi şeklinde uygulanır. Bu geriye doğru endükleme tekniği, son durumdan, bir önceki basamağın durumlarına doğru yapılır (Çetin [159]).

Dinamik programlama, problemleri aşamalara ayrıştırmanın çerçevesini oluşturan optimumluk ilkesine sahiptir. Optimizasyon problemine bağlı olarak aşamaların yapısı farklılık gösterdiğinden, dinamik programlama her bir aşamayı optimum kılmak için gerekli hesaplamaların ayrıntısını vermez ve bu ayrıntılar problem çözücüsü tarafından doğaçlama olarak gerçekleştirilip tasarlanır (Taha [155]).

4.3.2 Sezgisel Yöntemler

Gerçek yaşam problemlerinin çoğunda problemin çözüm uzayı sonsuz veya tüm çözümlerin değerlendirilemeyeceği kadar büyük olur. Bunun için kabul edilebilir bir sürede çözümlerin değerlendirilerek iyi bir çözümün bulunması gerekmektedir (Cura [21]).

En basit anlamıyla algoritma, bir problemin ideal çözümüne giden yoldur ve belirli bir görevi yerine getiren sonlu sayıdaki işlemler dizisinden oluşur. Bir problemin çözümünde onu çözecek olan program dilinden ziyade, problemin algoritması en önemli kısımdır. Programı çalıştıracak algoritma en iyi şekilde ortaya konduktan sonra, sıra kullanılacak olan programlama dilinin yapısına göre kodlama aşamasına gelir (Karaboğa [35]). Her algoritma girdi (sıfır veya daha fazla değer dışarıdan verilmeli), çıktı (en azından bir değer üretilmeli), açıklık (her işlem açık olmalı ve farklı anlamlar içermemeli), sonluluk (her türlü olasılık için algoritma sonlu adımda bitmeli) ve etkinlik (her komut kişinin kalem ve kağıt ile yürütebileceği kadar basit olmalı) gibi kriterleri sağlamalıdır.

Sezgisel algoritmalar yakınsama özelliğine sahiptirler, fakat uygulandıkları problemler için kesin bir çözüm sağlayamazlar. Algoritmalar sadece kesin çözüm yakınındaki iyi bir çözümü garanti edebilirler.

Optimizasyon problemlerine çözüm, kesin çözümü bulma işleminin tanımlanamadığı bir yapı söz konusu olduğunda sezgisel algoritmalar ile aranır. Sezgisel algoritmalar anlaşılabilirlik açısından karar verici için çok daha basit olduğundan ve öğrenme amaçlı ve kesin çözümü bulma işleminin bir parçası olarak kullanılabildiklerinden dolayı günümüzde oldukça sıklıkla kullanılan çözüm metotları halini almışlardır.

Literatürde var olan çalışmalar incelendiğinde, araç rotalama problemlerinin çözümünde sezgisel algoritmaların öne çıktığı ve bu problemlerin çözümü ile ilgili birçok sezgisel algoritmanın öne sürüldüğü görülür. Burada dikkat edilmesi gereken konu, araç rotalama probleminin çözümüne uygulanan bu sezgisel algoritmaların özellikleri ve bu özellikleri sayesinde araç rotalama problemine uygulanabilirlikleridir. Bu noktada Cordeau ve diğerlerinin [160] da yaptığı çalışma dikkat çekicidir. Yazarlara göre araç rotalama problemi için iyi sonuçlar veren bir sezgisel algoritma aşağıda verilen özelliklere sahip olmalıdır:

Kesinlik: Algoritmanın ürettiği değerin problemin optimum çözümüne olan oranıdır.

Hız: Ortaya konacak olan algoritmanın problem için çözüm bulma zamanıdır.

Basitlik: Sezgisel algoritmanın anlaşılmasının ve kodlanmasının kolaylığıdır.

Esneklik: İyi bir sezgisel algoritmanın çeşitli kısıtlar ile bir araya getirilerek gerçek sistemlere uygulanabilir nitelikte olmasıdır.

4.3.2.1 Klasik Sezgisel Algoritmalar

Klasik sezgisel algoritmalar genel olarak üç sınıfa ayrılabilir (Eryavuz ve Gencer [129]):

Yapısal (tur kurucu) sezgiseller: Tur kurucu sezgiseller, mümkün olmayan atamalarla çözüme başlar ve her defasında iki düğüm arasına bir dal ekleyerek mümkün çözüme ulaşırlar. Dal eklenirken araç kapasite kısıtına uyulup uyulmadığı kontrol edilir. Eklenecek dal, bazı maliyet tasarruflarına göre seçilir.

İyileştirmeli (tur geliştirici) sezgiseller: Tur geliştirici sezgiseller, bir mümkün çözümü başlangıç çözümü olarak alır ve o çözümü geliştirirler. Her bir iterasyonda dal kombinasyonları değiştirilir ve yapılan bu değişimin problemi mümkün çözüme ulaştırıp ulaştırmadığı, maliyeti düşürüp düşürmediği kontrol edilir.

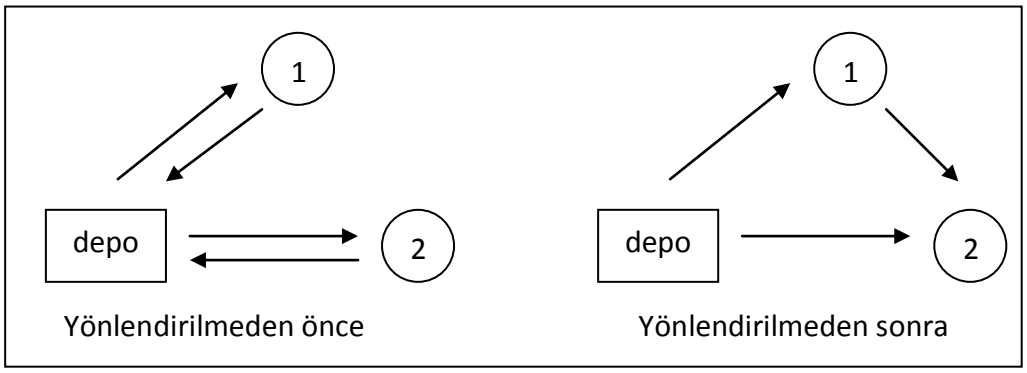
İki aşamalı sezgiseller: İki aşamalı sezgisellerin birinci aşamasında, düğümler araçlara kapasiteyi aşmayacak şekilde atanır. Sezgiselin ikinci aşamasında ise, her bir araç için gezgin satıcı problemi sezgiselleri kullanılarak rota oluşturulur.

4.3.2.1.1 Yapısal Sezgisel (Tur Kurucu) Algoritmalar

Tur kurucu algoritmalar, kapasiteli araç rotalama problemleri için çözüm arayan ilk sezgiseller arasındadır ve halen birçok çeşitli rotalama uygulamaları için çok sayıdaki yazılımın yürütülmesinde çekirdek yapı niteliğindedir. Tipik olarak bu algoritmalar, boş çözümle başlar ve tüm müşteriler rotalanana kadar her bir iterasyonda bir veya birden daha fazla müşterinin eklenmesi işleminin tekrarlanmasıyla rotalar oluşturur. Tur kurucu algoritmalar, müşterinin eklenmesi için elverişli rotaların sayısına bağlı olarak sıralı ve paralel olmak üzere ikiye ayrılır. Sıralı yöntemlerde bir zaman diliminde sadece bir rotada genişleme yapılırken, paralel yöntemlerde ise genişleme için eş zamanlı olarak birden fazla rota göz önüne alınır. Tur kurucu algoritmalar, sahip oldukları üç ana bileşene göre özelleşirler. Bu bileşenler, başlangıç kriteri, mevcut iterasyonda ekleme için hangi müşterilerin seçileceğini belirleyen seçim kriteri ve seçilen

müşterilerin mevcut rotalar içine yerleştirilmesine karar veren ekleme kriteridir (Cordeau vd. [161]). Yapısal sezgisel algoritmalar dört sınıfa ayrılır:

Tasarruf Algoritması: Clarke ve Wright tasarruf algoritması, araç rotalama probleminin çözümü için bilinen en iyi sezgisellerden biridir. Algoritma, Clarke ve Wright tarafından 1964 yılında geliştirilmiştir. Tasarruf algoritması, araç sayılarının (karar değişkeni) belirli olmadığı problemlere uygulanır ve hem yönlendirilmiş hem de yönlendirilmemiş problemler için eşit derecede en iyi şekilde çalışır. Tasarruf algoritmasının temel konsepti Şekil 4.6'da gösterilmiştir.



Şekil 4.6 Tasarruf algoritması konsepti (Eryavuz ve Gencer [129])

Tasarruf algoritmasında, (*0,…,i,0*) ve (*0,…,j,0*) şeklindeki iki rota, (*0,…,i,j,…,0*) şeklinde verilen tek bir rota içine fizibil olarak dönüştürüldüğünde, mesafe tasarrufu,

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{4.18}$$

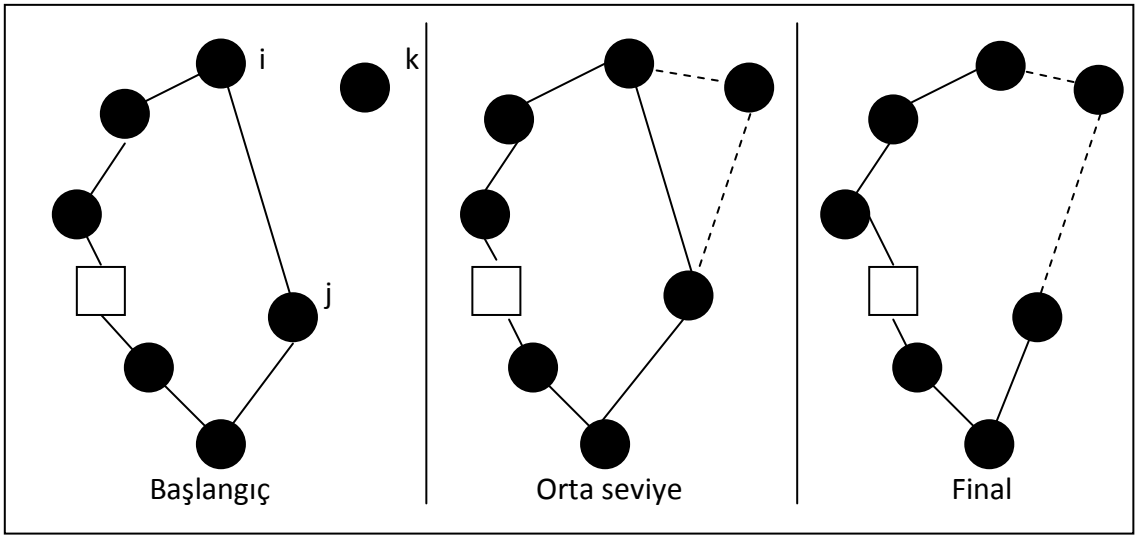
olarak ortaya çıkar. Tasarruf algoritmasında rotalar arka arkaya sıralı veya aynı anda paralel oluşturulabilir. Araç kapasitesi dolana kadar talep noktaları mevcut rotaya eklenirse, rotalar arka arkaya sıralı şekillenmiş olur. Her bir araç için aynı anda kısmi rotalar oluşturulursa, rotalar paralel şekillenir (Eryavuz ve Gencer [129]).

Eşleme Tabanlı Sezgisel Algoritma: Eşleme tabanlı sezgisel algoritma 1989 yılında Desrochers ve Verhoog tarafından literatüre kazandırılmıştır. Özellikle çok talep noktası olan kapasite kısıtlı araç rotalama problemleri için uygun olan bu yönteme göre tasarruf değeri s_{pq} bir kerede hesaplanmayıp, her iterasyonda, p ve q rotasının birleştirilmesinden elde edilir. S_k, k rotasına ait nokta kümesi ve $t(S_k)$ bu noktalara ait gezgin satıcı probleminin optimum çözümü ise tasarruf miktarı

$s_{pq} = t(S_p) + t(S_q) - t(S_p \cup S_q)$ denklemi ile bulunur. Eşleme tabanlı sezgisel algoritma ile elde edilen rotalar da kapasite kısıtı göz önüne alınarak s_{pq} değerlerine göre büyükten küçüğe göre eşlenerek birleştirilir (Van Breedam [162]).

Eşleme tabanlı sezgisel algoritma ile ilgili 1994 yılında Wark ve Holt tarafından geliştirilen bir başka yönteme göre ise eşleme ağırlıkları olarak *s* tasarruf miktarı değeri alınır. Rotaların her bir birleştirilme işleminden sonra eşleme ağırlıklarının hesaplandığı ve eğer tüm gruplar birbirleri ile eşlenirse rastgele olarak bazı grupların seçilerek ikiye bölündüğü bu yöntem iterasyonlar boyunca gruplardan oluşan bir ağaç gibi gelişerek problem için uygun bir çözüm üretir (Toth ve Vigo [135]).

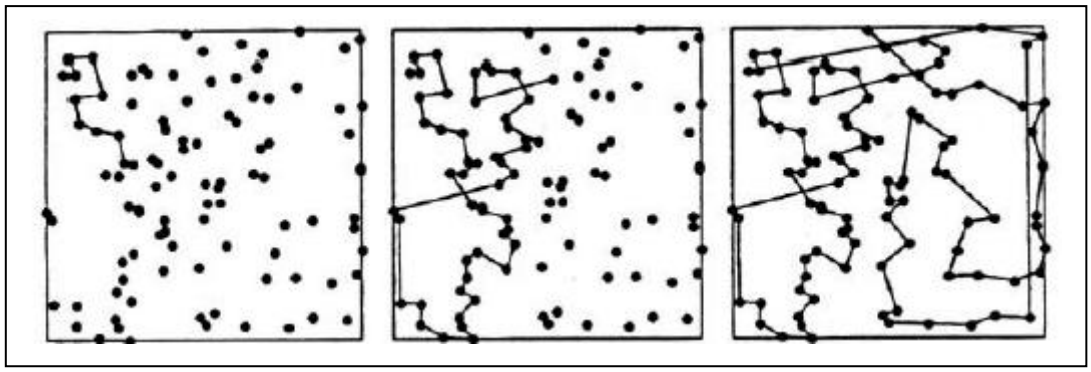
Ekleme Sezgisel Algoritması: Her periyotta bir müşteriyi rotaya yerleştirerek çözüm inşa eden ekleme sezgiselleri her periyotta ardışık ekleme sezgiselleri ile bir rota oluşturabilecekleri gibi paralel ekleme sezgiselleri ile de paralel olarak birçok veya bütün rotaları oluşturabilirler. Hangi müşterinin nereye ekleneceği gibi verilen kararlarla birbirinden ayrılan ekleme sezgiseli çeşitlerinde amaç toplam maliyeti en az etkileyen müşteriyi seçerek rotaya ekleyebilmektir. Ekleme sezgiseli algoritmalarında genel olarak *k* noktasından oluşan bir alt tur ele alınır, prosedürün seçim basamağı işlemi ile henüz alt turda yer almayan hangi noktanın alt tura eklenmesi gerektiği belirlenmeye çalışır. Şekil 4.7'de ekleme sezgiselinin uygulama örneği gösterilmiştir.



Şekil 4.7 Ekleme sezgiselinin uygulama örneği (Savelsbergh [163])

Araç sayısının değişken olarak alındığı yükleme kapasitesi bilinen kapasite kısıtlı araç rotalama problemi için geliştirilmiş bir yöntem olan ekleme sezgiseli algoritması başlangıç rotalarının oluşturulması ve bu rotalarla ilgili talep noktalarının rotaya mümkün olan en az maliyet ile eklenmesine dayanmaktadır. Eklemeli sezgisel algoritmalar ile ilgili iki önemli çalışma vardır. Bunlar ele alınan rotaya atanacak olan yeni noktanın, λ ve μ parametreleri ile belirli kurallara göre seçildiği Mole ve Jameson sıralı eklemeli sezgisel algoritması ve rotaların ilk olarak sıralı ve daha sonra da paralel şekilde ele alındığı Christofides, Mingozzi ve Toth sıralı eklemeli sezgisel algoritmasıdır (Toth ve Vigo [135]).

En Yakın Komşu Sezgiseli: En yakın komşu sezgiseli, gezgin satıcı probleminin çözümü için geliştirilmiş olan en basit tur kurucu sezgiseldir. Sezgiselin işleyişi başlangıçta rassal olarak bir şehrin seçilmesiyle başlar ve tam tur elde edilinceye kadar her adımda seçilen şehire en yakın şehrin seçilmesi ve bu şehrin kısmi çözüme eklenmesi ile devam eder (Aydemir [164]). Şekil 4.8'de en yakın komşu sezgiseli için uygulama örneği gösterilmiştir.



Şekil 4.8 En yakın komşu sezgiseli (Gambardella [128])

En Kısa Yol Algoritması: En kısa yol algoritması, sadece bir dağıtım noktasının ve bir varış noktasının olduğu şebekelerde rotalama işlemi yapmak için kullanılmaktadır. Bu yöntemde bağlantılar ve düğümlerle temsil edilen bir şebeke bulunmaktadır. Şebeke içerisinde bulunan düğümler maliyet, zaman, mesafe veya bunların bir karışımı olarak ifade edilebilen bağlantılar vasıtasıyla birbirine birleştirilir.

İki tür en kısa yol algoritması vardır. Bunlar Dijksta ve Floyd yöntemidir. Kaynak düğümüyle şebekedeki diğer bir düğüm arasındaki en kısa yolları belirlemek üzere

Dijkstra algoritması tasarlanırken, şebekedeki herhangi iki düğüm arasındaki en kısa yolun belirlenmesine izin veren Floyd algoritması daha genel bir yapıdadır (Taha [155]).

4.3.2.1.2 İyileştirmeli Sezgisel (Tur Geliştirici) Algoritmalar

Yerel arama algoritmaları, diğer sezgisellerle üretilen başlangıç çözümlerini geliştirmede sıklıkla kullanılırlar. Verilen çözümden başlanarak, daha iyi maliyetli olası komşu çözümlerin oluşturulmasında yerel arama metoduna, yay değiş tokuşu veya müşteri hareketi gibi basit değişiklikler uygulanır. Eğer gelişmiş bir çözüm bulunursa, mevcut çözüm bu bulunan yeni çözüm olur ve proses tekrarlanır. Aksi halde, yerel minimum tanımlanır (Cordeau vd. [161]).

Tek Rota İyileştirmeli Sezgisel Algoritma: Araç rotalama problemlerinin çözümü içinde yer alan ve en sık kullanılan çözüm yöntemlerinden biri de 1965 yılında Lin tarafından gezgin satıcı probleminin çözümü için geliştirilen ve tek bir rotanın iyileştirilmesi amacıyla kullanılan λ-opt metodudur. λ adet doğrunun (yol) bir rotadan çıkartılarak mümkün olan tüm permütasyonlarda bu doğrunun rotanın çeşitli noktalarına eklenmesiyle gerçekleştirilen ve mevcut çözümden daha iyi bir çözüm bulunmasının amaçlandığı bu yöntemde yeni rota çıktı olarak verilir. Genellikle birbirini kesmeyen doğrulardan oluşan bir rota kurmaya çalışılan λ-opt yönteminde algoritma adımları içerisindeki "opt" bir fonksiyon içine kodlandığı takdirde 3-opt algoritması opt(opt(rota)) ile, 4-opt algoritması ise opt(opt(opt(rota))) ile tekrarlamalı bir şekilde gösterilerek elde edilir. Genel olarak bir araç rotalama probleminde λ-opt algoritmasının $0\left(n^{\lambda}\right)$ sürede sonuç bulması beklenirken λ-opt algoritması üzerinde yapılabilecek olan değişiklikler ile bu süre kısaltılabilinir veya yöntemin bazı versiyonlarında başlangıç rotasından daha iyi sonucun elde edildiği durumda çözüm saklanır ve arama süreci devam ettirilir (Laporte vd. [165]).

Tek rota iyileştirmeli sezgisel algoritmalar arasında bir turdaki iki kenar silinerek turun iki parçaya ayrılması ve ardından bu parçaların ters çevrilerek birleştirilmesiyle oluşturulan 2-opt (Özkan vd. [166]), tur içerisindeki 3 kenar bağlantısının silinmesi ve bu kenarların olası başka yollar ile yeniden bir araya getirilmesiyle oluşturulan 3-opt [167], üçe kadar komşu düğümlerden oluşan ardışık dizinin kaldırılarak bu dizinin aynı

rota içerisinde başka bir yere eklenmesiyle oluşturulan or-opt ve algoritmanın her iterasyonunda kenar sayısına dinamik olarak karar verilmesiyle olşturulan Lin-Kermingham sezgiseli sayılabilir.

Çok Rota İyileştirmeli Sezgisel Algoritma: İyileştirme algoritmaları, araç rotaları içinde ya da arasında kenar veya düğümlerin sıralamalarının değiştirilmesi ile fizibil çözümün geliştirilmesi için uygulanır. Araç rotalama problemlerinin çözümü için kullanılan çok rota iyileştirmeli sezgiseller, çeşitli rotalar içinden alınan her bir araç rotası üzerinde eş zamanlı olarak yürütülürler.

Mevcut çözümün dışında komşuluk çözümüne göre oluşan rotalar arasındaki durakların mutasyonu olarak tanımlanabilen Van Breedam sezgiseli (Van Breedam [162]), döngüsel *k*-transferler kavramı temelline dayanan Thompson ve Psaraftis sezgiseli ve turların birbirinden izole edilmiş olarak düşünülmediği ve müşteriler ile yolların farklı turlar arasında değiştirildiği Kindervater ve Savelsbergh sezgiseli literatürde bulunan çok rota iyileştirmeli sezgisel algoritma örnekleridir.

4.3.2.1.3 İki Aşamalı Sezgisel Algoritmalar

İki aşamalı metotlar, araç rotalama problemi çözüm prosesini iki alt probleme ayıran parçalama temellidir. Bu alt problemler aşağıda verildiği gibidir (Cordeau vd. [161]):

Kümeleme: Her bir ilgili rotada müşterilerin alt setler içine bölünmesini belirler.

Rotalama: Her bir rotadaki müşterilerin sırasını belirler.

İlk Grupla Sonra Rotala Yöntemleri: İlk grupla sonra rotala yöntemlerinde, talep noktalarının bir kez temel gruplara bölünmesiyle oluşturulan her bir grup için rotalama yaparak, tam sonuç veren bir yöntemi sadeleştirerek ve çok sayıda nokta grubunun oluşturulması ile bu nokta gruplarından uygun olanlarını seçerek çözüm aranmaktadır.

Müşterilerin gruplar içerisine kümelenmesinde kullanılan bir metot olan süpürme algoritması, uygun bir şekilde tanımlanmış olan genelleştirilmiş atama problemi vasıtasıyla kümeleme adımını çözen Fisher ve Jaikumar atama tabanlı algoritması (Cordeau vd. [161]), kapasite kısıtlı problemlerin çözülmesi ile belirlenen çekirdekler ve bölüştürülmüş rotalar içine adım adım eklenen arta kalmış düğümleri kapsayan Bramel

ve Simchi-Levi lokasyon tabanlı algoritması (Laporte vd. [165]), tüm olası dallanma olanaklarını kullanmayarak dallanmanın sezgisel kurallar marifeti ile daraltıldığı budanmış dal ve sınır yöntemi, taç yaprakları olarak adlandırılan birçok rotanın oluşturulduğu ve bölünmüş problem setlerinin çözülmesiyle sonuç seçiminin yapıldığı taç yaprağı algoritması ve tabu rota özelliklerinden bazılarını içeren ve bir tabu arama uygulaması olan Taillard algoritması (Laporte vd. [165]) literatürde varolan iki aşamalı sezgisel algoritmalardır.

İlk Rotala Sonra Grupla Yöntemleri: Araç rotalama probleminin gezgin satıcı problemi gibi ele alındığı ilk rotala sonra grupla türündeki algoritmalarda kapasite kısıtları göz ardı edilerek her müşteriyi içinde barındıran büyük bir tur oluşturulur. Oluşturulan bu turun, uygun araç rotalarına bölünerek probleme çözüm bulunmaya çalışıldığı bu yöntem ile ilgili ilk çalışma 1983 yılında Beasley tarafından yapılmış ve Beasley ilk rotala sonra grupla yöntemlerinin standart bir en kısa yol problemi olduğunu ileri sürmüştür.

4.3.2.2 Metasezgisel Yöntemler

Metasezgisel yöntemler, araç rotalama problemleri gibi karmaşık yapıdaki kombinasyonel optimizasyon problemlerinin çözümü için geliştirilmiş özel sezgisellerdir. Metasezgisel ifadesi, literatürde ilk kez Fred Glover tarafından 1986 yılında ortaya atılmıştır (Gendreau vd. [168]).

Metasezgisel yöntemler, literatürde birçok değişik yapıdaki araç rotalama probleminin çözümüne uygulanmıştır. Klasik sezgisel yöntemler dikkate alındığında, metasezgisel yöntemlerin çözüm uzayının araştırılmasında daha etkili olduğu ve yerel optimumla sonlanmalarının daha az gerçekleştiği görülmektedir. Metasezgisel yöntemler, genel olarak aşağıda verilen üç temel alana ayrılırlar (Cordeau vd. [161]):

Yerel arama: Tavlama benzetimi, deterministik tavlama ve tabu arama tarafından içerilir.

Popülasyon arama: Genetik algoritma ve uyarlamalı hafıza prosedürü tarafından içerilir.

Öğrenme mekanizması: Sinir ağları ve karınca kolonisi optimizasyonu tarafından içerilir.

Metasezgisel yöntemler tipik olarak, gelişmiş komşuluk arama kurallarını, hafıza yapılarını ve çözüm kombinasyonlarını birleştirirler. Bu yöntemler ile üretilmiş olan çözümlerin kalitesi, genellikle klasik sezgisel yöntemlerle elde edilen sonuçlardan daha iyidir. Fakat metasezgisel yöntemler çözüm elde etmede daha fazla hesaplama zamanı gerektirmektedir. Ayrıca, metasezgisel yöntemlerdeki süreçler genellikle şartlara bağlıdır ve bu sezgiseller bazı zor durumların genelleştirilmesi için parametrelere ihtiyaç duyarlar. Metasezgisel yöntemler, iyileştirme süreçlerinden daha gelişmiş olmamakla beraber, klasik sezgisel yöntemlerin doğal bir iyileştirmesi olarak görülebilirler (Laporte vd. [165]).

Metasezgisel yöntemler, arama sürecine rehberlik eden stratejilerdir. Bu yöntemlerde amaç, en iyi ya da en iyiye yakın çözümleri bulmak için arama uzayını hızlı bir şekilde araştırmaktır. Metasezgisel yöntemler, basit yerel arama algoritmalarından karmaşık öğrenme süreçlerine kadar geniş bir yelpazeyi içermektedir. Metasezgisel yöntemler, yaklaşık algoritmalardır ve genellikle deterministik değildirler. Bu yöntemler, arama uzayındaki yerel en iyi tuzaklardan kurtulmak için çeşitli mekanizmaları kullanırlar. Öte yandan, metasezgisel yöntemler, probleme özgü değildirler ve üst seviye stratejiler tarafından kontrol edilen sezgisellerde probleme özgü bilgi kullanımına izin verirler. İleri seviye metasezgisel yöntemler ise, aramaya rehberlik etmesi amacıyla arama sırasında elde edilen bilgiyi (hafızayı) kullanırlar. Kısacası, bu yöntemler, farklı metotlar ile arama uzayının araştırılması için yüksek seviyedeki stratejilerdir (Aydemir [169]).

Tabu Arama Algoritması: Temel olarak Tabu Arama (TA) kavramı ilk olarak Glover tarafından 1986 yılında ortaya atılmıştır. Tabu arama algoritmasındaki temel yaklaşım, problemi son çözüme götüren adımın, dairesel hareketler yaratmasını engellemek için bir sonraki döngüde tekrarının yasaklanması veya cezalandırılmasıdır. Tabu arama yöntemi kısmen, benzer olaylarda rastlantısal işlemler gerçekleştirirken insan davranışının tutarsızlığa eğilimli olmasından etkilenmiştir. Tabu arama yöntemi, incelenmiş bir yol olmadığı sürece her çözümü araştırabilen bir süreçtir. Böylece yeni bir çözüm uzayının incelenmesi suretiyle yerel minimumdan kaçınılarak istenilen çözüme ulaşılabilmektedir. Algoritma, yerel minimuma doğru hareket ederek başlar, daha önce yapılmış hareketlere tekrar dönüş yapmayı engellemek için yöntem bir veya birden fazla tabu listesi tutar. Bu tabu listesinin orjinal amacı, önceden yapılmış bir

hareketin tekrarından çok tersine dönmesini önlemektir. Tabu listesi kronolojik bir yapıya sahiptir ve tabu arama belleğini biçimlendirir. Belleğin rolü algoritma ilerledikçe değişebilir. Başlangıçta hedef, çözüm uzayında kaba araştırma yapmak iken, aday konumlar belirlendikçe arama yerel optimum çözümü üretmeye daha fazla odaklanır (Cura [21]).

Literatür çalışmalarına bakıldığında araç rotalama problemlerine çözüm üretmek için tabu arama metasezgiselinin son yıllarda sıklıkla kullanıldığı görülmektedir. Cordeau ve diğerlerinin [146] da periyodik ve çok depolu araç rotalama problemine uygulandıkları tabu arama algoritması, Ho ve Haugland'ın [170] de zaman pencereli ve bölümlü dağıtımlı araç rotalama problemine uyguladıkları tabu arama algoritması, Leung ve diğerlerinin [171] de iki boyutlu yüklemeli araç rotalama problemi için önerdikleri genişletilmiş tabu arama ve yeni paketleme algoritması, Brandao'nun [172] de heterojen araç rotalama probleminin çözümünde kullandıkları tabu arama algoritması ve Montane ve Galvao'nun [173] de tabu arama algoritması ile çözüm aradıkları eşzamanlı toplama ve dağıtım hizmetli araç rotalama problemi bu çalışmalara örnek olarak verilebilir.

Tavlama Benzetimi Algoritması: Tavlama Benzetimi (TB), bir metalin soğuyarak ve donarak minimum enerjili kristal yapıya dönüşmesi ile daha genel bir sistemde minimumun araştırılması arasındaki benzerlikten yararlanır. Bu yaklaşımı daha iyi anlayabilmek için doğal tavlamanın esasları ortaya konmalıdır. Doğal tavlamada, herhangi bir katı madde erime noktasını aşıncaya kadar ısıtılır ve ardından katılaşıncaya kadar soğutulursa, bu katı maddenin yapısal özellikleri soğuma hızına bağlı olarak değişir. Örneğin, büyük kristaller çok yavaş soğutulacak olursa gelişmeler gözlenebilirken, hızlı soğutulma neticesinde yapılarında birçok bozulmaları barındırırlar. Yani, ısıtılan ve ardından belli bir hızla soğutularak en iyi biçimine ulaştırılmaya çalışılan bir madde, bir sistemdeki parçacık gibi algılanırsa bu tavlama sürecinden tavlama benzetimi elde edilmiş olunur (Cura [21]).

Literatür çalışmalarına bakıldığında araç rotalama problemlerinin çözümü için tavlama benzetimi metasezgiseli ile diğer sezgisellerin bir araya getirildikleri ve oluşturulan bu melez yapılar ile problemlere çözüm arandığı görülmektedir. Örneğin, Tan ve diğerleri

[131] de zaman pencereli araç rotalama probleminin çözümünde tavlama benzetimi, tabu arama ve genetik algoritma sezgisellerini kullanırken, Tavakkoli-Moghaddam ve diğerleri [174] te ise bağımsız rota uzunluklu kapasite kısıtlı araç rotalama probleminin çözümü için tavlama benzetimi algoritması temelli en yakın komşu arama ve tam sayılı programlama yöntemlerinden oluşan melez bir yapıyı önermişlerdir.

Yapay Sinir Ağları: Yapay Sinir Ağları (YSA), insan beyninin çalışma ilkelerinden ilham alınarak geliştirilmiş, ağırlıklı bağlantılar denilen tek yönlü iletişim kanalları vasıtası ile birbirleriyle haberleşen, her biri kendi hafızasına sahip birçok işlem elemanından oluşan paralel ve dağınık bilgi işleme yapılarıdır. Yapay sinir ağları gerçek dünyaya ait ilişkileri tanıyabilir, sınıflandırma, kestirim ve işlev uydurma gibi görevleri yerine getirebilir. Desen tanıma tekniğinin gerekliliği, gerçek dünya ile bilgisayar ilişkisinin başlaması ile ortaya çıkmıştır. Bu durum yapay sinir ağlarının çok güçlü örnek tanıma tekniği olarak ortaya çıkmasına ve gelişmesine neden olmuştur (Erdem ve Uzun [175]).

Parçacık Sürü Optimizasyonu: Parçacık Sürü Optimizasyonu (PSO) yaklaşımı, parçacıkların sürü halindeki toplu hareketinden yararlanan bir tekniktir. Söz konusu parçacıkların her birisi bir çözüm adayını temsil eder ve bir optimizasyon probleminin mümkün çözüm uzayını araştırmak için kullanılırlar. Her bir parçacık başlangıç aşamasında tesadüfi veya sezgisel olarak belirlenir ve daha sonra serbest harekete bırakılır. Her bir optimizasyon adımında, her parçacık kendi uygunluğunu ve çevresindekilerin uygunluğunu ölçer. Bu uygunluk, optimum değere olan uzaklık gibi düşünülebilir. Parçacık sürü optimizasyonu, ilk olarak Kennedy ve Eberhart tarafından 1995 yılında, kuş sürülerinin davranışlarından esinlenilerek geliştirilmiş popülasyon tabanlı bir stokastik optimizasyon tekniğidir. Bu algoritma, potansiyel çözümlerin oluşturduğu bir kümenin, belirli bir problemin uygun çözümüne yaklaşmak için evrime uğradığı popülasyon temelli bir algoritmadır. Bir optimizasyon tekniği olarak amacı, belirli bir çözüm uzayında tanımlanmış bir fonksiyonun global optimum noktasını bulmaktır (Cura [21]).

Literatür çalışmaları incelendiğinde parçacık sürü optimizasyonu yönteminin, araç rotalama problemlerinin çözümünde tek başına kullanılmaktansa diğer metasezgisellerle melez yapı oluşturularak kullanıldığı görülmektedir. Bu çalışmalara

örnek olarak, Ai ve Kachitvichyanukul'un [176] da eşzamanlı dağıtım ve toplama işlemlerini kapsayan araç rotalama problemi çözüm çalışması, yine aynı yazarların 2009 yılında gerçekleştirdikleri kapasiteli araç rotalama problemi çözüm çalışması, Marinakis ve diğerlerinin [177] de araç rotalama probleminin çözümü için önerdiği komşu arama sezgiseli ve parçacık sürü optimizasyondan oluşan çözümü önerisi ve Marinakis ve Marinaki'nin [178] de genetik algoritma ile entegre edilmiş parçacık sürü optimizasyon yöntemi çalışması verilebilir. Yazarlar bu çalışmalarla araç rotalama problemlerinin çözümü için yeni bir yaklaşım ortaya koymuş ve bu yaklaşımla aynı problem türleri için ayrı ayrı kullanılan genetik algoritma ve parçacık sürü optimizasyonu sonuçlarını geliştirmişlerdir.

Yapay Bağışıklık Sistemi Algoritması: Yapay Bağışıklık Sistemi (YBS), yapay sinir ağları ve genetik algoritmalarda olduğu gibi doğadan taklit edilerek ortaya çıkarılmış genel amaçlı bir sezgisel algoritmadır. Canlılardaki savunma mekanizması özetlenip modellenerek oluşturulmuş bu optimizasyon algoritmasının, mühendislik alanındaki birçok kompleks problemin çözümünde nasıl performans göstereceği incelenmiştir (Duman ve Akın [179]).

Literatür çalışmaları incelendiğinde yapay bağışıklık sistemi algoritmasının kullanım alanları içine araç rotalama problemlerinin yeni yeni girdiği görülmektedir. Bu noktada, 2009 yılında Masutti ve Castro tarafından [180] de gerçekleştirilen çalışmada gezgin satıcı problemini çözmek için bu algoritmanın kullanıldığı görülmektedir. Çalışmada kurulan her bir ağ, iki nöronla başlatılır ve yapay bağışıklık sisteminin temeli olan klonlama prosesi ile ağ içine yeni nöronlar yerleştirilir.

Diferansiyel Gelişim Algoritması: Diferansiyel Gelişim Algoritması (DG), Price ve Storn tarafından 1995 yılında geliştirilmiş, özellikle sürekli verilerin söz konusu olduğu problemlerde etkin sonuçlar verebilen, işleyiş ve operatörleri itibariyle genetik algoritmaya dayanan popülasyon temelli sezgisel optimizasyon tekniğidir. Temel olarak genetik algoritmaya dayanmaktadır. Aynı anda birçok noktada araştırma yapabilmekte ve iterasyonlar boyunca operatörler yardımıyla problemin çözümü için daha iyi sonuçlar araştırabilmektedir. Genetik algoritmadan farklı olarak değişkenler gerçek değerleriyle temsil edilmektedir. Ayrıca genetik algoritmanın diferansiyel gelişim

algoritmasında kullanılan operatörlerinden her biri tüm popülasyona sırayla uygulanmamaktadır. Kromozomlar tek tek ele alınmakta, rastgele seçilen diğer üç kromozomda kullanılarak yeni bir birey elde edilmektedir. Mevcut kromozomla elde edilen yeni kromozomun uygunlukları karşılaştırılarak uygunluğu daha iyi olan, yeni birey olarak bir sonraki popülasyona aktarılmaktadır. Diferansiyel gelişim algoritmasında, üretilen çözümlerin kalitesi, amaç fonksiyonuna ürettikleri değerle (uygunluk değeri) ölçülmektedir. Diferansiyel gelişim algoritmasının diğer sezgisellere göre önemli üstünlüğü kolayca kodlanabilmesidir. Diğer algoritmalar için binlerle ifade edilen satırdan oluşan kodlar söz konusu iken diferansiyel gelişim algoritması için yaklaşık 20 satırlık kod yeterli olmaktadır (Keskintürk [181]).

Literatür çalışmaları incelendiğinde diferansiyel gelişim algoritmasının araç rotalama problemlerine, sıklıkla uygulanmadığı görülmekle beraber, Erbao ve Mingyong'un [182] de yaptıkları araç rotalama probleminin çözümü için stokastik simülasyon ve diferansiyel gelişim algoritmasının entegre edildiği melez yapı çalışması bu algoritmanın araç rotalama problemlerinin çözümünde kullanılabilmesi açısından örnek olma niteliğindedir.

4.3.2.3 Araç Rotalama Problemi için Klasik Sezgisel Yöntemlerin Karşılaştırılması

Araç rotalama problemleri hakkında Cordeau ve arkadaşları tarafından yapılan ve Çizelge 4.1'de verilen değerlendirmede klasik sezgisel yöntemlerin karakteristikleri belirtilmiş ve genel olarak klasik sezgisel yöntemlerin hızlı sonuçlar vermelerine rağmen, bu yöntemler ile üretilen çözümlerin yüksek kalitede olmadığı ve bu yöntemleri gerçek hayat problemlerine uyarlamanın zor olduğu sonucu bulunmuştur (Cordeau vd. [160]).

Çizelge 4.1 Araç rotalama problemleri için klasik sezgisel yöntemlerin karşılaştırılması

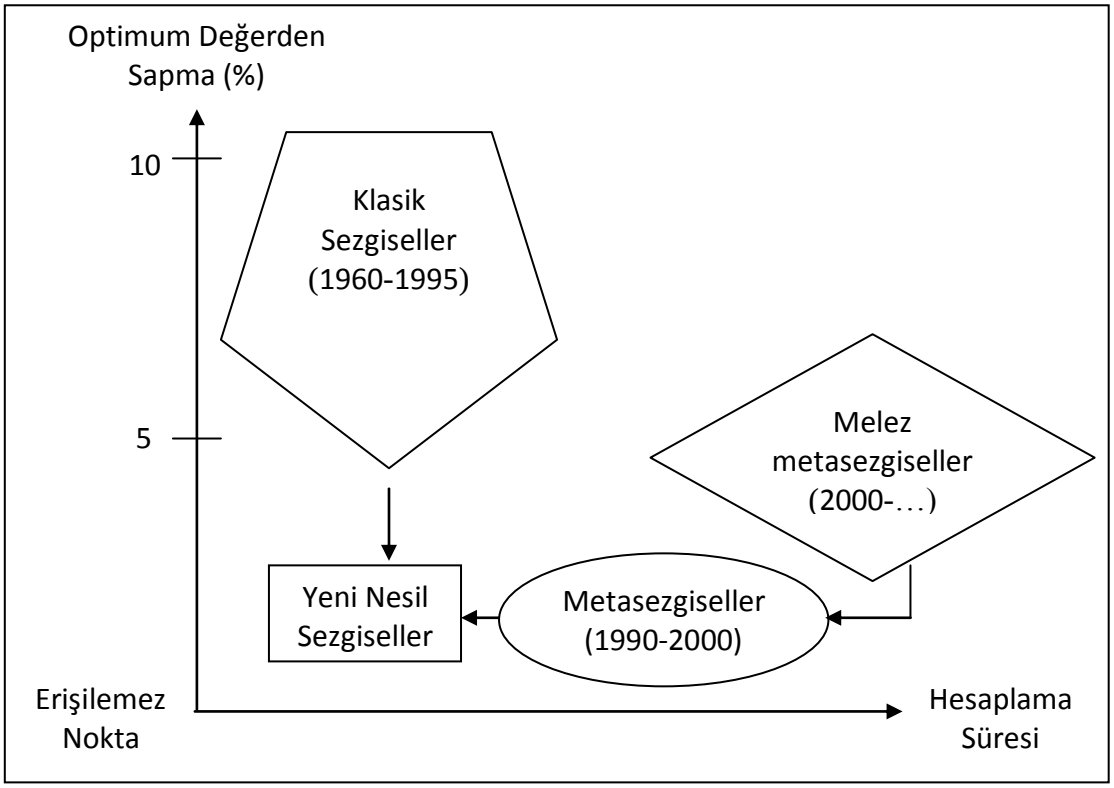
Yöntem	Kesinlik	Hız	Basitlik	Esneklik
Clarke ve Wright Tasarruf Algoritması	Düşük	Çok yüksek	Çok yüksek	Düşük
Eşleme Tabanlı Metotlar	Yüksek	Çok düşük	Düşük	Düşük
Süpürme Yöntemi	Düşük	Orta yüksek	Yüksek	Düşük
1 - Taç Yaprağı Yöntemi	Düşük	Yüksek	Orta	Orta
2 - Taç Yaprağı Yöntemi	Orta	Orta	Orta	Orta
Fisher ve Jaikumar Atama Tabanlı Yöntemi	-	Orta	Düşük	Düşük
Bramel ve Simchi-Levi Lokasyon Tabanlı Algoritması	Orta	Düşük	Düşük	Düşük

Çizelge 4.1'e göre araç rotalama problemlerini çözüm kesin yöntemlerden biri olan Clarke ve Wright tasarruf algoritması oldukça hızlı bir sürede çözüm bulurken problemde oluşabilecek değişikliklere adaptasyon sağyalamamaktadır. Bunun yanı sıra eşleme tabanlı metotlar oldukça zor olan yapılarına rağmen kesin sonuçlar bulmada başarılıdır. Bramel ve Simchi-Levi lokasyon tabanlı algoritması ise oldukça basit ve esnek olan yapısı ile daha uzun bir işlem süresi gerektirmektedir.

4.3.2.4 Araç Rotalama Problemi için Metasezgisel Yöntemlerin Karşılaştırılması

Metasezgisel yöntemler doğruluk, hız, basitlik ve esneklik kriterlerine göre değerlendirilebilirler (Laporte [183]). Metasezgisel algoritmalar müşteri sayısı birkaç yüz adete kadar olan araç rotalama problemleri için makul sürelerde yüksek kaliteli optimuma yakın sonuçlar üretmektedirler.

Metasezgisel yöntemlerin çözüm süreleri klasik sezgisel yöntemlere kıyasla biraz daha uzun olsa da, bu yöntemlerle üretilen çözüm sonuçları tatmin edici düzeydedir. Metasezgisel yöntemler arasında yapılan karşılaştırmalar ile de tabu arama algoritmalarının diğer yöntemlere göre hesaplama süresi ve çözüm kalitesi açısından genelde daha iyi sonuçlar verdiği bilinmekle birlikte araştırmacılar tarafından gerçeğe dönüştürülecek olan genetik algoritma ve karınca kolonisi optimizasyonu yöntemlerinden oluşturulacak olan melez yapılar ürettikleri sonuçlarla tabu arama algoritmasını yakalayacaktır. Şekil 4.9'da araç rotalama problemleri için geliştirilen sezgisel ve metasezgisel yöntemlerin çözüm kaliteleri ve hesaplama süreleri arasındaki ilişki gösterilmektedir.



Şekil 4.9 Araç rotalama problemleri için geliştirilen sezgisel yöntemler

Şekil 4.9'dan da anlaşılacağı üzere klasik sezgisellere göre metasezgiseller işlemlerin gerçekleştirilmesi için daha fazla zamana ihtiyaç duymalarına rağmen daha iyi sonuçlar elde ederler. Genellikle klasik sezgiseller en iyi çözüm değerinin % 2-10 arasında sapma ile çözüm verirken, en iyi metasezgisel yaklaşımlar ise % 0.5'ten daha düşük sapmalarla problemlere çözüm üretirler.

BÖLÜM 5

ÇOK DEPOLU ARAÇ ROTALAMA PROBLEMİNİN MODELLENMESİNDE VE ÇÖZÜMÜNDE GENETİK ALGORİTMA VE KARINCA KOLONİSİ OPTİMİZASYONU KULLANIMI

Bu bölümde çok depolu araç rotalama probleminin modellenmesi ve çözümü için genetik algoritma ile karınca kolonisi optimizasyonundan oluşturulacak melez bir yapı ortaya konmuştur.

Çalışmada sunulan model literatürde var olan problem setleri ile test edilmiş ve bulunan sonuçlar literatürde bilinen en iyi sonuçlarla karşılaştırılmıştır.

Uygulanan çözüm yönteminde, kullanılan genetik algoritmanın popülasyon büyüklüğü, çaprazlama olasılığı, mutasyon olasılığı ve kuşak aralığı kontrol parametreleri ile karınca kolonisi optimizasyonunun feromon değerleri ve mesafeye bağlı seçilebilirlik parametre değerleri üzerinde yapılan değişiklikler ile elde edilen sonuçlar üzerinde gerçekleşebilecek iyileştirmeler araştırılmıştır.

5.1 Problemin Tanımı

Araç rotalama problemleri kombinasyonel optimizasyon problemlerinin en önemli örneklerinden birisidir. Lojistik alanında önemli bir yer tutan bu problem türü belirli bir talebe sahip müşterileri kapsar. Temel araç rotalama probleminin yapısında tek bir depodan araçlar ayrılmakta ve müşteri taleplerini karşılayan bu araçlar tekrar depoya dönmektedirler. Her araç daha fazla yüklenemeyeceği belirli bir kapasite kısıtına sahiptir ve her bir müşterinin talebi yalnızca bir araç tarafından karşılanabilmektedir.

Araç rotalama probleminin türlerinden biri olan çok depolu araç rotalama probleminde ise araçların kullandığı birden fazla depo söz konusudur. Çok depolu araç rotalama problemleri yapısı gereği bulundurduğu birden fazla sayıdaki depo ile müşterilerine daha kısa sürede ve daha iyi hizmet sunabilmektedir. Bu nedenle, gerçek yaşam problemlerinin simülasyonuna çok depolu araç rotalama problem türü daha uygundur.

Çok depolu araç rotalama problemleri, gerçek yaşamda bir süpermarket zincirinin bu zincirdeki marketlerine tedarik sağlayan depolar veya kargo şirketlerinin dünyanın dört bir yanına yayılmış alım ve teslimatlarının iletildiği depolar olarak karşımıza çıkabilirler. Çok depolu araç rotalama problemleri günlük yaşamımızda bizlerle bu kadar iç içe olmasına rağmen var olan literatür çalışmalarına bakıldığında bu araç rotalama problem türünün diğer araç rotalama problem türleri kadar araştırılmadığı ortaya çıkmaktadır.

Bu nedenle bu tez çalışmasının konusu olarak çok depolu araç rotalama problemleri ve daha önce bu konuda hakkında bir arada kullanılarak bir çalışma yapılmış olmayan genetik algoritma ile karınca kolonisi optimizasyonu tekniği iç içe kullanılarak melez bir yapı ortaya konmuştur.

Çok depolu araç rotalama probleminin çözümü için önerilen bu melez yapıda iki aşamalı bir yöntem önerilmiş ve literatürde var olan genetik kümeleme yöntemi çok depolu araç rotalama probleminin birinci aşamasında müşterilerin depolara gruplanması için revize edilerek kullanılmış ve problemin ikinci aşamasında ise rotalama işlemini gerçekleştirmek için karınca kolonisi optimizasyonu yöntemi kullanılmıştır.

5.2 Problem ile İlgili Veriler

Bu tez çalışması kapsamında ortaya konan genetik algoritma ve karınca kolonisi optimizasyonu yöntemlerinden oluşturulan melez yapı literatürde var olan problem setleriyle test edilmiştir. Bu problem setleri Christofides ve Eilon tarafından 1969 yılında literatüre kazandırılmış (Thangiah ve Salhi [184]) ve aynı problem setleri Cordeau ve diğerlerinin [146] da periyodik ve çok depolu araç rotalama problemleri için ortaya koydukları tabu arama sezgiseli ile ilgili çalışmada kullanıldığı gibi yine

literatürdeki birçok önerilen yöntemin test edilmesinde ve bu yöntemlerin gerçekliklerinin sınanmasında kullanılmıştır. Literatürde yer alan bu problem setlerinin özellikleri aşağıda verildiği gibidir:

- Problem setlerinde, sayıları 50 ile 360 arasında değişen müşteriler vardır.
- Problem setlerinde, sayıları 2 ile 9 arasında değişen depolar vardır.
- Problem setlerinde, araç kapasite kısıtı vardır ve araçların kapasiteleri 80 ile 500 arasında değişir.

Tez çalışması içerisinde çok depolu araç rotalama probleminin çözümü için önerilen yöntemin test edilmesi için 23 farklı problem seti kullanılmıştır. Bu problem setlerinin özellikleri özet olarak Çizelge 5.1'de verilmiştir.

Çizelge 5.1 Problem setlerinin özellikleri [185]

Problemin adı	Müşteri sayısı	Depo sayısı	Araç kapasitesi
P01	50	4	80
P02	50	4	160
P03	75	5	140
P04	100	2	100
P05	100	2	200
P06	100	3	100
P07	100	4	100
P08	249	2	500
P09	249	2	500
P10	249	4	500
P11	249	5	500
P12	80	2	60
P13	80	2	60
P14	80	2	60
P15	160	4	60
P16	160	4	60
P17	160	4	60
P18	240	6	60
P19	240	6	60
P20	240	6	60
P21	360	9	60
P22	360	9	60
P23	360	9	60

5.3 Algoritmanın Temel Yapısı

Önerilen çözüm yaklaşımının uygulandığı problemde müşterilere hizmet sunan birden fazla sayıda depo söz konusudur. Problemin çözümü için amaç fonksiyonu her deponun kullanılarak tüm müşterilere sunulacak olan hizmetin minimum toplam seyahat mesafesi ile yapılabilmesidir. Burada, her müşteri sadece bir depodan hizmet alabilmekte ve yine her bir müşteri sadece bir araç tarafından ziyaret edilebilmektedir. Müşterilerin sahip oldukları talepler bir kerede karşılanır yani müşteri taleplerinin bölünmesi ve parça parça karşılanması mümkün değildir. Her araç bir depodan seyahatine başlar ve seyahati sonunda müşterilere hizmet ilettikten sonra yine seyahatine başladığı depoya döner. Bir araç birden fazla depodan yükleme yapamaz, sadece bir depodan mal alabilir. Depolarda müşteri taleplerini fazlasıyla karşılayabilecek kadar mal bulunur. Her bir müşterinin mal talep miktarı belirlidir ve birbirinden farklıdır. Müşterilere hizmet götüren araçların kapasiteleri de belirlidir, önceden bilinir ve problem tiplerine göre farklılık gösterir. Araçlar sahip oldukları bu araç kapasitelerinden daha fazla yüklenemezler. Bunların dışında araçların seyahatleri boyunca arızalanmadığı, mola vermediği ve araç yakıtlarının tükenmediği varsayılır.

Çok depolu araç rotalama problemlerinin çözümü için literatürde genel olarak iki aşamalı yaklaşımdan söz edilir. Bu iki aşamalı çözüm yaklaşımı ile gruplama ve rotalama işlemleri gerçekleştirilir problem ya önce grupla sonra rotala yaklaşımına göre veya önce rotala sonra grupla yaklaşımına göre çözülür.

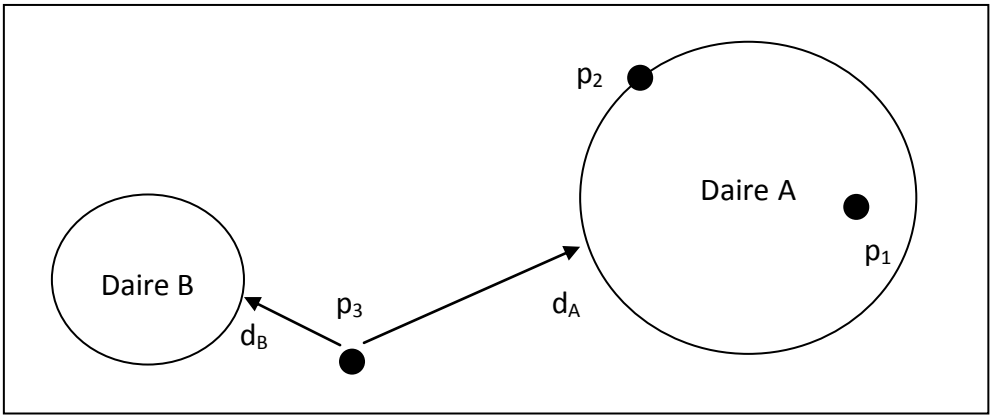
Önerilen çözüm yaklaşımı da iki aşamadan oluşmaktadır. Bu tez çalışması kapsamında çok depolu araç rotalama probleminin çözümü için önce grupla sonra rotala yaklaşımı kullanılmıştır. Çözüm yaklaşımının birinci aşamasında hangi müşterilerin hangi depolardan hizmet alacağının belirlendiği gruplama işlemi yapılırken, ikinci aşamada ise hangi depodan hizmet alacağı belirlenmiş olan müşterilerin o depo içinde rotalanması gerçekleştirilir. Bu iki aşama daha detaylı olarak aşağıda verildiği gibi açıklanabilir:

Birinci aşama (gruplama): Önerilen çözüm yaklaşımın ilk aşamasında gruplama işlemi yapılır ve gruplama ile hangi müşterilerin hangi depodan hizmet alacağı belirlenir. Müşterilerin depolara gruplanmasında kullanılabilecek birçok metot vardır. Bunlardan bir tanesi olan genetik kümeleme yöntemi Thangiah ve Salhi'nin [184] te ortaya

koydukları genetik kümeleme yönteminin çok depolu araç rotalama problemine uygulanmış halinin geliştirilmişidir.

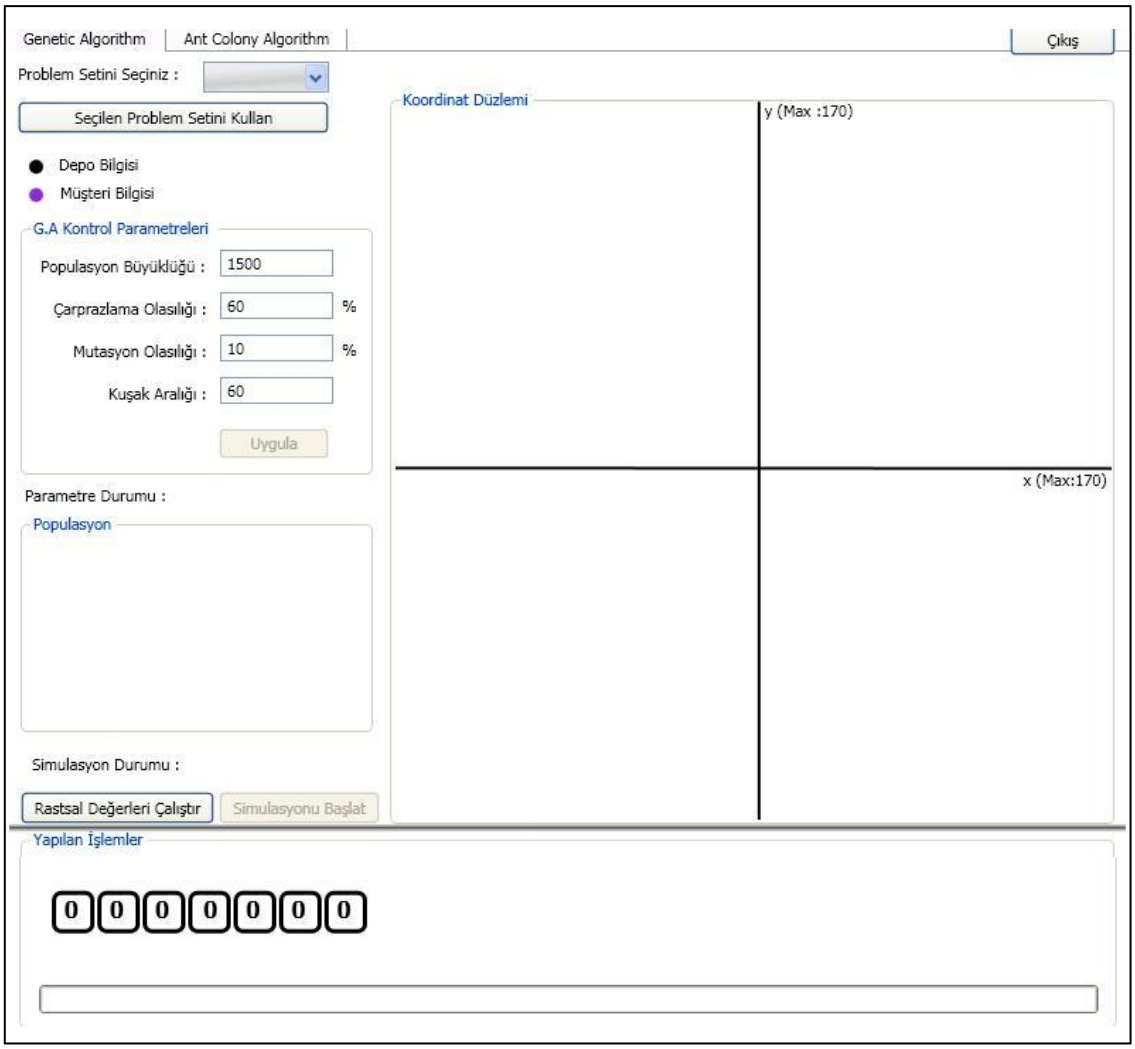
Thangiah ve Salhi'ye göre [184] te ilk rotaların özellikleri genetik algoritma kullanılarak kümeleme ile oluşturulur. Maliyet minimizasyonunun yapılmaya çalışıldığı amaç fonksiyonlu problem çözümünde kümeleme işlemi için geometrik şekiller kullanılır. Bu geometrik şekillerle elde edilen gruplar direkt olarak maliyet fonksiyonuna bağlanarak çok depolu araç rotalama problemine bir çözüm aranır.

Genetik kümeleme yönteminin revize edilmiş halinde ise müşterilerin gruplara ayrılması maliyet fonksiyonundan ayrılarak yapılır. Yöntemde koordinat düzlemindeki konumları belirli olan müşteriler ve depolar bu düzleme yerleştirilir. Thangiah ve Salhi [184] te önerilen yöntemde olduğu gibi depolardan rastlantısal olarak belirlenen çaplar ile müşterileri kapsayan ve birbirini kesmeyen daireler çizilir. Buna göre, bir müşteri ya bir dairenin içinde olabilir, ya bir dairenin üzerinde ya da dairenin dışında olabilir. Bu işlem örnek olarak Şekil 5.1'de gösterilmiştir. Şekil 5.1'de p_1, p_2 ve p_3 şeklinde ifade edilen üç tane müşteri ve *A* ve *B* şeklinde ifade edilen iki adet depo vardır. Üç müşterinin iki depoya dağıtımı için depo sayısı kadar yani iki tane daire çizilir. Bu daireler birbirini kesmez. Şekile göre, p_1 müşterisi *A* deposundan hizmet alır çünkü *A* dairesinin içindedir; p_2 müşterisi de yine *A* deposundan hizmet alır çünkü *A* dairesinin üzerindedir. p_3 müşterisinin ise hangi depodan hizmet göreceği bilinmemektedir. Çünkü müşteri ne *A* dairesinin ne de *B* dairesinin içerisindedir. Bu noktada p_3 müşterisinin hangi depodan hizmet alacağının belirlenmesi için müşterinin her iki dairenin çevrelerine olan uzaklıkları ölçülür ve p_3 müşterisi hangi daireye daha yakınsa bu müşteri o depoya atanır. Yani p_3 müşterisi ile *A* dairesinin kenarı arasındaki mesafe d_A ve *B* dairesinin kenarı arasındaki mesafe d_B ise, p_3 müşterisi $d_A \leq d_B$ olduğunda *A* deposuna, $d_B < d_A$ olduğunda ise *B* deposuna atanır.



Şekil 5.1 p_1, p_2 ve p_3 müşterilerinin A ve B daireleri ile olan ilişkisi

Önerilen yöntemde müşterilerin depolara gruplanması işlemi için genetik algoritma kullanılmıştır. Genetik kümeleme yönteminin ilk aşaması olan koordinat düzleminde rastlantısal dairelerin çizilmesi için Visual Studio.NET 2008, .NET Framework 3.5 ve C# programlarının sağlamış olduğu kütüphaneler ve The genetic algorithm optimization toolbox for MATLAB arayüzü birleştirilerek kullanılmıştır. Önerilen çözüm yönteminin genetik kümeleme için kullanılan arayüzü Şekil 5.2'de gösterilmiştir.



Şekil 5.2 Önerilen kullanıcı arayüzü

Müşterilerin depolara genetik kümeleme yöntemi ile gruplanmasının ardından genetik algoritma adımları işletilmeye başlanır. Müşteriler depolara gruplanır. Depo sayısına k dersek 2^k adet depoların sıralanma kombinasyonu elde edilir ve böylelikle popülasyon havuzu oluşturulur. Her bir popülasyon için amaç fonksiyonu olan ve toplam seyahat mesafesinin minimizasyonunun arandığı uygunluk değeri hesaplanır. Uygunluk değerlerine göre belirlenen bireyler arasında sırasıyla genetik algoritma operatörleri çalıştırılır. İkili (binary) şekilde kodlanan popülasyondan rulet tekeri seçimi ile çaprazlamaya sokulacak bireyler belirlenir ve popülasyona çaprazlama operatörü uygulanır. Çaprazlama işlemi, dairesel çaprazlama ile yapılır. Çaprazlama işleminin ardından mutasyon operatörü çalıştırılır. Mutasyon işlemi, komşu iki geni değiştirme ile yapılır. Mutasyon işleminin ardından yeniden uygunluk değeri hesaplanır. Eğer genetik

algoritmanın durma kuralına yani üretilen maksimum nesil sayısına ulaşılırsa elde edilen küme rotalama işlemi için karınca kolonisi optimizasyonu yöntemine verilir.

İkinci aşama (rotalama): Önerilen çözüm yönteminin ikinci aşaması rotalama aşamasıdır. Hangi müşterinin hangi depodan hizmet alacağı belirlendikten sonra müşterilerin hangi sıra ile hizmet alacakları belirlenir yani rotalar çizilir. Bu aşamada karınca kolonisi optimizasyonu yöntemi kullanılır. Karınca kolonisi optimizasyonu ile karıncaların çözüm aradıkları grupları oluşturan ve daha basit bir şekilde çözüme ulaşılmasını sağlayan aday listesi oluşturulur. Aday listesi bir karıncanın ziyaret edeceği müşterileri belirten bir listedir. Aday listesi oluşturulurken en yakın komşu sezgiseli kullanılır ve araç kapasiteleri göz önüne alınır. Araçların kapasitelerini geçmeyecek şekilde müşteri taleplerini hesaplayarak depoya en yakın olan müşteriden başlanarak aday listeleri oluşturulur. Her aday listesi mutlaka bir depo içerir ve listeler araçların güzergahına uygun olarak bir depodan başlar ve yine aynı depoda sona erer.

Aday listelerinin oluşturulmasının ardından karınca koloni sistemi kullanılarak rotalama işlemi gerçekleştirilir. Karınca koloni sisteminde müşteri sayısı kadar karınca depoda bulundurulur. Tabu listesi olarak, her bir karıncanın ziyaret ettiği şehirleri gösteren bir yasaklar listesi oluşturulur. Bu listede yer alan müşterileri karınca tekrar ziyaret edemeyeceğinden, karıncaların aynı müşteriyi birden fazla ziyaret etmeleri bu sayede engellenmiş olur. Karıncanın tabu listesi başlangıçta boştur.

Müşteri sayısı kadar karıncanın depoda konumlandırılmasının ardından aday listesindeki müşterilerin birbirleriyle ve depo ile aralarındaki tüm bağlantı yollarına başlangıç feromon değeri atanır. Başlangıç feromon değeri, depodan başlanarak aday listesindeki tüm müşterilerin en yakın komşu müşterisi şeklinde belirlenen tur uzunluğundan $\tau_0 = \left(n * L_{nn}\right)^{-1}$ formülü göz önüne alınarak hesaplanır. Formül ile elde edilen başlangıç feromon değeri tüm şehirler ve şehirlerle depo arasındaki yollara yapıştırılır. Başlangıç feromon değerinin belirlenmesinden sonra depodaki karıncalar rastlantısal olarak şehirlere yerleştirilir.

Karınca koloni sisteminde şehirlerde bulunan karıncalar gidecekleri bir sonraki müşteriyi seçmek için sözde-rastlantısal-orantılı-durum kuralını kullanırlar. Önerilen çözüm yönteminde bu kural,

$$s = \begin{cases} \arg \max_{(ij) \notin tabu_k} \left\{ \tau_{ij}^{\alpha} * \eta_{ij}^{\beta} \right\} & \text{eğer } q \le q_0 \\ P_{ij}^{k} & \text{aksi halde} \end{cases} \quad (3.20)$$

$$a_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in \mathrm{N}_i} \left[\tau_{il}(t)\right]\left[\eta_{il}\right]^{\beta}} \qquad \forall j \in \mathrm{N}_i \quad (3.21)$$

denklemleri ile işletilir. Bu kurala göre, üretilen rastlantısal sayı q değeri, q_0'dan büyük olduğunda karınca yeni çözümler arar, üretilen rastlantısal sayı q_0'dan küçük olduğunda ise karınca sahip olduğu bilgiler ile bir sonraki müşteriyi seçer. Karınca bir sonraki müşteriye geldiğinde, ziyaret ettiği bir önceki müşteri aday listesinden çıkarılır ve aynı şehrin tekrar ziyaret edilmesini engelleyen yasaklı şehirlerin bulunduğu tabu listesine eklenir. Karınca bundan sonra yoluna yukarıda anlatıldığı şekilde devam eder ve tüm müşteriler ziyaret edilene yani tabu listesi başlangıçtaki aday listesine eşit oluncaya kadar işlemler tekrarlanır. İşlemler bittiğinde aday listesi boş küme olur.

Karınca tüm müşterileri dolaştıktan ve tabu listesini doldurduktan sonra seyahatinde kullandığı en kısa yolların üzerine bir sonraki iterasyonda kullanılmak üzere gerekli koku maddesini yani feromonu bırakır. Çalışmada feromon güncellemesi her karınca turunu tamamladıktan sonra global olarak yapılır. Global feromon güncellemesinde, karıncaların gerçekleştirdikleri turlar uzunluklarına göre sıralanır ve sadece en kısa turu yapan karıncanın geçtiği yollardaki bağlantılar üzerinde feromon güncellemesi yapılır. Global feromon güncellemesi

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t+1) \quad (3.25)$$

$$\Delta\tau_{ij}(t+) = \begin{cases} 1/L_{best}(t+1) & (i,j) \text{ en iyi tura ait ise} \\ 0 & \text{aksi halde} \end{cases} \quad (3.7)$$

denklemlerine göre gerçekleştirilir. Yapılan global feromon güncellemesi ile karıncalar gerçekleştirilecek bir sonraki iterasyon için yeniden başlangıç noktasına yani depoya getirilirler. Karıncalar durma kuralı olan ve başlangıçta belirlenen maksimum iterasyon sayısına ulaşana kadar tüm açıklanan işlemleri tekrar ederler. Tüm bu işlemlerin

tamamlanmasının ardından her aday listesi ve gruplar için sonuç araç rotaları elde edilir.

Geliştirilen çözümde genetik algoritma ve karınca kolonisi optimizasyonu yöntemleri için kullanılan parametrelerin başlangıç değerleri Çizelge 5.2'de verilmiştir.

Çizelge 5.2 Başlangıç parametre değerleri

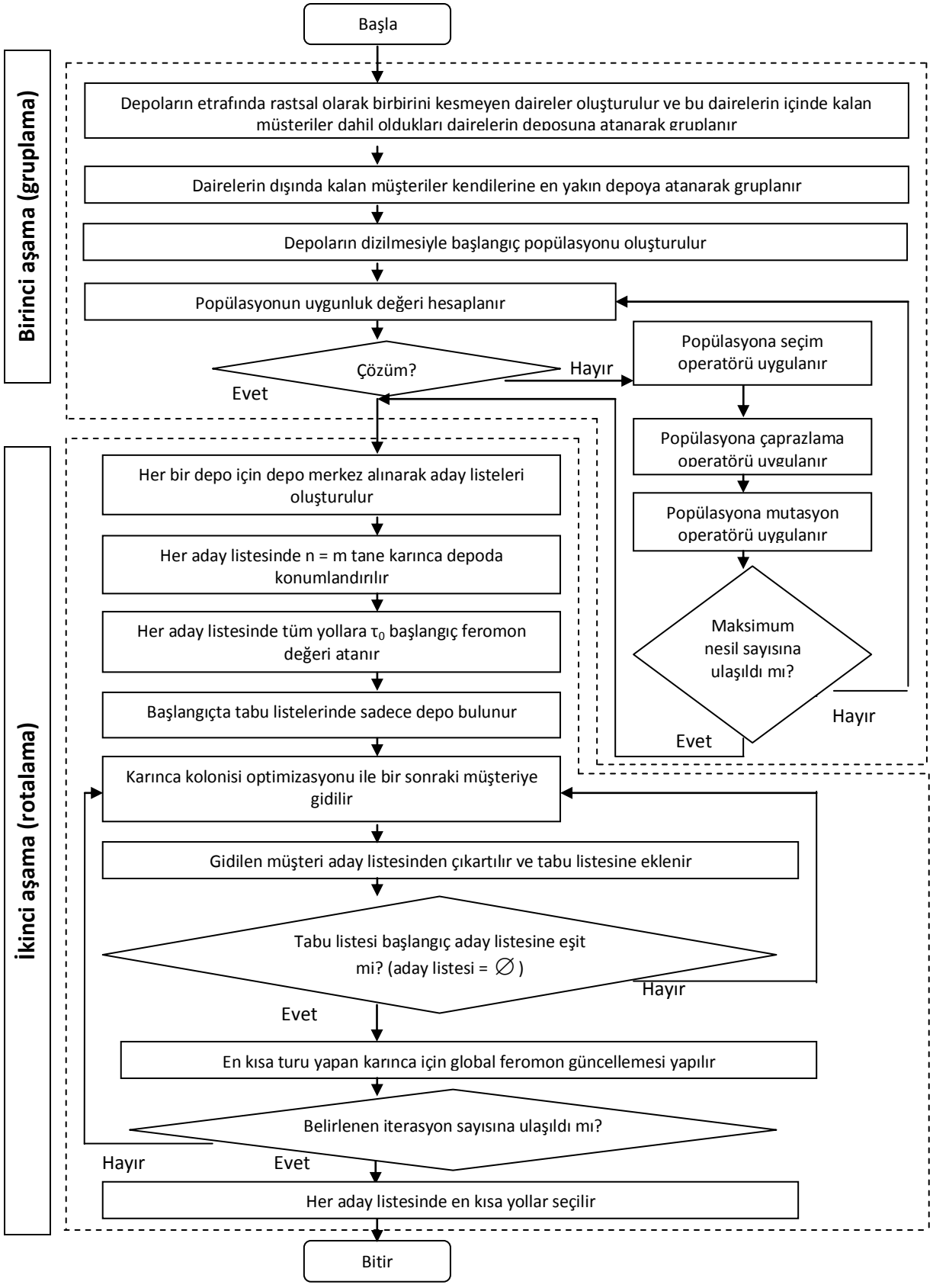
Popülasyon büyüklüğü (genetik algoritma)	1500
İterasyon sayısı (karınca kolonisi optimizasyonu)	2000
Çaprazlama oranı	%60
Mutasyon oranı	%10
Kuşak aralığı	60
q_0	0.9
α	1
β	5
ρ	0.1

5.4 Algoritmanın Akış Süreci

Çok depolu araç rotalama probleminin çözümü için önerilen genetik algoritma ve karınca kolonisi optimizasyonu yaklaşımlarının bir araya getirilmesiyle oluşturulmuş olan melez yapının özet kodu Şekil 5.3'de verilmiştir. Bu kod göz önüne alınarak oluşturulan algoritmanın akış süreci ise Şekil 5.4'de verilmiştir. Müşterilerin gruplama işleminin genetik algoritma ile, rotalama işleminin ise karınca kolonisi optimizasyonu ile yapıldığı bu yöntemin çözümü için bilgisayar yardımı alınmış ve önerilen çözüm C# programlama dili ile yazılmış, tüm veriler SQL programında muhafaza edilmiştir. Problem, çözüm için Intel® Core™ i7 extreme edition dört çekirdekli, 1 GB® ayrılmış bellekle NVIDIA Quadro FX3800M grafik kartı, 8 GB1333 MHz çift kanallı DDR3 bellek ve 500 GB 7200 RPM harddiske sahip olan bilgisayarda çalıştırılmıştır.

BEGIN
BÖLÜM 1.
P(0) dairelerini yapılandırmada rassal değerler tarafından başlangıç popülasyonu üret
Kullanıcı olarak parametre ayarla
"Popülasyon büyüklüğü" parametresini ayarla
"Çaprazlama oranı" parametresini ayarla
"Mutasyon oranı" parametresini ayarla
"Kuşak aralığı" parametresini ayarla
repeat
Kromozomları uygunluk fonksiyonu ile değerlendir
İki kromozoma seçim operatörü uygula
"Rulet tekeri seçim operatörü" uygula
Çaprazlama operatörü uygula
"Dairesel çaprazlama operatörü" uygula
Mutasyon operatörü uygula
"Komşu iki gen değiştirme mutasyon operatörü" uygula
Yavruları değerlendir
if
Yavru *P* popülasyonundaki en kötü kromozomdan daha iyiyse
then
En kötü kromozomu yavru ile değiştir
end if
Jenerasyon = Jenerasyon + l
until
"Üretilen maksimum jenerasyon sayısı" durma kuralına ulaşana kadar
En iyi kromozomu sonuç olarak seç ve 2. bölüme geç
BÖLÜM 2.
Kullanıcı olarak parametre ayarla
"İterasyon sayısı" parametresini ayarla,
"q_0" parametresini ayarla, "α" parametresini ayarla, "β" parametresini ayarla
"En yakın komşu metodu" ile aday listelerini belirle
repeat
Aday listesindeki müşteri sayısı kadar karıncayı depoda konumlandır
Başlangıç feromon değerini hesapla ($\tau_0 = \left(n * L_{nn}\right)^{-1}$), karıncaları rastsal olarak ata
repeat
Geçiş kuralını uygula
i = 0 ***to*** *n* ve *j = 0* ***to*** *m* için
Bir sonraki müşteriyi Denklem (3.20) ve Denklem (3.21)'ya göre seç
Seçilen müşteriye karıncayı gönder
Karınca tarafından gidilen şehri tabu-yasaklar listesine ekle
until
Başlangıç aday listesi = tabu listesi
Global feromon güncellemesi yap
k = 1 **to** m için $L_k(t)$ ve L_{best} hesapla
Feromonu Denklem (3.25) ve Denklem (3.7)'ye göre güncelle
until
"Maksimum iterasyon sayısı" durma kuralına ulaşana kadar
END.

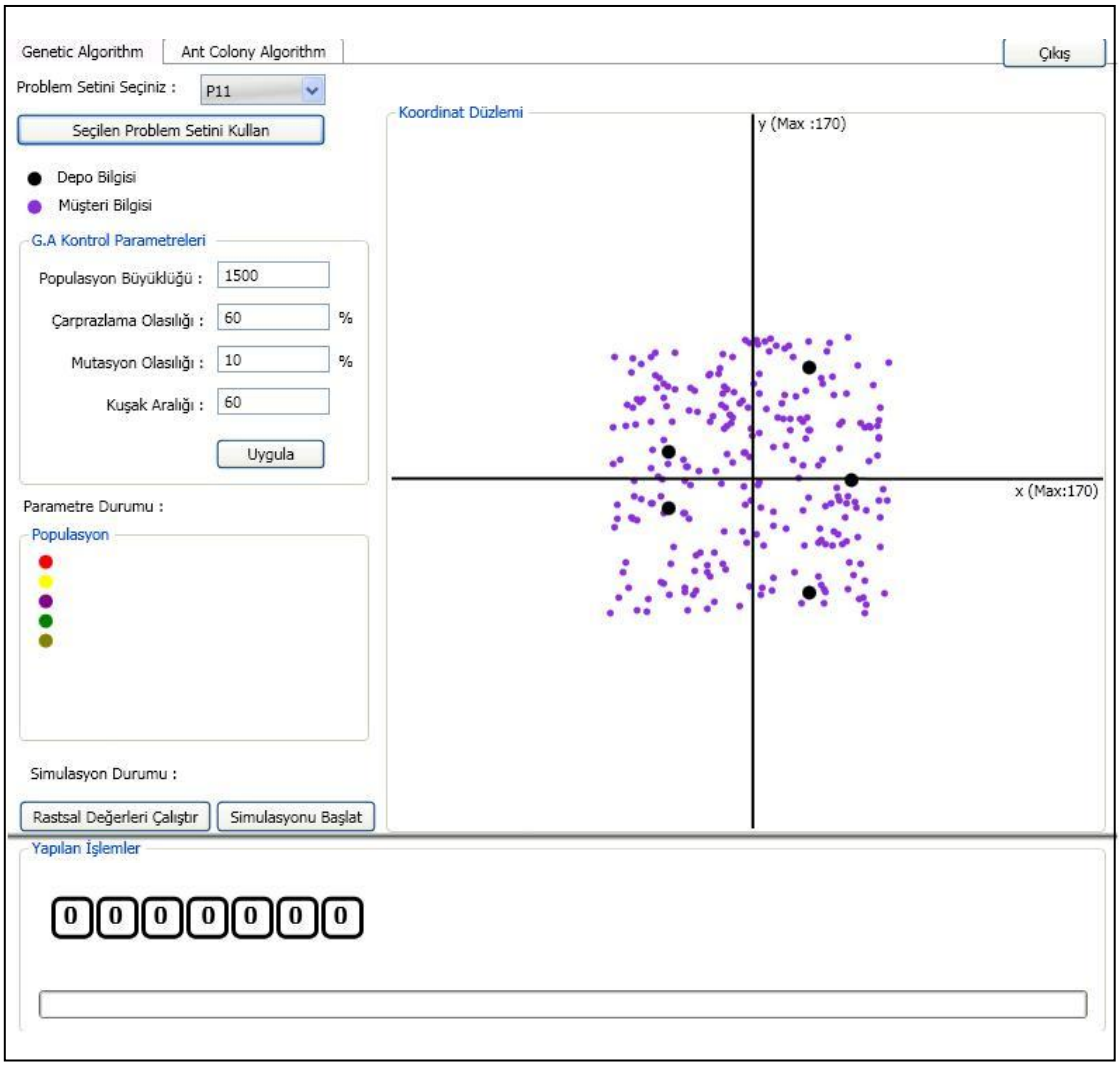
Şekil 5.3 Önerilen çözüm yaklaşımı için pseudo code



Şekil 5.4 Algoritmanın akış şeması

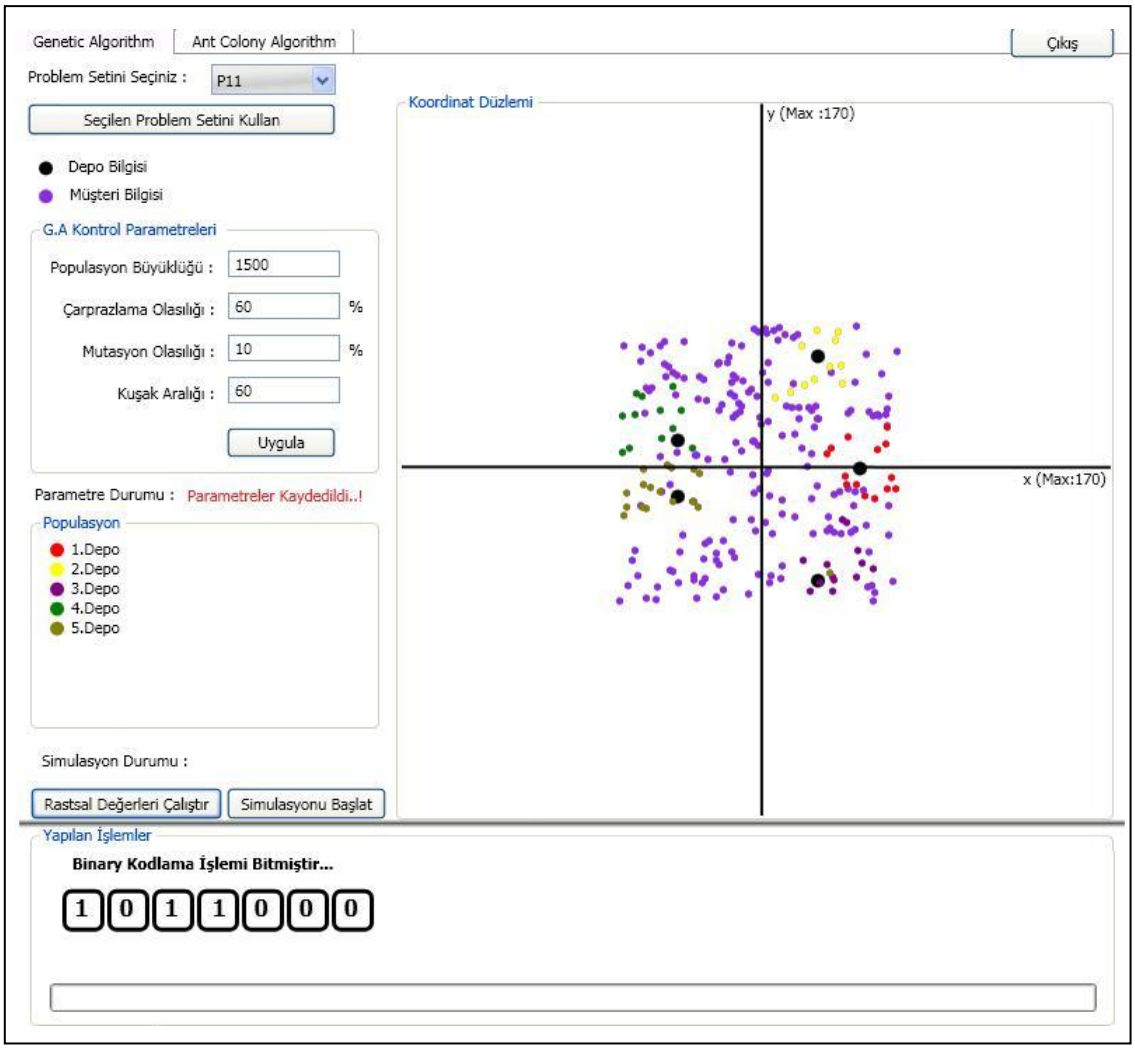
Geliştirilen yöntemi çözmek için oluşturulan programın daha kolay anlaşılması açısından kullanıcı arayüzü oluşturulmuştur. Bu arayüze yerleştirilen komutlar ile seçilen problem setine göre müşteriler ve depolar koordinatlarıyla ekrana gelmekte ve işlem adımları ekrana yansıtılmaktadır. Ayrıca arayüz üzerindeki pencereler sayesinde de programda kullanılan algoritmalara ait parametrelerin değiştirilmesi ve bu parametrelerin problemin çözümünü ne şekilde etkileyebileceğinin izlenmesi sağlanmaktadır.

Şekil 5.5'de 11. problem seti için genetik algoritma çalışmasının başlangıç durumu yani seçilen problem setinin depo ve müşteri bilgileri ile koordinatlar dahilinde ekrana yüklenmesi gösterilmiştir.



Şekil 5.5 Kullanıcı arayüzü ile verilerin ekrana yüklenmesi

Seçilen problem için depo ve müşteri koordinat bilgileri ekrana yüklendikten sonra, önerilen geometrik şekil temelli genetik kümeleme metodunun ilk adımı olan rastlantısal çap uzunluklarına sahip dairelerin çizimi gerçekleştirilir. Örnek olarak alınan 11. problem seti için beş ayrı depodan çizilen rastlantısal daireler Şekil 5.6'da gösterilmiştir.



Şekil 5.6 Kullanıcı arayüzü ile rastlantısal çap uzunluklu daire çizim işleminin ekran görüntüsü

Rassal çap uzunluklarına sahip dairelerin çizilmesinin ardından genetik kümeleme işlemini gerçekleştirecek olan tüm genetik algoritma operatörleri çalıştırılır. Geliştirilen genetik kümeleme çözüm yöntemi ile çok depolu araç rotalama problemi için rastsal olarak çizilen daireler ve bu daireler esas alınarak yapılan genetik algoritma işlemleri sonucu oluşturulan kümeler arası müşteri dağılımı rapor edilebilmektedir. Şekil 5.7'de 11. problem seti için tüm genetik algoritma işlemlerinin gerçekleştirilmesi sonucu elde edilen son müşteri dağılımı gösterilmiştir.

Şekil 5.7 Kullanıcı arayüzü ile birinci aşama(gruplama) çözüm sonucunun gösterimi

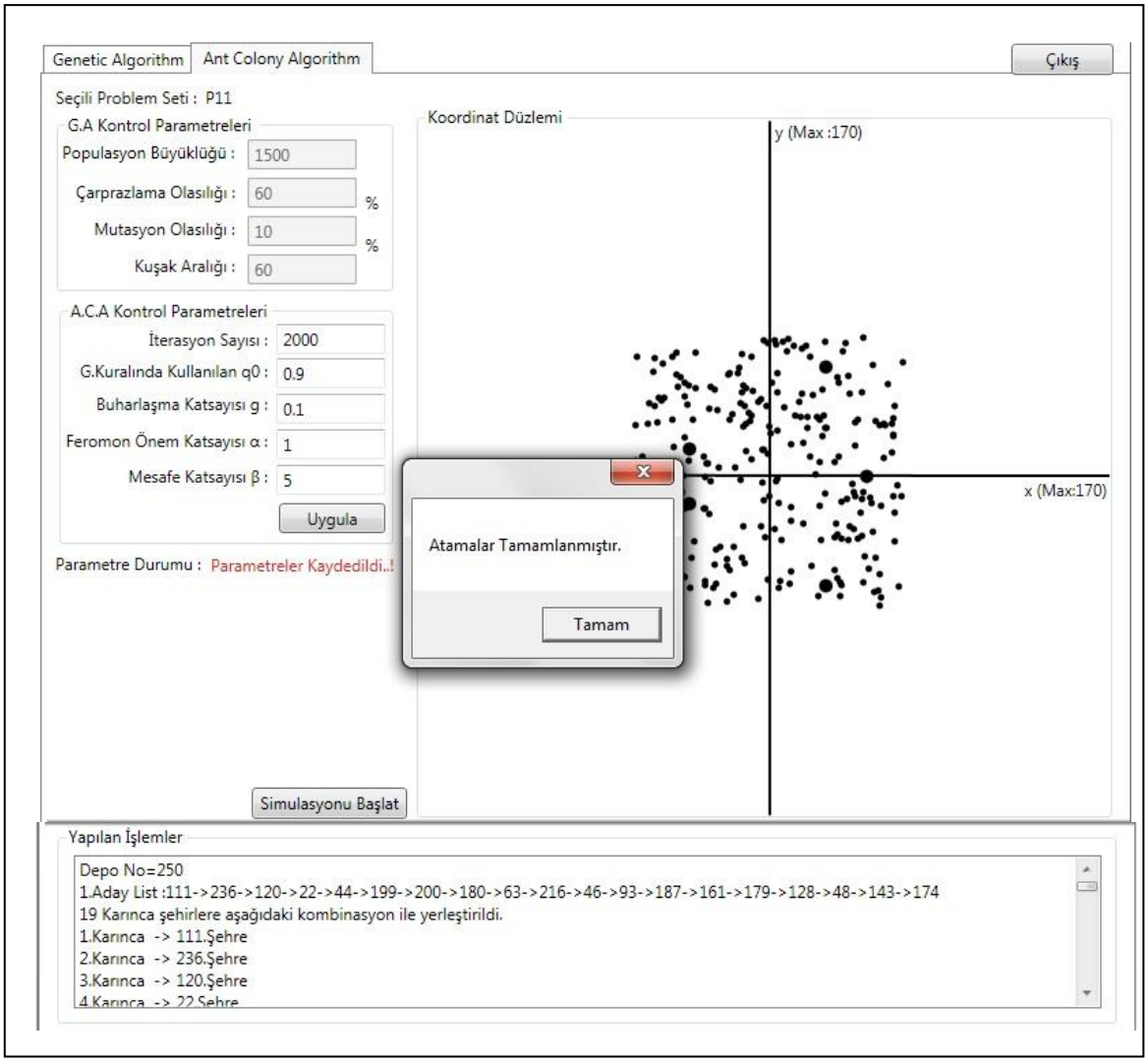
Geliştirilen kullanıcı arayüzü ile hangi müşterilerin hangi depolara gruplanacağı belirlendikten sonra genetik algoritma işlemleri tamamlanıp karınca kolonisi optimizasyonuna geçilir. Arayüzün bu kısımında da her bir grup farklı renklerle ifade edilerek hangi müşterilerin hangi depodan hizmet alacağı görsel olarak kullanıcının hizmetine sunulur. Karınca kolonisi sisteminin çalışma adımları da yine ekranda gösterilir.

İşlemlerin sonucunda program çıktı olarak karıncalar tarafından kat edilen toplam mesafeyi ve bu mesafeyi gerçekleştirdikleri rotayı verir.

Ekranın sol tarafında bulunan parametre pencerelerinin değiştirilmesi ile de mevcut çözüm üzerinde değişiklikler yapılarak bu değişikliklerin çözüm üzerine olan etkisi incelenebilir. Buna göre 11. problem seti için Şekil 5.8'de genetik algoritma (birinci aşama) sonucu elde edilen verilerin karınca kolonisi optimizasyonu (ikinci aşama)

arayüzüne atanması ve Şekil 5.9'da da aktarılan bu verilerle yapılan karınca kolonisi optimizasyonu işlemleri ile elde edilen nihai sonuç gösterilmiştir.

Şekil 5.8 Genetik algoritma (birinci aşama) sonucu elde edilen verilerin karınca kolonisi optimizasyonu (ikinci aşama) arayüzüne atanması



Şekil 5.9 Önerilen genetik algoritma ve karınca kolonisi optimizasyonu algoritma yapısının çalıştırılması ile nihai sonucun elde edilmesi

5.5 Hesaplama Bulguları

Tez çalışması kapsamında çok depolu araç rotalama problemlerinin çözümü için önerilen iki aşamalı çözüm yönteminin doğruluğunun testi için literatürde bulunan 23 problem seti kullanılmış, her iki aşama için karşılaştırma yapılan yöntem ve çalışmalar bu bölümde ele alınmış ve karşılaştırma değerleri çizelgeler halinde açıklanmıştır.

5.5.1 Elde Edilen Birinci Aşama Sonuçlarının Testi

Çok depolu araç rotalama problemlerinin çözümü için önerilen meta sezgisel yöntemin ilk aşaması olan müşterilerin hangi depolardan hizmet alacaklarının belirlendiği

gruplama işlemi genetik algoritma ile gerçekleştirilmiş ve bulunan sonuçlar en yakın komşu metodu ile karşılaştırılmıştır.

En Yakın Komşu Metodu ile Gruplama: En yakın komşu metodu ilk kez Karg ve Thompson tarafından ortaya atılmıştır ve tekli üretim yerleşimlerindeki sıralama işleri için en iyi metot olarak adlandırılmıştır (Monnot [186]). Gezgin satıcı probleminde kullanılan bu metotta, gezgin satıcı turuna bir şehirden başlar ve ardından seçeceği bir sonraki şehir için henüz ziyaret etmediği tüm şehirlerle olan mesafesine bakar, kendisine en yakın olan şehiri seçerek bir sonraki hareketinde bu şehire gider. Tüm şehirleri ziyaretinin ardından gezgin satıcı tekrar başladığı noktaya geri döner.

Bu tez çalışması kapsamında ele alınan en yakın komşu metodunda her bir problem seti için depolar ile müşteriler arasındaki tüm mesafeler hesaplanır ve müşteriler kendilerine en yakın olan depoya atanırlar. Depo ve müşteri olarak adlandırılan iki düğüm arasındaki mesafe öklid mesafe olarak alınır. İki düğüm arasındaki öklid mesafe Denklem (5.1)'de verildiği gibi düğümlerin koordinatlarına bağlı olarak hesaplanır.

$$A_{i,j} = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2} \tag{5.1}$$

En yakın komşu metodunun işlem adımları Şekil 5.10'da gösterilmiştir. Burada, *U* simgesi ile tanımlanan herhangi bir depoya atanmamış müşterilerin setidir.

BEGIN

{

compute $A_{l,I}$

$A_{l,i} = \sqrt{(x_i - x_l)^2 + (y_i - y_l)^2}$

$i \in U$, *i* herhangi bir depoya atanmamış şehir, $N \supset U$

repeat

{

if

{

min $\{d_i^l\}$ hesapla

then

i müşterisini *l* deposuna ata

}

end if

}

until

$U = \varnothing$ olana kadar

}

END.

Şekil 5.10 En yakın komşu metodu için işlem adımları

Önerilen Geometrik Şekil Temelli Genetik Kümeleme Metodu ile Gruplama: Tez çalışması kapsamında öne sürülen geometrik şekil temelli genetik kümeleme metodu ile gruplamada her bir problem setinde bulunan depolardan rassal çap uzunluklu daireler çizilir ve oluşturulan bu başlangıç çözümleri ile birlikte tüm genetik algoritma operatörleri çalıştırılarak gruplama sonuçları elde edilir.

5.5.1.1 Sonuçların Karşılaştırılması

Önerilen metot için depoların merkez alındığı rassal çap uzunluklu daireler çizilmiş ve genetik algoritma operatörlerinden popülasyon büyüklüğü 1500'e, çaprazlama oranı %60'a, mutasyon oranı %10'a ve son olarak da kuşak aralığı 60'a ayarlanarak işlemler gerçekleştirilmiştir. Çok depolu araç rotalama problemlerinin iki aşamalı çözümünün birinci aşaması olan müşterilerin hangi depolardan hizmet alacağının belirlendiği gruplama işlemi için önerilen geometrik şekil temelli genetik kümeleme metodu ile elde edilen sonuçlar Çizelge 5.3'ün 3. kolonunda gösterilmiştir.

Önerilen metot için yazılmış olan program üzerinde parametre etkinliğinin ve kabul edilen genetik operatör değerlerinin doğruluğunun testi için kullanıcı arayüzünde bulunan değerlerden çaprazlama oranı %3 olarak değiştirilmiş ve bu değişiklik sonucu elde edilen değerler de Çizelge 5.3'ün 4. kolonunda verilmiştir.

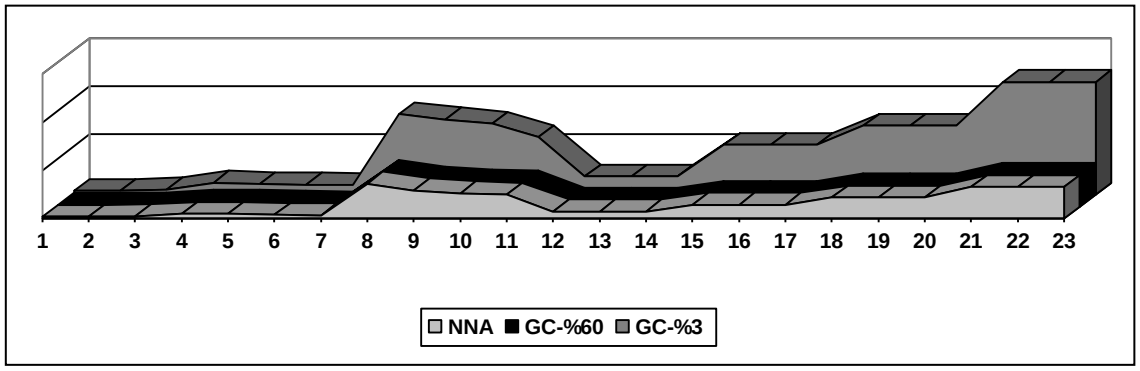
Önerilen metodun testi ise en yakın komşu metodu ile yapılmıştır. En yakın komşu metodu uygulanması ve adaptasyonu açısından kolay bir metot olmasına rağmen elle çözümünde yüksek işlem zamanları gerektirmektedir. Karşılaştırmanın yapılacağı 23 problem seti için en yakın komşu metodu ile elde edilen sonuçlar ise Çizelge 5.3'ün 2. kolonunda gösterilmiştir.

Çizelge 5.3 23 problem seti için en yakın komşu metodu ve önerilen genetik kümeleme metodu ile elde edilen toplam seyahat mesafesi değerlerinin karşılaştırılması

Problem	En yakın komşu metodu	Geometrik şekil temelli genetik kümeleme metodu (çaprazlama oranı %60)	Geometrik şekil temelli genetik kümeleme metodu (çaprazlama oranı %3)	Hata oranı (%)
01	708	708	1637	2.31
02	708	708	1637	2.31
03	904	904	2185	2.42
04	1926	1926	4911	2.55
05	1949	1949	4307	2.21
06	1500	1500	4095	2.73
07	1445	1445	3901	2.69
08	14472	14472	33286	2.30
09	11739	11739	31108	2.64
10	10434	10434	29319	2.81
11	9891	9891	23837	2.40
12	2924	2924	7456	2.54
13	2924	2924	7456	2.54
14	2924	2924	7456	2.54
15	5794	5794	20557	3.55
16	5794	5794	20557	3.55
17	5794	5794	20557	3.55
18	8691	8691	28680	3.29
19	8691	8691	28680	3.29
20	8691	8691	28680	3.29
21	13047	13047	46708	3.58
22	13047	13047	46708	3.58
23	13047	13047	46708	3.58

Çizelge 5.3'de görüldüğü gibi, önerilen genetik algoritmanın kullanıldığı gruplama metodu literatürden alınan 23 problem setinin tamamı için en yakın komşu metodu ile aynı sonuçları vermiştir. Aynı zamanda çizelgeden anlaşılacağı üzere genetik algoritmanın kontrol parametrelerinden biri olan çaprazlama oranında yapılan değişiklik de bulunan sonuçları olumsuz olarak etkilemekte ve %2.21 ile %3.58 arasında bir oranda hataya sebep olmaktadır. Bu hatayı ortadan kaldırmak için çaprazlama oranının %60 civarında seyretmesi gerektiği açıkça görülmektedir.

Tüm hesaplama ve karşılaştırmaların ardından farklı iki algoritma ve parametre değerleri ile elde edilen toplam seyahat mesafe uzunlukları Şekil 5.11'de gösterilmiştir.



Şekil 5.11 Birinci aşama sonuçlarının karşılaştırılması

5.5.2 Elde Edilen İkinci Aşama Sonuçları

Çok depolu araç rotalama problemlerinin çözümü için önerilen iki aşamalı algoritma yapısının birinci aşamasında gerçekleştirilen genetik kümelemenin ardından ikinci aşamada da karınca koloni sistemi yöntemi kullanılmış ve elde edilen toplam mesafe sonuçları Çizelge 5.4'de gösterilmiştir.

Çizelge 5.4 23 problem seti için kabul edilen başlangıç parametre değerlerine göre elde edilen sonuçlar

Problemin adı	Kabul edilen başlangıç parametre değerlerine göre elde edilen sonuçlar
P01	583.1
P02	483.4
P03	648.4
P04	1023.2
P05	761.6
P06	883.9
P07	904.6
P08	4507.8
P09	3976.9
P10	3727.7
P11	3677.9
P12	1302.2
P13	1302.6
P14	1351.5
P15	2518.8
P16	2561.8
P17	2721.2
P18	3793.3
P19	3825.4
P20	4101.9
P21	5692.8
P22	5732.4
P23	6160.7

5.5.2.1 Elde Edilen Sonuçların Bilinen En İyi Sonuçlar ile Karşılaştırılması

Çok depolu araç rotalama problemlerinin çözümü için önerilen genetik algoritma ve karınca kolonisi optimizasyonundan oluşan iki aşamalı programın sonuçları elde edilen sonuçların gerçekliğinin test edilmesi ve başarısının ortaya konması için literatürde var olan aynı konu üzerinde yapılan çalışma sonuçları ile karşılaştırılmış ve karşılaştırma sonuçları Çizelge 5.5'de, bu karşılaştırmanın grafik gösterimi ise Şekil 5.12'de verilmiştir.

Çizelge 5.5 Elde edilen sonuçların bilinen en iyi sonuçlar ile karşılaştırılması

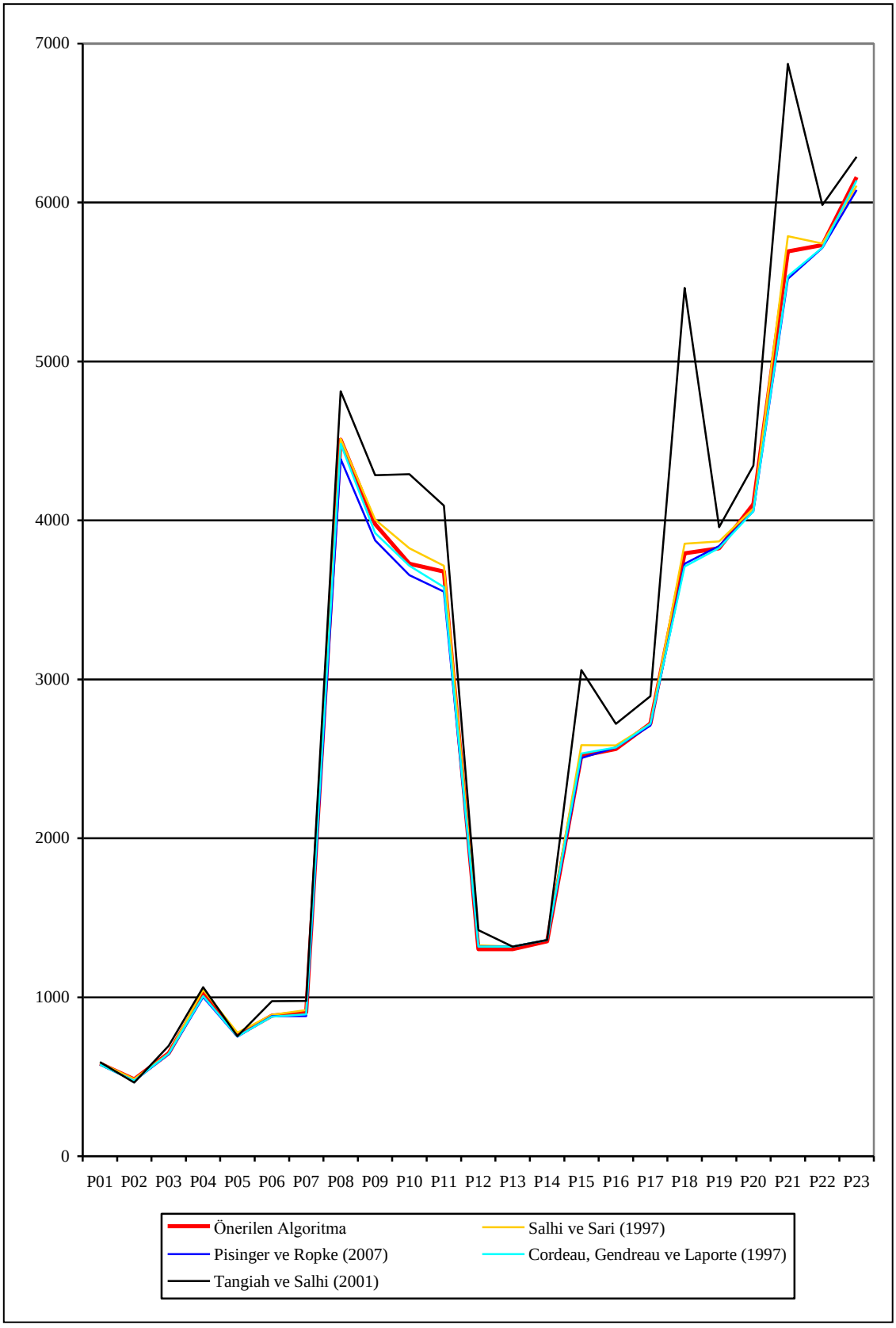
Problemin Adı	Önerilen algoritma sonuçları	SS (1997)	CGL (1997)	TS (2001)	PR (2007)	En iyi sonuçlar ile fark	Sapma (%)
P01	583.1	587.8	**576.9**	591.7	**576.9**	-6.2	(-)1.06
P02	483.4	484.6	473.9	**463.2**	473.5	-20.2	(-)4.17
P03	648.4	645.9	645.2	694.5	**641.2**	-7.2	(-)1.11
P04	1023.2	1047.9	1006.7	1062.4	**1001.6**	-21.6	(-)2.11
P05	761.6	777.2	753.3	754.8	**751.9**	-9.7	(-)1.27
P06	883.9	888.6	**877.8**	976.0	880.4	-6.1	(-)0.69
P07	904.6	918.9	891.9	976.5	**881.9**	-22.7	(-)2.50
P08	4507.8	4513.3	4482.4	4812.5	**4387.4**	-120.4	(-)2.67
P09	3976.9	4005.3	3920.9	4284.6	**3874.8**	-102.1	(-)2.56
P10	3727.7	3824.7	3714.7	4291.5	**3655.2**	-72.5	(-)1.94
P11	3677.9	3714.3	3580.8	4092.7	**3552.3**	-125.6	(-)3.41
P12	**1302.2**	1326.8	1318.9	1421.9	1318.9	+16.7	(+)1.28
P13	**1302.6**	1318.9	1318.9	1318.9	1318.9	+16.3	(+)1.25
P14	**1351.5**	1360.1	1360.1	1360.1	1360.1	+8.6	(+)0.63
P15	2518.8	2586.7	2534.1	3059.2	**2505.4**	-13.4	(-)0.53
P16	**2561.8**	2584.5	2572.2	2719.9	2572.2	+10.4	(+)0.40
P17	2721.2	2720.2	2720.2	2894.7	**2709.1**	-12.1	(-)0.44
P18	3793.3	3853.3	**3710.5**	5462.9	3727.6	-82.8	(-)2.18
P19	**3825.4**	3867.9	3827.1	3956.6	3839.4	+1.7	(+)0.04
P20	4101.9	4074.8	**4058.1**	4344.8	**4058.1**	-43.8	(-)1.06
P21	5692.8	5788.5	5535.9	6872.1	**5519.5**	-173.3	(-)3.04
P22	5732.4	5742.6	5716.0	5985.3	**5714.5**	-17.5	(-)0.30
P23	6160.7	6106.6	6139.7	6288.0	**6078.8**	-81.9	(-)1.32

SS: Salhi ve Sari (1997); CGL: Cordeau, Gendreau ve Laporte (1997); TS: Thangiah ve Salhi (2001); PR: Pisinger ve Ropke (2007).

Çizelge 5.5'de gösterilen sonuçlara göre önerilen algoritma ile literatürde var olan 23 problem setinin 5 tanesi için bilinen en iyi sonuçlardan daha iyi çözümler elde edilmiş ve önerilen algoritma ile elde edilen çözümlerin %0.30-%3.41 arasında bir oranda sapma gösterdiği belirlenmiştir. Önerilen algoritma ile daha iyi sonuçların elde edildiği problem setleri incelendiğinde bu problem setlerinin orta sayıda müşterisi bulunan ve araç kapasitesinin düşük olduğu problem grupları olduğu gözlenmiştir.

Önerilen algoritma ile elde edilen sonuçların karşılaştırıldığı çalışmalardan Salhi ve Sari [187] de eşzamanlı olarak müşterilerin depolara atanması, dağıtım rotalarının bulunması ve araçların filo bileşimlerinin sınırlandırılması problemlerinin çözülmesi için karmaşık tamsayılı lineer programlama yöntemi ile kurdukları formülasyonu çok-seviyeli bileşik arama sezgiseli ile çözmüşlerdir. Karşılaştırmanın yapıldığı ve dört problem seti için bilinen en iyi çözümleri veren Cordeau, Gendreau ve Laporte [146] da çalışmasında ise yazarlar periyodik araç rotalama problemi, periyodik gezgin satıcı

problemi ve çok depolu araç rotalama probleminin çözümünde tabu arama sezgiselini kullanmış ve çok depolu araç rotalama problemini periyodik araç rotalama probleminin özel bir vak'ası olarak ele alarak problemleri aynı metodoloji ile çözmüşlerdir. Literatürde var olan 23 problem setinden sadece 2.si için bilinen en iyi çözümü sağlayan Thangiah ve Salhi [184] te ise maliyet minimizasyonu amaçlı çok depolu araç rotalama problemlerinin çözümünde kümeleme işlemi için bu tez kapsamında önerilen algoritma yapısının birinci aşamasına da temel olma özelliğini taşıyan geometrik şekilleri kullanmış ve bu geometrik şekillerle elde edilen grupları direkt olarak maliyet fonksiyonuna bağlayarak çok depolu araç rotalama problemleri için çözüm elde etmişlerdir. Son olarak 2007 yılında yaptıkları çalışma ile 23 problem setinin 15 tanesi için bilinen en iyi çözümleri sunan Pisinger ve Ropke [188] de ise zaman pencereli araç rotalama problemi, kapasiteli araç rotalama problemi ve çok depolu araç rotalama probleminin çözümü için tüm problem çeşitlerini toplama ve dağıtım modeline çevirerek komşu aramanın genişletilmiş bir versiyonu olan uyarlanmış geniş komşu arama sezgiselini kullanmışlardır.



Şekil 5.12 23 problem seti için önerilen algoritma sonuçları ile literatürde var olan sonuçların karşılaştırılması

5.6 Bulunan Sonuçlar Üzerinde Yapılan İyileştirme Çalışmaları

Çok depolu araç rotalama problemlerinin çözümü için önerilen genetik algoritma ve karınca kolonisi optimizasyonu temelli algoritma yapısının çalıştırılması ile elde edilen sonuçlar algoritma parametre değerlerinin değiştirilmesi ile test edilebilir. Bu testler esnasında parametreler tek tek değiştirilebileceği gibi eşzamanlı olarak yapılacak birden fazla parametre değişimi ile de program çıktıları üzerinde iyileştirme çalışmaları gerçekleştirilebilinir.

5.6.1 Tek Parametre Değişimleri

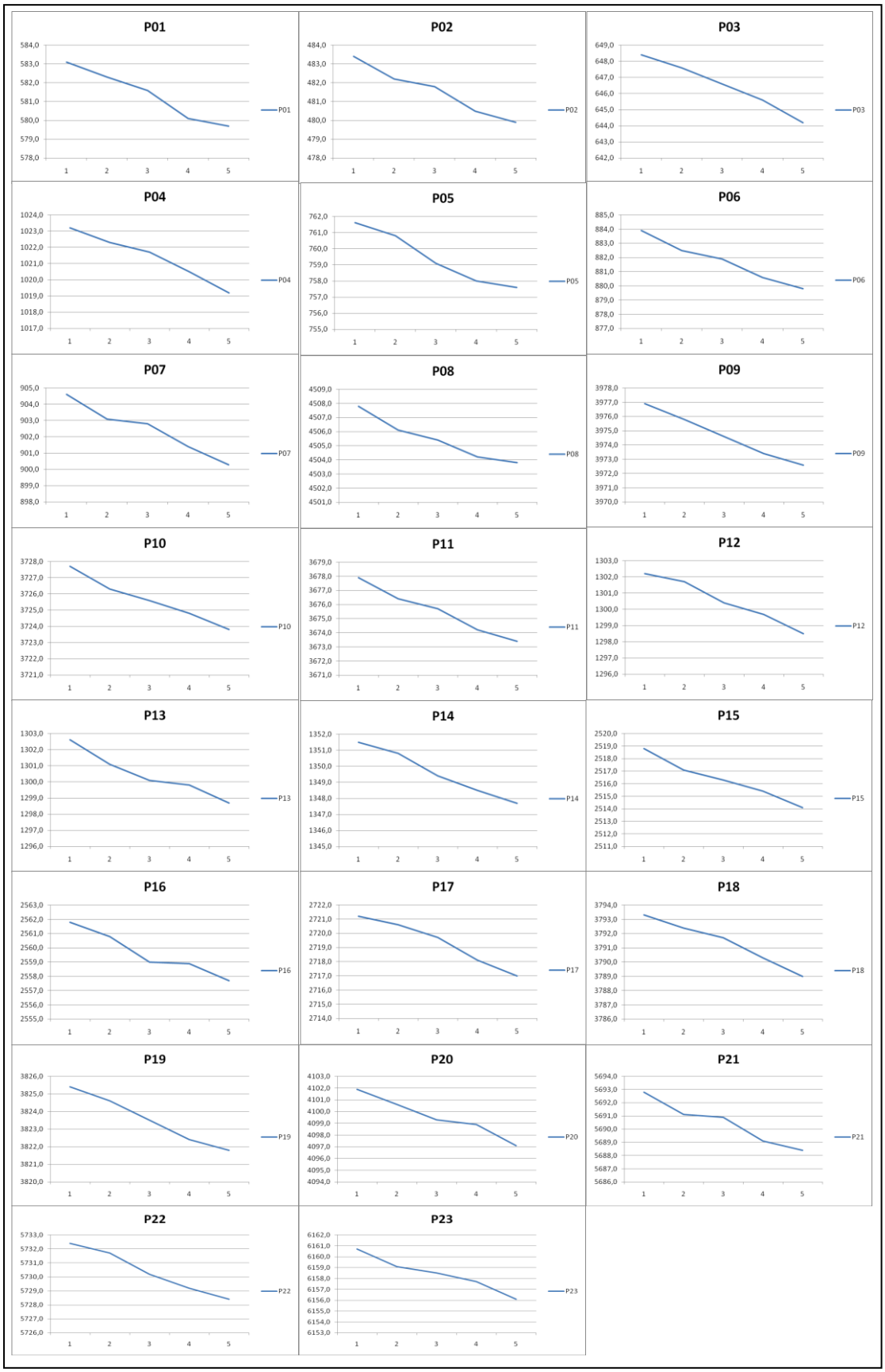
Bu doktora tezi kapsamında oluşturulan algoritma yapısının karınca kolonisi optimizasyonu kısımında yani çözümün ikinci aşamasında kullanılan α, β, ρ, q_0 ve iterasyon sayısı parametreleri üzerinde yapılan değişimler ve algoritma sonuçlarının bu değişimlere verdiği tepkiler bu bölüm altında incelenecektir.

5.6.1.1 α Parametresinin Değiştirilmesi

Karınca kolonisi optimizasyonunda yoldaki feromon miktarının önemini gösteren α parametresi baz alındığı 1 değerinden üst sınırı olan 5 değerine doğru yükseltilirken programın ürettiği sonuç çıktı değerlerinin küçüldüğü dolayısıyla α değerinin büyük seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu elde edilmiştir. α parametresi üzerinde yapılan değişikliklerle elde edilen sonuç değerleri Çizelge 5.6'da ve 23 problem seti üzerindeki sonuç değişiklik grafikleri ise Şekil 5.13'de gösterilmiştir.

Çizelge 5.6 α parametresinin değiştirilmesi ile elde edilen sonuçlar

α =	1	2	3	4	5
P01	*583.1*	582.3	581.6	580.1	579.7
P02	*483.4*	482.2	481.8	480.5	479.9
P03	*648.4*	647.6	646.6	645.6	644.2
P04	*1023.2*	1022.3	1021.7	1020.5	1019.2
P05	*761.6*	760.8	759.1	758.0	757.6
P06	*883.9*	882.5	881.9	880.6	879.8
P07	*904.6*	903.1	902.8	901.4	900.3
P08	*4507.8*	4506.1	4505.4	4504.2	4503.8
P09	*3976.9*	3975.8	3974.6	3973.4	3972.6
P10	*3727.7*	3726.3	3725.6	3724.8	3723.8
P11	*3677.9*	3676.4	3675.7	3674.2	3673.4
P12	*1302.2*	1301.7	1300.4	1299.7	1298.5
P13	*1302.6*	3101.1	1300.1	1299.8	1298.7
P14	*1351.5*	1350.8	1349.4	1348.5	1347.7
P15	*2518.8*	2517.1	2516.3	2515.4	2514.1
P16	*2561.8*	2560.8	2559.0	2558.9	2557.7
P17	*2721.2*	2720.6	2719.7	2718.1	2717.0
P18	*3793.3*	3792.4	3791.7	3790.3	3789.0
P19	*3825.4*	3824.6	3823.5	3822.4	3821.8
P20	*4101.9*	4100.6	4099.3	4098.9	4097.1
P21	*5692.8*	5691.1	5690.9	5689.1	5688.4
P22	*5732.4*	5731.7	5730.2	5729.2	5728.4
P23	*6160.7*	6159.1	6158.5	6157.7	6156.1



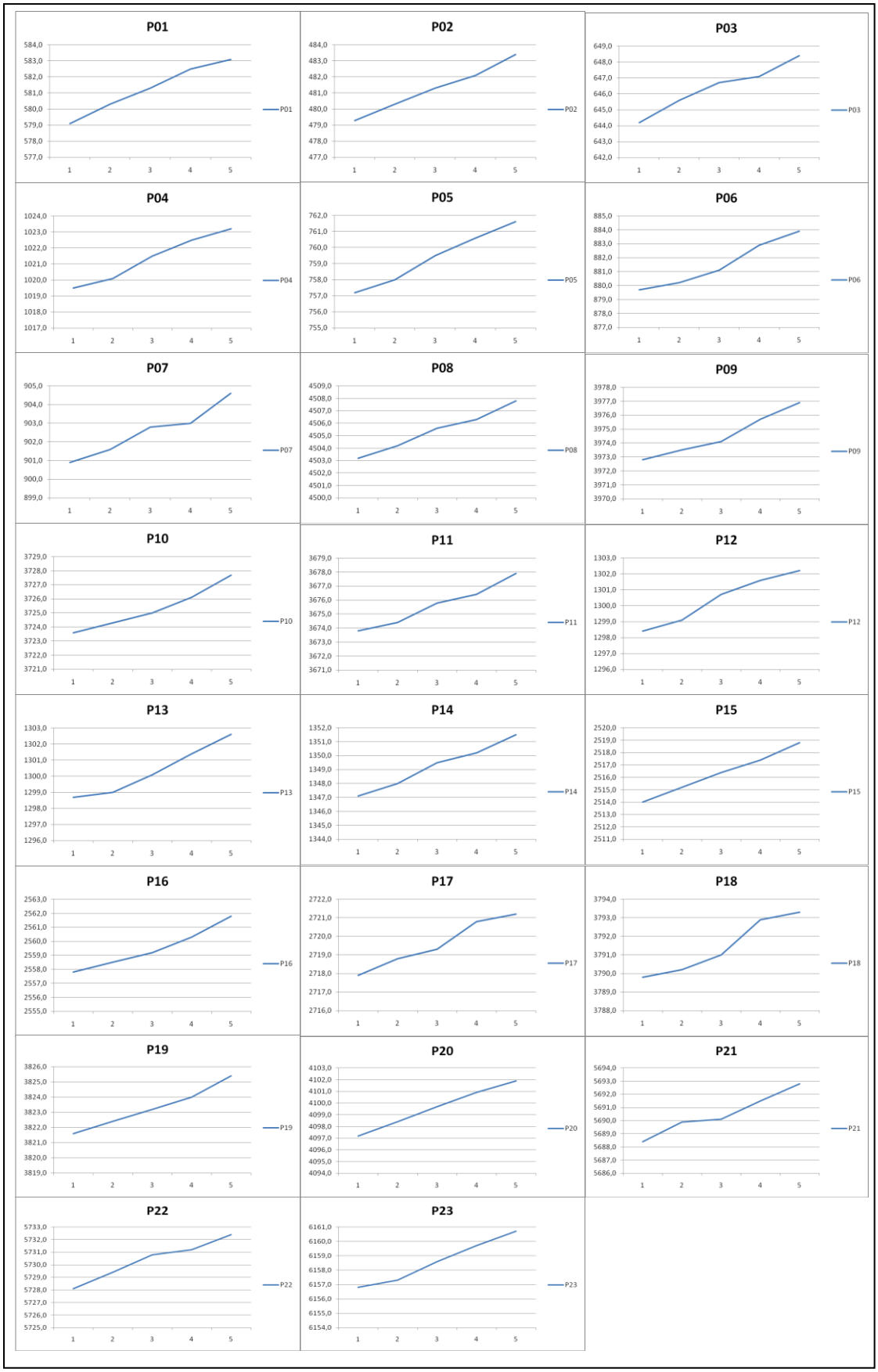
Şekil 5.13 α parametresinin değiştirilmesi ile elde edilen sonuçların 23 problem seti için grafik gösterimi

5.6.1.2 β Parametresinin Değiştirilmesi

Karınca kolonisi optimizasyonunda sezgiselliğin önemini gösteren ve karıncaları yakın mesafedeki şehirlere gitmeleri yönünde etkileyen β parametresi baz alındığı 5 değerinden alt sınırı olan 1 değerine doğru indirilirken programın ürettiği sonuç çıktı değerlerinin küçüldüğü dolayısıyla β değerinin küçük seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu elde edilmiştir. β parametresi üzerinde yapılan değişikliklerle elde edilen sonuç değerleri Çizelge 5.7'de ve 23 problem seti üzerindeki sonuç değişiklik grafikleri ise Şekil 5.14'de gösterilmiştir.

Çizelge 5.7 β parametresinin değiştirilmesi ile elde edilen sonuçlar

β =	1	2	3	4	5
P01	579.1	580.3	581.3	582.5	*583.1*
P02	479.3	480.3	481.3	482.1	*483.4*
P03	644.2	645.6	646.7	647.1	*648.4*
P04	1019.5	1020.1	1021.5	1022.5	*1023.2*
P05	757.2	758.0	759.5	760.6	*761.6*
P06	879.7	880.2	881.1	882.9	*883.9*
P07	900.9	901.6	902.8	903.0	*904.6*
P08	4503.2	4504.2	4505.6	4506.3	*4507.8*
P09	3972.8	3973.5	3974.1	3975.7	*3976.9*
P10	3723.6	3724.3	3725.0	3726.1	*3727.7*
P11	3673.8	3674.4	3675.8	3676.4	*3677.9*
P12	1298.4	1299.1	1300.7	1301.6	*1302.2*
P13	1298.7	1299.0	1300.1	1301.4	*1302.6*
P14	1347.1	1348.0	1349.5	1350.2	*1351.5*
P15	2514.0	2515.2	2516.4	2517.4	*2518.8*
P16	2557.8	2558.5	2559.2	2560.3	*2561.8*
P17	2717.9	2718.8	2719.3	2720.8	*2721.2*
P18	3789.8	3790.2	3791.0	3792.9	*3793.3*
P19	3821.6	3822.4	3823.2	3824.0	*3825.4*
P20	4097.2	4098.4	4099.7	4100.9	*4101.9*
P21	5688.4	5689.9	5690.1	5691.5	*5692.8*
P22	5728.1	5729.4	5730.8	5731.2	*5732.4*
P23	6156.8	6157.3	6158.6	6159.7	*6160.7*



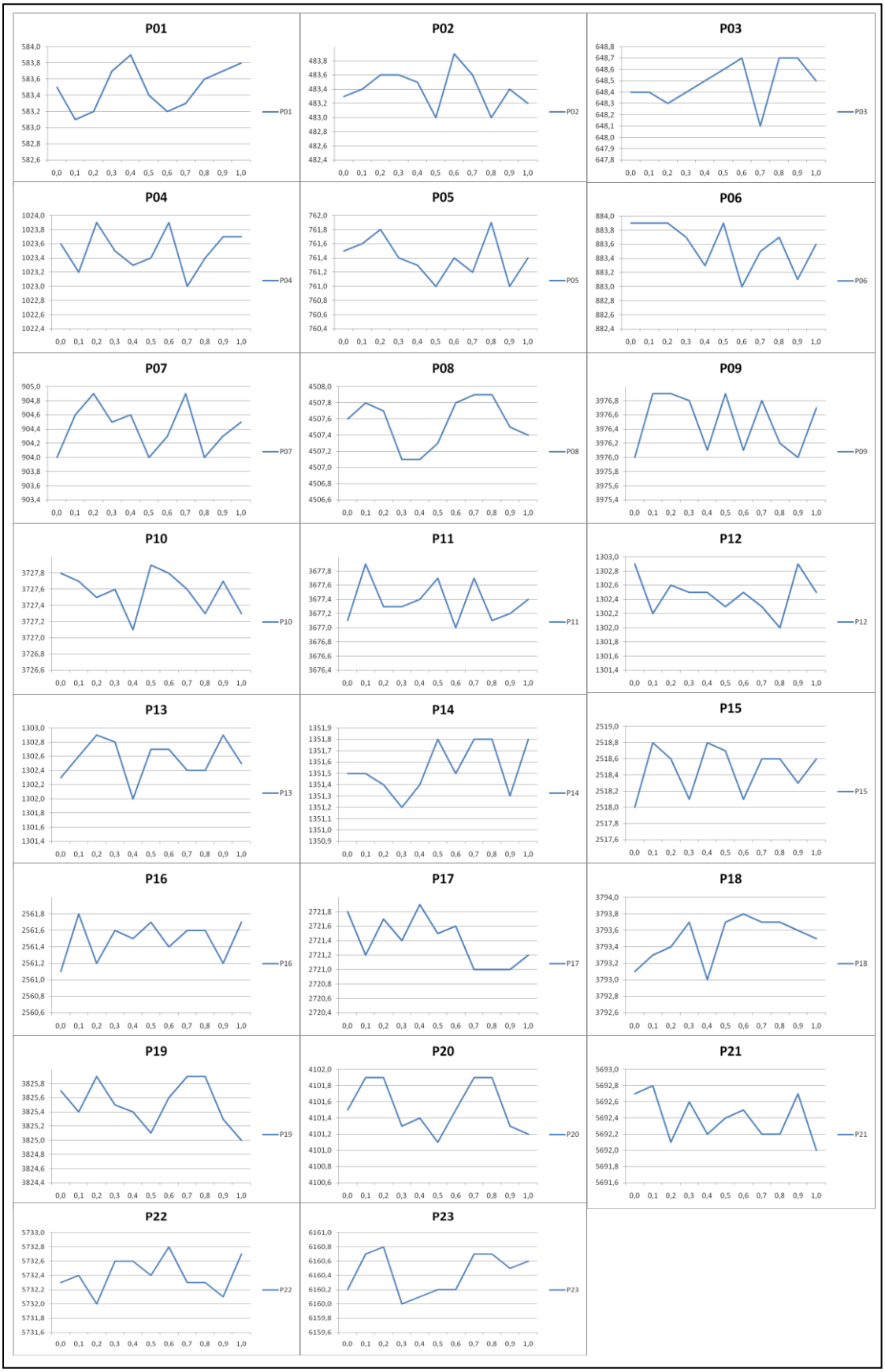
Şekil 5.14 β parametresinin değiştirilmesi ile elde edilen sonuçların 23 problem seti için grafik gösterimi

5.6.1.3 ρ Parametresinin Değiştirilmesi

Karınca kolonisi optimizasyonunda karıncaların geçtiği yollara bıraktıkları feromon maddesinin sınırsız büyümesini engellemek için kullanılan ρ buharlaşma katsayısı parametresi baz alındığı 0.1 değerinden üst sınırı olan 1.0 değerine doğru yükseltilirken programın ürettiği sonuç çıktı değerlerinin ondalık değerler oranında artıp azaldığı fakat grafiksel olarak tanımlanacak düzgün bir değişim göstermediği sonucu elde edilmiş ve ρ buharlaşma katsayısı parametresinin toplam mesafe sonuç değeri üzerinde anlamlı bir etkisi olmadığı ortaya konmuştur. ρ parametresi üzerinde yapılan değişikliklerle elde edilen sonuç değerleri Çizelge 5.8'de ve 23 problem seti üzerindeki sonuç değişiklik grafikleri ise Şekil 5.15'de gösterilmiştir.

Çizelge 5.8 ρ parametresinin değiştirilmesi ile elde edilen sonuçlar

ρ =	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	583.5	*583.1*	583.2	583.7	583.9	583.4	583.2	583.3	583.6	583.7	583.8
P02	483.3	*483.4*	483.6	483.6	483.5	483.0	483.9	483.6	483.0	483.4	483.2
P03	648.4	*648.4*	648.3	648.4	648.5	648.6	648.7	648.1	648.7	648.7	648.5
P04	1023.6	*1023.2*	1023.9	1023.5	1023.3	1023.4	1023.9	1023.0	1023.4	1023.7	1023.7
P05	761.5	*761.6*	761.8	761.4	761.3	761.0	761.4	761.2	761.9	761.0	761.4
P06	883.9	*883.9*	883.9	883.7	883.3	883.9	883.0	883.5	883.7	883.1	883.6
P07	904.0	*904.6*	904.9	904.5	904.6	904.0	904.3	904.9	904.0	904.3	904.5
P08	4507.6	*4507.8*	4507.7	4507.1	4507.1	4507.3	4507.8	4507.9	4507.9	4507.5	4507.4
P09	3976.0	*3976.9*	3976.9	3976.8	3976.1	3976.9	3976.1	3976.8	3976.2	3976.0	3976.7
P10	3727.8	*3727.7*	3727.5	3727.6	3727.1	3727.9	3727.8	3727.6	3727.3	3727.7	3727.3
P11	3677.1	*3677.9*	3677.3	3677.3	3677.4	3677.7	3677.0	3677.7	3677.1	3677.2	3677.4
P12	1302.9	*1302.2*	1302.6	1302.5	1302.5	1302.3	1302.5	1302.3	1302.0	1302.9	1302.5
P13	1302.3	*1302.6*	1302.9	1302.8	1302.0	1302.7	1302.7	1302.4	1302.4	1302.9	1302.5
P14	1351.5	*1351.5*	1351.4	1351.2	1351.4	1351.8	1351.5	1351.8	1351.8	1351.3	1351.8
P15	2518.0	*2518.8*	2518.6	2518.1	2518.8	2518.7	2518.1	2518.6	2518.6	2518.3	2518.6
P16	2561.1	*2561.8*	2561.2	2561.6	2561.5	2561.7	2561.4	2561.6	2561.6	2561.2	2561.7
P17	2721.8	*2721.2*	2721.7	2721.4	2721.9	2721.5	2721.6	2721.0	2721.0	2721.0	2721.2
P18	3793.1	*3793.3*	3793.3	3793.7	3793.0	3793.7	3793.8	3793.7	3793.7	3793.6	3793.5
P19	3825.7	*3825.4*	3825.4	3825.5	3825.4	3825.1	3825.6	3825.9	3825.9	3825.3	3825.0
P20	4101.5	*4101.9*	4101.9	4101.3	4101.4	4101.1	4101.5	4101.9	4101.9	4101.3	4101.2
P21	5692.7	*5692.8*	5692.8	5692.6	5692.2	5692.4	5692.5	5692.2	5692.2	5692.7	5692.0
P22	5732.3	*5732.4*	5732.4	5732.6	5732.6	5732.4	5732.8	5732.3	5732.3	5732.1	5732.7
P23	6160.2	*6160.7*	6160.8	6160.0	6160.1	6160.2	6160.2	6160.7	6160.7	6160.5	6160.6



Şekil 5.15 ρ parametresinin değiştirilmesi ile elde edilen sonuçların 23 problem seti için grafik gösterimi

5.6.1.4 q_0 Parametresinin Değiştirilmesi

Karınca kolonisi optimizasyonunda karıncaların gideceği şehri seçmelerini sağlayan sözde-rastlantısal-orantılı geçiş kuralında kullanılacak olan formülasyonu belirlemek için program tarafından üretilen rastlantısal q değeri ile karşılaştırılan q_o parametresi baz alındığı 0.9 değerinden alt sınırı olan 0.0 değerine doğru indirilirken programın ürettiği sonuç çıktı değerlerinin küçüldüğü dolayısıyla q_0 değerinin küçük seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu elde edilmiştir. q_0 parametresi üzerinde yapılan değişikliklerle elde edilen sonuç değerleri Çizelge 5.9'da ve 23 problem seti üzerindeki sonuç değişiklik grafikleri ise Şekil 5.16'da gösterilmiştir.

Çizelge 5.9 q_0 parametresinin değiştirilmesi ile elde edilen sonuçlar

q_0 =	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	574.3	575.7	576.5	577.1	578.9	579.5	580.2	581.0	582.0	*583.1*	584.3
P02	474.8	475.5	476.5	477.3	478.1	479.0	480.9	481.0	482.2	*483.4*	484.2
P03	639.1	640.4	641.6	642.1	643.9	644.7	645.7	646.5	647.8	*648.4*	649.5
P04	1014.1	1015.0	1016.6	1017.2	1018.5	1019.5	1020.5	1021.4	1022.5	*1023.2*	1024.6
P05	752.8	753.5	754.1	755.3	756.1	757.5	758.6	759.4	760.2	*761.6*	762.4
P06	874.9	875.7	876.5	877.4	878.7	879.3	880.4	881.2	882.9	*883.9*	884.3
P07	895.3	896.2	897.2	898.6	899.6	900.9	901.0	902.1	903.4	*904.6*	905.2
P08	4498.8	4499.5	4500.7	4501.8	4502.2	4503.7	4504.9	4505.9	4506.1	*4507.8*	4508.8
P09	3967.2	3968.7	3969.0	3970.1	3971.1	3972.8	3973.6	3974.7	3975.6	*3976.9*	3977.4
P10	3718.6	3719.8	3720.3	3721.4	3722.0	3723.5	3724.4	3725.9	3726.0	*3727.7*	3728.8
P11	3668.4	3669.9	3670.2	3671.7	3672.4	3673.1	3674.5	3675.8	3676.4	*3677.9*	3678.4
P12	1293.8	1294.0	1295.3	1296.2	1297.4	1298.8	1299.6	1300.6	1301.8	*1302.2*	1303.4
P13	1293.9	1394.9	1295.0	1296.7	1297.2	1298.4	1299.5	1300.6	1301.7	*1302.6*	1303.0
P14	1342.3	1343.5	1344.4	1345.0	1346.6	1347.3	1348.0	1349.2	1350.3	*1351.5*	1352.2
P15	2509.3	2510.2	2511.5	2512.6	2513.7	2514.5	2515.1	2516.7	2517.9	*2518.8*	2519.4
P16	2552.9	2553.4	2554.8	2555.5	2556.0	2557.9	2558.3	2559.2	2560.7	*2561.8*	2562.7
P17	2712.3	2713.5	2714.0	2715.6	2716.5	2717.1	2718.2	2719.3	2720.8	*2721.2*	2722.0
P18	3784.8	3785.1	3786.8	3787.6	3788.3	3789.8	3790.9	3791.8	3792.7	*3793.3*	3794.1
P19	3816.8	3817.9	3818.1	3819.8	3820.0	3821.2	3822.2	3823.4	3824.9	*3825.4*	3826.2
P20	4092.2	4093.5	4094.5	4095.8	4096.6	4097.6	4098.6	4099.5	4100.1	*4101.9*	4102.9
P21	5683.0	5684.3	5685.7	5686.7	5687.4	5688.5	5689.4	5690.4	5691.6	*5692.8*	5693.0
P22	5723.2	5724.6	5725.1	5726.7	5727.3	5728.8	5729.8	5730.8	5731.6	*5732.4*	5733.5
P23	6151.0	6152.3	6153.3	6154.2	6155.3	6156.7	6157.4	6158.5	6159.3	*6160.7*	6161.0

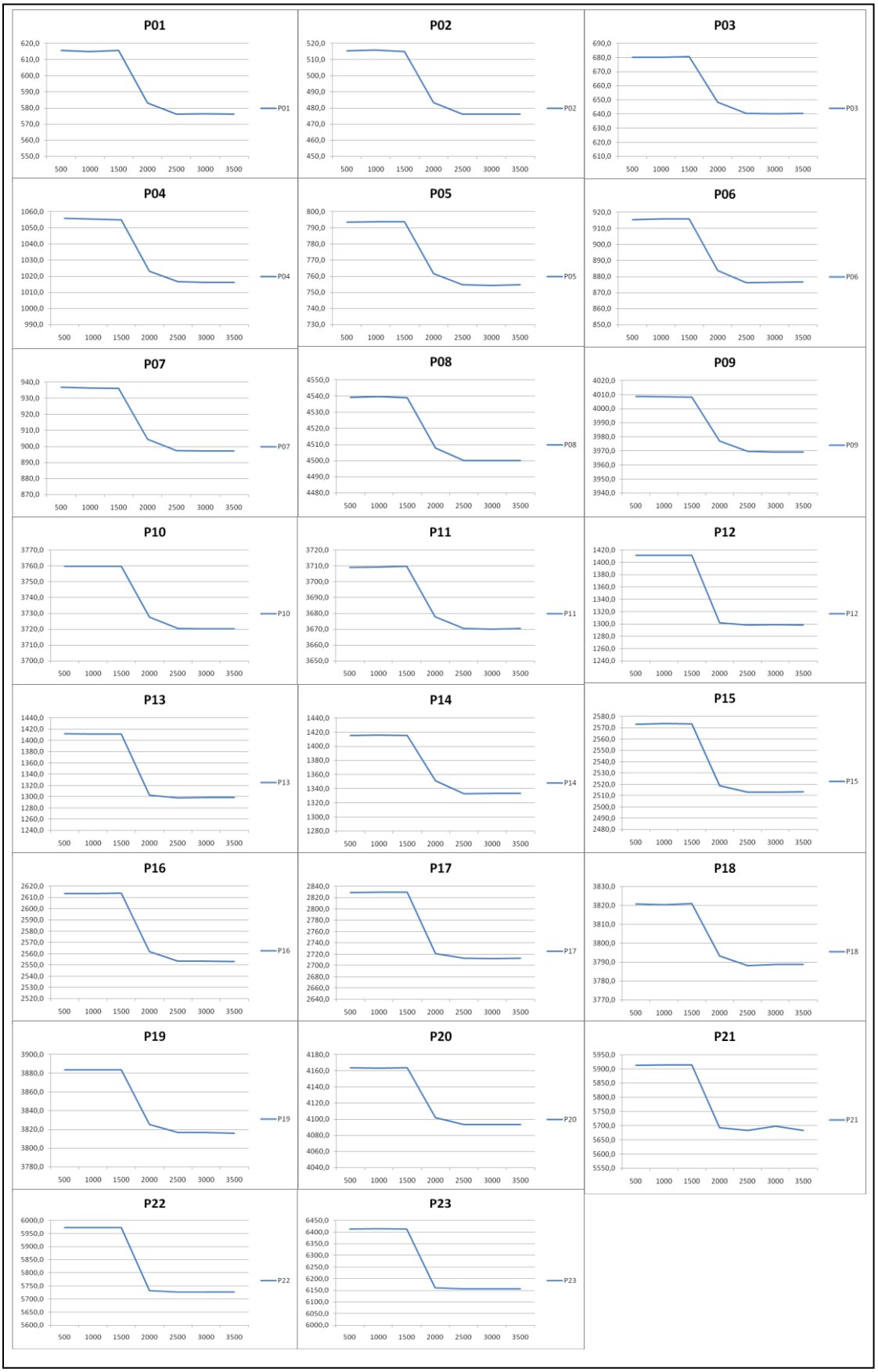
Şekil 5.16 q_0 parametresinin değiştirilmesi ile elde edilen sonuçların 23 problem seti için grafik gösterimi

5.6.1.5 İterasyon Sayısı Parametresinin Değiştirilmesi

Karınca kolonisi optimizasyonunda algoritma adımlarının tekrarlanacağı maksimum sayıyı gösteren iterasyon sayısı parametresi baz alındığı 2000 değerinin altında bulunan 500 değerine üzerinde de 3500 değerine kadar çalıştırıldığında programın 2000'in altındaki iterasyon sayılarında daha yüksek, 2000'in üzerindeki iterasyon sayılarında ise daha düşük sonuç çıktı değerleri ürettiği gözlenmiş ve iterasyon sayısı değerinin 2000'den büyük 3000 değeri civarında seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu elde edilmiştir. İterasyon sayısı parametresi üzerinde yapılan değişikliklerle elde edilen sonuç değerleri Çizelge 5.10'da ve 23 problem seti üzerindeki sonuç değişiklik grafikleri ise Şekil 5.17'de gösterilmiştir.

Çizelge 5.10 İterasyon sayısı parametresinin değiştirilmesi ile elde edilen sonuçlar

İterasyon sayısı =	500	1000	1500	2000	2500	3000	3500
P01	615.7	615.0	615.7	*583.1*	576.2	576.4	576.3
P02	515.4	515.8	515.0	*483.4*	476.3	476.3	476.2
P03	680.1	680.1	680.8	*648.4*	640.4	640.1	640.6
P04	1055.9	1055.4	1055.0	*1023.2*	1016.8	1016.2	1016.3
P05	793.5	793.7	793.7	*761.6*	754.7	754.3	754.8
P06	915.5	915.8	915.8	*883.9*	976.3	876.4	876.7
P07	936.9	936.3	936.0	*904.6*	897.4	897.2	897.1
P08	4539.1	4539.6	4539.1	*4507.8*	4500.1	4500.0	4500.0
P09	4008.8	4008.5	4008.2	*3976.9*	3976.7	3969.1	3969.1
P10	3759.8	3759.8	3759.7	*3727.7*	3720.5	3720.3	3720.3
P11	3709.0	3709.3	3709.8	*3677.9*	3670.4	3670.0	3670.4
P12	1411.2	1411.2	1411.4	*1302.2*	1298.3	1298.6	1298.4
P13	1411.9	1411.5	1411.1	*1302.6*	1298.1	1298.5	1298.9
P14	1415.4	1415.9	1415.2	*1351.5*	1333.0	1333.2	1333.5
P15	2573.0	2573.7	2573.4	*2518.8*	2513.0	2513.1	2513.5
P16	2613.5	2613.3	2613.9	*2561.8*	2553.4	2553.5	2553.0
P17	2829.0	2829.3	2829.4	*2721.2*	2712.9	2712.5	2712.6
P18	3820.7	3820.3	3820.9	*3793.3*	3788.1	3788.8	3788.7
P19	3883.4	3883.4	3883.7	*3825.4*	3816.8	3816.8	3816.1
P20	4163.7	4163.3	4163.7	*4101.9*	4093.3	4093.6	4093.3
P21	5913.2	5913.9	5913.9	*5692.8*	5683.5	5698.7	5683.2
P22	5972.8	5972.4	5972.9	*5732.4*	5726.8	5726.9	5726.9
P23	6413.1	6413.9	6413.3	*6160.7*	6156.6	6156.3	6156.0



Şekil 5.17 q_0 parametresinin değiştirilmesi ile elde edilen sonuçların 23 problem seti için grafik gösterimi

5.6.2 Eşzamanlı-Çapraz Parametre Değişimleri

Bu doktora tezi kapsamında oluşturulan algoritma yapısının karınca kolonisi optimizasyonu kısımında yani çözümün ikinci aşamasında kullanılan α, β, ρ, q_0 ve iterasyon sayısı parametreleri üzerinde yapılan bir parametrenin değiştirilerek diğer parametre değerlerinin sabit tutulduğu değişimler ve algoritma sonuçlarının bu değişimlere verdiği tepkiler bir önceki bölümde açıklanmıştır. Bu bölümde de, sonuç değerleri üzerinde anlamlı ve büyük bir etkisi olmayan ρ parametresi 0.1 değerine, iterasyon sayısı parametresi de 3000 değerine sabitlenerek α, β ve q_0 parametreleri üzerinde yapılan eşzamanlı-çapraz değişimler ve algoritma sonuçlarının bu değişimlere verdiği tepkiler incelenmiştir.

***İterasyon Sayısı = 3000, α = 1* ve *β = 5*:** Kabul edilen bu parametre değerlerine göre geçiş kuralı parametresi q_0 değeri 0.0 ile 1.0 değerleri arasında ne kadar küçük seçilirse algoritmanın problemin çözüm amacı olan daha düşük mesafeyi o derece sağladığı sonucu elde edilmiş ve yapılan bu eşzamanlı-çapraz parametre değişiklikleri ile elde edilen sonuç değerleri Çizelge 5.11'de gösterilmiştir.

Çizelge 5.11 İterasyon sayısı = 3000, α = 1 ve β = 5 iken q_0 parametresinin değiştirilmesi ile elde edilen sonuçlar

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	567.4	568.6	569.1	570.0	571.9	572.1	573.3	574.3	575.0	576.5	577.2
P02	467.6	468.8	469.7	470.2	471.3	472.7	473.3	474.7	475.6	476.8	477.0
P03	631.3	632.5	633.3	634.2	635.6	636.3	637.4	638.0	639.6	640.2	614.6
P04	1007.3	1008.7	1009.1	1010.3	1011.0	1012.6	1013.6	1014.2	1015.7	1016.6	1017.6
P05	745.2	746.7	747.5	748.5	749.2	750.1	751.3	752.6	753.5	754.1	755.0
P06	867.8	868.7	869.4	870.9	871.5	872.6	873.9	874.2	875.9	876.9	877.7
P07	888.5	889.8	890.2	891.8	892.7	893.4	894.7	895.6	896.7	897.0	898.8
P08	4491.5	4492.4	4493.1	4494.8	4495.4	4496.7	4497.7	4498.8	4499.6	4500.4	4501.0
P09	3960.6	3961.6	3962.0	3963.7	3964.0	3965.0	3966.3	3967.6	3968.7	3969.8	3970.3
P10	3711.2	3712.2	3713.9	3714.9	3715.8	3716.9	3717.2	3718.9	3719.8	3720.6	3721.2
P11	3661.1	3662.6	3663.4	3664.2	3665.8	3666.2	3667.9	3668.3	3669.7	3670.9	3672.1
P12	1298.1	1289.1	1291.7	1292.6	1293.3	1294.7	1295.5	1296.1	1296.6	1298.2	1299.1
P13	1289.7	1290.5	1291.8	1292.2	1293.8	1294.8	1295.4	1296.7	1297.7	1298.3	1299.9
P14	1324.3	1325.3	1326.9	1327.0	1328.0	1329.7	1330.8	1331.8	1332.2	1333.3	1334.5
P15	2504.8	2505.9	2506.1	2507.4	2508.5	2509.1	2510.3	2511.4	2512.3	2513.9	2514.7
P16	2544.7	2545.8	2546.1	2547.2	2548.7	2549.4	2550.7	2551.0	2552.6	2553.5	2554.5
P17	2703.3	2704.9	2705.2	2706.9	2707.6	2708.4	2709.2	2710.2	2711.6	2712.9	2713.8
P18	3779.9	3780.6	3781.6	3782.6	3783.4	3784.3	3785.0	3786.7	3787.3	3788.2	3789.4
P19	3807.1	3808.7	3809.3	3810.7	3811.9	3812.5	3812.8	3814.0	3815.2	3816.7	3817.9
P20	4084.2	4085.4	4086.5	4087.2	4088.4	4089.8	4090.5	4091.1	4092.2	4093.5	4094.5
P21	5674.3	5675.1	5676.6	5677.0	5678.2	5679.7	5680.5	5681.9	5682.9	5683.7	5684.5
P22	5717.6	5718.4	5719.0	5720.4	5721.2	5722.7	5723.2	5724.4	5725.4	5726.0	5727.0
P23	6147.4	6148.7	6149.3	6150.3	6151.5	5152.2	6153.3	6154.0	6155.8	6156.1	6157.7

***İterasyon Sayısı = 3000, α = 2* ve *β = 4*:** Kabul edilen bu parametre değerlerine göre geçiş kuralı parametresi q_0 değeri 0.0 ile 1.0 değerleri arasında ne kadar küçük seçilirse algoritmanın problemin çözüm amacı olan daha düşük mesafeyi o derece sağladığı sonucu elde edilmiş ve yapılan bu eşzamanlı-çapraz parametre değişiklikleri ile elde edilen sonuç değerleri Çizelge 5.12'de gösterilmiştir.

Çizelge 5.12 İterasyon sayısı = 3000, α = 2 ve β = 4 iken q_0 parametresinin değiştirilmesi ile elde edilen sonuçlar

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	565.7	566.9	567.2	568.8	569.3	570.1	571.5	572.6	573.8	574.8	575.9
P02	465.0	466.5	467.7	468.0	469.7	470.5	471.5	472.5	473.2	474.1	475.7
P03	629.7	630.6	631.8	632.0	633.6	634.4	635.0	636.5	637.9	638.3	639.5
P04	1005.2	1006.2	1007.5	1008.6	1009.6	1010.2	1011.7	1012.5	1013.7	1014.0	1015.2
P05	743.4	744.6	745.1	746.5	747.2	748.3	749.2	750.1	751.2	752.3	753.8
P06	865.2	866.3	867.3	868.7	867.3	870.0	871.3	872.9	873.0	874.4	875.4
P07	886.3	887.7	888.8	889.2	890.8	891.8	892.8	893.1	894.1	895.8	896.7
P08	4489.7	4490.9	4491.2	4492.8	4493.4	4494.3	4495.2	4496.7	4497.0	4498.5	4499.4
P09	3958.9	3959.7	3960.4	3961.5	3962.3	3963.5	3964.0	3965.4	3966.0	3967.5	3968.0
P10	3709.9	3710.7	3711.5	3712.4	3713.3	3714.9	3715.2	3716.2	3717.4	3718.6	3719.8
P11	3659.3	3660.0	3661.9	3662.8	3663.4	3664.0	3665.3	3666.0	3667.7	3668.1	3669.8
P12	1287.4	1288.3	1289.8	1290.6	1291.3	1292.2	1293.9	1294.1	1295.1	1296.8	1297.6
P13	1287.2	1288.9	1289.7	1290.3	1291.4	1292.4	1293.4	1294.2	1295.1	1296.7	1297.6
P14	1322.3	1323.7	1324.0	1325.3	1326.7	1327.5	1328.2	1329.0	1330.9	1331.9	1332.1
P15	2502.5	2503.4	2504.2	2505.1	2506.9	2507.4	2508.7	2509.4	2510.5	2511.9	2512.5
P16	2542.1	2543.7	2544.7	2545.2	2546.0	2574.0	2548.7	2549.4	2550.0	2551.4	2552.9
P17	2701.1	2702.0	2703.7	2704.0	2705.3	2706.0	2707.3	2708.6	2709.0	2710.4	2711.0
P18	3777.4	3778.1	3779.4	3780.2	3781.7	3782.2	3783.0	3784.5	3785.1	3786.7	3787.4
P19	3805.6	3806.7	3807.0	3808.1	3809.3	3810.4	3811.3	3812.5	3813.7	3814.1	3815.4
P20	4082.0	4093.2	4084.4	4085.4	4086.8	4087.1	4088.2	4089.2	4090.0	4091.1	4092.6
P21	5672.6	5673.4	5674.6	5675.6	5676.3	5677.8	5678.3	5679.3	5680.7	5681.0	5682.4
P22	5715.9	5716.2	5717.3	5718.3	5719.1	5720.3	5721.9	5722.0	5723.0	5724.6	5725.2
P23	6145.0	6146.7	6174.3	6148.3	6149.5	6150.9	6151.0	6152.3	6153.3	6154.1	6155.4

***İterasyon Sayısı = 3000, α = 3* ve *β = 3*:** Kabul edilen bu parametre değerlerine göre geçiş kuralı parametresi q_0 değeri 0.0 ile 1.0 değerleri arasında ne kadar küçük seçilirse algoritmanın problemin çözüm amacı olan daha düşük mesafeyi o derece sağladığı sonucu elde edilmiş ve yapılan bu eşzamanlı-çapraz parametre değişiklikleri ile elde edilen sonuç değerleri Çizelge 5.13'de gösterilmiştir.

Çizelge 5.13 İterasyon sayısı = 3000, α = 3 ve β = 3 iken q_0 parametresinin değiştirilmesi ile elde edilen sonuçlar

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	563.9	564.4	565.3	566.9	567.6	568.7	569.6	570.3	571.1	572.4	573.4
P02	463.7	646.7	465.6	466.3	467.2	468.3	469.6	470.0	471.3	472.0	473.3
P03	627.2	628.2	629.9	630.6	631.3	632.4	633.3	634.1	635.5	636.9	637.3
P04	1003.6	1004.7	1005.5	1006.8	1007.7	1008.3	1009.0	1010.0	1011.3	1012.2	1013.3
P05	741.1	742.8	743.9	744.3	745.9	746.2	747.5	748.1	749.8	750.1	751.4
P06	863.3	864.4	865.0	866.4	867.9	868.6	869.0	870.9	871.9	872.5	873.6
P07	884.2	885.1	886.0	887.0	888.1	889.1	890.3	891.8	892.8	893.7	894.1
P08	4487.3	4488.3	4489.2	4490.0	4491.0	4492.8	4493.7	4494.0	4495.6	4496.8	4497.7
P09	3956.2	3957.5	3958.7	3959.6	3960.6	3961.3	3962.8	3963.1	3964.6	3965.1	3966.0
P10	3707.9	3708.8	3709.8	3710.2	3711.7	3712.4	3713.8	3714.0	3715.1	3716.2	3717.1
P11	3657.0	3658.6	3659.9	3660.8	3661.7	3662.7	3663.7	3664.0	3665.7	3666.5	3667.4
P12	1285.8	1286.0	1287.9	1288.5	1289.3	1290.5	1291.1	1292.7	1293.0	1294.9	1295.4
P13	1285.3	1286.3	1287.1	1288.8	1289.8	1290.3	1291.3	12912.0	1293.7	1294.0	1295.9
P14	1320.1	1321.1	1322.6	1323.7	1324.1	1325.0	1326.1	1327.6	1328.2	1329.8	1330.4
P15	2500.7	2501.3	2502.0	2503.9	2504.4	2505.9	2506.9	2507.1	2508.9	2509.3	2510.2
P16	2540.4	2541.9	2542.2	2543.9	2544.1	2545.3	2546.6	2547.7	2548.2	2549.9	2550.3
P17	2699.3	2700.9	2701.7	2702.0	2703.7	2704.4	2705.4	2706.4	2707.1	2708.7	2709.7
P18	3775.7	3776.2	3777.6	3778.8	3779.6	3780.2	3781.2	3782.8	3783.9	3784.9	3785.2
P19	3803.3	3804.4	3805.4	3806.5	3807.2	3808.4	3809.1	3810.5	3811.4	3812.0	3813.9
P20	4080.4	4081.3	4082.9	4093.4	4084.3	4085.8	4086.9	4087.4	4088.8	4089.7	4090.5
P21	5670.6	5671.7	5672.4	5673.6	5674.0	5675.3	5676.0	5677.2	5678.9	5679.7	5680.2
P22	5713.6	5714.7	5715.6	5716.3	5717.5	5718.7	5719.9	5720.5	5721.2	5722.0	5723.6
P23	6143.2	6144.7	6145.3	6146.8	6147.0	6148.7	6149.8	6150.9	6151.6	6152.9	6153.9

***İterasyon Sayısı = 3000, α = 4* ve *β = 2*:** Kabul edilen bu parametre değerlerine göre geçiş kuralı parametresi q_0 değeri 0.0 ile 1.0 değerleri arasında ne kadar küçük seçilirse algoritmanın problemin çözüm amacı olan daha düşük mesafeyi o derece sağladığı sonucu elde edilmiş ve yapılan bu eşzamanlı-çapraz parametre değişiklikleri ile elde edilen sonuç değerleri Çizelge 5.14'de gösterilmiştir.

Çizelge 5.14 İterasyon sayısı = 3000, α = 4 ve β = 2 iken q_0 parametresinin değiştirilmesi ile elde edilen sonuçlar

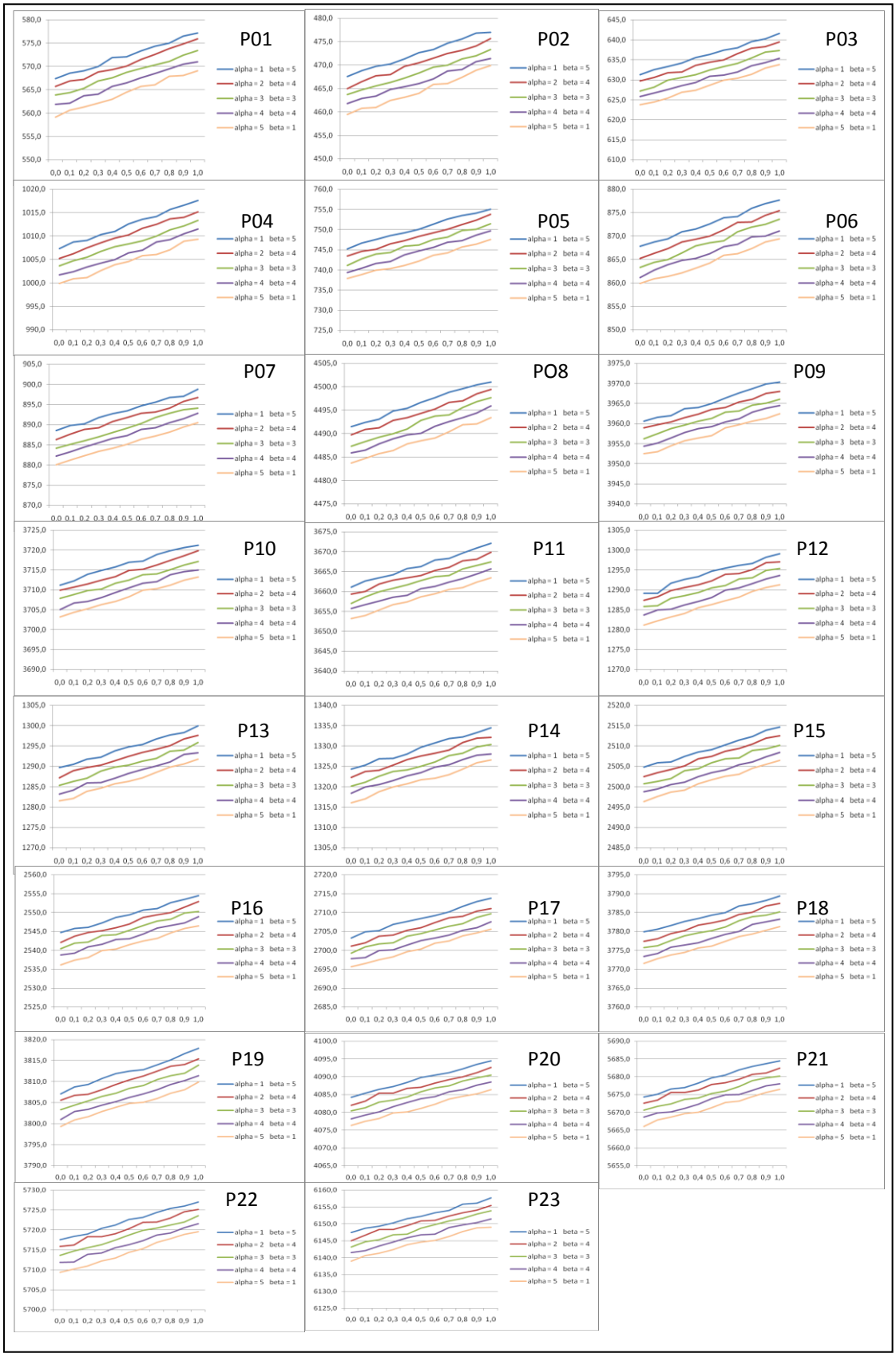
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	561.9	562.1	563.7	564.1	565.7	566.5	567.6	568.5	569.5	570.5	571.0
P02	461.8	462.9	463.4	464.8	465.4	466.1	467.1	468.7	469.1	470.7	471.4
P03	625.8	626.7	627.6	628.5	629.3	630.9	631.2	632.0	633.5	634.3	635.4
P04	1001.7	1002.4	1003.4	1004.2	1005.0	1006.4	1007.0	1008.7	1009.2	1010.5	1011.5
P05	739.5	740.4	741.6	742.1	743.7	744.7	745.6	746.9	747.2	748.6	749.7
P06	861.1	862.7	863.9	864.8	865.2	866.2	867.7	868.2	869.8	870.0	871.1
P07	882.2	883.3	884.5	885.5	886.6	887.3	888.8	889.2	890.5	891.5	892.8
P08	4485.9	4486.5	4487.8	4488.9	4489.7	4490.1	4491.6	4492.6	4493.4	4494.4	4495.9
P09	3954.4	3955.1	3956.4	3957.8	3958.7	3959.2	3960.4	3961.2	3962.8	3963.8	3964.5
P10	3705.0	3706.7	3707.1	3708.1	3709.3	3710.5	3711.7	3712.1	3713.8	3714.6	3715.0
P11	3655.7	3656.7	3657.6	3658.5	3659.0	3660.7	3661.3	3661.3	3663.2	3664.4	3665.7
P12	1283.7	1284.9	1285.1	1286.2	1287.1	1288.1	1289.9	1290.5	1291.6	1292.7	1293.6
P13	1283.2	1284.2	1285.1	1286.2	1287.1	1288.1	1289.9	1290.5	1291.6	1292.7	1293.6
P14	1318.4	1319.9	1320.5	1321.5	1322.7	1323.4	1324.8	1325.4	1326.7	1327.7	1328.0
P15	2498.8	2499.5	2500.5	2501.1	2502.5	2503.4	2504.1	2505.4	2506.1	2507.3	2508.4
P16	2538.8	2539.3	2540.9	2541.6	2542.9	2543.1	2544.3	2545.9	2546.6	2547.2	2548.9
P17	2697.8	2698.1	2699.9	2700.1	2701.4	2702.6	2703.3	2704.0	2705.3	2706.0	2707.5
P18	3773.4	3774.2	3775.8	3776.4	3777.0	3778.2	3779.2	3780.0	3781.9	3782.5	3783.2
P19	3801.0	3802.9	3803.4	3804.4	3805.2	3806.2	3807.1	3808.1	3809.3	3810.2	3811.4
P20	4078.2	4079.3	4080.1	4081.6	4082.8	4083.8	4084.4	4085.8	4086.4	4087.6	4088.5
P21	5668.7	5669.8	5670.1	5671.1	5672.3	5673.8	5674.9	5675.0	5676.2	5677.4	5678.0
P22	5711.9	5712.0	5713.9	5714.2	5715.6	5716.3	5717.3	5718.7	5719.2	5720.6	5721.6
P23	6141.5	6142.1	6143.4	6144.6	6145.9	6146.8	6147.0	6148.9	6149.7	6150.4	6151.4

***İterasyon Sayısı = 3000, α = 5* ve *β = 1*:** Kabul edilen bu parametre değerlerine göre geçiş kuralı parametresi q_0 değeri 0.0 ile 1.0 değerleri arasında ne kadar küçük seçilirse algoritmanın problemin çözüm amacı olan daha düşük mesafeyi o derece sağladığı sonucu elde edilmiş ve yapılan bu eşzamanlı-çapraz parametre değişiklikleri ile elde edilen sonuç değerleri Çizelge 5.15'de gösterilmiştir.

Çizelge 5.15 İterasyon sayısı = 3000, $\alpha = 5$ ve $\beta = 1$ iken q_0 parametresinin değiştirilmesi ile elde edilen sonuçlar

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P01	559.1	560.6	561.3	562.1	563.0	564.5	565.7	566.1	567.9	568.1	569.1
P02	459.5	460.8	461.0	462.5	463.1	464.0	465.9	466.1	467.4	468.9	469.9
P03	623.8	624.5	625.4	626.9	627.4	628.6	629.9	630.4	631.4	632.9	633.8
P04	999.9	1000.9	1001.1	1002.6	1003.9	1004.6	1005.8	1006.1	1007.1	1008.9	1009.3
P05	737.9	738.9	739.9	740.3	741.1	742.2	743.6	744.2	745.7	746.4	747.5
P06	859.9	860.9	861.4	862.1	863.1	864.2	865.9	866.1	867.3	868.7	869.4
P07	880.1	881.2	882.3	883.4	884.2	885.1	886.4	887.2	888.2	889.4	890.5
P08	4483.7	4484.7	4485.7	4486.4	4487.8	4488.5	4489.1	4490.4	4491.9	4492.1	4493.4
P09	3952.5	3953.0	3954.5	3955.7	3956.4	3957.0	3958.9	3959.7	3960.6	3961.3	3962.4
P10	3703.2	3704.4	3705.2	3706.3	3707.1	3708.3	3709.9	3710.3	3711.2	3712.4	3713.2
P11	3653.2	3654.0	3655.3	3656.7	3657.4	3658.6	3659.4	3660.5	3661.0	3662.3	3663.4
P12	1281.1	1282.2	1283.2	1284.1	1285.5	1286.3	1287.3	1288.2	1289.6	1290.6	1291.3
P13	1281.5	1282.1	1283.9	1284.1	1285.5	1286.3	1287.3	1288.2	1289.8	1290.6	1291.8
P14	1316.0	1317.0	1318.8	1319.9	1320.7	1321.7	1322.1	1323.0	1324.3	1325.9	1326.6
P15	2496.3	2497.6	2498.7	2499.2	2500.7	2501.7	2502.6	2503.1	2504.5	2505.5	2506.5
P16	2536.1	2537.4	2538.1	2539.9	2540.3	2541.5	2545.5	2543.2	2544.6	2545.8	2546.5
P17	2695.7	2696.5	2697.5	2698.3	2699.6	2700.3	2701.9	2702.5	2703.8	2704.6	2705.6
P18	3771.5	3772.8	3773.8	3774.5	3775.6	3776.1	3777.4	3778.6	3779.3	3780.3	3781.3
P19	3799.3	3800.9	3801.6	3802.9	3803.9	3804.8	3805.1	3806.0	3807.2	3808.1	3809.9
P20	4076.3	4077.5	4078.3	4079.8	4080.1	4081.1	4082.3	4083.7	4084.5	4085.2	4086.4
P21	5666.0	5667.9	5668.7	5669.6	5670.0	5671.3	5672.8	5673.2	5674.4	5675.6	5676.5
P22	5709.4	5710.2	5711.0	5712.2	5713.0	5714.4	5715.3	5716.9	5717.8	5718.9	5719.6
P23	6139.0	6140.6	6141.3	6142.4	6143.8	6144.6	6145.1	6146.3	6147.7	6148.8	6149.0

Sonuç değerleri üzerinde anlamlı ve büyük bir etkisi olmayan ρ parametresi 0.1 değerine, iterasyon sayısı parametresi de 3000 değerine sabitlenerek α, β ve q_0 parametreleri üzerinde yapılan eşzamanlı-çapraz değişimlerin 23 problem seti sonuçları üzerindeki grafiksel etkisi Şekil 5.18'de gösterilmiştir.



Şekil 5.18 ρ = 0.1, iterasyon sayısı = 3000 iken α, β ve q_0 parametrelerinin değiştirilmesi ile elde edilen sonuçların 23 problem seti için grafik gösterimi

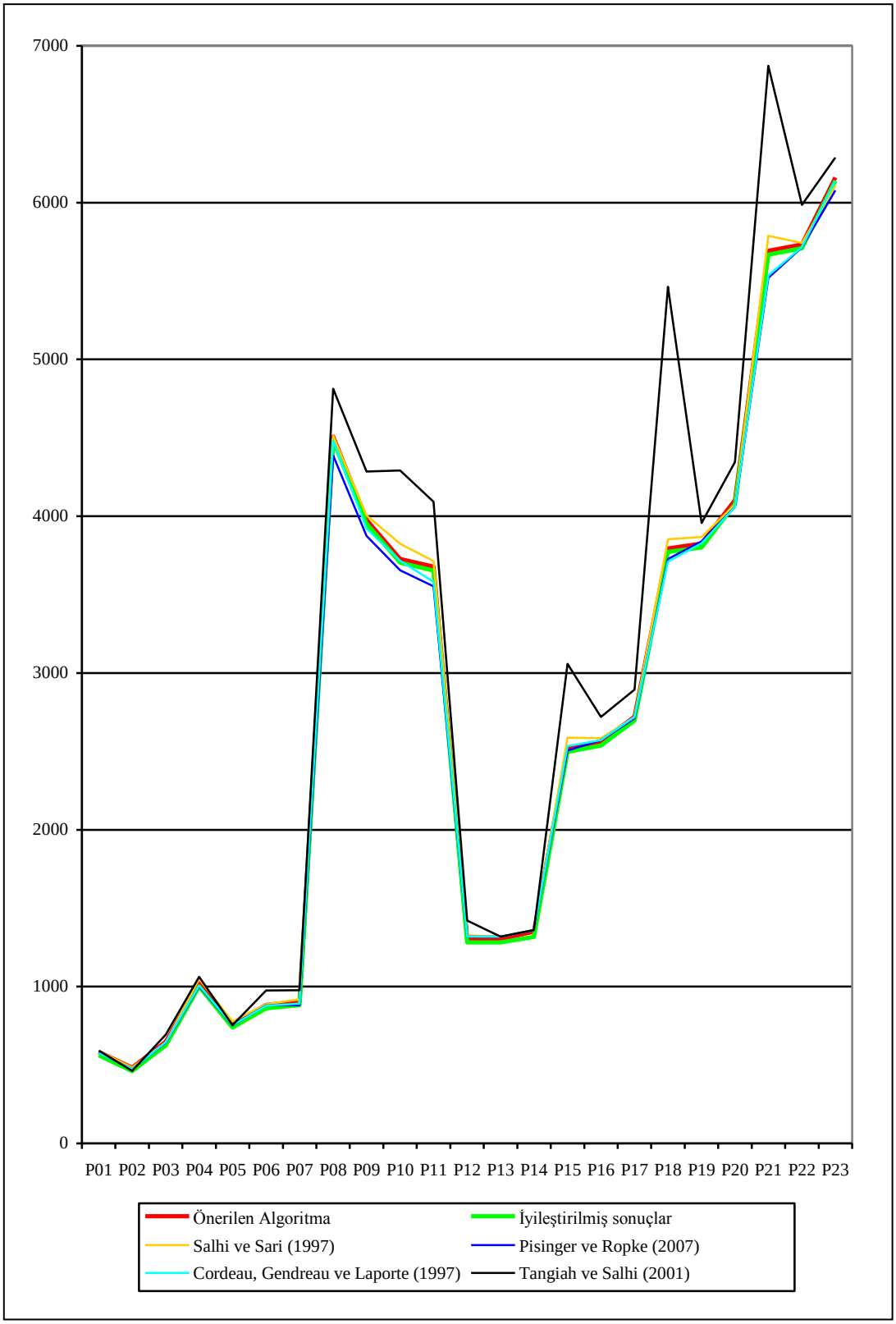
Önerilen çözüm yönteminin ikinci aşamasında gerçekleştirilen eşzamanlı-çapraz parametre değişimleri ile 23 problem seti için farklı sonuçlar elde edilmiş, bu yeni iyileştirilmiş sonuçlar literatürde var olan aynı konu üzerinde yapılan çalışma sonuçları ile yeniden karşılaştırılmış ve *iterasyon sayısı* = 3000, α = 5, β = 1, q_0 = 0.1 ve ρ = 0.1 değerleri ile elde edilen yeni karşılaştırma sonuçları Çizelge 5.16'da, bu karşılaştırmanın grafik gösterimi de Şekil 5.19'da verilmiştir.

Çizelge 5.16 Elde edilen iyileştirilmiş sonuçların bilinen en iyi sonuçlar ile karşılaştırılması

Problemin adı	Önerilen algoritma sonuçları	İyileştirilmiş sonuçlar	SS (1997)	CGL (1997)	TS (2001)	PR (2007)	En iyi Sonuçlar ile fark	Sapma (%)
P01	583.1	***560.6***	587.8	**576.9**	591.7	**576.9**	+16.3	(+)2.90
P02	483.4	***460.8***	484.6	473.9	**463.2**	473.5	+2.4	(+)0.52
P03	648.4	***624.5***	645.9	645.2	694.5	**641.2**	+16.7	(+)2.67
P04	1023.2	***1000.9***	1047.9	1006.7	1062.4	**1001.6**	+0.7	(+)0.06
P05	761.6	***738.9***	777.2	753.3	754.8	**751.9**	+13.0	(+)1.75
P06	883.9	***860.9***	888.6	**877.8**	976.0	880.4	+16.9	(+)1.96
P07	904.6	***881.2***	918.9	891.9	976.5	**881.9**	+0.7	(+)0.07
P08	4507.8	4484.7	4513.3	4482.4	4812.5	**4387.4**	-97.3	(-)2.16
P09	3976.9	3953.0	4005.3	3920.9	4284.6	**3874.8**	-78.2	(-)1.97
P10	3727.7	3704.4	3824.7	3714.7	4291.5	**3655.2**	-49.2	(-)1.32
P11	3677.9	3654.0	3714.3	3580.8	4092.7	**3552.3**	-101.7	(-)2.78
P12	**1302.2**	***1282.2***	1326.8	1318.9	1421.9	1318.9	+36.7	(+)2.86
P13	**1302.6**	***1282.1***	1318.9	1318.9	1318.9	1318.9	+36.8	(+)2.87
P14	**1351.5**	***1317.0***	1360.1	1360.1	1360.1	1360.1	+43.1	(+)3.27
P15	2518.8	***2497.6***	2586.7	2534.1	3059.2	**2505.4**	+7.8	(+)0.31
P16	**2561.8**	***2537.4***	2584.5	2572.2	2719.9	2572.2	+34.8	(+)1.37
P17	2721.2	***2696.5***	2720.2	2720.2	2894.7	**2709.1**	+12.6	(+)0.46
P18	3793.3	3772.8	3853.3	**3710.5**	5462.9	3727.6	-62.3	(-)1.65
P19	**3825.4**	***3800.9***	3867.9	3827.1	3956.6	3839.4	+26.2	(+)0.68
P20	4101.9	4077.5	4074.8	**4058.1**	4344.8	**4058.1**	-19.4	(-)0.47
P21	5692.8	5667.9	5788.5	5535.9	6872.1	**5519.5**	-148.4	(-)2.61
P22	5732.4	***5710.2***	5742.6	5716.0	5985.3	**5714.5**	+4.3	(+)0.07
P23	6160.7	6140.6	6106.6	6139.7	6288.0	**6078.8**	-61.8	(-)1.00

SS: Salhi ve Sari (1997); CGL: Cordeau, Gendreau ve Laporte (1997); TS: Thangiah ve Salhi (2001); PR: Pisinger ve Ropke (2007).

Çizelge 5.16'da gösterilen sonuçlara göre önerilen algoritma içinde yapılan eşzamanlı-çapraz parametre değişimleri ile literatürde var olan 23 problem setinin 15 tanesi için bilinen iyi sonuçlardan daha iyi çözümler elde edilmiş ve önerilen algoritma ile elde edilen çözümlerin %0.47-%2.61 arasında bir oranda sapma gösterdiği belirlenmiştir.



Şekil 5.19 23 problem seti için iyileştirilmiş sonuçlar ile literatürde var olan sonuçların karşılaştırılması

5.7 Uygulama Sonuçları

Yapılan bu doktora tez çalışması, optimizasyon problemlerine ışık tutan iki metasezgisel yöntemi, Genetik Algoritmaları ve Karınca Kolonisi Optimizasyonunu göz önüne olarak yeni bir melez algoritma tabanlı yapının oluşturulmasını sağlamış ve oluşturulan bu yeni melez yapı günlük hayatımızda sıklıkla karşımıza çıkan çok depolu araç rotalama probleminin çözümünde kullanılmıştır.

Önerilen çözüm yaklaşımı iki aşamalı olarak gerçekleştirilmiş ve tez çalışması kapsamında çok depolu araç rotalama probleminin çözümü için önce grupla sonra rotala yaklaşımı kullanılmıştır. Çözüm yaklaşımının birinci aşamasında hangi müşterilerin hangi depolardan hizmet alacağının belirlendiği gruplama işlemi genetik algoritma ile yapılırken, ikinci aşamada ise hangi depodan hizmet alacağı belirlenmiş olan müşterilerin o depo içinde rotalanması karınca kolonisi optimizasyonu ile gerçekleştirilmiştir.

Önerilen çözüm yaklaşımının uygulandığı çok depolu araç rotalama probleminde müşterilere hizmet sunan birden fazla sayıda depo olduğu belirlenmiş ve problemin çözümü için amaç fonksiyonu her deponun kullanılarak tüm müşterilere sunulacak olan hizmetin minimum toplam seyahat mesafesi ile yapılabilmesi olarak yapılandırılmıştır. Burada, her müşterinin sadece bir depodan hizmet alabildiği, her bir müşterinin sadece bir araç tarafından ziyaret edilebildiği, müşterilerin sahip oldukları taleplerin bir kerede karşılanabildiği yani müşteri taleplerinin bölünmesi ve parça parça karşılanmasının mümkün olmadığı ve her aracın bir depodan seyahatine başladığı ve seyahati sonunda da müşterilere hizmet ilettikten sonra yine aynı depoya döndüğü kabul edilmiştir.

Çalışma kapsamında ortaya konan yeni algoritma yapısı literatürde var olan 23 çok depolu araç rotalama problem seti üzerinde test edilmiş ve bu çalışma ile elde edilen sonuç ve değerlendirmeler aşağıdaki gibi sıralanmıştır:

- Önerilen algoritma yapısının birinci aşamasında hangi müşterilerin hangi depolardan hizmet alacağının belirlenmesinde kullanılan genetik algoritma ile elde edilen değerler en yakın komşu sezgiseli ile karşılaştırılmış ve önerilen çözüm yöntemiyle genetik algoritma parametrelerinden biri olan çaprazlama oranı %60 civarında seçildiğinde en iyi sonuçları verdiği bulunmuştur.

- Önerilen algoritma yapısının ikinci aşama işlem adımlarının gerçekleştirilmesi ile birlikte elde edilen nihai çözümlerden 5 tanesinin literatürde var olan aynı problem setleri için bilinen sonuçlardan daha iyi sonuçlar verdiği bulunmuştur. Bu 5 problem setinin daha çok orta sayıda müşterisi bulunan ve araç kapasitesinin düşük olduğu problem grupları olduğu ortaya konmuştur. Diğer problem setleri için de önerilen algoritma ile elde edilen çözümlerin %0.30-%3.41 arasında sapma gösterdiği belirlenmiştir.
- Karınca kolonisi optimizasyonunda yoldaki feromon miktarının önemini gösteren α parametresi baz alındığı 1 değerinden üst sınırı olan 5 değerine doğru yükseltilirken programın ürettiği sonuç çıktı değerlerinin küçüldüğü dolayısıyla α değerinin büyük seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu bulunmuştur.
- Karınca kolonisi optimizasyonunda sezgiselliğin önemini gösteren ve karıncaları yakın mesafedeki şehirlere gitmeleri yönünde etkileyen β parametresi baz alındığı 5 değerinden alt sınırı olan 1 değerine doğru indirilirken programın ürettiği sonuç çıktı değerlerinin küçüldüğü dolayısıyla β değerinin küçük seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu elde edilmiştir.
- Karınca kolonisi optimizasyonunda karıncaların geçtiği yollara bıraktıkları feromon maddesinin sınırsız büyümesini engellemek için kullanılan ρ buharlaşma katsayısı parametresi baz alındığı 0.1 değerinden üst sınırı olan 1.0 değerine doğru yükseltilirken programın ürettiği sonuç çıktı değerlerinin ondalık değerler oranında artıp azaldığı fakat grafiksel olarak tanımlanacak düzgün bir değişim göstermediği sonucu elde edilmiş ve ρ buharlaşma katsayısı parametresinin toplam mesafe sonuç değeri üzerinde anlamlı bir etkisi olmadığı ortaya konmuştur.
- Karınca kolonisi optimizasyonunda karıncaların gideceği şehri seçmelerini sağlayan sözde-rastlantısal-orantılı geçiş kuralında kullanılacak olan formülasyonu belirlemek için program tarafından üretilen rastlantısal q değeri ile karşılaştırılan q_o parametresi baz alındığı 0.9 değerinden alt sınırı olan 0.0

değerine doğru indirilirken programın ürettiği sonuç çıktı değerlerinin küçüldüğü dolayısıyla q_0 değerinin küçük seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu elde edilmiştir.

- Karınca kolonisi optimizasyonunda algoritma adımlarının tekrarlanacağı maksimum sayıyı gösteren iterasyon sayısı parametresi baz alındığı 2000 değerinin altında bulunan 500 değerine üzerinde de 3500 değerine kadar çalıştırıldığında programın 2000'in altındaki iterasyon sayılarında daha yüksek, 2000'in üzerindeki iterasyon sayılarında ise daha düşük sonuç çıktı değerleri ürettiği gözlenmiş ve iterasyon sayısı değerinin 2000'den büyük 3000 değeri civarında seçilmesinin problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucu bulunmuştur.
- Sonuç değerleri üzerinde anlamlı ve büyük bir etkisi olmayan ρ parametresi 0.1 değerine, iterasyon sayısı parametresi de 3000 değerine sabitlenerek α, β ve q_0 parametreleri üzerinde yapılan eşzamanlı-çapraz değişimler ile yeni çözümler elde edilmiş ve önerilen algoritma yapısında parametre değerleri $\alpha = 5$, $\beta = 1$ ve $q_0 = 0.1$ olarak seçildiğinde problemin çözüm amacı olan daha düşük mesafeyi sağladığı sonucuna varılmıştır.
- Yapılan eşzamanlı-çapraz parametre değişimleri ile literatürde var olan 23 problem setinin 15 tanesi için bilinen iyi sonuçlardan daha iyi çözümler elde edildiği ve önerilen algoritma ile elde edilen çözümlerin %0.47-%2.61 arasında bir oranda sapma gösterdiği bulunmuştur. Bu 15 problem setinin küçük ve orta büyüklükteki müşteri sayısı olan problem grupları olduğu ortaya konmuştur.

Bu sonuç ve değerlendirmelerden hareketle, genetik algoritma ve karınca kolonisi optimizasyonunun bir araya getirilmesi ile oluşturulan algoritma yapısı müşteri sayısı, depo sayısı, müşteri ve depo konumları, araç kapasitesi gibi farklı özellikteki 23 problem setine uygulanmış ve önerilen çözüm yönteminin küçük ve orta büyüklükteki müşteri sayılarına sahip ve araç kapasitesi nispeten daha küçük olan problem setleri için iyi çözümler ürettiği gerçeği kabul edilmiştir.

BÖLÜM 6

SONUÇ

Gerçek yaşam problemlerinin çoğunda problemin çözüm uzayı sonsuz veya tüm çözümlerin değerlendirilemeyeceği kadar büyük olur. Bunun için kabul edilebilir bir zaman dilimi içinde ortaya konacak olan çözümlerin değerlendirilerek iyi bir çözümün bulunması gerekmektedir. Bu amaçla ortaya konan tekniklerden ikisi olan Genetik Algoritma ve Karınca Kolonisi Optimizasyonu bu doktora tez çalışmasının konusudur. Çalışmada, ana amaç doğrultusunda, literatürde var olan metasezgisel yöntemlerin kullanım yerleri, özellikleri ve algoritma yapıları incelenmiş, literatürde var olmayan yeni bir melez algoritma yapısı ortaya konmuş ve bu melez yapının çok depolu araç rotalama problemine uygulanması ile de bulunan algoritma test edilmiş ve algoritmanın gerçekliği kanıtlanmıştır.

Optimizasyon problemlerine çözüm arayan ve arama sürecine rehberlik eden stratejilerden olan genetik algoritma ve karınca kolonisi optimizasyonu algoritmalarının literatürde farklı problem türleri için uygulamaları vardır. Bu problemlere örnek olarak gezgin satıcı problemleri, araç rotalama problemleri ve çizelgeleme problemleri verilebilir. Yapılan literatür taraması ışığında esneklik ve oluşabilecek değişikliklere kolayca adapte edilebilen karınca kolonisi optimizasyonu ile arama uzayında en iyinin hayatta kalması ilkesine göre en iyi çözümü arayan genetik algoritma metasezgisellerinin optimizasyon problemleri arasında en az araç rotalama problemlerine uygulandığı saptanmıştır. Buradan hareketle, tez çalışmasının bu problem türü üzerine yoğunlaştırılması uygun görülmüştür.

Literatürde araç rotalama problemlerini çözmek için geliştirilmiş çok sayıda algoritma mevcuttur. Fakat uzun yıllardır yapılan araştırma çalışmaları ile elde edilen tüm problem yapılarına uygun optimum bir çözüm bulunamadığından bu yöndeki çalışmalar hızla devam etmektedir. Bu tez çalışması kapsamında da araştırmalar, aynı problem türüne ayrı ayrı uygulanmasına rağmen her iki tekniğin bir arada kullanılarak melez bir algoritma yapısının ortaya konmadığı genetik algoritma ve karınca kolonisi optimizasyonu metasezgiselleri ile devam etmiştir.

Çalışmada birden fazla sayıda depodan müşterilere hizmet sunan bir araç rotalama problemi çeşidi olan çok depolu araç rotalama problemi ele alınmış ve önerilen algoritma literatürde var olan bilinen en iyi sonuçlara sahip müşteri sayısı, depo sayısı ve araç kapasitesi özellikleri ile birbirinden farklılaşan 23 problem seti için çalıştırılmıştır.

Önerilen algoritma ile çalıştırılan çözüm prosesi iki aşamadan oluşturularak ilk aşamada gruplama işlemi yapılmış ve gruplama işlemi ile hangi müşterilerin hangi depodan hizmet alacağı genetik algoritma ile belirlenmiştir. Çözüm prosesinin ikinci aşamasında ise rotalama işlemi yapılmış ve rotalama işlemi ile müşterilerin hangi sıra ile hizmet alacakları karınca kolonisi optimizasyonu ile belirlenmiştir. Önerilen çözüm yöntemi C# programlama dili ile yazılmış, tüm veriler SQL programında tutulmuş ve program çözüm üretmek için Intel® Core™ i7 extreme edition dört çekirdekli, 1 GB® ayrılmış bellekle NVIDIA Quadro FX3800M grafik kartı, 8 GB1333 MHz çift kanallı DDR3 bellek ve 500 GB 7200 RPM harddiske sahip olan bir bilgisayarda çalıştırılmıştır.

Önerilen, çalıştırılan ve kullanıcı arayüzünde bulunan parametre değerlerinin değiştirilmesi ile iyileştirilen çözümler ile literatürde var olan 23 problem setinin 15 tanesi için bilinen en iyi sonuçlardan daha iyi sonuç değerleri elde edilmiş ve diğer 8 problem seti için de bilinen en iyi sonuçlara %0.47-%2.61 sapmayla yakın sonuçlar bulunmuştur.

Sonuç olarak, tedarik zinciri içerisinde ve lojistikte önemli bir karar alanı olan taşıma ve ürünlerin müşterilere ulaştırılması konusunda yapılan bu tez çalışmasında genetik algoritma ve karınca kolonisi optimizasyonu metasezgisellerinin bir araya getirilmesi ile ortaya konan yeni algoritma yapısının küçük ve orta büyüklükteki müşteri sayılarına

sahip ve araç kapasitesi nispeten daha küçük olan problem setleri için iyi çözümler ürettiği ortaya konmuştur.

Çalışma, yapılabilecek parametre değişiklikleri ile hem farklı araç rotalama problemlemi çeşitlerine hem de farklı optimizasyon problemlerine uyarlanabilecek niteliktedir.

KAYNAKLAR

[1] Kahaner, D., Moler, C. ve Nash, S., (1989). Numerical Methods and Software, Englewood Cliffs, NJ:Prentice-Hall.

[2] Bhatti, M.A., (2000). Practical Optimization Methods with Mathematica Applications, Springer-Verlag New York, Inc.

[3] Collette, Y. ve Siarry, P., (2003). Multiobjective Optimization Principles and Case Studies, Springer-Verlag Berlin Heidelberg.

[4] Deb, K., (2004). Optimization for Engineering Design: Algorithms and Examples, Prentice Hall, New Delhi, India.

[5] Atlas, M., (2008). "Çok Amaçlı Programlama Çözüm Tekniklerinin Sınıflandırılması", Anadolu Üniversitesi Sosyal Bilimler Dergisi, 8(1):47-68.

[6] Ramkumar, B., Schoen, M.P. ve Lin, F., (2011). "Hybrid Enhanced Continuous Tabu Search and Genetic Algorithm for Parameter Estimation in Colored Noise Environments", Expert Systems with Applications, 38:3909–3917.

[7] Yu, S., Ding, C. ve Zhu, K., (2011). "A Hybrid GA-TS Algorithm for Open Vehicle Routing Optimization of Coal Mines Material", Expert Systems with Applications, doi: 10.1016/j.eswa.2011.02.108.

[8] Tseng, L.-Y. ve Chen S.-C., (2006). "A Hybrid Metaheuristic for the Resource-Constrained Project Scheduling Problem", European Journal of Operational Research, 175:707-721.

[9] Lee, Z.-J., Su, S.-F., Chuang, C.-C. ve Liu, K.-H., (2008). "Genetic Algorithm with Ant Colony Optimization (GA-ACO) for Multiple Sequence Alignment", Applied Soft Computing, 8:55-78.

[10] Kuo, R.J. ve Lin, L.M., (2010). "Application of a Hybrid of Genetic Algorithm and Particle Swarm Optimization Algorithm for Order Clustering", Decision Support Systems, 49:451–462.

[11] Abd-El-Wahed, W.F., Mousa, A.A. ve El-Shorbagy, M.S., (2011). "Integrating Particle Swarm Optimization with Genetic Algorithms for Solving Nonlinear

Optimization Problems", Journal of Computational and Applied Mathematics, 235:1446–1453.

[12] Chen, P.-H. ve Shahandashti, S.M., (2009). "Hybrid of Genetic Algorithm and Simulated Annealing for Multiple Project Scheduling with Multiple Resource Constraints", Automation in Construction, 18:434–443.

[13] Shen, Q., Shi, W.-M. ve Kong, W., (2008). "Hybrid Particle Swarm Optimization and Tabu Search Approach for Selecting Genes for Tumor Classification Using Gene Expression Data", Computational Biology and Chemistry, 32:53–60.

[14] Zhang, G., Shao, X., Li, P. ve Gao, L., (2009). "An Effective Hybrid Particle Swarm Optimization Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem, Computers & Industrial Engineering, 56:1309–1318.

[15] Ongsakul, W. ve Bhasaputra, P., (2002). "Optimal Power Flow with FACTS Devices by Hybrid TS/SA Approach, Electrical Power and Energy Systems, 24:851-857.

[16] Swarnkar, R. ve Tiwari, M.K., (2004). "Modeling Machine Loading Problem of FMSs and its Solution Methodology Using a Hybrid Tabu Search and Simulated Annealing-Based Heuristic Approach", Robotics and Computer-Integrated Manufacturing, 20:199–209.

[17] Ji, J., Hu, R., Zhang, H. ve Chunnian, L., (2011). "A Hybrid Method for Learning Bayesian Networks Based on Ant Colony Optimization", Applied Soft Computing, doi:10.1016/j.asoc.2011.01.009.

[18] Shelokar, P.S., Siarry, P., Jayaraman, V.K. ve Kulkarni, B.D., (2007). "Particle Swarm and Ant Colony Algorithms Hybridized for Improved Continuous Optimization". Applied Mathematics and Computation, 188:129-142.

[19] He, Q. ve Wang, L., (2007). "A Hybrid Particle Swarm Optimization with a Feasibility-Based Rule for Constrained Optimization", Applied Mathematics and Computation, 186:1407–1422.

[20] Liu, B., Wang, L., Liu, Y., Qian, B. ve Jin, Y.-H., (2010). "An Effective Hybrid Particle Swarm Optimization for Batch Scheduling of Polypropylene Processes", Computers and Chemical Engineering, 34:518–528.

[21] Cura, T., (2008). Modern Sezgisel Teknikler ve Uygulamarı, Papatya Yayıncılık Eğitim, 1. Basım, İstanbul.

[22] Gen, M. ve Cheng, R., (2000). "Genetic Algorithms and Engineering Optimization", A Wiley-Interscience Publication, John-Wiley & Sons, Inc.

[23] Toğan, V. ve Daloğlu, A., (2006). "Genetik Algoritma ile Üç Boyutlu Kafes Sistemlerin Şekil ve Boyut Optimizasyonu", İMO Teknik Dergi, 251:3809-3825.

[24] Mitchell, M., (1996). An Introduction to Genetic Algorithms, A Bradford Book, The MIT Pres, Cambridge, Massachusetts, Londra, İngiltere.

[25] Wang, C.-H. ve Lu, J.-Z., (2009). "A Hybrid Genetic Algorithm that Optimizes Capacitated Vehicle Routing Problems", Expert Systems with Application, 36:2921–2936.

[26] Er, H., Çetin, M.K. ve Çetin, E.İ., (2005). "Finansta Evrimsel Algoritmik Yaklaşımlar: Genetik Algoritma Uygulamaları", Akdeniz İ.İ.B.F. Dergisi, 10:73-94.

[27] Michalewicz, Z., (1996). Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag Berlin Heidelberg, New York.

[28] Emel, G.G. ve Taşkın, Ç., (2002). "Genetik Algoritmalar ve Uygulama Alanları", Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, (21), 1:129-152.

[29] Yapay Zeka-Org, http://www.yapay-zeka.org/modules/wiwimod/index.php?page =GA, 25.11.2010.

[30] Eggermont, J., (2007). Genetic Programming, http://www.win.tue.nl/ipa/archive/falldays2007/HandoutEggermont.pdf, 31.08.2010.

[31] Kuo, R.J. ve Lin, L.M., (2010). "Application of a Hybrid of Genetic Algorithm and Particle Swarm Optimization Algorithm for Order Clustering", Decision Support Systems, 49(4):451-462.

[32] Marel Obitko sitesi, http://www.obitko.com/tutorials/genetic-algorithms/selection.php, 12.09.2010.

[33] Engin, O. ve Fığlalı, A., (2002). "Akış Tipi Çizelgeleme Problemlerinin Genetik Algoritma Yardımı ile Çözümünde Uygun Çaprazlama Operatörünün Belirlenmesi", Doğuş Üniversitesi Dergisi, 2002/6:27-35.

[34] Kaya, İ., (2006). "Çok Aşamalı Proseslerde Örnek Hacminin Belirlenmesi Üzerine Bir Model ve Genetik Algoritmalar Yardımıyla Çözüm Önerisi", Selçuk Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, 15:435-457.

[35] Karaboğa, D., (2004). Yapay Zeka Optimizasyon Algoritmaları, Atlas Yayın Dağıtım, 1. Basım, İstanbul.

[36] Akcayol, M.A., (2008). Zeki Optimizasyon Teknikleri–Genetik Algoritma, Galatasaray Üniversitesi Bilgisayar Mühendisliği Bölümü, Ders Notları, İstanbul.

[37] Samanlıoğlu, F., Ferrell, W.G.Jr. ve Kurz, M.E., (2008). "A Memetic Random-Key Genetic Algorithm for a Symmetric Multi-Objective Traveling Salesman Problem", Computers and Industrial Engineering, 55:439-449.

[38] Yang, J., Wu, C., Lee, H.P. ve Liang, Y., (2008). "Solving Traveling Salesman Problems Using Generalized Chromosome Genetic Algorithm", Progress in Natural Science, 18:887-892.

[39] Xing, L.-N., Chen, Y.-W., Yang, K.-W., Hou, F., Shen, X.-S. ve Cai, H.-P., (2008). "A Hybrid Approach Combining an Improved Genetic Algorithm and Optimization Strategies for the Asymmetric Traveling Salesman Problem", Engineering Applications of Artificial Intelligence, 21:1370-1380.

[40] Jeon, G., Leep, H.R. ve Shim, J.Y., (2007). "A Vehicle Routing Problem Solved by Using a Hybrid Genetic Algorithm", Computers and Industrial Engineering, 53:680-692.

[41] Bae, S.-T., Hwang, H.S., Cho, G.-S. ve Goan, M.-J., (2007). "Integrated GA-VRP Solver for Multi-Depot System", Computers and Industrial Engineering, 53:233-240.

[42] Alba, E. ve Dorronsoro, B., (2006). "Computing Nine New Best-So-Far Solutions for Capacitated VRP with a Cellular Genetic Algorithm", Information Processing Letters, 98:225-230.

[43] Cheng, C.-B. ve Wang, K.-P., (2009). "Solving a Vehicle Routing Problem with Time Windows by a Decomposition Technique and a Genetic Algorithm", Expert Systems with Applications, 36:7758–7763.

[44] Liu, L. ve Li, Y., (2006). "The Fuzzy Quadratic Assignment Problem with Penalty: New Models and Genetic Algorithm", Applied Mathematics and Computation, 174:1229-1244.

[45] Drezner, Z., (2008). "Extensive Experiments with Hybrid Genetic Algorithms for the Solution of the Quadratic Assignment Problem", Computers and Operations Research, 35:717-736.

[46] Pezzella, F., Morganti, G. ve Ciaschetti, G., (2008). "A Genetic Algorithm for the Flexible Job-Shop Scheduling Problem", Computers and Operations Research, 35:3202-3212.

[47] Gao, J., Sun, L. ve Gen, M., (2008). "A Hybrid Genetic and Variable Neighborhood Descent Algorithm for Flexible Job Shop Scheduling Problems", Computers and Operations Research, 35:2892-2907.

[48] Essafi, I., Mati, Y. ve Dauzere-Peres, S., (2008). "A Genetic Local Search Algorithm for Minimizing Total Weighted Tardiness in the Job-Shop Scheduling Problem", Computers and Operations Research, 35:2599-2616.

[49] Vilcot, G. ve Billaut, J.-C., (2008). "A Tabu Search and a Genetic Algorithm for Solving a Bicriteria General Job Shop Scheduling Problem", European Journal of Operational Research, 190:398–411.

[50] Li, M., Yu, B. ve Qi, M., (2006). "PGGA: A Predictable and Grouped Genetic Algorithm for Job Scheduling", Future Generation Computer Systems, 22:588-599.

[51] Chou, F.-D., (2009). "An Experienced Learning Genetic Algorithm to Solve the Single Machine Total Weighted Tardiness Scheduling Problem", Expert Systems with Applications, 36:3857–3865

[52] Chang, P.-C., Hsieh, J.-C. ve Liu, C.-H., (2006a). "A Case-Injected Genetic Algorithm for Single Machine Scheduling Problems with Release Time", International Journal of Production Economics, 103:551-564.

[53] Chang, P.C., Chen, S.H. ve Mani, V., (2009). "A Hybrid Genetic Algorithm with Dominance Properties for Single Machine Scheduling with Dependent Penalties", Applied Mathematical Modelling, 33:579–596.

[54] Kılıç, M., Ulusoy, G. ve Şerifoğlu, F.S., (2008). "A Bi-Objective Genetic Algorithm Approach to Risk Mitigation in Project Scheduling", International Journal of Production Economics, 112:202-216.

[55] Shadrokh, S. ve Kianfar, F., (2007). "A Genetic Algorithm for Resource Invesment Project Scheduling Problem, Tardiness Permitted with Penalty", European Journal of Operational Research, 181(1):86-101.

[56] Gonçalves, J.F., Mendes, J.J.M. ve Resende, M.G.C., (2008). "A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem", European Journal of Operational Research, 189:1171-1190.

[57] Valls, V., Ballestin, F. ve Quintanilla, S., (2008). "A Hybrid Genetic Algorithm for the Resource-Constrained Project Scheduling Problem", European Journal of Operational Research, 185:495-508.

[58] Nagano, M.S., Ruiz, R. ve Lorena, L.A.N., (2008). "A Constructive Genetic Algorithm for Permutation Flowshop Scheduling", Computers and Industrial Engineering, 55:195–207.

[59] Chen, J.-S., Pan, J.C.-H. ve Lin, C.-M., (2008). "A Hybrid Genetic Algorithm for the Re-Entrant Flow-Shop Scheduling Problem", Expert Systems with Applications, 34:570-577.

[60] Cheng, B.-W. ve Chang, C.-L., (2007). "A Study on Flowshop Scheduling Problem Combining Taguchi Experimental Design and Genetic Algorithm", Expert Systems with Applications, 32:415-421.

[61] Mabrouk, B.B., Hasni, H. ve Mahjoub, Z., (2009). "On a Parallel Genetic-Tabu Search Based Algorithm for Solving the Graph Colouring Problem", European Journal of Operational Research, 197:1192-1201.

[62] Lu, Z. ve Hao, J.-K., (2010). "A Memetic Algorithm for Graph Coloring", European Journal of Operational Research, 203:241–250.

[63] Lin, C.-M. ve Gen, M., (2008). "Multi-Criteria Human Resource Allocation for Solving Multi-Stage Combinatorial Optimization Problems Using Multi-Objective Hybrid Genetic Algorithm", Expert Systems with Applications, 34:2480-2490.

[64] Lee, K.-Y., Roh, M.-I. ve Jeong, H.-S., (2005). "An Improved Genetic Algorithm for Multi-Floor Facility Layout Problems Having inner Structure Walls and Passages", Computers and Operations Research, 32:879-899.

[65] Wang, M.-J., Hu, M.H., ve Ku, M.-Y., (2005). "A Solution to the Unequal Area Facilities Layout Problem by Genetic Algorithm", Computers in Industry, 56:207-220.

[66] Aiello, G., Enea, M. ve Galante G., (2006). "A Multi-Objective Approach to Facility Layout Problem by Genetic Search Algorithm and Electre Method", Robotics and Computer-Integrated Manufacturing, 22:447-455.

[67] Levitin, G., Rubinovitz, J. ve Shnits, B., (2006). "A Genetic Algorithm for Robotic Assembly Line Balancing", European Journal of Operational Research, 168:811-825.

[68] Kim, Y.K., Song, W.S. ve Kim J.H., (2009). "A Mathematical Model and a Genetic Algorithm for Two-Sided Assembly Line Balancing", Computers and Operations Research, 36:853-865.

[69] Mansouri, S.A., (2005). "A Multi-Objective Genetic Algorithm for Mixed-Model Sequencing on JIT Assembly Lines", European Journal of Operational Research, 167:696-716.

[70] Defersha, F.M. ve Chen, M., (2008). "A Linear Programming Embedded Genetic Algorithm for an İntegrated Cell Formation and Lot Sizing Considering Product Quality", European Journal of Operational Research, 187:46-69.

[71] Chang, P.-T., Yao, M.-J., Huang, S.-F. ve Chen, C.-T., (2006b). "A Genetic Algorithm for Solving a Fuzzy Economic Lot-Size Scheduling Problem", International Journal of Production Economics, 102:265-288.

[72] Dorigo, M. ve Stützle, T., (2004). Ant Colony Optimization, Massachusetts Institute of Technology, The MIT Press.

[73] Yağmahan, B. ve Yenisey, M.M., (2008). "Ant Colony Optimization for Multi-Objective Flow Shop Scheduling Problem", Computers and Industrial Engineering, 54:411–420.

[74] Dalkılıç, G. ve Türkmen, F., (2002). "Karınca Kolonisi Optimizasyonu", Yüksek Performanslı Bilişim Sempozyumu-YPBS'2002, Bildiriler Kitabı, Kocaeli.

[75] Dorigo, M., Di Caro, G. ve Gambardella, L.M., (1999). "Ant Algorithms for Discrete Optimization", Artificial Life, 5(2):137-172.

[76] Alaykıran, K. ve Engin, O., (2005). "Karınca Kolonileri Metasezgiseli ve Gezgin Satıcı Problemleri Üzerinde Bir Uygulama", Gazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi, 1:69-76.

[77] Rizzoli, A.E., Oliverio, F., Montemanni, R. ve Gambardella, L.M., (2005). Ant Colony Optimisation for Vehicle Routing Problems: From Theory to Applications, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA-report).

[78] Dorigo, M., Maniezzo, V. ve Colorni, A., (1996). "The Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):1-13.

[79] Gambardella, L.M. ve Dorigo, M., (1995). "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem", 12th International Conference on Machine Learning, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann, 252-260.

[80] Gambardella, L.M. ve Dorigo, M., (2000). "An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem", INFORMS Journal on Computing, 12(3):237-255.

[81] Söyler, H. ve Keskintürk, T., (2007). "Karınca Kolonisi Algoritması ile Gezen Satıcı Probleminin Çözümü", 8. Türkiye Ekonometri ve İstatistik Kongresi, İnönü Üniversitesi, Malatya.

[82] Maniezzo, V., Gambardella, L.M. ve De Luigi F., (2003). "Ant Colony Optimization", New Optimization Techniques in Engineering, by Onwubolu, G. C., and B. V. Babu, Springer-Verlag Berlin Heidelberg, 101-117.

[83] Reimann, M., Doerner, K. ve Hartl, R.F., (2004). "D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem", Computers and Operations Research, 31:563-591.

[84] Keskintürk, T. ve Söyler, H., (2006). "Global Karınca Kolonisi Optimizasyonu", Gazi Üniversitesi Mühendislik ve mimarlık Fakültesi Dergisi, 21(4):689-698.

[85] Sitarz, S., (2009). "Ant Algorithms and Simulated Annealing for Multicriteria Dynamic Programming", Computers and Operations Research, 36:433-441.

[86] McKendall, A.R.Jr. ve Shang, J., (2006). "Hybrid Ant Systems for the Dynamic Facility Layout Problem", Computers and Operations Research, 33:790-803.

[87] Huang, K.-L. ve Liao, C.-L., (2008). "Ant Colony Optimization Combined with Taboo Search for the Job Shop Scheduling Problem", Computers and Operations Research, 35:1030-1046.

[88] Dorigo, M. ve Gambardella, L.M., (1997). "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computation, 1(1):53-66.

[89] Chen, C.-H. ve Ting, C.-J., (2005). "A Hybrid Ant Colony System for Vehicle Routing Problem with Time Windows", Journal of the Eastern Asia Society for Transportation Studies, 6:2822-2836.

[90] Baykoç, Ö.F. ve İşleyen, S.K., (2007). "Simetrik Gezgin Satıcı Problemi için Etkin Bir Tekrarlı Yerel Arama", Teknoloji Dergisi, 10(2):99-106.

[91] Demirten, M., (2004). Heuristic Algoritmalar, Gebze İleri Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümü, Analysis of Algorithms ders notu, http://www.bilmuh.gyte.edu.tr/BIL523/presentations/mdemirten/heuristic.ppt, 08.09.2009.

[92] Garcia-Martinez, C., Cordon, O. ve Herrera, F., (2007). "A Taxonomy and an Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-Criteria TSP", European Journal of Operational Research, 180:116-148.

[93] Cheng, C.-B. ve Mao, C.-P., (2007). "A Modified Ant Colony System for Solving the Travelling Salesman Problem with Time Windows", Mathematical and Computer Modelling, 46:1225-1235.

[94] Donati, A.V., Montemanni, R., Casagrande, N., Rizzoli, A.E. ve Gambardella, L.M., (2008). "Time Dependent Vehicle Routing Problem with a Multi Ant Colony System", European Journal of Operational Research, 185:1174-1191.

[95] Mazzeo, S. ve Loiseau, I., (2004). "An Ant Colony Algorithm for the Capacitated Vehicle Routing", Electronic Notes in Discrete Mathematics, 18:181-186.

[96] Gajpal, Y. ve Abad, P.L., (2009). “Multi-Ant Colony System (MACS) for a Vehicle Routing Problem with Backhauls”, European Journal of Operational Research, 196:102-117.

[97] Fuellerer, G., Doerner, K.F., Hartl, R.F. ve Iori, M., (2009). “Ant Colony Optimization for the Two-Dimensional Loading Vehicle Routing Problem”, Computers and Operations Research, 36:655-673.

[98] Zhang, T., Tian, W.-X., Zhang, Y.-J. ve Liu, S.-X., (2008). “Improved Ant Colony System for VRPSPD with Maximum Distance Constraint”, Systems Engineering - Theory and Practice, 28(1):132-140.

[99] Gajpal, Y. ve Abad, P.L., (2009). “An Ant Colony System (ACS) for Vehicle Routing Problem with Simultaneous Delivery and Pickup”, Computers and Operations Research, 36:3215-3223.

[100] Saremi, H.Q., Abedin, B. ve Kermani, A.M., (2008). “Website Structure Improvement: Quadratic Assignment Problem Approach and Ant Colony Meta-Heuristic Technique”, Applied Mathematics and Computation, 195:285-298.

[101] Demirel, N.Ç. ve Toksarı, M.D., (2006). “Optimization of the Quadratic Assignment Problem Using an Ant Colony Algorithm”, Applied Mathematics and Computation, 183:427-435.

[102] Silva, C.A., Sousa, J.M.C. ve Runkler, T.A., (2008). “Rescheduling and Optimization of Logistic Processes Using GA and ACO”, Engineering Applications of Artificial Intelligence, 21:343–352.

[103] Seçkiner, S.U. ve Kurt, M., (2008). “Ant Colony Optimization for the Job Rotation Scheduling Problem”, Applied Mathematics and Computation, 201:149–160.

[104] Silva, C.A., Sousa, J.M., Runkler, T. ve Sa da Costa, J.M.G., (2005). “A Logistic Process Scheduling Problem: Genetic Algorithms or Ant Colony Optimization”, International Federation of Accountants, 1-6.

[105] Heinonen, J. ve Pettersson, F., (2007). “Hybrid Ant Colony Optimization and Visibility Studies Applied to a Job-Shop Scheduling Problem”, Applied Mathematics and Computation, 187:989-998.

[106] Rossi, A. ve Dini, G., (2007). “Flexible Job-Shop Scheduling with Routing Flexibility and Separable Setup Times Using Ant Colony Optimisation Method”, Robotics and Computer-Integrated Manufacturing, 23:503-516.

[107] Liao, C.-J. ve Juan, H.-C., (2007). “An Ant Colony Optimization for Single-Machine Tardiness Scheduling with Sequence-Dependent Setups”, Computers and Operations Research, 34:1899-1909.

[108] Cheng, T.C.E., Lazarev, A.A. ve Gafarov, E.R., (2009). “A Hybrid Algorithm for the Single-Machine Total Tardiness Problem”, Computers and Operations Research, 36:308-315.

[109] Gajpal, Y. ve Rajendran, C., (2006). “An Ant-Colony Optimization Algorithm for Minimizing the Completion-Time Variance of Jobs in Flowshops”, International Journal of Production Economics, 101:259-272.

[110] Yağmahan, B. ve Yenisey, M.M., (2006). “Akış Tipi Çizelgeleme Problemi için KKE Parametre Eniyileme”, İTÜ Dergisi/D Mühendislik, (5), Kısım 2(2):133-141.

[111] Lin, B.M.T., Lu, C.Y., Shyu, S.J. ve Tsai, C.Y., (2008). “Development of New Features of Ant Colony Optimization for Flowshop Scheduling”, International Journal of Production Economics, 112:742–755.

[112] Bui, T.N., Nguyen, T.V.H., Patel, C.M. ve Phan, K.-A.T., (2008). “An Ant-Based Algorithm for Coloring Graphs”, Discrete Applied Mathematics, 156:190-200.

[113] Dowsland, K.A. ve Thompson, J.M., (2008). “An Improved Ant Colony Optimisation Heuristic for Graph Colouring”, Discrete Applied Mathematics, 156:313-324.

[114] Chaharsooghi, S.K. ve Kermani, A.H.M., (2008). “An Effective Ant Colony Optimization Algorithm (ACO) for Multi-Objective Resource Allocation Problem (MORAP)”, Applied Mathematics and Computation, 200:167–177.

[115] Yin, P.-Y. ve Wang, J.-Y., (2006). “Ant Colony Optimization for the Nonlinear Resource Allocation Problem”, Applied Mathematics and Computation, 174:1438-1453.

[116] Montemanni, R., Smith, D.H. ve Gambardella, L.M., (2008). “A Heuristic Manipulation Technique for the Sequential Ordering Problem”, Computers and Operations Research, 35:3931-3944.

[117] Baykasoğlu, A., Dereli, T. ve Sabuncu, İ., (2006). “An Ant Colony Algorithm For Solving Budget Constrained and Unconstrained Dynamic Facility Layout Problems”, The International Journal of Management Science-Omega, 34:385-396.

[118] Solimanpur, M., Vrat, P. ve Shankar, R., (2005). “An Ant Algorithm for the Single Row Layout Problem in Flexible Manufacturing Systems”, Computers and Operations Research, 32:583-598.

[119] Hani, Y., Amodeo, L., Yalaoui, F. ve Chen, H., (2007). “Ant Colony Optimization for Solving an Industrial Layout Problem”, European Journal of Operational Research, 183:633-642.

[120] Simaria, A.S. ve Vilarinho, P.M., (2009). “2-ANTBAL: An Ant Colony Optimisation Algorithm for Balancing Two-Sided Assembly Lines”, Computers and Industrial Engineering, 56:489–506.

[121] Bautista, J. ve Pereira, J., (2007). “Ant Algorithms for a Time and Space Constrained Assembly Line Balancing Problem”, European Journal of Operational Research, 177:2016-2032.

[122] Pitakaso, R., Almeder, C., Doerner, K.F. ve Hartl, R.F., (2007). “A MAX-MIN Ant System for Unconstrained Multi-Level Lot-Sizing Problems”, Computers and Operations Research, 34:2533-2552.

[123] Su, C.-T. ve Wong, J.-T., (2008). “Design of a Replenishment System for a Stochastic Dynamic Production/Forecast Lot-Sizing Problem under Bullwhip Effect”, Expert Systems with Applications, 34:173-180.

[124] Gencer, C. ve Yaşa, Ö., (2007). “Ulaştırma Komutanlığı Ring Seferlerinin Eş Zamanlı Dağıtım Toplama Karar Destek Sistemi”, Gazi Üniversitesi, Mühendislik-Mimarlık Fakültesi Dergisi, 22(3):437-449.

[125] Alabaş, Ç. ve Dengiz, B., (2004). “Yerel Arama Yöntemlerinde Yöre Yapısı: Araç Rotalama Problemine Bir Uygulama”, Yöneylem Araştırması/Endüstri Mühendisliği - XXIV Ulusal Kongresi, Bildiri Kitabı.

[126] Yu, B., Yang, Z. ve Yao, B., (2009). “An Improved Ant Colony Optimization for Vehicle Routing Problem”, European Journal of Operational Research, 196:171–176.

[127] Ombuki, B. ve Hanshar, F., (2004). An Effective Genetic Algorithm for the Multi-Depot Vehicle Routing Problem, Brock University, Department of Computer Science, Technical Report CS-04-10.

[128] Gambardella, L.M., (2000). Vehicle Routing Problems (VRPs), Metaheuristics Network, Technische Universiteit Eindhoven, Kısa ders notları.

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

[130] TMMOB İnşaat Mühendisleri Odası, http://www.e-kutuphane.imo.org.tr/pdf/11451.pdf, 23.09.2010.

[131] Tan, K.C., Lee, L.H. ve Ou, K., (2001). “Heuristic Methods for Vehicle Routing Problem with Time Windows”, Artificial Intelligence in Engineering, 15:281-295.

[132] Montemanni, R., Gambardella, L.M., Rizzoli, A.E. ve Donati A.V., (2002). “A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System”, Technical Report IDSIA-23-02, IDSIA.

[133] Brandao, J., (2004). “A Tabu Search Algorithm for the Open Vehicle Routing Problem”. European Journal of Operational Research, 157:552-564.

[134] Lacomme, P., Prins, C. ve Sevaux, M., (2006). “A Genetic Algorithm for a Bi-Objective Capacitated Arc Routing Problem”, Computers and Operations Research, 33:3473-349.

[135] Toth, P., Vigo, D., (2002). “The Vehicle Routing Problem”, Society for Industrial and Applied Mathematics, Philadelphia.

[136] Bouthillier, A.L. ve Crainic, T.G., (2005). “A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows”, Computers and Operations Research, 32:1685-1708.

[137] Dondo, R. ve Cerda, J., (2007). “A Cluster-Based Optimization Approach for the Multi-Depot Heterogeneous Fleet Vehicle Routing Problem with Time Windows”, European Journal of Operational Research, 176:1478-1507.

[138] Demirel, T., Demirel, N.Ç. ve Taşdelen, B., (2007). "Time Dependent Vehicle Routing Problem with Fuzzy Travelling Times under Different Traffic Conditions", Journal of Multiple-Valued Logic & Soft Compputing, 14:387-400.

[139] Russell, R.A. ve Chiang, W.-C., (2006). "Scatter Search for the Vehicle Routing Problem with Time Windows", European Journal of Operational Research, 169:606-622.

[140] Wang, X. ve Regan, A.C., (2009). "On the Convergence of a New Time Window Discretization Method for the Traveling Salesman Problem with Time Window Constraints", Computers and Industrial Engineering, 56:161–164.

[141] Kallehauge, B., Larsen, J. ve Madsen, O.B.G., (2006). "Lagrangian Duality Applied to the Vehicle Routing Problem with Time Windows", Computers and Operations Research, 33:1464-1487.

[142] NEO Network and Emerging Optimization, http://neo.lcc.uma.es/radi-aeb/WebVRP/, 16.09.2010.

[143] Ganesh, K. ve Narendran, T.T., (2007). "CLOVES: A Cluster-and-Search Heuristic to Solve the Vehicle Routing Problem with Delivery and Pick-Up", European Journal of Operational Research, 178:699-717.

[144] Nagy, G. ve Salhi, S., (2005). "Heuristic Algorithms for Single and Multiple Depot Vehicle Routing Problems with Pickups and Deliveries", European Journal of Operational Research, 162:126-141.

[145] Fan, J., Wang, X. ve Chen, Q., (2007). "A Heuristic Algorithm for Multiple Depots Vehicle Routing Problem with Backhauls", Fourth International Conference on Fuzzy Systems and Knowledge Discovery-FSKD'2007, 0-7695-2874-0/07, IEEE.

[146] Cordeau, J.-F., Gendreau, M. ve Laporte, G., (1997). "A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems", Networks, John Wiley and Sons, Inc., 30:105-119.

[147] Mingozzi, A., (2005). The Multi-Depot Periodic Vehicle Routing Problem, J.-D. Zucker and L. Saitta (Eds.): SARA, Springer-Verlag Berlin Heidelberg, LNAI 3607, 347-350.

[148] Ho, W., Ho, G.T.S., Ji, P. ve Lau, H.C.W., (2008). "A Hybrid Genetic Algorithm for the Multi-Depot Vehicle Routing Problem", Engineering Applications of Artificial Intelligence, 21:548–557.

[149] Irnich, S., (2000). "A Multi-Depot Pickup and Delivery Problem with a Single Hub and Heterogeneous Vehicles", European Journal of Operational Research, 122:310-328.

[150] Ochi, L.S., Vianna, D.S., Drummond, L.M.A. ve Victor, A.O., (1998). "A Parallel Evolutionary Algorithm for the Vehicle Routing Problem with Heterogeneous Fleet", Future Generation Computer Systems, 14:285-292.

[151] Belfiore, P. ve Yoshizaki, H.T.Y., (2009). "Scatter Search for a Real-Life Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Split Deliveries in Brazil", European Journal of Operational Research, 199:750–758.

[152] Kallehauge, B., (2008). “Formulations and Exact Algorithms for the Vehicle Routing Problem with Time Windows”, Computers and Operations Research, 35:2307-2330.

[153] Çevik, O., (2006). “Tam Sayılı Doğrusal Programlama ile İşgücü Planlaması ve Bir Uygulama”, Afyon Kocatepe Üniversitesi, İktisadi ve İdari Bilimler Fakültesi Dergisi, 8(1):157-171.

[154] Wikipedia İnternet Ansiklopedisi, http://en.wikipedia.org/wiki/Branch_and_cut, 25.09.2010.

[155] Taha, H., (2005). Yöneylem Araştırması, Prentice-Hall, Inc., 6. Basımdan çeviri, Literatür Yayınları, Sayı 43.

[156] Rousseau, L.-M., Gendreau, M. ve Feillet, D., (2007). “Interior Point Stabilization for Column Generation”, Operations Research Letters, 35:660-668.

[157] Jin, M., Liu, K. ve Ekşioğlu, B., (2008). “A Column Generation Approach for the Split Delivery Vehicle Routing Problem”, Operations Research Letters, 36:265-270.

[158] Cardoen, B., Demeulemeester, E. ve Belien, J., (2009). “Optimizing a Multiple Objective Surgical Case Sequencing Problem”, International Journal of Production Economics, 119:354-366.

[159] Çetin, E., (2005). “Dinamik Programlama ile Sınır Tenörü Optimizasyonu”, Doğu ve Güneydoğu Anadolu Maden Kaynaklarının Değerlendirilmesi Sempozyumu, 139-143.

[160] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.Y. ve Semet, F., (2002). “A Guide to Vehicle Routing Heuristics”, The Journal of the Operational Research Society, 53(5):512-522.

[161] Cordeau, J.-F., Laporte, G., Savelsbergh, M.W.P. ve Vigo, D., (2007). “Vehicle Routing”, C. Barnhart and G. Laporte (Eds.), Handbook in OR and MS, doi: 10.1016/S0927-0507(06)14006-2, 14:367-428.

[162] Van Breedam, A., (2001). “Comparing Descent Heuristics and Metaheuristics for the Vehicle Routing Problem”, Computers and Operations Research, 28:289-315.

[163] Savelsbergh, M., (2003). Vehicle Routing and Scheduling, The Logistics Institute, Georgia Institute of Technology, http://www2.isye.gatech.edu/~mwps/presentations/ VRP_part1.pdf, 05.10.2009.

[164] Aydemir, A., (2008a). Çözüm Kurucu Sezgiseller, Başkent Üniversitesi, END407 Sezgisel Yöntemler Ders notları, http://www.baskent.edu.tr/~ayyuce/END407Ders2.pdf, 05.10.2010.

[165] Laporte, G., Gendreau, M., Potvin, J.-Y. ve Semet, F., (2000). “Classical and Modern Heuristics for the Vehicle Routing Problem”, International Transactions in Operational Research, 7:285-300.

[166] Özkan, B., Çevre, U. ve Uğur, A., (2008). “Melez Bir Eniyileme Yöntemi ile Rota Planlama”, Akademik Bilişim 2008, Çanakkale Onsekiz Mart Üniversitesi, Çanakkale, Bildiri no: 41.

[167] Wikipedia İnternet Ansiklopedisi, http://en.wikipedia.org/wiki/3-opt, 06.10.2010.

[168] Gendreau, M., Potvin, J.-Y., Braysy, O., Hasle, G. ve Lokketangen, A., (2007). “Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography”, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT).

[169] Aydemir, A., (2008b). Metasezgisel Yöntemler, Başkent Üniversitesi, END543 Sezgisel Optimizasyon Ders notları, http://www.baskent.edu.tr/~ayyuce/END407%20Ders3a.pdf, 09.10.2010.

[170] Ho, S.C. ve Haugland, D., (2004). “A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows and Split Deliveries”, Computers & Operations Research, 31:1947-1964.

[171] Leung, S.C.H., Zhou, X., Zhang, D. ve Zheng, J., (2011). “Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem”, Computers & Operations Research, 38:205–215.

[172] Brandao, J., (2011). “A Tabu Search Algorithm for the Heterogeneous Fixed Fleet Vehicle Routing Problem”, Computers & Operations Research, 38:140–151.

[173] Montane, F.A.T. ve Galvao, R.D., (2006). “A Tabu Search Algorithm for the Vehicle Routing Problem with Simultaneous Pick-Up and Delivery Service”, Computers and Operations Research, 33:595-619.

[174] Tavakkoli-Moghaddam, R., Safaei, N. ve Gholipour, Y., (2006). “A Hybrid Simulated Annealing for Capacitated Vehicle Routing Problems with the Independent Route Length”, Applied Mathematics and Computation, 176:445–454.

[175] Erdem, O.A. ve Uzun, E., (2005). “Yapay Sinir Ağları ile Türkçe Times New Roman, Arial ve El Yazısı Karakterleri Tanıma”, Gazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi, 20(1):13-19.

[176] Ai, T.J. ve Kachitvichyanukul, V., (2009). “A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery”, Computers and Operations Research, 36:1693-1702.

[177] Marinakis, Y., Marinaki, M. ve Dounias, G., (2010). “A Hybrid Particle Swarm Optimization Algorithm for the Vehicle Routing Problem”, Engineering Applications of Artificial Intelligence, 23:463–472.

[178] Marinakis, Y. ve Marinaki, M., (2010). “A Hybrid Genetic–Particle Swarm Optimization Algorithm for the Vehicle Routing Problem”, Expert Systems with Applications, 37:1446–1455.

[179] Duman, E. ve Akın, E., (2006). "Yapay Bağışıklık Sistemi ile Bulanık Yaklaştırıcı Kural Tabanı Optimizasyonu", Elektrik-Elektronik-Bilgisayar Sempozyumu-ELECO'2006, Elektronik Bildirileri Kitabı.

[180] Masutti, T.A.S. ve Castro, L.N., (2009). "Neuro-Immune Approach to Solve Routing Problems", Neurocomputing, 72:2189-2197.

[181] Keskintürk, T., (2006). "Diferansiyel Gelişim Algoritması", İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, yıl 5, 9-2006/1:85-99.

[182] Erbao, C. ve Mingyong, L., (2009). "A Hybrid Differential Evolution Algorithm to Vehicle Routing Problem with Fuzzy Demands", Journal of Computational and Applied Mathematics, 231:302-310.

[183] Laporte, G., (2004). "Metaheuristics for the Vehicle Routing Problem: Fifteen Years of Research", XII Congreso Latino-Iberoamericano de Investigación de Operaciones y Sistemas, La Havane, Cuba.

[184] Thangiah, S.R. ve Salhi, S., (2001). "Genetic Clustering: An Adaptive Heuristic for the Multidepot Vehicle Routing Problem", Applied Artificial Intelligence, 15:361-383.

[185] HEC Montreal, http://neumann.hec.ca/chairedistributique/data/mdvrp/old/, 07.09.2009

[186] Monnot, J., (2002). "Approximation Results Toward Nearest Neighbor Heuristic", Yugoslav Journal of Operations Research, 12(1):11-16.

[187] Salhi, S. ve Sari, M., (1997). "A Multi-Level Composite Heuristic for the Multi-Depot Vehicle Fleet Mix Problem", European Journal of Operational Research, 103:95-112.

[188] Pisinger, D. ve Ropke, S., (2007). "A General Heuristic for Vehicle Routing Problems", Computers and Operations Research, 34(8):2403-2435.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı:	Ganimet Nilay YÜCENUR
Doğum Tarihi ve Yeri:	04.04.1981 / İstanbul
Yabancı Dili:	İngilizce
E-posta:	nserbest@yildiz.edu.tr

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Y. Lisans	Endüstri Mühendisliği	Yıldız Teknik Üniversitesi	2006
Lisans	Endüstri Mühendisliği	Yıldız Teknik Üniversitesi	2004
Lise	Fen/Matematik	Çemberlitaş Kız Lisesi	1999

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2005-…	Yıldız Teknik Üniversitesi/Endüstri Mühendisliği Bölümü	Araştırma Görevlisi

YAYINLARI

Makale

1. **Yücenur, G.N.** ve Demirel, N.Ç., "A New Geometric Shape Based Genetic Clustering Algorithm for the Multi-Depot Vehicle Routing Problem", *Expert Systems with Applications,* Kabul Tarihi: 05.03.2011.

2. **Yücenur, G.N.**, Vayvay, Ö. ve Demirel, N.Ç., (2011), "Suppplier Selection Problem in Global Supply Chains by AHP and ANP Approaches under Fuzzy Enivronment", *International Journal of Advanced Manufacturing Technology*, doi: 10.1007/s00170-011-3220-y.

3. Demirel, N.Ç., **Yücenur, G.N.**, Demirel, T. ve Muşdal, H., "Risk-Based Evaluation of Turkish Agricultural Strategies using Fuzzy AHP and Fuzzy ANP", *International Journal of Human and Ecological Risk Assessment*, Kabul Tarihi: 02.11.2011.

4. **Yücenur, G.N.**, Demirel, N.Ç., Ceylan, C. ve Demirel, T. (2011), "Hizmet Değerinin Müşterilerin Davranışsal Niyetleri Üzerindeki Etkisinin Yapısal Eşitlik Modeli ile Ölçülmesi", *Doğuş Üniversitesi Dergisi*, 12(1):156-168.

5. **Yücenur, G.N.**, Demirel, N.Ç. ve Demirel, T. (2010), "Türkiye Ekonomisinde Stratejik Politika Seçimi için SWOT Analizi ile Entegre Edilmiş Bulanık AHP/Bulanık ANP", *Mühendislik ve Fen Bilimleri Dergisi,* Sigma 28:275-286.

6. Demirel, T., Muşdal, H., Demirel, N.Ç. ve **Yücenur, G.N.**, (2009), "Multi-Criteria Evaluation of Land Cover Policies Using Fuzzy AHP and Fuzzy ANP: The Case of Turkey", *International Journal of Human and Ecological Risk Assessment,* 15(4):746-764.

7. **Serbest, G.N.** ve Vayvay, Ö., (2008), "Selection of the most suitable distribution channel using fuzzy Analytic Hierarchy Process in Turkey", *International Journal of Logistics Systems and Management,* 4(5):487–505.

Bildiri

1. Demirel, N.Ç. ve **Yücenur, G.N.**, (2010), "Selection of the Development Strategy by using AHP and ANP Processes under Fuzzy Environment", *FLINS Conference on Foundations and Applications of Computational Intelligence,* sy. 451-457, 02-04 Ağustos 2010, Çin.

2. **Yücenur, G.N.** ve Demirel, N.Ç., (2010), "Türkiye'de Ekonomik Krize Karşı Alınabilecek Önlemlerin Seçiminde Bulanık AHP ve Bulanık ANP Yöntemleri", *30.Ulusal Yöneylem Araştırması ve Endüstri Mühendisliği Kongresi – YA'EM2010,* sy. 359-360 (Özet Basım), 30 Haziran-02 Temmuz 2010, İstanbul, Türkiye.

3. Demirel, T., Yılmaz, Ş., Demirel, N.Ç. ve **Yücenur, G.N.**, (2009), "Literature Review about Multi-Depot Vehicle Routing Problem and Its Solution Methods", *7th International Logistics and Supply Chain Congress – LMSCM2009,* sy. 444-451, 05-06 Kasım 2009, İstanbul, Türkiye.

4. **Yücenur, G.N.** ve Demirel, N.Ç., (2009), "The Best Insurance Company Selection by AHP and ANP Methodologies with Fuzzy

Numbers in Logistic Sector", *7th International Logistics and Supply Chain Congress – LMSCM2009,* sy. 182-189, 05-06 Kasım 2009, İstanbul, Türkiye.

5. **Yücenur, G.N.** ve Demirel, N.Ç., (2009), "A Structural Equation Model for Measuring Service Quality in Passenger Transportation", *International Conference on Prospects for Research in Transport and Logistics on a Regional-Global Perspective,* sy. 125-131, 12-14 Şubat 2009, İstanbul, Türkiye.

6. Demirel, N.Ç. ve **Yücenur, G.N.**, (2008), "Selection of the most Suitable City for a Nuclear Power Plant bu using Fuzzy AHP and Fuzzy ANP Methodology", *FLINS Conference,* sy. 1069-1074, 21-24 Eylül 2008, Madrid, İspanya.

7. **Yücenur, G.N.** ve Demirel, N.Ç., (2008), "Selection of Water Utilization Strategy with Fuzzy AHP and Fuzzy ANP Methodologies under Global Warming Risk", *12th International Resarch/Expert Conference-TMT2008, "Trends in the Development of Machinery and Associated Technology",* sy. 617-620, 26-30 Ağustos 2008, İstanbul, Türkiye.

8. **Yücenur, G.N.** ve Demirel, N.Ç., (2008), "Tekstil Sektöründe Müşteri Talepleri için Bulanık Kalite Fonksiyonu Göçerimi Uygulaması". *2th Ulusal Sistem Mühendisliği Kongresi - 21. Yüzyılda Sistem ve İnovasyon,* sy. 22-26, 06-08 Şubat 2008, İstanbul, Türkiye.

9. **Yücenur, G.N.** ve Demirel, N.Ç., (2008), "Bulanık Analitik Hiyerarşi Prosesi Metodu ile Turizm Endüstrisinde Strateji Seçimi". *2th Ulusal Sistem Mühendisliği Kongresi - 21. Yüzyılda Sistem ve İnovasyon,* sy. 291-295, 06-08 Şubat 2008, İstanbul, Türkiye.

10. Demirel, N.Ç. ve **Serbest, G.N.**, (2007), "A Fuzzy Model for Selection of the Storage Place in Logistics Sector". *5th International Logistics and Supply Chain Congress –LMSCM2007,* sy. 446-452, 08-09 Kasım 2007, İstanbul, Türkiye.

11. Demirel, N.Ç. ve **Serbest, G.N.**, (2007), "Selection of Turkish Agricultural Strategy with Fuzzy AHP Methodology". *1th International Conference on Risk Analysis and Crisis Response - RACR2007,* sy. 127-131, 25-26 Eylül 2007, Shanghai, Çin.

12. **Serbest, G.N.**, Demirel, N.Ç. ve Aydın, N., (2007), "Bulanık Analitik Hiyerarşi Prosesi Metodu ile Katı Atıklar için En Uygun Tesis Yerinin Seçimi". *27.Ulusal Yöneylem Araştırması ve Endüstri Mühendisliği Kongresi – YA'EM2007,* sy. 1237-1242, 02-04 Temmuz 2007, İzmir, Türkiye.

13. **Serbest, G.N.**, Demirel, T. ve Demirel, N.Ç., (2007), "A Fuzzy Model For Evaluation of Department Performance". *3th International Conference on Business, Management and Economics – ICBME2007,* CD Basım, 13-17 Haziran 2007, İzmir, Türkiye.

14. Demirel, N.Ç. ve **Serbest, G.N.**, (2007), "Strategy Selection Process for Companies after a Partnership with Fuzzy AHP Methodology". *3th International Conference on Business, Management and Economics – ICBME2007,* CD Basım, 13-17 Haziran 2007, İzmir, Türkiye.

15. Demirel, N.Ç., **Serbest, G.N.** ve Aydın, N., (2006), "A Proposed Model to Measure Service Quality in Passenger Transportation". *4th International Logistics and Supply Chain Congress – ILC2006,* sy. 322-327, 29-30 Kasım/01 Aralık 2006, İzmir, Türkiye.

16. Demirel, N.Ç., **Serbest, G.N.** ve Demirel, T., (2006), "Oyun Teorisi ile Tekstil Sektöründe Rekabetçi İş Stratejileri". *26.Ulusal Yöneylem Araştırması ve Endüstri Mühendisliği Kongresi – YA'EM2006,* sy. 406-409, 03-05 Temmuz 2006, Kocaeli, Türkiye.

17. Demirel, N.Ç. ve **Serbest, G.N.**, (2006), "Personnel Empowerment in Human Resources Management". *5th International Symposium on Intelligent Manufacturing Systems – IMS2006,* sy. 776-783, 29-31 Mayıs 2006, Sakarya, Türkiye.

18. Demirel, N.Ç. ve **Serbest, G.N.**, (2006), "A Fuzzy Decision Making System for Selection of the Best Gelcoat Manufacturer". *5th International Symposium on Intelligent Manufacturing Systems – IMS2006,* sy. 767-775, 29-31 Mayıs 2006, Sakarya, Türkiye.

19. Demirel, N.Ç., **Serbest, G.N.** ve Demirel, T., (2005), "Supplier Selection in Food Sector Using a Fuzzy Model". *3th International Logistics & Supply Chain Congress 2005 – LMSCM2005,* sy. 81-85, 23-24 Kasım 2005, İstanbul, Türkiye.

20. Demirel, N.Ç. ve **Serbest, G.N.**, (2005), "Implementation of Quality Function Deployment in Ergonomics". *11th International Symposium on Quality Function Deployment – ISQFD2005,* sy. 167-174, 26-30 Eylül 2005, İzmir, Türkiye.

21. **Serbest, G.N.** ve Demirel, N.Ç., (2005), "A Fuzzy Approach to Determination of the Supplier in TQM". *9th International Research/Expert Conference on "Trends in the development of machinery and associated Technology" – TMT2005,* sy. 347-350, 26-30 Eylül 2005, Antalya, Türkiye.

22. **Serbest, G.N.** ve Demirel, N.Ç., (2005), "A Fuzzy Model for Single Facility Layout Problem". *35th International Conference of Computers & Industrial Engineering – CIE2005,* sy. 1723-1729, 19-22 Haziran 2005, İstanbul, Türkiye.