EŞ ZAMANLI TOPLA DAĞIT ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜMÜ İÇİN BİR HİBRİT META SEZGİSEL ALGORİTMA ÖNERİSİ

Pamukkale Üniversitesi
Sosyal Bilimler Enstitüsü
Doktora Tezi
İşletme Anabilim Dalı
Genel İşletme Doktora Programı

Tayfun ÖZTAŞ

Danışman: Doç. Dr. Ayşegül TUŞ

Temmuz 2021
DENİZLİ

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

İmza
Tayfun ÖZTAŞ

ÖNSÖZ

Bu tezin hazırlanma sürecinde değerli vakitlerini ayırarak beni yönlendiren danışman hocam Doç. Dr. Ayşegül TUŞ başta olmak üzere tez savunma jüri üyeleri Prof. Dr. Arzu ORGAN, Prof. Dr. Muhsin ÖZDEMİR, Doç. Dr. Kenan KARAGÜL ve Doç. Dr. Yusuf ŞAHİN hocalarıma en içten teşekkürlerimi sunarım. Tez yazım sürecini en başından beri takip eden Doç. Dr. Esra AYTAÇ ADALI hocama da destekleri sebebiyle teşekkür ederim.

Doktora eğitim hayatım boyunca her türlü zorluğu birlikte aştığımız sevgili eşim ve aynı zamanda meslektaşım olan Dr. Gülin Zeynep ÖZTAŞ'a göstermiş olduğu sabır, destek, anlayış ve bana olan güveni için ne kadar teşekkür etsem az kalır. Yaşadığım her olayda benimle sevinip, benimle üzülen, dualarını hiç esirgemeyen ve bugünlere gelmemde en büyük destekçilerim olan aile bireylerime şükranlarımı sunarım. Ailemden sonra bana en büyük destek ve anlayışı sunarak hayatımı kolaylaştırmak için ellerinden geleni yapan değerli dostlarımı tanıdığım için kendimi çok şanslı hissediyorum. İyi ki varsınız!

Son olarak, tez yazım sürecinde verilerini benimle paylaşan Prof. Dr. Jan Dethloff ve Prof. Dr. Fermín Alfredo Tang Montané'ye teşekkür ederim.

ÖZET

EŞ ZAMANLI TOPLA DAĞIT ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜMÜ İÇİN BİR HİBRİT META SEZGİSEL ALGORİTMA ÖNERİSİ

ÖZTAŞ, Tayfun
Doktora Tezi
İşletme ABD
Genel İşletme Doktora Programı
Danışman: Doç. Dr. Ayşegül TUŞ

Temmuz 2021, ix+137 Sayfa

Araç rotalama problemleri, belirli sayıda araç kullanılarak müşteriler ve depo arasında taşıma taleplerinin karşılanması ile ilgilenmektedir. Problemin sahip olduğu özelliklere göre literatürde birçok araç rotalama problemi türü bulunmaktadır. Bu tezde, müşterilerin eş zamanlı olarak hem dağıtım hem de toplama talepleri karşılanan Eş Zamanlı Topla Dağıt Araç Rotalama problemi (EZTDARP) ele alınmıştır. Söz konusu problem, kesikli (kombinatoryal) bir optimizasyon problemidir. Bu nedenle problem boyutu büyüdükçe çözülmesi zorlaşmaktadır. Bu tür problemleri kısa sürede kesin yöntemlerle çözmek mümkün olmadığı için meta sezgisel yöntemlere ihtiyaç duyulmaktadır.

Meta sezgisel yöntemler, bir optimizasyon probleminin arama uzayında çeşitli yaklaşımlarla arama yaparak daha iyi çözümlere ulaşmayı hedefler. Bu yaklaşımlarla kısa sürelerde optimal ya da optimale yakın çözümlere ulaşmak mümkün olabilmektedir. Kesin yöntemler ile optimal sonuçları elde etmenin mümkün olmadığı ya da çok uzun süre beklemenin gerektiği durumlarda meta sezgisel yöntemleri kullanmak mantıklı bir seçim olmaktadır.

Bu tezde yinelemeli yerel arama, değişken komşuluk iniş ve eşik kabul meta sezgisellerine dayanan ILS-RVND-TA olarak isimlendirilen hibrit bir algoritma önerilerek seçilen araç rotalama probleminin çözülmesi amaçlanmıştır. Yinelemeli yerel arama, çözümler üzerinde sarsım yoluyla değişiklikler yaparak arama uzayının farklı bölgelerini keşfeden ve bu sayede yerel optimuma takılmamaya çalışan bir meta sezgiseldir. Değişken komşuluk iniş, çözümlerin birden fazla komşuluğunda arama yaparak yoğun bir şekilde daha iyi çözümler arayan bir meta sezgiseldir. Eşik kabul ise daha kötü çözümlerin kabul edilmesine izin veren bir meta sezgiseldir. Önerilen algoritma, belirli test problemleri kullanılarak bulgular analiz edilmiştir.

Anahtar Kelimeler: Optimizasyon, Araç Rotalama Problemleri, Meta Sezgiseller, Eş Zamanlı Topla Dağıt Araç Rotalama Problemi, Yinelemeli Yerel Arama Algoritması, Değişken Komşuluk İniş Algoritması, Eşik Kabul Algoritması

ABSTRACT

A HYBRID METAHEURISTIC ALGORITHM PROPOSAL FOR VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERIES

ÖZTAŞ, Tayfun
Doctoral Thesis
Business Administration Department
PhD. in Business Administration
Adviser of Thesis: Assoc. Prof. Dr. Ayşegül TUŞ

July 2021, ix +137 Pages

Vehicle routing problems deal with the meeting of transportation demands between customers and the depot by using a certain number of vehicles. There are many sub-types of vehicle routing problems in the literature depend on the characteristics of the problem. In this thesis, the Vehicle Routing Problem with Simultaneous Pickup and Deliveries (VRPSPD), where customers have both distribution and collection demands are met simultaneously, is addressed. The problem in question is a discrete (combinatorial) optimization problem. For this reason, solving this problem is getting more difficult as the size of the problem is increasing. Since it is not possible to solve such problems with exact methods in a short time, metaheuristics methods are required.

Metaheuristics aim to reach good solutions by seeking the search space of an optimization problem with various approaches. With this approach, it is possible to reach optimal or near-optimal solutions in a short time. It is a reasonable choice to use metaheuristics in cases where it is not possible to obtain optimal results with exact methods or when it is necessary to wait too long.

It is aimed to solve the selected vehicle routing problem by proposing a hybrid algorithm based on Iterative Local Search, Variable Neighborhood Descent, and Threshold Acceptance metaheuristics and called as ILS-RVND-TA in this thesis. Iterative local search is a metaheuristic that explores different regions of the search space by making changes with a mechanism named perturbation on solutions, thus trying not to be trapped to the local optimum. Variable neighborhood descent, on the other hand, is a metaheuristic that searches more than one neighborhood of solutions intensely for improving solutions. Threshold acceptance is a metaheuristic that allows worse solutions to be accepted. The proposed algorithm was tested on certain test problems and the findings were analyzed.

Keywords: Optimization, Vehicle Routing Problems, Metaheuristics, Vehicle Routing Problem with Simultaneous Pickup and Deliveries, Iterative Local Search Algorithm, Variable Neighborhood Descent Algorithm, Threshold Acceptance Algorithm

İÇİNDEKİLER

ÖNSÖZ i
ÖZET ii
ABSTRACT iii
İÇİNDEKİLER iv
ŞEKİLLER DİZİNİ vi
TABLOLAR DİZİNİ vii
EKLER DİZİNİ viii
SİMGE VE KISALTMALAR DİZİNİ ix
GİRİŞ 1

BİRİNCİ BÖLÜM
ARAÇ ROTALAMA PROBLEMLERİ

1.1. Araç Rotalama Probleminin Matematiksel Modeli 6
1.2. Araç Rotalama Problemlerinin Sınıflandırılması 7
1.2.1. Ağ (Yol) Yapısına Göre Araç Rotalama Problemleri 8
1.2.2. Ulaştırma Taleplerinin Tipine Göre Araç Rotalama Problemleri 8
1.2.3. Rota İçi Kısıtlara Göre Araç Rotalama Problemleri 10
1.2.4. Filo ve Depo Özelliklerine Göre Araç Rotalama Problemleri 10
1.2.5. Rotalar Arası Kısıtlara Göre Araç Rotalama Problemleri 11
1.2.6. Optimizasyon Amaçlarına Göre Araç Rotalama Problemleri 11
1.3. Araç Rotalama Problemlerinin Çözüm Yöntemleri 11
1.3.1. Kesin Yöntemler 12
1.3.1.1. Dal sınır yöntemleri 12
1.3.1.2. Dal kesme yöntemleri 12
1.3.2. Sezgisel Yöntemler 12
1.3.2.1. Rota kurucu sezgiseller 13
1.3.2.2. Rota iyileştirici sezgiseller 13
1.3.2.3. İki aşamalı sezgiseller 13
1.3.3. Meta Sezgisel Yöntemler 14
1.4. Eş Zamanlı Topla Dağıt Araç Rotalama Problemleri 14
1.4.1. EZTDARP'nin Matematiksel Modeli 15
1.4.2. Probleme İlişkin Literatür Taraması 17

İKİNCİ BÖLÜM
KARMAŞIKLIK KURAMI VE ARAÇ ROTALAMA PROBLEMLERİNİN ÇÖZÜMÜNDE KULLANILAN META SEZGİSELLER

2.1. Karmaşıklık Kuramı 32
2.2. Optimizasyon Problemleri ve Bunların Çözüm Yöntemleri 35
2.3. Meta Sezgisel Kavramı ve Temel Özellikleri 38
2.4. Meta Sezgisel Yöntemlerin Sınıflandırılması 40
2.5. Araç Rotalama Problemlerinde Kullanılan Meta Sezgiseller 41
2.5.1. Tek Çözüm Temelli Yöntemler 41
2.5.1.1. Yinelemeli yerel arama 41

2.5.1.2. Açgözlü rassallaştırılmış uyarlanabilir arama prosedürü....44
2.5.1.3. Tabu arama....46
2.5.1.4. Değişken komşuluk arama....48
2.5.1.5. Tavlama benzetimi....51
2.5.1.6. Yönlendirilmiş yerel arama....53
2.5.1.7. Geniş komşuluk arama....56
2.5.2. Popülasyon Temelli Yöntemler....57
2.5.2.1. Genetik algoritma....57
2.5.2.2. Dağınık arama....60
2.5.2.3. Karınca kolonisi optimizasyonu....62
2.5.2.4. Parçacık sürü optimizasyonu....65

ÜÇÜNCÜ BÖLÜM
HİBRİT YİNELEMELİ YEREL ARAMA ALGORİTMASI İLE EŞ ZAMANLI TOPLA DAĞIT ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜLMESİ

3.1. Önerilen Algoritma....68
3.1.1. Başlangıç Çözümün Oluşturulması....70
3.1.2. Yerel Arama Aşaması....70
3.1.2.1. Rota içi komşuluk yapıları....72
3.1.2.2. Rotalar arası komşuluk yapıları....75
3.1.3. Sarsım....78
3.1.4. Kabul Kriteri....82
3.2. Sayısal Analiz....83
3.2.1. Test Problemleri....84
3.2.2. Hesaplama Sonuçları....86
3.2.2.1. Dethloff (2001) test problemlerine ilişkin sonuçlar....86
3.2.2.2. Salhi ve Nagy (1999) test problemlerine ilişkin sonuçlar....91

SONUÇ....97
KAYNAKLAR....104
EKLER....124

ŞEKİLLER DİZİNİ

Şekil 1. Örnek Bir Araç Rotalama Probleminin Çözümüne Ait Rotalar 5
Şekil 2. Araç Rotalama Problemlerinin Sınıflandırılması 7
Şekil 3. Optimizasyon Problemlerinin Sınıflandırılması 37
Şekil 4. Türev Yardımıyla Fonksiyonun Maksimum Değerinin Belirlenmesi 37
Şekil 5. Optimizasyon Yöntemlerinin Sınıflandırılması 38
Şekil 6. Araç Rotalama Problemlerinde Kullanılan Bazı Meta Sezgisellerin Özellikleri 41
Şekil 7. Yinelemeli Yerel Arama Algoritmasına Ait Sözde-Kod 44
Şekil 8. Açgözlü Rassallaştırılmış Uyarlanabilir Arama Prosedürüne Ait Sözde-Kod 46
Şekil 9. Tabu Arama Algoritmasına Ait Sözde-Kod 48
Şekil 10. Değişken Komşuluk Arama Algoritmasına Ait Sözde-Kod 51
Şekil 11. Tavlama Benzetimi Algoritmasına Ait Sözde-Kod 53
Şekil 12. Yönlendirilmiş Yerel Arama Algoritmasına Ait Sözde-Kod 55
Şekil 13. Geniş Komşuluk Arama Algoritmasına Ait Sözde-Kod 57
Şekil 14. Bitlerle Oluşturulmuş Bir Kodlama Örneği 58
Şekil 15. Çaprazlama Operatörünün Kullanıma İlişkin Örnek 59
Şekil 16. Bir Kromozomda Mutasyon Operatörünün Kullanımına Örnek 59
Şekil 17. Genetik Algoritmaya Ait Sözde-Kod 60
Şekil 18. Dağınık Arama Algoritmasına Ait Sözde-Kod 62
Şekil 19. Karınca Kolonisi Optimizasyon Algoritmasına Ait Sözde-Kod 65
Şekil 20. Parçacık Sürü Optimizasyonu Algoritmasına Ait Sözde-Kod 67
Şekil 21. ILS-RVND-TA Algoritmasına Ait Sözde-Kod 69
Şekil 22. Yerel Arama Aşamasına Ait Sözde-Kod 71
Şekil 23. Parça Tersine Çevirme Komşuluk Yapısının Uygulama Örneği 73
Şekil 24. Or-Opt Komşuluk Yapısının Uygulama Örneği 73
Şekil 25. 2-Opt Komşuluk Yapısının Uygulama Örneği 74
Şekil 26. Değişim Komşuluk Yapısının Uygulama Örneği 74
Şekil 27. Ekleme Komşuluk Yapısının Uygulama Örneği 75
Şekil 28. Değiş Tokuş (1-1) Komşuluk Yapısının Uygulama Örneği 75
Şekil 29. Öteleme (1-0) Komşuluk Yapısının Uygulama Örneği 76
Şekil 30. Değiş Tokuş (2-2) Komşuluk Yapısının Uygulama Örneği 76
Şekil 31. Öteleme (2-0) Komşuluk Yapısının Uygulama Örneği 77
Şekil 32. Değiş Tokuş (2-1) Komşuluk Yapısının Uygulama Örneği 77
Şekil 33. Çaprazlama Komşuluk Yapısının Uygulama Örneği 78
Şekil 34. Çapraz Değişim Sarsım Mekanizmasının Uygulama Örneği 80
Şekil 35. Sarsım Aşamasına Ait Sözde-Kod 81
Şekil 36. CMT3X ve CMT11X Problemlerinde Müşterilerin Konumları 93
Şekil 37. Algoritmaların Sonuçlarının Karşılaştırılması 94
Şekil 38. Ortalama Çözüm Değerleri ve Hesaplama Süreleri Açısından Algoritmaların Karşılaştırılması 95

TABLOLAR DİZİNİ

Tablo 1. Bilginin Çıkış Şekli ve Bilinirliğine Göre Sınıflandırma 10
Tablo 2. Dethloff (2001)'un Modeline Ait Gösterimler 15
Tablo 3. Literatür Taramasına İlişkin Özet Tablo 30
Tablo 4. Karmaşıklık Düzeyleri ve Bunların İsimlendirmeleri 34
Tablo 5. MaxIt Parametresinin Ortalama Çözüm Değeri ve Süre Üzerindeki Etkisi 83
Tablo 6. SCA Problemlerine İlişkin En İyi Çözüm Değerleri 87
Tablo 7. CON Problemlerine İlişkin En İyi Çözüm Değerleri 88
Tablo 8. SCA Problemleri İçin Detaylı Sonuçlar 89
Tablo 9. CON Problemleri İçin Detaylı Sonuçlar 90
Tablo 10. Salhi ve Nagy (1999) Problemlerine İlişkin En İyi Çözüm Değerleri 92
Tablo 11. Salhi ve Nagy (1999) Problemlerine İlişkin Detaylı Sonuçlar 94

EKLER DİZİNİ

Ek 1. SCA Problemleri İçin Belirlenen Rastgele Çekirdek Değerleri 125
Ek 2. CON Problemleri İçin Belirlenen Rastgele Çekirdek Değerleri..... 126
Ek 3. CMT Problemleri İçin Belirlenen Rastgele Çekirdek Değerleri 127
Ek 4. SCA Problemleri İçin Algoritmaların En İyi Çözüm Değerleri ve Ortalama Hesaplama Süreleri 128
Ek 5. CON Problemleri İçin Algoritmaların En İyi Çözüm Değerleri ve Ortalama Hesaplama Süreleri 129
Ek 6. CMT Problemleri İçin Algoritmaların En İyi Çözüm Değerleri ve Ortalama Hesaplama Süreleri 130
Ek 7. SCA Problemleri İçin Elde Edilen Rotalar (1. Kısım) 131
Ek 8. SCA Problemleri İçin Elde Edilen Rotalar (2. Kısım) 132
Ek 9. CON Problemleri İçin Elde Edilen Rotalar (1. Kısım)..... 133
Ek 10. CON Problemleri İçin Elde Edilen Rotalar (2. Kısım)..... 134
Ek 11. CMT Problemleri İçin Elde Edilen Rotalar (1. Kısım)..... 135
Ek 12. CMT Problemleri İçin Elde Edilen Rotalar (2. Kısım)..... 136

SİMGE VE KISALTMALAR DİZİNİ

D_i	Dağıtım Miktarı
P_i	Toplama Miktarı
A	Yay
BEÇ	Literatürde Bilinen En İyi Çözüm Değeri
C	Araç Kapasitesi
CPU	İşlemci (Central Process Unit)
E	Kenar
EÇ	Elde Edilen En İyi Çözüm Değeri
EZTDARP	Eş Zamanlı Topla Dağıt Araç Rotalama Problemi
G	Çizge
ILS	Yinelemeli Yerel Arama (Iterated Local Search)
m	Araç Sayısı
n	Müşteri Sayısı
$N(s)$	Komşu Çözüm
O	Karmaşıklık Düzeyi
RVND	Rastgele Komşuluk Sıralamalı Değişken Komşuluk İniş (Random neighborhood ordering Variable Neighborhood Descent)
Q	Toplam Talep
s	Çözüm
S	Çözüm Uzayı
T	Sıcaklık
TA	Eşik Kabul (Threshold Acceptance)
V	Düğüm

GİRİŞ

Eski dönemlerde medeniyetlerin bilgi birikimine bağlı olarak ortaya çıkan her türlü ürün; genellikle savaş, ticaret gibi olayların beraberinde getirdiği etkileşimler sonucunda başka medeniyetler tarafından tanınma ve benimsenme imkânı bulmuştur. Bu nedenle de ortaya çıkan bir yenilik, dünyanın geri kalanına hemen yayılma şansı bulamamıştır. Buna karşın, bilim ve teknolojinin sağladığı imkânlar sayesinde günümüz insanı, ihtiyaç duyduğu şeylere çok daha zahmetsizce ve sıradan bir şekilde ulaşabilmektedir. Örneğin, geçmişi göreceli olarak çok eski bir döneme dayanmayan telefon ve bunun insan hayatına etkisi incelenecek olursa; telefonun icadından önceki dönemde farklı bölgeler arasındaki iletişim, alıcıya ulaşması haftalar ya da aylar süren mektuplarla sağlanmıştır. Telefonun yeni yaygınlaştığı dönemlerde ise iletişim, santralin aranan kişiyi hatta bağlama süresi değişkenlik gösterse de mektupla iletişime göre hızlanmıştır. İçerisinde bulunduğumuz modern dönemde ise sahip olduğumuz cihazlarla dünyanın herhangi bir yerinde bulunan kişilere anlık olarak sesli ve görüntülü olarak ulaşmak mümkün hale gelmiştir. Benzer örnekleri ticaret için de vermek mümkündür. Eski zamanlarda ticaret yolları üzerinden uzun sürelerde müşterilerine ulaşan ürünler, günümüzde aynı gün içerisinde ya da birkaç gün gibi çok daha kısa sürelerde müşterilerine ulaşabilmektedir.

Son kullanıcı açısından zahmetsiz görünen bu işlemlerin arkasında ise son derece karmaşık süreçler bulunmaktadır. Müşterinin, sipariş vermesi ve siparişini teslim alması arasında geçen sürede bilgi paylaşımı, planlama, tedarik, üretim ve lojistik gibi süreçler bulunmaktadır. Genel başlıklar altında bir araya getirilen bu süreçlerin, en iyi şekilde yönetilmesi, tarafların bu süreci en kârlı şekilde tamamlamalarının temel koşuludur. Benzer işlemlerin hizmet alımı için de geçerli olduğunu belirtmekle birlikte, en iyi ifadesi ile de optimuma atıfta bulunulduğu açıktır.

Mal veya hizmet talebinde bulunan kişi ya da kuruluşların ihtiyaçlarının karşılanması sürecinde yer alan faaliyetlerden lojistik, bu tezin temel konusunu oluşturmaktadır. Mal veya hizmetlerin bir noktadan başka bir noktaya ulaştırılması çok boyutlu bir süreçtir. Öncelikle ulaştırma, alıcı ve satıcı açısından çok önemli bir maliyet kalemidir. Ham madde ve ara mamullerin satın alınması ile başlayan ve nihai ürünün tüketiciye teslim edilmesi ile sonuçlanan bir süreçte malın nakliyesi, ürün fiyatının çok büyük bir kısmını oluşturabilmektedir. Ayrıca bu sürecin gerçekleşmesini sağlayacak

araçların ve personelin temin edilmesi de büyük bir yatırımı gerektirmektedir. Ulaştırma faaliyetinin zaman açısından uzunluğu, müşteri memnuniyetini etkileyen önemli faktörlerden birisidir. İçerisinde bulunduğumuz hızlı tüketim çağında teslimat süresinin uzaması, yoğun şikayetlere neden olmaktadır ve bu şikayetler, müşterilerin daha hızlı teslimat yapan alternatifleri seçmesi ile sonuçlanmaktadır. Çevresel anlamda ise ulaştırma faaliyetlerinin hava kirliği, ses kirliliği ve aşırı karbon salınımı nedeniyle ortaya çıkan küresel ısınma gibi ekolojik sonuçları bulunmaktadır. Geniş bir perspektiften bakıldığında, bu faaliyetler ile ilgili olumsuz sonuçları ortadan kaldırabilecek ya da etkisini bir nebze olsun azaltabilecek potansiyel araştırma alanlarının bulunduğu kolayca görülmektedir. Bu tezin konusu da söz konusu araştırma problemi düşünülerek belirlenmiştir.

Bu tezde, literatürde bilim insanları tarafından yaygın şekilde çalışılan araç rotalama problemlerinin bir türü ele alınmıştır. Söz konusu problem, en genel haliyle merkezi bir depoda bulunan ürünlerin belirli sayıda araç kullanılarak çeşitli konumlarda bulunan müşterilere ulaştırılmasını ve araçların depoya dönmesini ele almaktadır. Araç rotalama problemleri aslında çok çeşitli alt problemlere sahip geniş bir problem ailesidir. Bu alt problemler, problemin en genel halinde bahsedilen depo, araç ve müşteri gibi bileşenlerin karakteristik özelliklerine bağlı olarak ortaya çıkmaktadır. Bu tezde, müşterilerin teslimat işlemine ek olarak aynı zamanda kendisinden bir toplama işleminin de yapılmasını istediği bir alt problem türü olan Eş Zamanlı Topla Dağıt Araç Rotalama Problemi (EZTDARP) ele alınmıştır.

Araç rotalama problemleri, problem boyutunun büyümesi ile çözülmesi zorlaşan problem türlerinden birisidir. Probleme ilişkin sonuçları kesin bir şekilde sunabilen yöntemler, yalnızca küçük boyutlu problemlerde işe yaradığı için daha büyük boyutlu problemlerde farklı yaklaşımlara ihtiyaç duyulmaktadır. Meta sezgisel yöntemler, kesin yöntemlerin çözüm bulamadığı ya da kabul edilebilir zaman dilimi içerisinde çözüm üretemediği problemler için kısa sürede optimal ya da optimale yakın çözüm üretebilen faydalı araçlardır. Bu tür yaklaşımların kullanılması ile problemlerin çözümü için çok uzun süreler beklemek yerine çok daha kısa sürede daha iyi çözümler elde etmek mümkün olmaktadır. Bu tezde de ele alınan probleme meta sezgisel yöntemler ile çözüm bulunması amaçlanmıştır.

Tezin geri kalan kısmı şu şekilde organize edilmiştir: Birinci bölümde, öncelikle araç rotalama problemi incelenmiştir. Problemin matematiksel modeli (Dethloff, 2001) sunularak problemin alt türlerine dair detaylı bir sınıflandırma yapılmıştır. Araç rotalama problemlerinin çözümünde kullanılan yöntemler de sınıflandırıldıktan sonra tezin asıl ilgi alanına giren EZTDARP daha detaylı incelenmiştir ve bu probleme ilişkin literatür taraması sunulmuştur.

Tezin ikinci bölümünde, karmaşıklık kuramı, optimizasyon problemleri ve bunların çözüm yöntemlerinden bahsedilmiştir. Sonrasında ise meta sezgisel yöntemlere ilişkin bir sınıflandırma yapılarak bu sınıflandırma ışığında araç rotalama problemlerinin çözümünde literatürde öne çıkan meta sezgisel yöntemler genel olarak ele alınmıştır.

Tezin üçüncü bölümünde ise ele alınan problemin çözümü için önerilen hibrit meta sezgisel algoritmasına özgü bilgiler verilmiştir. Önerilen algoritmanın test edildiği problem kümeleri, başlangıç çözümü elde etmek için kullanılan sezgisel, yerel arama aşamasında kullanılan hem rota içi hem de rotalar arası komşuluk yapıları, arama uzayının çeşitli bölgelerinde arama yapılmasını sağlayan sarsım (perturbation) mekanizmaları tanıtılmıştır. Önerilen algoritmanın karakteristik özellikleri tanıtıldıktan sonra seçilen test problemlerine ilişkin sonuçlar verilerek literatürde bilinen en iyi çözümlerle karşılaştırmalar yapılmıştır.

Sonuç bölümünde ise tezin genel bir fotoğrafı çekilerek, önerilen meta sezgisel algoritmanın bir değerlendirmesi yapılmıştır. Algoritmanın güçlü ve zayıf olduğu yönleri belirtilerek sonraki çalışmalar için olası çalışma alanlarına ilişkin öneriler paylaşılmıştır.

BİRİNCİ BÖLÜM
ARAÇ ROTALAMA PROBLEMLERİ

İşletme kavramı için literatürde çok sayıda tanım yer almaktadır. Yapılan tanımlar birbirinden farklı olsa da işletmeler, faaliyet alanı ayrımı gözetilmeksizin mal veya hizmet üreterek fayda yaratır. Fayda yaratma; şekil değişikliği, zaman değişikliği, yer değişikliği ve mülkiyet değişikliği gibi farklı yollarla gerçekleştirilmektedir (Mucuk, 2016: 3). Sayılan fayda yaratma yollarından yer değişikliği, mal veya hizmetin müşterilere sunulabilmesi için farklı bölgeler arasında taşımacılık faaliyetlerini gerektirmektedir. Ekonomik faaliyetlerin etkin biçimde yürütülmesinde söz konusu taşıma işlemlerinin rolü önemlidir. Taşıma faaliyetleri, gelişmiş ülkelerin ekonomilerinde sahip olduğu konum nedeniyle araştırmalara yoğun bir şekilde konu olmaktadır (Fisher, 1995: 1). Taşımacılık faaliyetlerinde kullanılan araçlar ile ilgili ortaya çıkan planlama sorunları ise optimizasyon alanında araç rotalama problemleri olarak ele alınmaktadır.

Araç rotalama problemleri; üretici, depo ve müşteri şeklinde üç taraflı bir yapıya sahiptir (Keskintürk vd., 2015: 78). Ürünlerin müşterilere ulaştırılabilmesi için gerekli rotaların belirlenmesi, işletmeler için önemli ve zor bir problemdir. Bu problem, literatürde araç rotalama problemi (vehicle routing problem) olarak isimlendirilmektedir. Araç rotalama problemi, merkezi bir depodan coğrafi olarak dağılmış durumda olan müşterilere optimal dağıtım ya da toplama rotalarının tasarımından oluşur ve araç kapasitesi, rota uzunluğu, zaman penceresi, müşteriler arası öncelik ilişkileri gibi kısıtlara sahiptir (Laporte, 2007: 811). Araç rotalama problemi, literatürde ilk kez Dantzig ve Ramser (1959) tarafından benzin istasyonlarına gidecek kamyonların rotasını optimize etmeyi amaçlayan "kamyon sevkiyat problemi – truck dispatching problem" olarak ele alınmıştır. Araç rotalama ifadesini içeren ilk çalışma ise Golden vd. tarafından 1972 yılında yapılmıştır (Eksioglu vd., 2009: 1473).

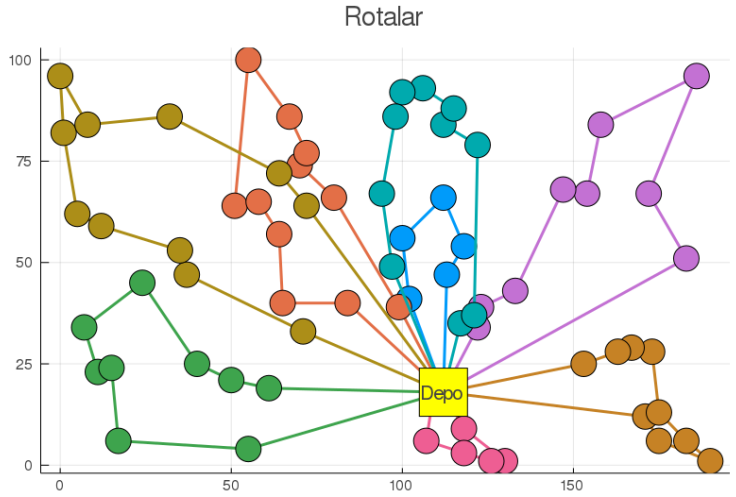
Gezgin satıcı problemi, n adet noktadan oluşur ve satıcı, başlangıç noktasından başlayarak diğer noktaları dolaşıp başlangıç noktasına geri döner. Bu problemin amacı, satıcının geriye kalan $n-1$ noktayı en kısa toplam mesafeyi kat ederek ya da en kısa sürede dolaşarak başlangıç noktasına geri dönmesidir (Flood, 1956: 61). Araç rotalama problemlerinde her bir araç, merkezi bir depodan başlayarak müşterileri sırasıyla ziyaret eder ve tekrar depoya döner. Bu şekilde hizmet sunan her bir araç, gezgin satıcı

problemlerini (traveling salesman problem) anımsatmaktadır. Araç rotalama problemi, gezgin satıcı probleminin daha fazla araç ve kısıt içeren versiyonudur (Keskintürk vd., 2015: 79). Araç rotalama problemi, gezgin satıcı problemini genelleştirdiğinden literatürde *NP-Zor* (*NP-Hard*) sınıfında yer almaktadır (Cordeau vd., 2007: 368). Burada kullanılan *NP-Zor* terimi, söz konusu problemin polinom zamanda çözülemeyeceği anlamına gelmektedir (Lenstra ve Kan, 1981: 222).

Laporte ve Osman, çizge (graph) teorisini kullanarak araç rotalama problemlerini şu şekilde tanımlamıştır (Laporte ve Osman, 1995: 228):

> "$G = (V, E \cup A)$ biçiminde bir çizge tanımlansın. Burada $V = \{v_1, \dots, v_n\}$ düğüm (vertex) kümesini, $A = \{(v_i, v_j) : i \neq j, v_i, v_j \in V\}$ yönlendirilmiş yay (arcs) kümesini, $E = \{(v_i, v_j) : i < j, v_i, v_j \in V\}$ ise yönlendirilmemiş kenar (edges) kümesini ifade etmektedir. Çoğu problemde A veya E kümesi boştur. Sadece yaylardan oluşan problemler yönlendirilmiş veya asimetrik olarak, sadece kenarlardan oluşan problemler ise yönlendirilmemiş ya da simetrik olarak isimlendirilir. Hem kenar hem de yay içeren problemler ise karma olarak isimlendirilir. Düğümler arasındaki yaylar ya da kenarlar, düğümler arasındaki seyahat maliyeti ya da seyahat uzunluğu ile ilişkilidir ve genelde c_{ij} ile gösterilir."

Düğümler arasındaki seyahatin simetrik olup olmaması ise *i* düğümünden *j* düğümüne yapılacak seyahat uzunluğunun ya da seyahat maliyetinin, *j* düğümünden *i* düğümüne yapılacak seyahatin uzunluğuna ya da seyahatin maliyetine eşit olup olmaması ile ilgilidir. Şekil 1'de örnek bir araç rotalama probleminin çözümüne ait bir çizge örneği yer almaktadır. Bu şekilde x ve y eksenleri, müşterilerin bulunduğu konumların koordinatlarını gösterirken kullanılan her bir renk ise farklı bir rotayı ifade etmektedir.



Şekil 1. Örnek Bir Araç Rotalama Probleminin Çözümüne Ait Rotalar

Günümüz dünyasının araç rotalama problemleri, gerçek hayat zorluklarını daha iyi yansıtmayı amaçladığı için problemi ilk zamanlarında ele alan Dantzig ve Ramser (1959)'in, Clarke ve Wright (1964)'ın sunduğu modellerden son derece farklıdır (Braekers vd., 2016: 300). Gerçek hayatta karşılaşılan araç rotalama problemleri; asimetrik seyahat maliyetleri, heterojen araç filosu, değişken araç sayısı, hizmet süresi ile ilgili kısıtlamalar, öncelikli müşteriler, müşterilere yapılacak ziyaretlerin periyodik olması, müşterilerin stok miktarı ve problemin karmaşık amaçlara sahip olması gibi zorlukları içerebilir (Fisher, 1995: 2-3). Burada belirtilenlere benzer hususlar sebebiyle, literatürde araç rotalama probleminin farklı alt problem türleri bulunmaktadır. Laporte, bu nedenle araç rotalama probleminin, bir problem sınıfı olarak kabul edilebileceğini belirtmektedir (Laporte, 2009: 408).

1.1. Araç Rotalama Probleminin Matematiksel Modeli

Simetrik bir araç rotalama problemi $G = (V, E)$ çizgesi üzerinde tanımlansın. Bu çizgede $V = \{0, \dots, n\}$ düğümler kümesidir. 0 düğümü depoyu, diğer düğümler ise müşterileri temsil etmek üzere her müşteri, negatif olmayan q_i talebe sahiptir. Depoda ise birbirleriyle aynı özelliklere sahip *m* adet araç bulunmaktadır ve her birinin kapasitesi *Q* kadardır. Toplam seyahat maliyetini minimum yapacak rotaların belirleneceği bir probleme ilişkin varsayımlar ise şu şekildedir (Cordeau vd., 2007: 368-369):

i. Her müşteri, bir rota ile sadece bir kere ziyaret edilir.

ii. Her rota, depodan başlar ve depoda biter.

iii. Bir rotadaki müşterilerin toplam talebi, araç kapasitesi *Q*'dan fazla olamaz.

iv. Her bir rotanın uzunluğu, önceden belirlenen *L* limitini aşamaz (rota uzunluğu ile ilgili herhangi bir kısıtlama varsa).

Laporte vd. (1985) tarafından önerilen modelden kaynağını alan iki indisli araç akış formülasyonu Eşitlik (1.1) – (1.5)'te gösterildiği gibidir (Laporte, 2007: 812):

$$Min\ z = \sum_{[i,j] \in E} c_{ij} x_{ij}$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = 2m \qquad (1.1)$$

$$\sum_{i < k} x_{ik} + \sum_{j > k} x_{kj} = 2 \qquad (k \in V \setminus \{0\}) \qquad (1.2)$$

$$\sum_{\substack{i\in S, j\notin S \\ ya\ da \\ i\notin S, j\in S}} x_{ij} \geq 2b(S) \qquad (S \subseteq V \setminus \{0\}) \tag{1.3}$$

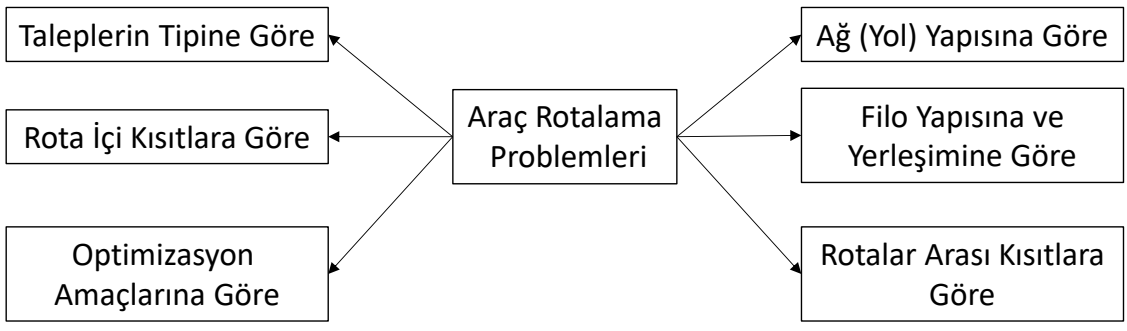
$$x_{ij} = 0\ ya\ da\ 1\ (i,j \in V \setminus \{0\}) \tag{1.4}$$

$$x_{0j} = 0{,}1\ ya\ da\ 2\ (j \in V \setminus \{0\}) \tag{1.5}$$

Yukarıda yer alan modelde, S düğüm kümesini, $b(S)$ ise bütün S düğümlerinin ziyaret edilebilmesi için gerekli araç sayısının alt sınırını ifade eder. Bu değer, genellikle $\lceil \sum_{i\in S} q_i/Q \rceil$ olarak kullanılır. Eşitlik (1.1), her turun depodan başlayarak yine depoda bitmesini sağlar. Eşitlik (1.2) ise her bir düğüme sadece bir aracın uğrayarak, buradan başka bir düğüme gitmesini sağlar. Eşitlik (1.3) ise alt turların elenmesini ve kapasite kısıtlarının ihlal edilmemesini sağlar. Eşitlik (1.4), karar değişkenlerinin 0 veya 1 tamsayı değerlerine sahip olması gerektiğini belirtirken; Eşitlik (1.5), tek müşteriden oluşan ve depo ile j müşterisi arasındaki rotayı belirtir. Bu modele benzer olarak ürün akış ve küme bölümleme formülasyonları da mevcuttur (Laporte vd., 1985: 1052; Laporte, 2007: 812; Laporte, 2009: 410; Çetin, 2013: 31).

1.2. Araç Rotalama Problemlerinin Sınıflandırılması

Araç rotalama problemi, ortaya çıktığı günden itibaren farklı türde gerçek dünya problemlerini ele almaktadır. Benzer karakteristiklere sahip problemleri bir araya getirerek sınıflandırma yapılabilir; ancak literatürde genel olarak kabul görmüş bir sınıflandırma biçimi yoktur. Genel olarak, problemler Şekil 2'de gösterildiği gibi sınıflandırılabilir.



Şekil 2. Araç Rotalama Problemlerinin Sınıflandırılması
Kaynak: Irnich vd. (2014: 8)

1.2.1. Ağ (Yol) Yapısına Göre Araç Rotalama Problemleri

Araç rotalama probleminde ziyaret edilecek noktalar arasındaki mesafelerin ya da ulaşım maliyetlerinin yapısına göre problem, simetrik ya da asimetrik olarak isimlendirilir. Eğer bir müşteriden başka bir müşteriye gitmenin maliyeti ya da uzunluğu, tam tersi yöndeki yolculuğun maliyetine ya da uzunluğuna eşitse bu problem, simetrik bir araç rotalama problemi olarak isimlendirilir. Aksi taktirde, asimetrik araç rotalama problemi söz konusudur. Yoldaki trafik durumunun dikkate alındığı durumlarda ise problem ayrıca stokastik bir yapı kazanabilir.

Araç rotalama probleminde araçlar, bütün müşterilere hizmet sunduktan sonra başlangıç noktası olan depoya dönüp dönmemelerine göre de sınıflandırılabilir. Araç başlangıç noktasına dönüyorsa kapalı araç rotalama problemi, başlangıç noktasına dönmüyorsa açık araç rotalama problemi olarak isimlendirilir.

1.2.2. Ulaştırma Taleplerinin Tipine Göre Araç Rotalama Problemleri

Araç rotalama problemlerinde müşterilere yalnızca dağıtım yapılmaz. Bazı durumlarda müşterilerden çeşitli ürünlerin toplanması da gerekebilir. Örneğin, bir beyaz eşya markasının servis işlemlerini yürüten bir işletme, kurulum yapmak amacıyla müşterilere beyaz eşya götürebileceği gibi farklı müşterilerden de arızalanmış beyaz eşyaları toplayabilir. Benzer şekilde, gıda ya da geri dönüşüm sektörlerinde de benzer durumlar sıklıkla söz konusu olmaktadır. Gerçek hayat uygulamalarında dağıtım ve toplama işi genelde aynı araç tarafından yapıldığı için araç rotalama problemleri, işlem sırasına göre aşağıdaki gibi sınıflandırılabilir (Atmaca, 2012: 13):

i. *Önce dağıtım sonra toplama:* Öncelikle dağıtımı yapılacak müşterilere gidilir. Dağıtım işlemi tamamlandıktan sonra toplama işlemine geçilir. Eğer bir müşterinin hem dağıtım hem de toplama talebi varsa problemin yapısı gereği müşteri, birden fazla ziyaret edilebilir.

ii. *Karışık dağıtım toplama:* Dağıtım ve toplama işleminin belirli bir sırası yoktur. Bu problemde de müşteri, birden fazla ziyaret edilebilir.

iii. *Eş zamanlı dağıtım ve toplama:* Müşterinin dağıtım ve toplama talebi varsa aynı ziyarette iki işlem birlikte yapılır. Örneğin, gazete dağıtımı yapan bir işletmenin

aracı, gittiği büfeye günün gazetesini bırakırken, önceki günün satılmayan gazetelerini toplayabilir.

Müşterilerden gelen talebe göre bir başka sınıflandırma da toplama ve dağıtım noktalarının farklılığına göredir. Müşteriler, farklı noktalar arasında toplama ve dağıtım işleminin yapılmasını istiyorsa (Psaraftis, 1980: 131) buna çoktan-çoğa (many-to-many) araç rotalama problemi denir. Noktalar arasında yapılan taşıma işlemi, insan taşımacılığını gerektiriyorsa bu problem, kapıdan kapıya araç rotalama problemi (dial-a ride problem) olarak isimlendirilir (Cordeau ve Laporte, 2007: 29).

Müşterilerden gelen talepler, belirli bir düzende tekrarlanıyorsa periyodik araç rotalama problemi (periodic vehicle routing problem) ve stok rotalama problemi (inventory routing problem) söz konusu olur. Periyodik araç rotalama probleminde planlama günlük bazda yapılmaz, bunun yerine müşterilere belirli periyodlarda teslimat yapılacak şekilde planlama yapılır. Bu problemde, müşterilerin teslimatlarını belirlenen periyotta yapabilmek için bir planlama yapılır ve aynı zamanda kullanılan her bir aracın rotasının toplam kat edilen mesafeyi minimize edecek şekilde olması istenir (Alonso vd., 2008: 964). Stok rotalama problemi, belirli bir planlama dönemi içinde tek bir ürünün tek bir tesisten bir müşteri kümesine tekrarlayan biçimde dağıtımı ile ilgilenir. Amaç, planlama döneminde müşterinin stoksuz kalmayacak şekilde ortalama dağıtım maliyetinin minimum yapılmasıdır (Campbell vd., 1998: 96).

Bütün görevlerin tek bir araç tarafından bir kerede yapıldığı varsayımı söz konusu olduğunda hizmetler bölünmemektedir (Irnich vd., 2014: 12). Bazı durumlarda ise maliyet (uzaklık) odaklı bir yaklaşımla talepte bulunan noktaya verilecek hizmet, birden fazla araç arasında bölünür. Bu tarz problemlere bölünmüş dağıtım araç rotalama problemi (split delivery vehicle routing problem) denir (Dror ve Trudeau, 1990: 384).

Araç rotalama problemleri, içerisinde bulunulan sistemde bilginin deterministik olarak ya da stokastik bir şekilde ortaya çıkması ya da planlama aşamasında ihtiyaç duyulan bilginin mevcut olup olmamasına göre sınıflandırılabilir. Bu sınıflandırma Tablo 1'de gösterildiği gibidir.

Tablo 1. Bilginin Çıkış Şekli ve Bilinirliğine Göre Sınıflandırma

		Bilginin kesinliği	
		Deterministik Girdi	Stokastik Girdi
Bilgi Değerlendirmesi	Girdiler önceden biliniyor	Statik ve deterministik	Statik ve stokastik
	Girdiler zaman içerisinde değişiyor	Dinamik ve deterministik	Dinamik ve stokastik

Kaynak: Pillac vd. (2013: 2)

1.2.3. Rota İçi Kısıtlara Göre Araç Rotalama Problemleri

Talepte bulunan müşterilere hizmet sunmak için gerekli rotaların belirlenmesinde dikkate alınan kısıtlara göre araç rotalama problemleri sınıflandırılabilir. Sadece kapasite ile ilgili kısıtlara sahip araç rotalama problemleri, kapasite kısıtlı araç rotalama problemi (capacitated vehicle routing problem) olarak isimlendirilmektir. Bölüm 1.1'de Eşitlik (1.1) – (1.5)'te gösterilen model, kapasite kısıtlı bir araç rotalama problemidir. Benzer şekilde rota uzunluğu ile ilgili kısıtlar da söz konusu olabilir.

Araç rotalama probleminin varsayımları arasında belirli bir planlama periyodunda her aracın bir kez kullanılması dolaylı olarak yer almaktadır. Araç sayısı m önceden biliniyorsa ve müşterilerin toplam talep miktarı Q göreli olarak küçük ise, araç rotaları belirlendikten sonra bazı rotalar aynı araca atanarak daha az sayıda araç ile aynı dağıtım işlemi yapılabilir (Taillard vd., 1996: 1065). Bu tür problemlere çoklu araç kullanımlı araç rotalama problemi (vehicle routing problem with multiple use of vehicles) denir.

Müşterilere hizmet verilirken zaman unsuru planlamalara dâhil edilerek zaman pencereleri kullanılabilir. Zaman pencereleri kullanıldığında müşterilere hizmet verilebilecek en erken ve en geç zaman noktaları belirlenerek rotalama işlemi yapılır. Bu problemlerde müşterilerin talepleri ve zaman pencereleri kesin olarak bilinir ve problem, zaman pencereli araç rotalama problemi (vehicle routing problem with time windows) olarak isimlendirilir (Bräysy ve Gendreau, 2005: 104-105).

1.2.4. Filo ve Depo Özelliklerine Göre Araç Rotalama Problemleri

Rotalama işleminde kullanılan araçların özellikleri ve depo sayısı ile ilgili literatürde değişik tipte problemler yer almaktadır. Bunlardan birisi, birden fazla depoya sahip olunan çok depolu araç rotalama problemidir (multi-depot vehicle routing problem). Bu problemin amacı, bütün müşterilere araç kapasitelerini aşmayacak ve

toplam seyahat uzunluğunu minimum yapacak şekilde hizmet sunmaktır (Lim ve Wang, 2005: 397).

Aynı özelliklere sahip *m* adet aracın kullanılması araç rotalama probleminin varsayımlarından biridir. Gerçek hayatta bazen bu varsayım geçerli olmayabilir. Eğer birbirinden farklı özelliklerde araçlar söz konusu ise heterojen ya da karma filolu araç rotalama problemi (heterogeneous or mixed fleet vehicle routing problem) ortaya çıkmaktadır. Literatürde değişik tipleri bulunmakla birlikte, bunlar filo sınırlaması (sınırlı/sınırsız) ve maliyet (değişken/sabit) temellidir (Penna vd., 2013: 202).

1.2.5. Rotalar Arası Kısıtlara Göre Araç Rotalama Problemleri

Rotalar arası süre, teslimat miktarı gibi farklılıkları azaltmak için "*dengeleme*" kısıtları, belirli bir depoya atanacak araç sayısı gibi global olarak sınırlı kaynakların paylaşımı ile ilgili "*rotalar arası kaynak kısıtları*" ve farklı tipte araçlar söz konusu olduğunda bu araçların faaliyetleri arasındaki "*senkronizasyon*" ile ilgili kısıtları (Irnich vd., 2014: 19-20) içeren araç rotalama problemleri bu grup altında incelenebilir.

1.2.6. Optimizasyon Amaçlarına Göre Araç Rotalama Problemleri

Ele alınan araç rotalama problemi yalnızca maliyet, süre, hizmet verilen müşteri sayısı gibi tek bir özelliğe odaklanıyorsa, bu problemlere tek amaçlı araç rotalama problemleri denir. Benzer şekilde, birden fazla amacın öncelik sırasına göre optimize edildiği problemlere hiyerarşik amaçlı araç rotalama problemleri ve birden fazla amacın aynı anda optimize edildiği problemlere ise çok amaçlı araç rotalama problemleri denir (Irnich vd., 2014: 21-22).

1.3. Araç Rotalama Problemlerinin Çözüm Yöntemleri

Araç rotalama problemleri, gezgin satıcı probleminin genelleştirilmiş hali olduğu için *NP-Zor* sınıfında yer alır. Bu nedenle problem çözülürken belirli boyutlara kadar kesin yöntemler (exact methods) ile çözüm değeri elde edilebilir. Problemin büyüklüğü arttıkça kesin yöntemler yerine sezgisel (heuristics) ve meta sezgisel (metaheuristics) yöntemler kullanılarak optimal ya da optimale yakın çözüm değerleri elde edilebilir. Bu bölümde araç rotalama problemlerinde kullanılan çözüm yöntemleri genel olarak ele alınacaktır.

1.3.1. Kesin Yöntemler

Kesin yöntemler kullanılarak araç rotalama problemleri çözüldüğünde optimal çözüm elde edilebilmektedir; ancak kesin yöntemlerin kullanılabilirliği, problem boyutu ile doğrudan ilişkilidir. Araç rotalama problemlerinde dal sınır ve dal kesme yöntemleri, sıklıkla kullanılan kesin yöntemlerdendir.

1.3.1.1. Dal sınır yöntemleri

Araç rotalama problemi için oluşturulan matematiksel modelde düğümlerin birbiriyle bağlantısı ve kapasite ile ilgili kısıtlar gevşetilerek problemler çözülür. Kısıtlarda yapılan bu gevşetme işlemi, alt sınır değerlerinin elde edilmesi için kullanılır (Toth ve Vigo, 2002: 29-30). Problemin simetrik olması durumunda yayılım ağaçları (spanning trees) ve b-eşleştirme (b-matching) gevşetme kullanılır (Toth ve Vigo, 1998: 15). Asimetrik problemlerde ise atama problemi (assignment problem) gevşetmesi kullanılır (Laporte vd., 1992: 473).

1.3.1.2. Dal kesme yöntemleri

Dal kesme yöntemi, dal sınır yönteminin bir uzantısıdır. İki yöntem arasındaki temel fark, arama ağacının düğümlerindeki çözümlerin işlenme şeklidir. Dal kesme algoritmasında, gerektiğinde doğrusal programlama gevşetmesini güçlendirmek için ek kısıtlamalar kullanılır (Gansterer vd., 2018: 363). Elde edilen çözüm optimal değil ise kesirli değere sahip değişkene alt ve üst sınır eklenerek dallandırma işlemi yapılır ve problem, iki yeni probleme dönüştürülür. Bu iki probleme ait çözümden daha iyi olan, asıl problem için optimal çözüm olacaktır. Kesme düzlemi ile arama (sayma) ağacının bütünleşmesi, dal kesme yönteminin özünü oluşturur (Naddef ve Rinaldi, 2002: 56).

1.3.2. Sezgisel Yöntemler

Sezgisel yöntemler; problemin boyutuna bağlı olarak çözüm elde etmek zorlaştığında kullanılan, optimal veya optimale yakın çözüm veren yaklaşımlardır. Araç rotalama problemlerinde rota kurucu (route construction) sezgiselleri, rota iyileştirme (route improvement) sezgiselleri ve iki aşamalı (two-phase) sezgiseller kullanılmaktadır.

1.3.2.1. Rota kurucu sezgiseller

Rota kurucu sezgiseller, uygun çözüm elde edilinceye kadar sırayla yayları (kenarları) seçer. Yayların (kenarların) seçiminde, maliyet minimizasyonu ve araç kapasite kısıtlarının ihlal edilmemesi gözetilir (Fisher, 1995: 7). Clarke ve Wright (1964) tarafından geliştirilen tasarruf algoritması (savings algorithm), rota kurucu sezgiseller arasında en bilindik yöntemlerin başında gelmektedir. Daha sonra geliştirilen rota kurucu sezgiseller, genel olarak bu algoritmayı iyileştirmişlerdir. Tasarruf algoritmasının dışında en yakın komşuluk sezgiseli, rastgele rota oluşturma ve ekleme tabanlı yöntemler gibi yaklaşımlarla rotalar oluşturulabilmektedir.

1.3.2.2. Rota iyileştirici sezgiseller

Rota iyileştirici sezgiseller, uygun bir başlangıç çözüm ile aramaya başlayarak, aramanın her yinelemesinde bu çözümü iyileştirebilmek için çeşitli kombinasyonlar dener. Denenen her kombinasyonun uygunluğu ve çözüm değeri kontrol edilir (Eryavuz ve Gencer, 2001: 141). Rota içindeki komşuluklar ya da rotalar arasındaki komşuluklar kullanılıp değişiklikler yapılarak çözümdeki iyileşme araştırılır (Cordeau vd., 2007: 379). Lin (1965) tarafından önerilen λ-opt sezgiseli, sık kullanılan rota iyileştirici sezgisellerdendir. Söz konusu sezgiselde, λ adet kenar çözümden silinir ve bunların yerleri değiştirilerek yeni çözüm elde edilir. İşlem kolaylığı açısından genellikle λ değeri, 2 ya da 3 olarak kullanılır.

1.3.2.3. İki aşamalı sezgiseller

İki aşamalı sezgisellerde öncellikle gruplama, daha sonra rotalama işlemi yapılmaktadır. Gruplama aşamasında araç kapasitelerinin aşılmamasına dikkat edilirken, rotalama aşamasında gezgin satıcı problemi çözüm yöntemleri kullanılarak uygun çözüm aranır. Süpürme algoritması (sweeping algorithm) ve petal algoritması, en sık kullanılan iki aşamalı sezgisellerdendir. Süpürme algoritmasında noktalar, koordinat düzleminde ele alınır. Saat yönünde ve saat yönünün tersinde en küçük açı yapan nokta seçilir ve hareket edilen yöne göre kapasiteyi aşmayacak şekilde süpürme işlemi yapılarak noktalar rotaya dâhil edilir. Kapasite sınırına ulaşıldığında, benzer şekilde diğer rotalar oluşturulur. Daha sonra rotalar arasında değişiklikler yapılarak toplam mesafe azaltılmaya çalışılır (Gillet ve Miller, 1974: 342). Petal algoritması ise süpürme algoritmasının uzantısıdır ve küme

bölümleme biçiminde bir problemi çözerek nihai bir seçim yapar (Laporte vd., 2000: 288).

1.3.3. Meta Sezgisel Yöntemler

Meta sezgiseller, belirli bir tür problemi sezgisel bir yaklaşım ile ele alabilmek için genel bir yapı ve strateji rehberliği sağlayan çözüm yöntemleridir (Hillier ve Lieberman, 2015: 617). Meta sezgisel yöntemler, çözüm uzayını çeşitli yaklaşımlarla araştırarak optimum ya da optimuma yakın çözümler elde etmektedir. Araç rotalama problemlerinin çözümünde kullanılan meta sezgisel yöntemlere dair detaylı bilgiler, ikinci bölümde yer almaktadır.

1.4. Eş Zamanlı Topla Dağıt Araç Rotalama Problemleri

Bir araç rotalama probleminde, merkezi depodan müşterilere ileri yönlü (linehaul) mal taşıma ve müşterilerin bulunduğu konumlardan da merkezi depoya geri yönlü (backhaul) taşıma hizmetleri sunuluyorsa bu tür problemlere, geri taşımalı araç rotalama problemleri denilmektedir. Söz konusu iki yönlü hizmetin sunulma biçimine göre Bölüm 1.2.2'de yapılan sınıflandırmaya göre önce dağıtım sonra toplama, karma toplama dağıtma ve eş zamanlı toplama dağıtma şeklinde üç tür problem söz konusudur. Bu tezde toplama ve dağıtım faaliyetlerinin eş zamanlı olarak yapıldığı araç rotalama problemleri ele alınmıştır. Tezin bu kısmında Eş Zamanlı Topla Dağıt Araç Rotalama Problemi (EZTDARP) ile ilgili bilgiler ve söz konusu problem ile ilgili literatür taraması verilmiştir.

EZTDARP, müşterilerden merkezi depoya bir taşıma faaliyetini içerdiği için tersine lojistik ile ilişkilidir. Bu problem, akla ilk olarak dolu şişelerin teslim edilip, boş şişelerin teslim alındığı içecek sektörünü akla getirmektedir; ancak teslim alınan ürünün aynı veya benzer şekilde tekrar kullanılabildiği, geri dönüşüm yoluyla farklı şekillerde kullanılabildiği ya da ürünün, çevreye zarar verme ihtimaline karşın denetleyici/düzenleyici makamlar tarafından toplanılmasının zorunlu tutulduğu durumlarda da ortaya çıkmaktadır. Müşterilerden merkezi depoya geri taşıma işleminin ürünleri tekrar kullanma imkânı tanıması sebebiyle bu araç rotalama problemlerinin çevre üzerinde olumlu etkiye sahip olduğunu söylemek mümkündür.

1.4.1. EZTDARP'nin Matematiksel Modeli

EZTDARP, Bölüm 1.1'de yer alan kapasite kısıtlı araç rotalama probleminden farklı özelliklere sahiptir. Tezin bu kısmında problemin daha iyi anlaşılabilmesi için literatürde yaygın bir şekilde kullanılan Dethloff (2001)'un formülasyonu Eşitlik (1.6) – (1.14)'te verilmiştir. Söz konusu formülasyonda 0 indisi, depoyu ifade etmekte olup kullanılan gösterimler ve bunların açıklamaları Tablo 2'de gösterildiği gibidir.

Tablo 2. Dethloff (2001)'un Modeline Ait Gösterimler

Gösterim	Gösterimin açıklaması
J	Bütün müşteri konumlarının kümesi
J_0	Bütün düğümlerin kümesi (bütün müşteriler ve depo)
V	Bütün araçların kümesi
C	Araç kapasitesi
c_{ij}	Düğümler arası uzaklık $i \in J_0, j \in J_0, i \neq j; c_{ii} = M, i \in J, c_{00} = 0$
D_j	j müşterisine yapılacak dağıtım miktarı $(j \in J)$
n	Düğüm sayısı, $n = \lvert J_0 \rvert$
P_j	j müşterisinden yapılacak toplama miktarı $(j \in J)$
M	Büyük bir sayı, $M = max\{\sum_{j\in J}(D_j + P_j), \sum_{i\in J_0}\sum_{j\in J_0, j\neq i} c_{ij}\}$
l'_v	v aracının depodan ayrılırken yükü $(v \in V)$
l_j	j müşterisine hizmet verildikten sonra aracın yükü $(j \in J)$
π_j	Alt turları yasaklamak için kullanılan değişken, $j \in J$ düğümünün rotadaki pozisyonu olarak yorumlanabilir
x_{ijv}	v aracının $i \in J_0$'den direkt olarak $j \in J_0$'e gidip gitmediğini gösteren 0-1 değerli değişken $(v \in V)$

$$\text{Min } z = \sum_{i\in J_0}\sum_{j\in J_0}\sum_{v\in V} c_{ij}x_{ijv}$$

$$\sum_{i\in J_0}\sum_{v\in V} x_{ijv} = 1 \ (j \in J) \tag{1.6}$$

$$\sum_{i\in J_0} x_{isv} = \sum_{j\in J_0} x_{sjv} \ (s \in J, v \in V) \tag{1.7}$$

$$l'_v = \sum_{i\in J_0}\sum_{j\in J} D_j x_{ijv} \ (v \in V) \tag{1.8}$$

$$l_j \geq l'_v - D_j + P_j - M(1 - x_{0jv}) \ (j \in J, v \in V) \tag{1.9}$$

$$l_j \geq l_i - D_j + P_j - M(1 - \sum_{v\in V} x_{ijv}) \ (i \in J, j \in J, j \neq i) \tag{1.10}$$

$$l'_v \leq C \ (v \in V) \tag{1.11}$$

$$l_j \leq C \ (j \in J) \tag{1.12}$$

$$\pi_j \geq \pi_i + 1 - n\left(1 - \sum_{v \in V} x_{ijv}\right) \ (i \in J, j \in J, j \neq i) \quad (1.13)$$

$$\pi_j \geq 0 \quad (1.14)$$

$$x_{ijv} \in \{0,1\} \ (i \in J_0, j \in J_0, v \in V)$$

Yukarıda verilmiş olan modelin amaç fonksiyonu incelendiğinde, müşterilere hizmet verecek en kısa mesafeli rotanın arandığı görülmektedir. x_{ijv}, v aracı ile i müşterisinden j müşterisine gidiyorsa 1 değerine; aksi taktirde 0 değerini alan bir karar değişkenidir. Eşitlik (1.6)'da yer alan ilk kısıt, müşterilerin yalnızca 1 kere ziyaret edilmesini gerektirmektedir. Eşitlik (1.7)'de yer alan kısıt ise her bir müşterinin bulunduğu konuma hizmet vermek için gelen araç ile bu konumdan ayrılan aracın aynı araç olmasını sağlamaktadır. Eşitlik (1.8)'de yer alan kısıt, depodan ayrılan bir aracın yapacağı dağıtımların toplamına eşit olan yükünü göstermektedir. Eşitlik (1.9)'da yer alan kısıt, depodan ayrılan bir aracın ilk müşterisine hizmet verdikten sonraki yükünü ifade etmektedir. Eşitlik (1.10) ise Eşitlik (1.9)'a benzer şekilde, aracın seyir halindeyken yükünü ifade etmektedir. Eşitlik (1.11) ve (1.12), araç yükünün hiçbir zaman (ilk müşteriye hizmet verdikten sonra ya da seyir halinde iken) araç kapasitesini aşmamasını sağlamaktadır. Eşitlik (1.13), alt tur oluşmasını engelleyen bir kısıttır ve Eşitlik (1.14), bunu sağlayan π_j değişken değerinin 0 ya da daha büyük olmasını sağlamaktadır.

Model bir bütün olarak düşünüldüğünde, Bölüm 1.1'de verilen temel modelden farklı olarak bir aracın hizmet vermek için uğradığı her bir müşteride dağıtım işlemine ek olarak toplama işlemi de yaptığı görülmektedir. Bu toplama ve dağıtım işlemlerinin sonucunda aracın sahip olduğu toplam yükün, rotanın uygunluğu açısından hiçbir şekilde araç kapasitesini aşmaması için ilgili kısıtlar eklenmiştir. Problemin sahip olduğu bu özellik, toplam kat edilen mesafeyi minimum yapacak araç rotalarını belirlemenin zorluğuna ek olarak, müşterilerin toplama ve dağıtım hizmetlerine olan taleplerine bağlı olarak araç yükünde ortaya çıkacak dalgalanmaların kontrolü gibi bir zorluğu da beraberinde getirmektedir. Söz konusu dalgalanmalar sebebiyle rotanın herhangi bir noktasında araçtaki yükün, araç kapasitesini aşmaması gerektiği gibi aşırı ihtiyatlı yaklaşımla oluşturulan rotalar sonucunda araçlarda kapasitenin çok altında yük miktarlarının ortaya çıkması da rota maliyetleri açısından izlenilmesi gereken bir husustur.

1.4.2. Probleme İlişkin Literatür Taraması

Tezin bu bölümünde ele alınan probleme ilişkin yapılan literatür taramasından elde edilen bulgular paylaşılmıştır. Literatür taraması yapılırken Web of Science, Scopus, Google Akademik gibi önde gelen bilimsel veri tabanlarından faydalanılmıştır. Söz konusu veri kaynaklarında tarama, makaleler üzerinden yapılmıştır. Tarama, 1989 yılından günümüze kadar geçen süreyi kapsamaktadır ve tezin içeriği ile ilişkili olan çalışmaları içermektedir. Bu kısımda ele alınan çalışmalar, tezin kapsamına uygun şekilde sınırlandırıldığı için problem ile ilgili daha fazla bilgi, Koç vd. (2020)'nin literatür taramasından edinilebilir.

EZTDARP, literatürde ilk olarak Min (1989) tarafından ele alınmıştır. Bu çalışmada merkezi bir kütüphane ve 22 kütüphane şubesi arasında kütüphane materyallerinin eş zamanlı olarak 2 adet eşit kapasiteli kamyon ile toplanıp dağıtıldığı bir problem olarak ele alınmıştır. Problem, müşterilerin coğrafi olarak kümelenmesi, şoförlerin bu kümelere atanması ve kamyon kapasiteleri gözetilerek rota yapısının oluşturulması şeklinde üç aşamalı olarak çözülmüştür. Yaşanan bu gelişmeye karşın bilim insanları, uzun bir süre boyunca bu probleme pek ilgi göstermemiştir.

Salhi ve Nagy (1999), geri yönlü taşımalı araç rotalama problemleri için bir ekleme sezgiseli (insertion heuristic) geliştirmeyi ve bu sezgiseli birden fazla deponun olduğu durumlara uyarlamayı amaçlamıştır. Geliştirilen sezgisel ile geri yönlü taşımalı araç rotalama problemleri için iyi başlangıç çözümler elde edilmesi hedeflenmektedir. Önerilen sezgisel üç farklı şekilde ekleme yapmaktadır: (i) Her seferinde 1 geri taşıma müşterisinin eklenmesi, (ii) Her seferinde 2 geri taşıma müşterisinin eklenmesi, (iii) Geri taşıma müşterilerinin küme olarak eklenmesidir. Çoklu depo durumu için ise öncelikle ileri yönlü taşıma müşterileri sınırda yer alan (iki depo arasında yaklaşık yarı yolda olan) müşteriler ve sınırda olmayan müşteriler şeklinde ikiye bölünür ve sınırda olmayan müşteriler en yakın depoya atanır. Depolar için araç rotalama problemleri çözülür ve sınırda olan müşterilerden her seferinde biri araçlara eklenir. Daha sonrasında ise sezgiseller aynı şekilde kullanılır. Bu çalışmada önerilen sezgiseli uygulamak için oluşturulan problem kümeleri, literatürde yaygın bir şekilde kıyaslama problemleri olarak kullanılmaktadır.

Dethloff (2001), EZTDARP için bir ekleme sezgiseli önermiştir. Sadece toplam mesafeyi kriter olarak alan ekleme sezgiselinin araç yükündeki dalgalanmayı dikkate almadığını belirterek, artık kapasite kriterini, radyal fazla yük kriterini ve bunların kombinasyonlarını dikkate alan ekleme sezgiselini geliştirmiştir. Bu çalışmada da önerilen sezgiseli test etmek için oluşturulan problem kümesi, literatürde Salhi ve Nagy (1999)'nin problem kümesi gibi literatürde yaygın bir şekilde kıyaslama problemleri olarak kullanılmaktadır.

Montané ve Galvão (2006), EZTDARP çözümünde kullanılmak üzere bir tabu arama algoritması geliştirmiştir. Rotalar arası 3 prosedür (yer değiştirme – relocation, takas – interchange ve çaprazlama – crossover), rota içinde ise 2-opt prosedürü kullanılarak komşu çözümler elde edilmiştir. Yoğunlaştırma ve çeşitlendirme, sıklık cezalandırma (uzun dönemli hafıza) ile sağlanmıştır. Tabu arama için çeşitli gruplandırma ve rotalama stratejilerinin kombinasyonundan oluşan dört sezgisel yöntem ile başlangıç çözümler elde edilmiştir. Geliştirilen algoritma, Pascal dilinde kodlanmıştır.

Chen ve Wu (2006), EZTDARP çözümünün elde edilebilmesinde hibrit bir yaklaşım benimsemiştir. Başlangıç çözümler, ekleme temelli bir sezgisel ile elde edilmiştir. Kayıttan kayıta seyahat (record-to-record travel), tabu listeleri ve rota iyileştirme prosedürlerinden oluşan hibrit yaklaşımla çözüm üretilmiştir. Komşu çözümlerin elde edilebilmesi için rotalar arasında 2-exchange (değişim), değiş tokuş/öteleme (swap/shift); rota içinde ise 2-opt ve Or-opt prosedürleri kullanılmıştır. Algoritma, C dilinde kodlanmıştır.

Ropke ve Pisinger (2006), çeşitli geri yönlü araç rotalama problemlerini çözmek için birleştirilmiş bir sezgisel önermiştir. Bu sezgisel, geniş komşuluk arama sezgiselinin geliştirilmiş halidir. Algoritma, başlangıç çözümü basit ekleme ve pişmanlık ekleme sezgiselleri şeklinde iki sınıf altında gruplandırılan sezgiseller ile elde etmiştir. Aramanın çeşitlendirme ve yoğunlaştırma süreçleri, silme ve ekleme sezgiselleri kullanılarak yürütülmüştür. İzleme ve öğrenme tabakası ile bu sezgisellerin seçilme olasılıkları değiştirilmiştir. Her geniş komşuluk araması adımından sonra amaç fonksiyonu, tavlama benzetimi paradigması ile kontrol edilmiştir. Geliştirilen algoritma, C++ programlama dili kullanılarak uygulanmıştır.

Wassan vd. (2008), EZTDARP söz konusu olduğunda araçlardaki yük miktarının dalgalandığını, yani her müşteride artabileceğini ya da azalabileceğini belirterek problem çözümünde kullanılacak algoritmaların bu dalgalanmalarla başa çıkabilecek özellikte olması gerektiğini ifade etmiştir. Bu zorluğu aşmak için reaktif tabu arama (reactive tabu search) algoritması önerilmiştir. Başlangıç çözümler değiştirilmiş süpürme algoritması ile elde edilmiştir. İyileştirme aşamasında ise öteleme, değiş tokuş ve tersine çevirme (reverse) prosedürleri kullanılmıştır. Geliştirilen algoritma, Fortran dili ile kodlanmıştır.

Ai ve Kachitvichyanukul (2009), EZTDARP için bir formülasyon ve parçacık sürü optimizasyonu algoritması önermiştir. Söz konusu formülasyon, literatürdeki birkaç formülasyonun genelleştirilmiştir halidir. Önerilen algoritmada, arama değişkenleri için kesikli değerler kullanmak yerine reel değerler kullanmaktadır ve herhangi bir yerel arama yöntemi olmaksızın uygulanmaktadır. Oluşturulan rotalar, 2-opt prosedürü ile iyileştirilmiştir. Algoritma, C# dili ile kodlanmıştır.

Gajpal ve Abad (2009), karınca koloni sistemini kullanarak EZTDARP için çözüm yöntemi geliştirmeyi amaçlamıştır. Önerilen algoritmanın aynı zamanda geri taşımalı ve karma yüklü araç rotalama problemini de çözebildiği belirtilmiştir. Algoritma, başlangıç çözümü en yakın komşuluk (nearest neighborhood) sezgiseli ile elde etmektedir. Komşu çözümler ise 2-opt, ekleme/takas (insertion/interchange) ve çok rotalı alt yol değişim (sub-path exchange multi-route) prosedürleri ile elde edilmektedir. Algoritma, C dilinde kodlanmıştır.

Karlaftis vd. (2009), toplama, dağıtma ve son teslim zamanı kısıtlarına sahip bir araç rotalama problemini formüle etmişlerdir. Rotaların kurulması için hibrit bir genetik algoritma kullanarak özel bir konteyner gemi filosu için problemi çözmüşlerdir. Algoritma için gerekli başlangıç popülasyon, rastgele oluşturulmuştur ve ebeveynler uyum değerine göre sıralanarak seçilmiştir. Önerilen algoritma, Ege Denizi'nde bulunan 25 ada arasında uygulanmıştır. Maksimum hizmet süresi 40 saat, hizmet için gerekli gemi sayısı ise 5 olarak belirlenmiştir.

Zachariadis vd. (2009), tabu arama ve yönlendirilmiş yerel arama yöntemlerini bir araya getiren hibrit bir meta sezgisel ile EZTDARP için çözüm elde etmeye çalışmıştır. Rotalar, maliyet tasarrufuna (cost savings) dayanan sezgisel ile kurulmuştur. Komşu

çözümler ise müşteri yer değiştirme, müşteri değişimi, rota değişimi 1, rota değişimi 2 prosedürleri ile elde edilmiştir. Algoritma, C# dilinde kodlanmıştır.

Zachariadis vd. (2010), yüksek kaliteli çözümler elde edebilmek için umut verici çözümlerin özelliklerini toplayan ve bunları bir araya getiren uyarlamalı hafıza algoritması geliştirmiştir. Önerilen strateji, aramayı çözüm uzayının farklı bölgelerine yönlendirmek için uyarlamalı hafızadan çıkarılan rotalama bilgi miktarını sistematik olarak maksimize etmek için yenilikçi bir hafıza mekanizması kullanmaktadır. Başlangıç çözüm, ağırlıklı tasarruf sezgiseli kullanılarak elde edilmiştir. Elde edilen çözümler, tabu arama meta sezgiseli kullanılarak ve bunun her yinelemesinde (1-0) değişim, (1-1) değişim ve 2-opt operatörleri aracılığı ile rastgele şekilde iyileştirilmiştir. Algoritma, C# dilinde uygulanmıştır.

Çatay (2010), EZTDARP için yeni bir tasarruf tabanlı görünürlük fonksiyonu ve feromon güncelleme prosedürü kullanan bir karınca kolonisi algoritması önermiştir. Başlangıç rota, en yakın komşu sezgiseli kullanılarak oluşturulmuştur. Komşu çözümlerin elde edilmesi için rota içinde ve rotalar arasında hareket (move), değiş tokuş prosedürlerine dayanan dört mekanizma kullanılmıştır. Önerilen algoritma, C++ dili kullanılarak kodlanmıştır.

Mingyong ve Erbao (2010), genel bir karma tam sayılı programlama modeli kurmuştur. Söz konusu model, zaman penceresine sahiptir. Bazı klasik araç rotalama problemleri, bu modelin özel bir durumudur. Bu problemin çözümü için geliştirilmiş diferansiyel gelişim algoritması (improved differential evolution algortihm) önerilmiştir. Başlangıç çözüm, rastgele şekilde oluşturulmuştur. Mevcut algoritmadan farklı olarak geliştirilmiş diferansiyel gelişim operatörleri ve mutasyon ile arama yapılmıştır. Uygun olmayan çözümler, cezalandırma tekniği ile cezalandırılmıştır. Algoritma, MATLAB kullanılarak kodlanmıştır.

Subramanian vd. (2010), EZTDARP çözümünde paralel bir yaklaşım kullanmıştır. Paralel algoritma, bir yinelemeli yerel arama çerçevesine entegre edilmiş rastgele komşuluk sıralamalı (random neighborhood ordering) değişken komşuluk iniş prosedürlerinden oluşan çoklu başlangıçlı bir sezgisel tarama içine gömülüdür. Geliştirilen algoritma, ana kısım ve üç temel kısımdan oluşmaktadır. İlk kısım, araç sayısını tahmin etmektedir, ikinci kısım otomatik kalibrasyon parametresi ile ilgilidir,

üçüncü kısım ise optimizasyon aşamasıdır. Başlangıç çözüm, açgözlü ekleme (greedy insertion) sezgiseli ile kurulmuştur. Komşu çözümlerin elde edilmesi için altı rotalar arası (bunlardan beş tanesi, λ-değişime dayanmaktadır, bir tanesi ise çapraz değişimdir), dört tane de rota içi (2-opt, Or-opt, değişim, tersine çevirme) prosedür kullanılmıştır. Algoritma, C++ kullanılarak kodlanmıştır.

Souza vd. (2011), yinelemeli yerel arama, değişken komşuluk iniş ve GENIUS sezgiseline dayanan ve GENILS şeklinde adlandırdıkları hibrit bir sezgisel önererek EZTDARP'yi çözmeye çalışmıştır. Yöntemde başlangıç çözümler, ekleme sezgiseline dayalı 3 sezgisel kullanılarak oluşturulmuştur. Komşu çözümler aranırken 7 komşuluk operatörüne (öteleme (1-0), öteleme (2-0), değiş tokuş (1-1), değiş tokuş (2-1), değiş tokuş (2-0), M2-opt, kOr-opt) sahip bir komşuluk yapısı kullanılmıştır. Önerilen algoritma, C++ dilinde kodlanmıştır.

Tasan ve Gen (2012), EZTDARP'yi daha etkin ve etkili çözmek için permütasyona dayalı bir gösterim kullanan ve uygunluğu garanti eden genetik algoritma temelli bir yaklaşım önermiştir. Geliştirilen algoritmanın performansını değerlendirmek için literatürden çeşitli problemler karma tam sayılı problemler olarak formüle edilmiştir. Önerilen algoritmanın sonuçları ile CPLEX çözücüsünün sonuçları karşılaştırılmış ve önerilen algoritma ile daha iyi sonuçlar elde edildiği raporlanmıştır.

Wang ve Chen (2012), zaman pencereli bir EZTDARP'yi ele almıştır. Problemin *NP* yapısından dolayı, çözüm prosedürünü hızlandırmak için en ucuz ekleme (the cheapest insertion method) sezgiselinin çeşitlerine sahip bir sezgisel ile ortak-evrim genetik algoritması (co-evolution genetic algorithm) önerilmiştir. Önerilen genetik algoritma, 2 popülasyona sahiptir. Popülasyon I, çeşitlendirme amacı için kullanılırken Popülasyon II ise evrimsel yoğunlaşma için kullanılmıştır. Popülasyon I, önerilen genetik algoritmanın geniş arama yeteneğini sürdürebilmek için üç operatör kullanmaktadır: yeniden üreme, çaprazlama ve seçim. Popülasyon II ise hızlıca yüksek kaliteli çözümlere ulaşmayı ve bunları iyileştirmeyi dört operatörle amaçlamaktadır: yeniden üreme, yerel iyileştirme, çaprazlama, mutasyon ve seçim. İki tip yerel iyileştirme kullanılmıştır: tekrar ekleme iyileştirme ve değiş tokuş iyileştirme. CPLEX küçük problemleri çözebildiği için karma tam sayılı olarak modellenen problemlerden 9'unda önerilen algoritma ile bir karşılaştırma yapılabilmiştir. CPLEX, yalnızca 5 problemde çözüm bulabilmiştir; genetik algoritma ise daha kısa sürede optimal çözüm elde etmiştir.

Jun ve Kim (2012), EZTDARP için rota kurma, rota iyileştirme ve sarsım prosedürlerinden oluşan bir sezgisel algoritma önermiştir. İyi başlangıç çözümler oluşturabilmek için yeni bir süpürme algoritması geliştirilmiştir. Rota iyileştirme aşamasında değişim (*m, n*), çaprazlama operatörleri rotalar arasında; 2-opt ve Or-opt operatörleri ise rota içinde kullanılmıştır. Sarsım aşamasında yerel optimumlardan kaçınabilmek için yıkma ve tamir etme operatörleri kullanılmıştır. Önerilen algoritma, C++ dilinde programlanarak uygulanmıştır.

Goksal vd. (2013), yerel arama aşamasında değişken komşuluk inişin kullanıldığı parçacık sürü optimizasyonuna dayalı sezgisel bir çözüm önermiştir. Sürü çeşitliliğini korumak için tavlama benzeri strateji uygulanmıştır. İyi çözümler elde etmek için arama uzayında parçacık sürü optimizasyonu uygulanırken değişken komşuluk iniş, her yinelemede popülasyondan rastgele seçilen çözümleri iyileştirmek için kullanılmıştır. Arama süreci için gerekli başlangıç çözüm, en yakın komşuluk sezgiseli ile elde edilmiştir. Yerel arama sürecinde ise çaprazlama, değiş tokuş, öteleme, 2-opt, değişim, tersine çevirme prosedürleri kullanılmıştır. Önerilen algoritma, C dilinde kodlanmıştır.

Liu vd. (2013), sağlık bakımı lojistiğinde karşılaşılan bir araç çizelgeleme problemini ele almıştır. Çalışmada iki ayrı tam sayılı programlama modeli; problemin çözümü için ise genetik algoritma ve tabu aramadan oluşan hibrit bir meta sezgisel önerilmiştir. Genetik algoritma; kromozom permütasyonu, ayırma prosedürü ve yerel aramaya dayanmaktadır. Tabu arama ise hastaların rota atamalarına, arttırılmış maliyet fonksiyonuna, rota yeniden optimizasyonuna ve özelliğe dayalı tabu yıkma düzeylerine dayanmaktadır. Genetik algoritmada başlangıç çözüm, sezgisel çözümlerle elde edilmiş iyi çözümlere ait kromozomlarla ve rastgele oluşturulmuş kromozomların kombinasyonu ile elde edilmiştir. Dört kurucu sezgiselden üçü tasarruf sezgiseli temellidir ve sonuncu sezgisel ise en yakın komşu sezgiselidir. Uygun kromozomlar değişim, yer değiştirme, 2-opt değişim ve 2-opt* değişim komşuluk hareketleri ile iyileştirilmektedir. Tabu aramada elde edilen çözümler, en yakın ekleme ile tekrar optimize edilmektedir. Algoritma, C dilinde kodlanmıştır.

Hezer ve Kara (2013), EZTDARP çözümünde bakteriyel besin arama optimizasyonu algoritması kullanmıştır. Başlangıç çözüm, en yakın komşuluk sezgiseli ile elde edilmiştir. Üreme yöntemi olarak çaprazlama seçilmiştir. Çözümlerin oluşturulmasında hafıza tablosu kullanılmıştır. Algoritma, Visual Basic dilinde

kodlanmıştır. Algoritma için gerekli parametre değerleri, varyans analizi ve Duncan testi yapılarak belirlenmiştir.

Kassem ve Chen (2013), zaman pencereli bir EZTDARP'yi ele almıştır. Problem için karma tam sayılı bir programlama modeli de önerilmiştir. Problem *NP-Zor* yapıda olduğu için sezgisel bir çözüm yaklaşımı önerilmiştir. Başlangıç çözüm, ekleme temelli bir sezgisel yaklaşımla elde edilmiştir; çözümleri iyileştirmek için ise tavlama benzetimi kullanılmıştır. Komşu çözümler; müşteri değiş tokuş, kenar değiş tokuş ve ekleme operatörleri ile elde edilmiştir. Algoritma, MATLAB kullanılarak kodlanmış ve LINGO ile de problemler çözülmeye çalışılmıştır. Küçük ölçekli problemlerde her iki yaklaşım da optimal çözümleri elde ederken, problem boyutu artınca LINGO pek başarılı olamamıştır.

Yousefikhoshbakht vd. (2014), karmaşık EZTDARP'nin çözülebilmesi için değiştirilmiş tabu arama ve elit karınca sistemi meta sezgisellerini bir araya getirmiştir. Tabu listesinin uzunluğu çeşitlendirme politikası için minimum değerde, yoğunlaştırma politikası için maksimum değerde tutulmuştur. Başlangıç çözüm elde etmek için en yakın komşu ekleme sezgiseli kullanılmıştır. Önerilen yöntem, 2-opt ve değişim komşuluk yapılarına sahiptir. Her yinelemede bulunun en iyi çözümü iyileştirmek için her bir rotada gezgin satıcı problemi gibi elit karınca sistemi kullanılmıştır. Algoritma, MATLAB kullanılarak kodlanmıştır.

Avci ve Topaloglu (2015), karma ve eş zamanlı topla dağıt problemleri için bir uyarlamalı yerel arama çözüm yaklaşımı geliştirmiştir. Önerilen iki aşamalı hibrit yöntemde, tavlama benzetiminden ilham alan bir algoritma ile değişken komşuluk iniş meta sezgiseli birlikte kullanılmaktadır. İlk aşamada, rastgele bir çözüm oluşturulduktan sonra, hibrit uyarlanabilir eşik kabul yöntemi ve değişken komşuluk iniş algoritması ile çeşitlendirme ve yoğunlaşma faaliyetleri yürütülmektedir. İkinci aşamada ise ilk aşamada elde edilen çözümü iyileştirmek ve yerel optimuma hapsolmamak için sarsım mekanizması ile değişken komşuluk iniş birlikte kullanılmıştır. Komşu çözümlerin elde edilmesi için rota içinde bitişik değiş tokuş, genel değiş tokuş, tek ekleme, blok ekleme ve 2-opt prosedürleri; rotalar arasında ise öteleme, değiş tokuş prosedürleri kullanılmıştır. Algoritma, MATLAB kullanılarak kodlanmıştır.

Li vd. (2015), gerçek hayatta karşılaşılan çok depolu EZTDARP ile ilgili çok fazla çalışma olmadığını ve hatta bu problemi çözmek için meta sezgisel kullanan herhangi bir çalışma olmadığını belirtmiştir. Bu çalışmada, yinelemeli yerel arama temelli meta sezgisel söz konusu problem için geliştirilmiştir. Aramayı güçlendirmek için iyileştirme ve sarsım adımlarında uyarlanabilir komşuluk seçim mekanizmasından faydalanılmıştır. Aramayı çeşitlendirmek için yeni sarsım operatörleri önerilmiştir. Söz konusu algoritmada çoklu depo kullanımı ile müşteriye cevap verme süresi azaltılması ve hizmet düzeyinin arttırılması amaçlanmıştır. Algoritmada elde edilen çözümün yeni olup olmadığını takip etmek için uzun dönemli bir hafıza kullanılmıştır. Başlangıç çözüm, tasarruf algoritmasının değiştirilmiş haliyle elde edilmiştir. Komşuluk yapısı için değiş tokuş, öteleme, 2-opt, tersine çevirme ve Or-opt gibi operatörler kullanılmıştır. Sarsım aşamasında basit açgözlü ekleme sezgiseli, pişmanlık ekleme sezgiselleri kullanılmıştır. Algoritma, Visual Basic'te kodlanmıştır.

Polat vd. (2015), zaman limitli EZTDARP'yi ele almıştır. Bu problem için karma tam sayılı optimizasyon modeli ve klasik tasarruf sezgiseli, değişken komşuluk arama ve sarsım mekanizması ile kombine edilmiş sarsım temelli komşuluk arama algoritması önerilmiştir. Ele alınan problemde, araçlar merkez depoya son teslim süresi bitmeden dönmek zorundadır. Başlangıç çözüm elde etmek için tasarruf algoritması kullanılmıştır. Değişken komşuluk arama, başlangıç çözümü iyileştirmek için kullanılmıştır. Sarsım mekanizması ise yerel optimumdan kaçınmak için kullanılmıştır. Rota içinde 3-opt, değiş tokuş, ekleme ve 2-opt; rotalar arasında ise değişim, çaprazlama, öteleme ve yer değiştirme komşuluk yapıları kullanılmıştır. Karma tam sayılı model, IBM CPLEX çözücüsünde çözülmüştür. Meta sezgisel ise MATLAB ve C++ kullanılarak uygulanmıştır.

Wang vd. (2015), genel karma tam sayılı programlama modeli ile araçların toplam maliyeti ve araçların seyahat maliyetinden oluşan rota maliyetlerini minimize etmeyi amaçlamıştır. Bu problemi çözmek için ekleme temelli artık kapasite ve radyal fazla yük sezgiselini içeren paralel tavlama benzetimi algoritması kullanılmıştır. Problemde, kesin zaman penceresi (hard time windows) ele alınmıştır. Zaman pencereli EZTDARP, coğrafi olarak dağınık farklı müşterilere hizmet etmek için bir depoya yerleştirilmiş homojen veya homojen olmayan araçlar filosunu içerir. Başlangıç çözüm, artık kapasite ve radyal fazla yük algoritması ile oluşturulmuştur. Komşu çözümleri elde etmek için ise Or-opt,

2-opt*, k-takas ve değiş tokuş/öteleme iyileştirme yöntemleri kullanılmıştır. Algoritma, Java dilinde geliştirilmiştir.

Avci ve Topaloglu (2016), heterojen filoya sahip bir EZTDARP ele almıştır. Bu problemin çözülebilmesi için hibrit bir yerel arama algoritması geliştirilmiştir. Algoritmada, monoton olmayan eşik ayarlama stratejisi, tabu arama ile entegre edilmiştir. Kullanılan eşik fonksiyonu, uyarlamalı bir yapıya sahiptir. Eşik kabul etme algoritması, tavlama benzetiminin deterministik versiyonu olarak görülerek, yerel optimumdan kaçınmak için uyarlanabilir ve monoton olmayan farklı soğutma stratejileri kullanılmıştır. Ancak eşik kabul etme algoritması, arama deneyimi ile ilgili herhangi bir bilgi tutmadığı için kısa dönemli hafızayı tutan tabu listesinden faydalanılmıştır. Algoritmada rastgele oluşturulan başlangıç çözüm kullanılmıştır ve komşu çözümlerin elde edilmesinde bitişik değiş tokuş, genel değiş tokuş, tek ekleme ve 2-opt prosedürleri kullanılmıştır. Geliştirilen algoritmanın etkililiği, genetik algoritma ile karşılaştırılmıştır. Algoritmalar, MATLAB kullanılarak kodlanmıştır.

Kalayci ve Kaya (2016), EZTDARP için karınca koloni sistemi ve değişken komşuluk arama temelli hibrit bir meta sezgisel geliştirmiştir. Değişken komşuluk arama, yoğun bir yerel arama sağlamasına rağmen hafıza yapısına sahip olmadığı için bu eksikliğin karınca koloni sisteminin uzun dönemli hafıza yapısı kullanılarak minimize edilebileceği ve algoritmanın genel performansını arttırabileceği belirtilmiştir. Başlangıç çözüm, tasarruf algoritmasının geliştirilmiş hali olan bir formül ile oluşturulmuştur. Komşu çözümlerin elde edilebilmesi için rotalar arasında değişim, çaprazlama ve öteleme; rota içinde ise değiş tokuş, ekleme, 2-opt temelli prosedürler kullanılmıştır. Algoritma, MATLAB'ta modellenip test edilmiştir ve C++ dilinde hız nedeniyle tekrar kodlanmıştır.

Mu vd. (2016), EZTDARP için dört farklı komşuluk yapısına sahip bir tavlama benzetimi algoritması önermiştir. Algoritma için gerekli başlangıç çözüm, Dethloff (2001) tarafından önerilen ekleme tabanlı sezgiselle oluşturulmuştur. Komşu çözümleri elde edebilmek için 2-opt, Or-opt, değiş tokuş/öteleme ve 2-opt* operatörleri kullanılmıştır. Algoritma, Java dilinde kodlanmıştır.

Belgin vd. (2018), iki aşamalı bir EZTDARP'yi ele almıştır. Problemin ilk aşamasında aynı araçlarla depodan ara depolara, ikinci aşamada ise ara depolardan

müşterilere eş zamanlı olarak toplama ve dağıtma işlemleri yapılmaktadır. Problemi çözmek için ilk olarak düğüm bazlı model önerilmiştir ve literatürden üç geçerli çözüm, modeli güçlendirmek için uyarlanmıştır. İkinci olarak ise problemin zorluğu nedeniyle değişken komşuluk iniş ve yerel aramaya dayalı hibrit bir yöntem orta ve geniş ölçekli problemleri çözmek için geliştirilmiştir. Başlangıç çözüm, en yakın komşuluk sezgiseli ile elde edilmiştir. Komşu çözümler ise öteleme, değiş tokuş, değişim, 2-opt, ekleme, uydu değişimi ve uydu değiş tokuş prosedürleri ile elde edilmiştir. Tek aşamalı ve iki aşamalı dağıtım sistemlerini karşılaştırmak için bir süpermarket zincirine değişken komşuluk iniş-yerel arama uygulanarak algoritmanın gerçek dünya problemlerine uygulanabilir olduğu gösterilmiştir. Matematiksel model ve gevşetmeler, CPLEX ile çözülmüştür. Değişken komşuluk iniş-yerel arama ise C++ dilinde kodlanmıştır.

Gong vd. (2018), çok amaçlı bir EZTDARP'yi ele almıştır. Problemde, minimum yakıt tüketimi, minimum bekleme süresi ve en kısa teslimat mesafesi optimize edilmeye çalışılan amaçlardır. Söz konusu problemde, üç tip müşteri bulunmaktadır. Bunlar; dağıtıcılar, geri dönüştürücüler ve tedarikçilerdir. Çözüm yöntemi olarak iki aşamalı BEG-NSGA II algoritması kullanılmıştır. İki aşamalı optimizasyon mekanizması ile NSGA II'nin dezavantajları giderilmeye çalışılmıştır. Algoritma, MATLAB kullanılarak kodlanmıştır. Önerilen modelin geçerliliği ve uygunluğu, bir simülasyon ve üç test örneği ile doğrulanmıştır.

Majidi vd. (2018), EZTDARP'yi kullanarak kirlilik rotalama ile ilgilenmiştir. Problemde, yakıt tüketiminin ve emisyonların minimize edilmesi amaçlanmıştır. Doğrusal olmayan karma tam sayılı programlama modeli ve uyarlanabilir geniş komşuluk arama sezgiseli, yeni ekleme çıkarma operatörleri ile bu problem için önerilmiştir. Paralel yerleştirmeye dayalı kurucu sezgisel yöntemi, başlangıç çözümü bulmak için önerilmiştir. Komşu çözümlerin elde edilmesinde 9 tipte yıkma operatörü, 3 tipte onarma operatörü kullanılmıştır. Algoritma, MATLAB kullanılarak kodlanmıştır.

Hof ve Schneider (2019), standart EZTDARP'nin yanında zaman kısıtlı, zaman pencereli, dağıtım ve toplama işleminin ayrı ziyaretlerde yapılabilmesine izin veren, daha önceden çalışılmamış kısıtlı karma bölünebilir topla dağıt ve zaman pencereli bölünebilir dağıtım toplama problemlerini ele almıştır. Uyarlanabilir geniş komşuluk araması algoritması ve rota birleştirme (path relinking) yaklaşımı birlikte kullanılarak hibrit bir meta sezgisel geliştirilmiştir. Algoritma, Java dilinde uygulanmıştır.

Zhang vd. (2019), çoktan çoğa EZTDARP'yi ele almıştır. Bu problemin diğer problemlerden farkları şu şekilde belirtilmiştir: Müşterilerden toplanan ürünler diğer müşterilere dağıtılmaktadır, çoklu ürün vardır ve bunların sayısı 10 bine kadar çıkabilmektedir. Bölüm bazlı değerlendirme planı ve ileri havuz yönetme yöntemi ile uyarlamalı hafıza programlama temelli algoritma önerilmiş ve matematiksel bir model de geliştirilmiştir. Büyük boyutlu örneklerde çözüm havuzunu başlatmak için pişmanlık ekleme sezgiseli kullanılmıştır. Komşu çözümlerin elde edilmesi için silme, blok takas, 2-opt temelli operatörler kullanılmıştır. Singapur'da yer alan bir hızlı moda perakende satıcısına ait gerçek veri ve 96 yeni oluşturulmuş örnek üzerinde algoritma test edilmiştir. Geliştirilen matematiksel model, CPLEX'de çözülmüştür; algoritma ise C++'da kodlanmıştır.

Qin vd. (2019), lojistik ulaştırma maliyetlerini düşürmek ve düşük karbon ekonomisine tepki vermek için EZTDARP'yi karbon vergisi politikasını dikkate alarak çalışmayı amaçlamıştır. Çalışmada, karbon emisyonlarını maliyet olarak içeren minimum toplam maliyetli amaç fonksiyonuna sahip bir matematiksel optimizasyon modeli geliştirilmiştir. Problemi çözmek için uyarlamalı genetik tepe tırmanma algoritması tasarlanmıştır. Arama süreci için gerekli başlangıç popülasyon, rastgele oluşturulmuştur. Ebeveyn seçiminde turnuva seçimi uygulanmıştır. Genetik algoritma ile oluşturulan en iyi bireyler, komşuluk araması için tepe tırmanma işleminden geçirilmiştir. Çalışmada, tepe tırmanma işlemini gerçekleştirmek için gen yer değiştirme (transposition) operatörü kullanılmıştır. Algoritma, MATLAB kullanılarak kodlanmıştır ve literatürden edinilen kıyaslama problemleri ile karşılaştırılmıştır. Sonrasında ise, bir bira işletmesinin verileri kullanılarak bir deney yapılmıştır. Makul bir karbon vergisi aralığının karbon emisyonlarını etkili bir şekilde azaltabileceği bulunmuştur, ancak çevresel faydalar elde etmek için ekstra ekonomik maliyetler ödenmesi gerektiği de tespit edilmiştir.

Aydoğdu ve Özyörük (2020), dinamik yapıda olan bir EZTDARP'yi ele almıştır. Söz konusu problem için matematiksel bir model geliştirilmiş ve literatürdeki kıyaslama problemleri ile karşılaştırma yapılarak modelin etkinliği değerlendirilmiştir. Söz konusu modelin problem boyutu büyüdükçe çözüm üretmede üstel şekilde daha çok zamana ihtiyaç duyduğu belirtilerek rassal iteratif yerel arama değişken komşu iniş isminde bir algoritma geliştirilmiştir. Başlangıç çözüm, en yakın komşuluk sezgiseli ile elde edilmiştir. Komşu çözümlerin elde edilmesi için gerekli prosedürler, permütasyon seçim

algoritması ile belirlenmiştir ve komşulukların sırası, rastgele şekilde değiştirilmiştir. Komşu çözüm elde etmek için rota içinde Or-opt, 2-opt, değiştirme, ters çevirme ve ekleme prosedürleri; rotalar arasında ise λ-yer değiştirme, k-kaydırma, çaprazlama prosedürleri kullanılmıştır. Uygun olmayan rotaların tamiri için ise tersine çevirme ve araç kapasitesini aşmayan en yakın müşteriden başlayarak rotaya ekleme prensiplerine sahip sezgisel kullanılmıştır. Matematiksel model, CPLEX kullanılarak çözülmüştür.

Olgun vd. (2021), müşterilerin dağıtım ve toplama taleplerini eş zamanlı olarak karşılarken yakıt tüketimini minimize etmeyi amaçlayan yeşil EZTDARP'yi ele almıştır. Söz konusu problem; yinelemeli yerel arama meta sezgiselini üst seviye strateji, değişken komşuluk sezgiselini ise düşük seviye strateji olarak kullanan bir hiper sezgisel (hyper heuristic) ile çözülmüştür. Algoritma için gerekli başlangıç çözüm, en yakın komşuluk sezgiseli ile elde edilmiştir. Yerel arama aşamasında 2-opt, değişim ve ekleme operatörleri rota içi komşuluk yapısı olarak; değiş tokuş (1-1), öteleme (1-0), değiş tokuş (2-1), öteleme (2-0) ve çaprazlama operatörleri ise rotalar arası komşuluk yapısı olarak kullanılmıştır. Söz konusu operatörler, seçim fonksiyonu ile sergilenen geçmiş performansa göre seçilmiştir. Algoritma, C++ kullanılarak kodlanmıştır. Geliştirilen matematiksel model ise CPLEX ile çözülmüştür.

Tezin bu bölümünde şimdiye kadar bahsedilen çalışmalar, yalnızca taşınan yükün miktarını dikkate almaktadır ve tek boyutludur. Literatürde müşterilerin toplama ve dağıtım taleplerini eş zamanlı olarak optimal şekilde karşılayabilmek için taşınan yükün fiziksel boyutlarını (en, boy, derinlik) hesaba katan çalışmalar da mevcuttur. Zachariadis vd. (2016; 2017) 2 boyutlu yükleme kısıtlarına sahip EZTDARP'yi ele alırken; Koch vd. (2018) ise 3 boyutlu yükleme kısıtlarına sahip zaman pencereli bir EZTDARP'yi ele almıştır. Bu tezin kapsamı, tek boyutlu problemlerden oluştuğu için yükleme kısıtlı problemlere genel olarak değinilmiştir ve tezin bundan sonraki kısmı tek boyutlu problemler ile ilgili bilgiler içermektedir.

Literatür taramasına ilişkin sonuçları özetlemek gerekirse, ele alınan problemi çözmek için kullanılan meta sezgiseller incelendiğinde 36 çalışmanın 21'inde hibrit meta sezgisellerin kullanıldığı görülmüştür. Bu durum, bilgi birikiminin artmasıyla birlikte rekabetçi çözüm yöntemleri geliştirebilmenin, yani bazı konularda iyi olan bir yöntem kullanmak yerine birden fazla konuda iyi olan farklı yöntemleri bir araya getirerek bir sinerji oluşturma zorunluluğunun bir sonucudur. Kullanılan hibrit yöntemler

incelendiğinde değişken komşuluk arama, değişken komşuluk iniş, tabu arama, genetik algoritma yöntemlerinin öne çıktığı görülmektedir. Söz konusu yöntemlere bakıldığında öne çıkan yöntemlerden sadece genetik algoritmanın popülasyona dayalı meta sezgisel olduğu, diğer yöntemlerin ise yerel aramaya dayalı meta sezgiseller olduğu tespit edilmiştir. Bu bulguya paralel olarak incelenen çalışmalardan 31 tanesi teorik, kalan 5 çalışma ise uygulamalı çalışmalardır. Teorik çalışmalarda geliştirilen algoritmaları test etmek için literatürden edinilen kıyaslama problemlerine ait veriler kullanılırken; uygulamalı çalışmalarda ise gerçek dünya verisi kullanılmıştır. Uygulanan alanlar ise deniz taşımacılığı, evde sağlık hizmetleri, geri dönüşüm, hızlı moda, süpermarket zincirleri ve yeniden imalat olarak tespit edilmiştir. Taranan çalışmalarla ilgili özet bilgi Tablo 3'te verilmiştir.

Meta sezgisel yöntemler kullanılarak problemlere ait arama uzaylarında optimal çözüm aranırken önemli hususlardan biri, aramanın başladığı başlangıç çözümdür. Bu nedenle aramaya çözüm kalitesi yüksek başlangıç çözümlerden başlamak, optimal çözümlere daha kısa sürede ulaşmayı sağlayacaktır. Taranan çalışmalarda, birden fazla yöntem kullanılabildiği bilgisi göz önünde bulundurularak hangi yöntemlerle başlangıç çözümlerin oluşturulduğu incelendiğinde rastgele başlangıç çözümün, en yakın komşuluk sezgiselinin ve tasarruf sezgiselinin öne çıktığı görülmektedir. Rastgele başlangıç çözümü kullanan meta sezgisellerin, genel olarak popülasyon temelli meta sezgiseller olduğunu söylemek mümkündür. Bu yöntemlerin çok sayıda bireysel çözüm içerdiği düşünüldüğünde sonuçların makul olduğu söylenebilir.

Meta sezgisel yöntemlerin optimal çözüme ulaşma süresini etkileyen bir husus da algoritmaların hangi programlama dillerinin kullanılarak uygulandığıdır. İncelenen çalışmalara bakıldığında C, C++, C# gibi diller ve özellikle de son yıllarda MATLAB'ın öne çıktığı tespit edilmiştir. Geliştirilen matematiksel modellerin çözümünde CPLEX'in sıklıkla kullanıldığını da belirtmek gerekmektedir.

Tablo 3. Literatür Taramasına İlişkin Özet Tablo

Çalışma	Hibrit mi?	Kullanılan Meta Sezgisel(ler)	Başlangıç Çözüm Nasıl Elde Edildi?
Olgun vd. (2021)	Evet	Yinelemeli Yerel Arama, Değişken Komşuluk İniş	En Yakın Komşuluk Sezgiseli
Aydoğdu ve Özyörük (2020)	Evet	Yinelemeli Yerel Arama, Değişken Komşuluk İniş	En Yakın Komşuluk Sezgiseli
Qin vd. (2019)	Evet	Genetik Algoritma, Tepe Tırmanma	Rastgele
Zhang vd. (2019)	Evet	Uyarlamalı Hafıza Programlama, Değişken Komşuluk Arama	Pişmanlık Ekleme Sezgiseli
Hof ve Schneider (2019)	Evet	Uyarlanabilir Geniş Komşuluk Arama, Yol Yeniden Bağlama	Tasarruf Sezgiseli
Majidi vd. (2018)	Evet	Uyarlanabilir Geniş Komşuluk Arama, Tavlama Benzetimi	Paralel Ekleme Sezgiseli
Gong vd. (2018)	Evet	Arı Evrimsel Algoritması Kılavuzlu NSGA-II	Rastgele
Belgin vd. (2018)	Evet	Değişken Komşuluk İniş, Yerel Arama	En Yakın Komşuluk Sezgiseli
Kalayci ve Kaya (2016)	Evet	Karınca Koloni Sistemi, Değişken Komşuluk Arama	Tasarruf Sezgiseli
Avci ve Topaloglu (2016)	Evet	Eşik Kabul, Tabu Arama	Rastgele
Mu vd. (2016)	Hayır	Paralel Tavlama Benzetimi	Artık Kapasite ve Radyal fazla Yük Ekleme Sezgiseli
Wang vd. (2015)	Hayır	Paralel Tavlama Benzetimi, Genetik Algoritma	Artık Kapasite ve Radyal fazla Yük Ekleme Sezgiseli
Polat vd. (2015)	Evet	Değişken Komşuluk Arama, Değişken Komşuluk İniş	Tasarruf Sezgiseli
Li vd. (2015)	Hayır	Yinelemeli Yerel Arama	Tasarruf Sezgiseli
Avci ve Topaloglu (2015)	Evet	Eşik Kabul, Değişken Komşuluk Arama	Rastgele
Yousefikhoshbakht vd. (2014)	Evet	Tabu Arama, Elit Karınca Sistemi	En Yakın Komşuluk Ekleme Sezgiseli
Hezer ve Kara (2013)	Hayır	Bakteriyel Besin Arama Optimizasyonu	En Yakın Komşuluk
Liu vd. (2013)	Hayır	Tabu Arama, Genetik Algoritma	Tasarruf Sezgiseli, En Yakın Komşu Sezgiseli, Rastgele
Goksal vd. (2013)	Evet	Parçacık Sürü Optimizasyonu, Değişken Komşuluk İniş	En Yakın Komşuluk Sezgiseli, Rastgele
Kassem ve Chen (2013)	Hayır	Tavlama Benzetimi	Ekleme Sezgiseli

Wang ve Chen (2012)	Hayır	Genetik Algoritma	En Ucuz Ekleme Sezgiseli
Tasan ve Gen (2012)	Hayır	Genetik Algoritma	Rastgele
Jun ve Kim (2012)	Hayır	Yinelemeli Yerel Arama	Süpürme Sezgiseli
Souza vd. (2011)	Evet	Yinelemeli Yerel Arama, Değişken Komşuluk İniş	Ekleme Sezgiseli
Subramanian vd. (2010)	Evet	Rastgele Sıralamalı Değişken Komşuluk İniş, Yinelemeli Yerel Arama	Açgözlü Ekleme Sezgiseli
Mingyong ve Erbao (2010)	Hayır	Geliştirilmiş Diferansiyel Gelişim Algoritması, Genetik Algoritma	Rastgele
Çatay (2010)	Hayır	Karınca Koloni Algoritması	En Yakın Komşuluk Sezgiseli
Zachariadis vd. (2010)	Evet	Uyarlamalı Hafıza, Tabu Arama	Tasarruf Sezgiseli
Zachariadis vd. (2009)	Evet	Tabu Arama, Yönlendirilmiş Yerel Arama	Tasarruf Sezgiseli
Karlaftis vd. (2009)	Evet	Genetik Algoritma	Rastgele
Gajpal ve Abad (2009)	Hayır	Karınca Koloni Sistemi	En Yakın Komşuluk Sezgiseli
Ai ve Kachitvichyanukul (2009)	Hayır	Parçacık Sürü Optimizasyonu	Rastgele
Wassan vd. (2008)	Hayır	Reaktif Tabu Arama	Süpürme Sezgiseli
Chen ve Wu (2006)	Evet	Kayıttan Kayıta Seyahat, Tabu Arama	Ekleme Sezgiseli
Montané ve Galvão (2006)	Hayır	Tabu Arama	Gruplama, Rotalama
Ropke ve Pisinger (2006)	Evet	Geniş Komşuluk Arama, Tavlama Benzetimi	Ekleme, Silme

İKİNCİ BÖLÜM
KARMAŞIKLIK KURAMI VE ARAÇ ROTALAMA PROBLEMLERİNİN ÇÖZÜMÜNDE KULLANILAN META SEZGİSELLER

Gündelik hayatta sıklıkla kullanılan problem kelimesi, Fransızca (problème) kökenli olup *"teoremler veya kurallar yardımıyla çözülmesi istenen soru, mesele"* anlamına gelmektedir (Türk Dil Kurumu, 2006) ve bir "*soru kümesini"* ifade etmektedir (Say, 2018: 36). Karşılaşılan problemin zorluk seviyesi, probleme çözüm bulma biçimini belirlemektedir. Basit seviyedeki problemler; geçmiş tecrübelerden faydalanma, akıl yürütme gibi bilişsel süreçlerle kısa zaman içerisinde kolaylıkla çözülebilir. Zor seviyedeki problemler ise daha teknik bilgi kullanımını gerektirir ve tanımda da belirtildiği gibi çeşitli teoremler veya kurallar, çözümün bulunmasına yardım eder. Basit problemleri çözmek için insan zekâsı yeterli olurken daha zor problemlerde bilgisayarlardan faydalanılır. Problemlerin zorluk derecesinin belirlenmesinde karmaşıklık kuramından (complexity theory) faydalanılmaktadır.

2.1. Karmaşıklık Kuramı

Bilimsel anlamda problemlerin zor olup olmadıklarının belirlenmesinde öznel yerine nesnel ölçütlerin kullanılması, daha doğru bir yaklaşım olacaktır. Bu konuda Chopard ve Tomassini (2018), bir problemi çözebilmek için ihtiyaç duyulan hesaba dayalı (computational) kaynak miktarının, problemin zorluğu olduğunu ve hesaplama aracının, geleneksel bir bilgisayar olduğu varsayıldığında ihtiyaç duyulan zamanın ve bellek alanının, ilgili kaynaklar olduğunu ifade etmiştir (Chopard ve Tomassini, 2018: 1).

Bir karar problemi, olası durumların bir alt kümesinin belirtilmesi ile ilgilenerek verilen durumun söz konusu kümede olup olmadığını tespit etmek için kullanılmaktadır (Goldreich, 2010: 6). Karmaşıklık kuramı, cevabı evet ya da hayır olan karar problemleri ile ilgilenmektedir (Talbi, 2009: 11). Bu problemin çözümü için programların (en kötü ihtimalle) harcayacağı zaman ve bellek miktarı, karmaşıklık kuramının temel konusunu oluşturmaktadır (Nilsson, 2010: 408). Karar problemlerinin karmaşıklığı belirlenirken çözümdeki en kötü durum (worst-case) davranışının kullanılmasının nedeni; problemlerin yapısı, bilgisayarlar arasındaki farklılıklar gibi hususlar sebebiyle çözüm sürelerinin değişkenlik göstermesidir (Papadimitriou ve Steiglitz, 1998: 158).

Hesaplanabilir problemler, iki ayrık olmayan sınıftan birine aittir: (1) Hesaplama zamanının, örnek problemin büyüklüğüne göre bir polinom tarafından sınırlandırıldığı problem sınıfı: *P*- polinom, (2) Doğru cevabın böyle bir zamanda kontrol edilebildiği problem sınıfı: *NP*- deterministik olmayan polinom (Chopard ve Tomassini, 2018: 2). Verilen bir örnek için bir çözümün doğru olup olmadığını test etmek kolaysa, o zaman verilen örneğe de çözüm bulmak kolay mıdır sorusu söz konusu olduğunda karmaşıklık kuramı alanında sıklıkla ele alınan $P = NP$ midir sorusu ortaya çıkmaktadır (Goldreich, 2002: 4). Bu sorunun yanıtı henüz bulunamamıştır; ancak Clay Mathematics Institute[1] tarafından çözümü bulana 1 milyon dolar ödül vaat edilmiştir (Jaffe, 2006: 657).

NP sınıfındaki problemler, polinom zaman sınırlı bir fonksiyon tarafından yine bu sınıftan bir probleme dönüştürülebiliyorsa, dönüştürülen bu probleme *NP-Tam* (*NP-Complete*) denir (Homer ve Selman, 2011: 129). $P = NP$ olması halinde *NP-Tam* sınıfında yer alan optimizasyon problemleri kolayca ve etkin bir şekilde çözülebilecek demektir (Fortnow, 2009: 80). Bu iyimserliğe karşın, bilim insanları henüz ispatlanmasa da $P \neq NP$ olduğuna inanmaktadır ve $P \subseteq NP$ ilişkisi söz konusudur (Moore ve Mertens, 2011: 96).

NP sınıfında olmak zorunda olmayan ama en az bu sınıftaki problemler kadar zor olan problemler kümesi ise *NP-Zor* (*NP-Hard*) problemler olarak isimlendirilmektedir (Homer ve Selman, 2011: 147). Çoğu *NP-Zor* optimizasyon problemi, karar problemi olmadığı için *NP-Tam* sınıfına girmeseler de bu problemlerin çözümleri tanımda belirtildiği gibi *NP-Tam* sınıfındaki problemler kadar zordur (Chopard ve Tomassini, 2018: 8).

Bir algoritmanın etkinliği, *T* fonksiyonu ile ölçülür ve $T(n)$, *n* uzunluğunda bir girdi verildiğinde algoritmanın yaptığı maksimum temel işlem sayısına eşittir (Arora ve Barak, 2009: 3). Algoritmanın, girdisinin belirli bir değere ulaştığında ya da sonsuza yaklaştığında asimptotik davranışı ise büyük-*O* (Big-Oh) gösterimi ile incelenir. Bu gösterim, fonksiyonun büyüme hızını gösterir ve aynı büyüme sınıfına sahip fonksiyonlar aynı *O* gösterimi ile ifade edilir. $T(n) = O(g(n))$ eşitliği, c gibi bir sabit mevcut ise $T(n) \leq c.g(n)$ anlamına gelir ve $g(n)$, *T(n)* için bir üst sınır ifade eder (Devi vd., 2011: 844). Bu gösterim, tam olarak hesaplama süresini vermese de algoritmaların birbirleri ile

[1] Detaylı bilgi için: https://www.claymath.org/millennium-problems/p-vs-np-problem

karşılaştırılmasını mümkün kılar. Algoritmalar karşılaştırılırken karşılaşılabilecek *O* gösterimleri ve bunların isimleri, Tablo 4'te gösterildiği gibidir.

Tablo 4. Karmaşıklık Düzeyleri ve Bunların İsimlendirmeleri

***O* Gösterimi**	**İsim**	**Değerler (1)**	**Değerler (2)**
$O(1)$	Sabit	1	1
$O(log_b n)$	Logaritmik (Herhangi b için)	1	1,3010
$O(n)$	Doğrusal	10	20
$O(nlog_b n)$	*"en log en"*	10	13,010
$O(n^2)$	Kuadratik	100	400
$O(n^3)$	Kübik	1000	8000
$O(c^n)$	Üstel (Herhangi c için)	1024	1048576
$O(n!)$	Faktöriyel	$3{,}629 \times 10^6$	$2{,}433 \times 10^{18}$

Kaynak: Downey (2012: 21)'den faydalanılarak hazırlanmıştır.
Değerler (1) sütununda n = 10, b = 10, c = 2 olarak alınmıştır. Değerler (2) sütununda ise n = 20 iken diğer değerler aynıdır.

Tablo 4 incelendiğinde, iki tür karşılaştırma yapılabilir. İlk olarak aynı sütunda hareket edilebilir. Örneğin, Değerler (1) sütununda dikey biçimde hareket edilecek olursa tablonun girdi sayısının (10) aynı olmasına rağmen $O(1)$'den $O(n!)$'e doğru hesaplama için gerekli zaman miktarının ciddi şekilde arttığı görülmektedir. İkinci olarak ise belirli bir karmaşıklık düzeyinde girdi sayısının etkisi karşılaştırılabilir. Buna örnek olarak ise girdi sayısı 2 kat arttırılarak 10'dan 20'ye çıkarıldığında logaritmik karmaşıklıkta hesaplama için gerekli zaman yaklaşık olarak 1,3 kat artarken, kuadratik karmaşıklıkta 4 kat, üstel karmaşıklıkta ise 1024 kat artmaktadır. Bu durum da bazı problemlerin girdi miktarındaki çok küçük bir artışın hesaplamalar için gerekli zaman miktarını ciddi şekilde arttırması nedeniyle "zor" olarak nitelendirilmesinin güzel bir örneğidir. Zor bir problem söz konusu olduğunda çözüm için uzun süre beklemek yerine çok daha kısa sürede çözümü yaklaşık olarak elde etmek daha mantıklı bir seçim olabilir. Meta sezgiseller, yaklaşık çözüm değerlerini elde edebilen (optimal ya da bilinen en iyi çözüm değerlerini tam olarak da elde edebilen) yöntemlerdir ve tezin ilerleyen kısımlarında bunlardan bahsedilecektir.

2.2. Optimizasyon Problemleri ve Bunların Çözüm Yöntemleri

Kaynaklar ve ihtiyaçlar, insanların hayatlarını sürdürdükleri ekonomik çevrenin temel iki ögesidir. Kaynaklar doğası gereği sınırlı iken, ihtiyaçlar ise sınırsızdır. Bu nedenle kıt kaynakların sınırsız olan ihtiyaçları karşılayabilmesi için rasyonel kararlar alınması gerekmektedir. Karar alınırken kullanılan süreçler aslında optimizasyon işlemidir. Optimizasyon için literatürde yapılmış tanımlardan bazıları şu şekildedir:

- *"Optimizasyon, verilen bir durumda en iyi çözümü arayan bir tekniktir"* (Datta vd., 2019: 1).

- *"Optimizasyon, fiziksel gerçekliğin modelleri olan bazı matematiksel olarak tanımlanmış problemlere 'en iyi' çözümü belirleme bilimi olabilir"* (Fletcher, 2000: 3).

- *"Optimizasyon modelleri (matematiksel programlar olarak da isimlendirilir) problem seçeneklerini karar değişkenleri olarak ifade eder. Olası karar seçenekleri üzerinde değişken değerlerinin limitlerini ifade eden kısıtlar altında karar değişkenlerinin amaç fonksiyonunu minimize ya da maksimize eden değerlerini arar"* (Rardin, 2017: 4).

Yukarıda verilen optimizasyon tanımları bütünleştirilecek olursa optimizasyon, belirli bir gerçekliğe dayanan bir matematiksel modelin amaçlar doğrultusunda en iyi çözümü -fayda sağlayan hususlar için en büyük değer; maliyete yol açan hususlar için ise en küçük değer- verecek şekilde çözülmesidir. Söz konusu model, alınacak kararları ifade eden karar değişkenlerinden, bu kararlar üzerinde doğrudan sınırlayıcı bir özelliğe sahip -kaynakları ifade eden- kısıtlardan ve alınan kararların sonuçlarını ifade eden amaç fonksiyonundan oluşmaktadır. Göreceli olarak yakın tarihte yapılan bu tanımlara karşın optimizasyon problemlerinin kökenleri, Öklid'in bir doğru ile nokta arasındaki minimum uzaklığı değerlendirdiği M.Ö. 300 yılına kadar gitmektedir. Fermat, Newton, Leibniz, Euler, Lagrange ve Cauchy gibi ünlü bilim insanları, optimizasyon alanına katkılarda bulunan diğer önemli isimlerdendir (Arora, 2015: 1). Bilgisayar alanındaki gelişmeler sayesinde 1940'lı ve 1950'li yıllarda optimizasyon alanında önemli gelişmeler yaşanmıştır (Fletcher, 2000: 3). Dantzig tarafından geliştirilen simpleks yöntemi, bir optimizasyon modeli olan doğrusal programlama modellerinin geniş kesimlerce

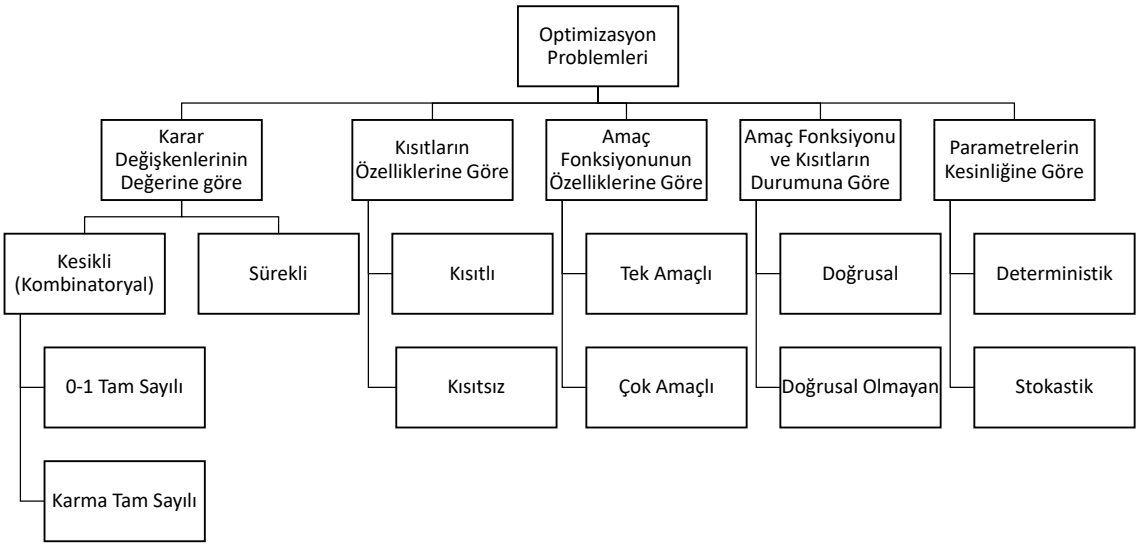
çözülebilmesini sağlamıştır (Nash, 2000: 31). Bahsedilen optimizasyon modellerinin genel bir matematiksel gösterimi, Eşitlik (2.1)'de verildiği gibidir.

$$\begin{aligned} &\min(maks)\, f(x_i) \;\; (i = 1, \dots, n) \\ &h_j(x_i) \leq b_j \;\; (j = 1, \dots, m) \\ &x_i > 0 \end{aligned} \tag{2.1}$$

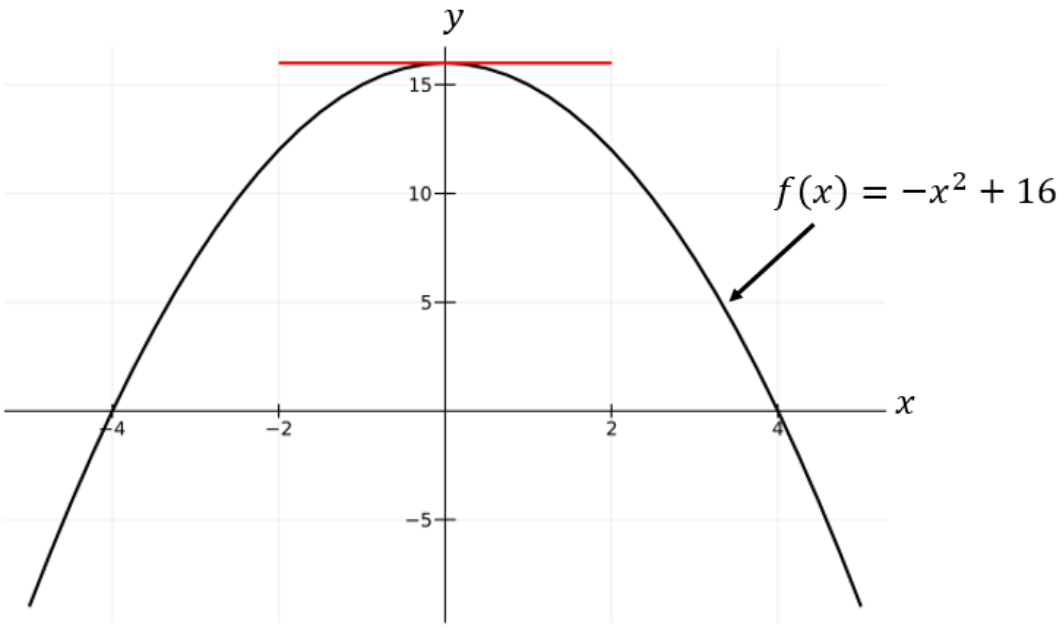
Optimizasyon modellerinin sahip olduğu unsurlara ait özelliklerin her biri, farklı türde problem ailelerini ifade etmektedir. Sınıflandırmaya karar değişkenlerinin özelliklerinden başlanacak olursa, karar değişkenleri herhangi bir reel değer alıyorsa sürekli optimizasyon problemleri; tam sayı değeri alıyorsa kesikli optimizasyon problemleri olarak iki ayrı sınıf elde edilir. Kesikli optimizasyon problemleri, bazı kaynaklarda kombinatoryal optimizasyon problemleri olarak da isimlendirilmektedir. Bununla birlikte karar değişkenleri, yalnızca 0 veya 1 değerini alabiliyorsa 0-1 tam sayılı optimizasyon problemleri; bazı değişkenler sadece kesikli değerler, bazı değişkenler ise sürekli değerler alabiliyorsa bu problemlere karma tam sayılı optimizasyon problemleri denilmektedir. Kısıtların özelliklerine göre ise problemde herhangi bir kısıt yoksa kısıtsız optimizasyon problemi, bir ve daha fazla kısıt var ise kısıtlı optimizasyon problemi şeklinde sınıflandırma yapılır. Amaç fonksiyonu açısından ise eğer tek bir amaç fonksiyonu varsa tek amaçlı optimizasyon problemleri, eş zamanlı olarak karşılanması gereken birden fazla amaç fonksiyonu varsa çok amaçlı optimizasyon problemleri söz konusu olmaktadır. Amaç fonksiyonu ve kısıtların durumuna göre ise doğrusal ve doğrusal olmayan optimizasyon problemleri biçiminde sınıflandırma yapmak mümkündür. Modeldeki parametreler kesin olarak biliniyorsa deterministik optimizasyon problemi, belirli bir olasılık dağılımına uyduğu biliniyorsa stokastik optimizasyon problemi söz konusudur. Anlatılan bu sınıflandırmalar, Şekil 3'te görsel olarak verilmiştir.

Optimizasyon problemlerinde, amaç fonksiyonun değerinin maksimum ya da minimum olması istenir. Basit fonksiyonların maksimum ya da minimum değerleri bulunurken matematiğin önemli konularından türevden faydalanılır. Bir fonksiyonun türevi, aslında değişim miktarını ifade eder. Eğer bir fonksiyonun türevi $f'(x) = 0$ ise bu nokta, kritik nokta olarak isimlendirilir. Fonksiyon, $[a, b]$ gibi bir aralıkta tanımlandıysa ve fonksiyon sürekli ise fonksiyonun maksimum ya da minimum değeri, aralığın uç noktalarında veya türevin sıfır olduğu kritik noktada bulunabilir. Kritik nokta hakkında

karar vermek için ikinci türevden faydalanılır ve $f''(x) > 0$ ise minimum, $f''(x) < 0$ ise maksimum değerden bahsedilir (Yang, 2018: 31; Lange, 2013: 3). Örneğin, Şekil 4'te basit bir fonksiyon olan $f(x) = -x^2 + 16$ fonksiyonunun grafiği verilmiştir. Bu fonksiyonun birinci türevi $f'(x) = -2x$'tir ve bu türevin sıfıra eşit olduğu 0 noktasında çizilmiş teğet, şekilde gösterilmiştir. Kritik nokta hakkında karar verebilmek için fonksiyonun ikinci türevine bakıldığında $f''(x) = -2$ olduğundan, 0 noktasının bu fonksiyon için maksimum değere sahip olan nokta olduğu belirlenmiştir.

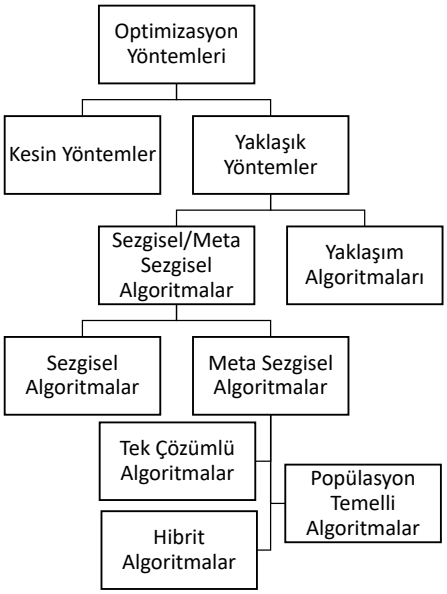


Şekil 3. Optimizasyon Problemlerinin Sınıflandırılması



Şekil 4. Türev Yardımıyla Fonksiyonun Maksimum Değerinin Belirlenmesi

Birden fazla değişkenin söz konusu olduğu kısıtsız optimizasyon problemlerinde ise kısmi türevleri de kullanan Hessian matristen faydalanılmaktadır. Bu matris, pozitif belirli ise kritik noktalar, yerel minimumdur. Matris, negatif belirli ise kritik noktalar, bu durumda maksimum olmaktadır (Yang, 2018: 33). Bu iki durum da söz konusu değilse herhangi optimallik durumu yoktur (Sezen, 2007: 395). Benzer şekilde problemde kısıtlar söz konusu ise kısıtların eşitlik biçiminde olduğu durumlarda Lagrange çarpanları, eşitsizlik biçiminde olduğu durumlarda ise Kuhn-Tucker koşulları gibi yöntemler kullanılmaktadır (Winston, 2004: 702-703). Belirtilen problem türleri ve bunların çözüm yöntemleri incelendiğinde, her optimizasyon problem türünün kendine özgü bir çözüm yöntemi olduğu görülmektedir. Bu çözüm yöntemlerinin hepsini incelemek mümkün olmadığı için Şekil 5'te optimizasyon yöntemleri genel başlıklar halinde verilmiştir.



Şekil 5. Optimizasyon Yöntemlerinin Sınıflandırılması
Kaynak: Makhadmeh vd. (2019: 9)

2.3. Meta Sezgisel Kavramı ve Temel Özellikleri

Meta sezgisel terimi, literatüre F. Glover tarafından kazandırılmıştır. Meta sezgisel arama yöntemleri, belirli optimizasyon problemlerini çözmek için temel sezgisellerin tasarlanmasında yol gösterici stratejiler olarak kullanılabilecek üst düzey genel metodolojiler (şablonlar) olarak tanımlanabilir (Talbi, 2009: 1). Yapılan bu tanımdan meta sezgisellerin probleme özgü olmasından ziyade genel karakteristiklere sahip oldukları ve problemin çözülebilmesi için daha alt düzey sezgiselleri (hatta aynı düzeyde meta sezgiselleri) yönlendiren yaklaşımlar olduğu söylenebilir. Buradan da

sezgisellerin probleme özgü oldukları buna karşın, meta sezgisellerin daha genel bir yapıya sahip oldukları sonucu çıkarılmaktadır. Bu çıkarımı destekleyen başka bir tanım da Sörensen (2015: 6) tarafından yapılmıştır:

> "…bununla birlikte, genel anlamda, bir meta-sezgisel bir algoritma değildir, yani bir yemek tarifi gibi takip edilmesi gereken bir eylemler dizisi değildir. Aksine, sezgisel optimizasyon algoritmalarını tasarlamak için kullanılabilecek tutarlı bir fikir, kavram ve operatör kümesidir. Benzetme üzerinden devam edilecek olursa genel anlamda bir meta sezgisel -tabu arama, genetik algoritmalar ve tavlama benzetimi vb. meta sezgisellerden bahsettiğimizde kullandığımız bu anlamdır- 'Fransız', 'Çin' ya da 'Cajun' gibi pişirme stilidir ve 'spaghetti carbonara `a la Antonio Carluccio' gibi bir yemek tarifi değildir".

Söz konusu tanımda meta sezgisellerin belirli kalıplara sahip algoritmalardan ziyade başlı başına özgün bir tarz olduğu belirtilmektedir. Bu tanım, meta sezgisellerin ilham aldıkları çıkış noktaları ve çözümü arama şekilleri düşünüldüğünde doğrudur, ancak belirli bir problemi çözebilmek için gerekli kuralları, işlemler kümesini adım adım uygulamak gerektiğinden bu tezde, meta sezgisellerin algoritma yapısına sahip olduğu kabul edilmiştir.

Bir meta sezgisel, çeşitli zor optimizasyon problemlerini yaklaşık olarak çözmek için tasarlanan ve her bir problem özelinde detaylı uyarlama gerektirmeyen bir algoritmadır (Boussaïd vd., 2013: 82). Bu nedenle meta sezgiseller, söz konusu problemi çözecek kesin yöntemlerin bulunmadığı ya da bu yöntemlerin yetersiz kaldığı durumlarda kullanılan ve optimale yakın sonuçlar veren tekniklerdir. Problemi çözebilecek kesin yöntemlerin mevcut olması halinde meta sezgisellerin kullanılması ise mantıklı bir seçim değildir.

Meta sezgisellerin problem özelinde uyarlama yapılmasına ihtiyaç duymamasına karşın karmaşık kombinatoryal problemleri çözmek için geliştirilen özel sezgiseller, her problem için genelleştirmeye ihtiyaç duymuştur ve farklı tür problemler için genelleştirme yapmak her zaman için mümkün olmamıştır (Ólafsson, 2006: 635). Meta sezgisellerin temel özellikleri, aşağıda gösterilen maddelerdeki gibidir (Blum ve Roli, 2003: 270-271):

- "Meta sezgiseller, arama sürecini 'yönlendiren' stratejilerdir.
- Amaç, arama uzayını etkin biçimde keşfederek optimal (ya da optimale yakın) çözümler bulmaktır.

- Meta sezgiselleri oluşturan alt bileşenler, basit yerel arama tekniklerinden karmaşık öğrenme süreçlerine kadar değişiklik göstermektedir.
- Meta sezgisel algoritmalar, yaklaşık çözüm verir ve genelde deterministik değildir.
- Arama uzayının sınırlı alanlarında sıkışıp kalmamak için mekanizmalar içerebilir.
- Meta sezgisellerin temel kavramları, soyut düzeyde bir tanımlamaya izin verir.
- Meta sezgiseller, probleme özgü değildir.
- Meta sezgiseller, alana özgü bilgiyi, üst düzey strateji ile kontrol edilen sezgiseller biçiminde kullanabilir.
- Günümüzde daha gelişmiş meta sezgiseller, aramayı yönlendirmek için arama deneyimini (bir çeşit hafıza biçiminde bulunur) kullanabilir."

2.4. Meta Sezgisel Yöntemlerin Sınıflandırılması

Meta sezgisel yöntemlerin sınıflandırılması gerektiğinde kriter olarak kullanılabilecek birçok yaklaşım bulunmaktadır. Bu yaklaşımlar, ilham kaynağı, hafıza kullanımı, kararların yapısı, çözüm arama şekli biçiminde olabilmektedir. Genel olarak sınıflandırma kriterleri ve bunların esas aldıkları prensipler şu şekildedir (Talbi, 2009: 25):

- ***Doğadan ilham alma–Doğadan ilham almama:*** Canlıların davranışlarından ya da bunların yaşam döngülerindeki doğal olaylardan ilham alınıp alınmadığı ile ilgilidir.

- ***Hafıza kullanımı-Hafızasız yöntemler:*** Çözümü arama sürecinde ortaya çıkan bilgiden daha sonra faydalanılıp faydalanılmadığı ile ilgilidir.

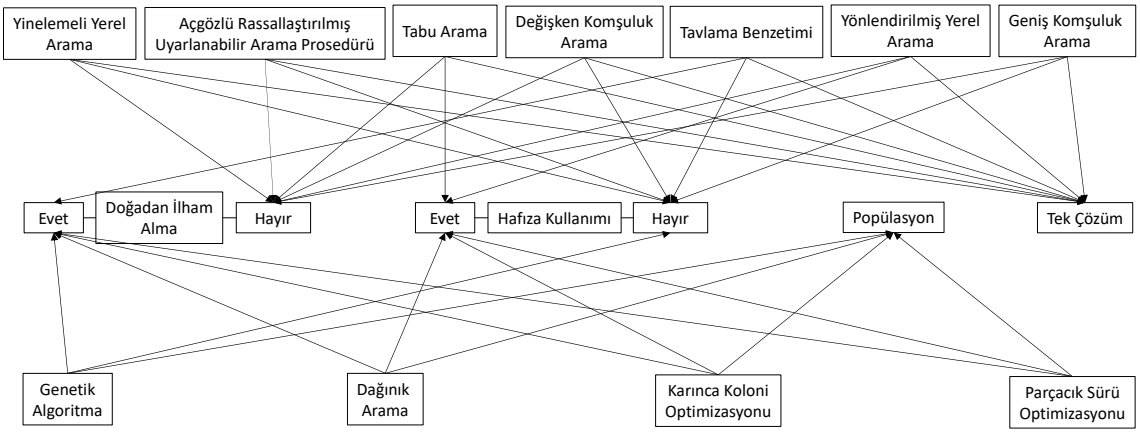
- ***Deterministik-Stokastik:*** Çözüm arama sürecinde kararların rastgele verilip verilmemesi ile ilgilidir.

- ***Popülasyon temelli arama-Tek çözüm temelli arama:*** Nihai çözüm aranırken tek bir çözümün ya da birden fazla çözümün bir araya gelerek oluşturduğu popülasyonun kullanılması ile ilgilidir.

- ***Yinelemeli-Açgözlü:*** Aramanın mevcut bir çözüm üzerinde değişiklikler yapılarak (yinelemeli) ya da bir çözümün sıfırdan oluşturularak (açgözlü) yapılması ile ilgilidir.

2.5. Araç Rotalama Problemlerinde Kullanılan Meta Sezgiseller

Tezin bu kısmında, araç rotalama problemlerinin çözümünde öne çıkan meta sezgiseller ele alınmıştır. Bölüm 2.4'te yapılan sınıflandırma ışığında meta sezgisellerin genel özellikleri ve çalışma şekillerini gösteren sözde-kodları (pseudo-code) verilmiştir. Şekil 6'da bu meta sezgisellere ait özellikler verilmiştir.



Şekil 6. Araç Rotalama Problemlerinde Kullanılan Bazı Meta Sezgisellerin Özellikleri

2.5.1. Tek Çözüm Temelli Yöntemler

Tek çözüm temelli yöntemler, bir başlangıç çözümden yola çıkarak arama uzayını araştırır. Bu yöntemler, bazı prosedürler kullanarak başlangıç çözüm üzerinde değişiklikler yapar ve bir aday çözüm elde eder. Elde edilen aday çözümün kalitesi, yerel aramanın sezgisel şekilde söz konusu bölgeye yoğunlaşmasını ya da arama uzayının farklı bir bölgesinde arama yapmaya devam etmesini belirler. Belirli bir durdurma koşulu altında bu işlemler tekrarlanarak arama yapılır ve optimal ya da optimale yakın çözümün elde edilmesi beklenir. İzleyen alt başlıklarda bu sınıfa giren meta sezgisellerin karakteristik özellikleri ele alınmıştır.

2.5.1.1. Yinelemeli yerel arama

Yinelemeli yerel arama (iterated local search), Lourenço vd. (2001) tarafından geliştirilen ve kombinatoryal optimizasyon problemlerinde kullanılan meta sezgisellerden birisidir. Yinelemeli yerel arama algoritmasında gömülü bir sezgisel ile yinelemeli olarak bir dizi çözüm oluşturulur (Lourenço vd., 2019: 130). Algoritmanın temel fikri, çözümler üzerinde değişiklikler yaparak yerel optimum uzayında rastgele yürüyüş yapmaktır (Pan ve Ruiz, 2012: 33). Bu yaklaşımı ile yinelemeli yerel arama

algoritmasında, bir başlangıç çözümden değişiklikler ile yeni çözümler elde edilmektedir ve bu çözüm dizisine, zincir çözümler de denilmektedir.

Olası çözümler uzayı S'de arama yaparken her yinelemede rastgele seçilen bir noktadan başlayarak arama yapma fikri, çok sayıda örnek içeren uzaylarda etkin olmayan bir yaklaşımdır. Yinelemeli yerel arama, bunun yerine daha önceden bulunan çözümlerin sarsımı ile oluşturulan yeni başlangıç çözümler elde ederek zincir çözümler oluşturma fikrinden faydalanır, böylelikle başlangıç çözümlerin yanlı örneklemesi yapılmış olur (Stützle ve Ruiz, 2018: 580). Yinelemeli yerel arama algoritması, 4 temel bileşenden oluşmaktadır. Bunlar:

i. Başlangıç Çözüm,

ii. Yerel Arama,

iii. Sarsım,

iv. Kabul kriteri.

Algoritma, bir başlangıç çözümden başlar ve sırasıyla yerel arama ve sarsım adımlarında bu çözümü değişikliğe uğratarak yeni çözümler elde eder. Elde edilen çözümler, kabul kriterinde değerlendirilerek yöntemin bir sonraki yinelemede harekete başlayacağı nokta belirlenir.

Yinelemeli yerel arama için gerekli başlangıç çözüm, rastgele ya da açgözlü (greedy) bir kurucu sezgisel ile oluşturulabilir (Leivadeas vd., 2013: 1079). Optimal ya da optimale yakın çözümler aranırken bu noktalara yakın bir noktadan aramaya başlamanın, genel olarak çok daha uzak bir noktadan aramaya başlamaya göre kısa sürede nihai hedefe ulaştıracağı göz önünde bulundurularak başlangıç çözümü üretecek mekanizma seçilmelidir. Bu nedenle başlangıç çözümlerin, mümkün olduğunca fazla bilgi kullanması gerekmektedir (Kramer, 2014: 45). Yeterince iyi seçilmemiş bir başlangıç çözüm, yerel aramayı optimum olmaktan uzak ya da başka bir deyişle kalitesi düşük olan yeni çözümlere yönlendirebileceği için algoritmanın performansı üzerinde olumsuz etkide bulunabilir.

Yerel arama için gerekli olan temel içerik, bir s çözümünden komşuluktaki daha iyi bir çözüme akıllıca bir geçişe izin veren komşuluk yapısıdır (Lourenço vd., 2019:

132). Bir aday çözüm olan $s \epsilon S$'nin komşuluğu $N(s)$, s'ye uygulanacak spesifik değişiklikler ya da hareketler ile elde edilebilecek bütün aday çözümler tarafından tanımlanır (Stützle ve Ruiz, 2018: 581). Yerel arama, algoritmanın işleyişi üzerinde önemli bir etkiye sahiptir. Bu nedenle yerel arama için değişik sezgiseller kullanılabilir (Lourenço vd., 2001: 4).

Sarsım, bir çözümü oluşturan bileşenler üzerinde değişiklik yapma biçiminde düşünülebilir. Değişiklik sayısı ölçülebildiği için sarsım da ölçülebilirdir. Sarsımın amacı, geçerli aday çözüme yerel arama aşamasında yapılan değişikliklerden daha fazla değişiklik uygulayarak yerel optimumdan ya da arama uzayının belirli bölgelerinden kaçmaktır (Stützle ve Ruiz, 2018: 584). Sarsım, algoritmanın yerel optimumda sıkışıp kalmasını engelleyerek arama uzayının farklı bölgelerinin keşfedilebilmesini sağlar. Elde edilen ara çözümlere uygulanan sarsım çok güçlü olursa (çok fazla çözüm bileşeni değiştirilirse) mevcut çözüm özelliklerini kaybederek farklı bir çözüme dönüşeceği için algoritma, rastgele arama yapmaya başlayacaktır; sarsım çok zayıf olursa (çok az çözüm bileşeni değiştirilirse) çok da farklı olmayan yeni bir çözüm ortaya çıkacağı için algoritma, dar bir bölgede arama yapmak zorunda kalacaktır.

Geçerli komşu çözümün sonraki yineleme için kabul edilip edilmeyeceği, kabul kriteri bileşeni tarafından belirlenir (Toksari, 2016: 778). Bu bileşen, sadece daha iyi çözümleri kabul edebileceği gibi arama sürecinde daha fazla çeşitlilik olması adına hafıza yapıları kullanarak belirli sayıda yinelemede çözüm değerinde herhangi iyileşme olmaması durumunda, yinelemeli yerel arama sürecini yeniden başlatabilir (Leivadeas vd., 2013: 1079). Lourenço vd. (2001) tarafından verilen bilgiler ışığında uyarlanan yinelemeli yerel arama algoritmasına ait sözde-kod, Şekil 7'de gösterildiği gibidir.

```
prosedür Yinelemeli Yerel Arama
    s0 = BaşlangıçÇözümOluştur
    s* = YerelArama(s0)
    tekrar et
        s' = Sarsım(s*)
        s*' = YerelArama(s')
        s* = KabulKriteri(s*,s*')
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 7. Yinelemeli Yerel Arama Algoritmasına Ait Sözde-Kod

2.5.1.2. Açgözlü rassallaştırılmış uyarlanabilir arama prosedürü

Feo ve Resende (1989) tarafından geliştirilen açgözlü rassallaştırılmış uyarlanabilir arama prosedürü (greedy randomized adaptive search procedure), kombinatoryal optimizasyon problemlerinde kullanılan meta sezgisel yöntemlerden birisidir. Yöntem, çoklu-başlangıç (multi-start) yapan, yinelemeli bir sürece sahiptir (Shen vd., 2011: 464). Açgözlü rassallaştırılmış uyarlanabilir arama prosedürü çalışırken iki temel aşama vardır (López-Sánchez vd., 2019: 297):

i. Açgözlü bir fonksiyon kullanılarak rassallaştırılmış başlangıç çözüm oluşturma,

ii. Yerel optimuma ulaşmak için iyileştirme.

Bu iki aşama, maksimum izin verilen CPU (işlemci) süresi, maksimum yineleme sayısı, başarılı iki iyileşme arasındaki maksimum yineleme sayısı gibi durdurma koşulları sağlanıncaya kadar yinelemeli olarak devam eder (Consoli vd., 2009: 443).

Başlangıç çözüm oluşturma aşamasına boş bir çözüm ile başlanır ve yinelemeli olarak her seferinde bir tane uygun çözüm oluşturulur (Festa, 2002: 7). Her çözüm oluşturma yinelemesinde, eklenecek sıradaki unsurun belirlenmesi, açgözlü fonksiyonla ilgili olan aday listesinde sıralama yapılarak belirlenir. Bu fonksiyon, her bir unsurun seçilmesinin faydasını ölçer (Feo ve Resende, 1995: 111). Açgözlü fonksiyonun ölçtüğü fayda, aslında maliyet minimizasyonu problemlerinde çözüme girmeye aday bir unsurun toplam maliyeti ne kadar arttıracağının belirlenmesidir. Amaç fonksiyonunun yapısı

nedeniyle en az maliyet artışına neden olacak unsurların seçilmesi daha yüksek fayda sağlayacaktır.

Açgözlü rassallaştırılmış uyarlanabilir arama prosedürü, sahip olduğu açgözlü fonksiyonun işlevi doğrultusunda en yüksek faydayı sağlayacak ya da başka bir deyişle maliyeti en az arttıracak en iyi adayları bir listede tutar. Bu liste, kısıtlanmış aday listesi (restricted candidate list) olarak isimlendirilir (Feo ve Resende, 1995: 111). Yöntemin olasılıksal yönü, söz konusu listeden bir adayın seçilmesinden gelmektedir. Aday listesinden bütün adayların seçilme olasılıkları, genellikle birbirine eşit olmakla birlikte seçilen adayın en iyi aday olmasına gerek yoktur (Festa ve Resende, 2009: 2); ancak başka bir yaklaşımla yanlı örnekleme de yapılabilir (Anagnostopoulos vd., 2010: 1062). Algoritma her çalıştırıldığında bu olasılıksal yönü nedeniyle farklı çözüm değerleri üretmektedir ve prosedürün rastgeleliği, çözüm uzayının farklı bölümlerinden çözümlerin elde edilebilmesine izin vermektedir (Consoli vd., 2009: 443). Seçilen unsur, kısmi çözüme eklendikten sonra ise aday unsur kümesi güncellenir ve maliyet artışları tekrar değerlendirilir (Resende ve Ribeiro, 2014: 290). Kısıtlanmış aday listesine üyelik, sıraya veya adayların göreli kalitelerine dayanarak iki şekilde belirlenebilir (Resende, 2008: 298).

Çözüm oluşturma aşamasında elde edilen uygun çözüm, yüksek (düşük) amaç fonksiyonu değerine sahip olabileceğinden optimal çözüme ulaşabilmek için yerel arama aşamasına ihtiyaç duyulmaktadır. Yerel arama aşamasında, kurulan çözüm geliştirilmeye çalışılır ve tanımlanan komşuluk yapısına göre yerel optimal olan bir çözüm üretilir (Pardalos vd., 1998: 400). Bu aşama, komşulukta daha iyi bir çözüm olmadığında sona erer (Binato vd., 2001: 249). Bu sona erme koşulundan, belirli bir problem ve bunun bir çözümü için tanımlanmış olan komşuluk yapısında daha iyi bir çözüm bulunmuyorsa yerel optimal bir çözüme ulaşıldığı sonucu çıkarılmaktadır. Optimalliğe ulaşmada kullanılan yerel arama için başarının anahtarı, uygun komşuluk yapısının seçimi, etkin komşuluk arama teknikleri, maliyet fonksiyonunun hızlı değerlendirilmesi ve çözüm kurma aşaması ile ilgili olarak başlangıç çözümün kalitesidir (Festa, 2002: 8). Yerel arama algoritmaları, keyfi bir başlangıç noktasından başladığında üstel zaman gerektirebilir, bu nedenle daha iyi başlangıç çözüm verecek yaklaşımlarla arama süreci etkin bir şekilde yürütülebilir (Feo ve Resende, 1995: 112). Feo ve Resende (1995) tarafından verilen bilgiler ışığında uyarlanan sözde-kod, Şekil 8'de gösterildiği gibidir.

```
prosedür Açgözlü Rassallaştırılmış Uyarlanabilir Arama
    girdi α
    tekrar et
        çözüm = AçgözlüRassalÇözümOluştur(α)
        çözüm = YerelArama(çözüm)
        Eğer f(çözüm) < f(eniyiçözüm)
            eniyiçözüm = çözüm
        sonlandır
    durdurma koşulu sağlanana kadar
    en iyi çözümü ver
sonlandır
```

Şekil 8. Açgözlü Rassallaştırılmış Uyarlanabilir Arama Prosedürüne Ait Sözde-Kod

2.5.1.3. Tabu arama

Tabu arama (tabu search), Glover (1986) tarafından geliştirilen ve kombinatoryal problemlerde kullanılan bir meta sezgisel yöntemdir. Yöntemin isminde geçen "tabu" kelimesi, bir Polinezya dili olan Tongan'dan gelmektedir ve anlam olarak kutsal oldukları için dokunulamayan şeyleri ifade eder (Glover ve Laguna, 1998: 624). Tabu kelimesi, algoritma içerisinde de benzer şekilde bir yasaklama işlevine sahiptir. Tabu aramayı yerel arama yöntemlerinden ayıran özelliği, hafıza yapısına sahip olmasıdır (Lucay vd., 2019: 6). Tabu arama, ödünleşim (performans indeksi ya da amaç fonksiyonu) bilgisini problem uzayında aramaya kılavuzluk etmek için kullanır. Böylelikle düzgün olmayan, sürekli olmayan ve türevi alınmayan amaç fonksiyonları kolaylıkla ele alınabilir (Abido, 2002: 472).

Tabu aramanın temel prensibi, bir yerel optimum ile karşılaşıldığında, iyileştirmeyen hareketlere izin vererek yerel aramayı sürdürmektir; hafıza kullanımı ile önceden ziyaret edilen çözümlere dönmek engellenir (Gendreau ve Potvin, 2010: 44). Bu stratejiler ile tabu arama, hem yerel optimuma sıkışmaktan kurtulmaktadır hem de arama sürecinin sonucunda daha iyi bir çözüme ulaşabilmek için kötü kaliteye sahip çözümlerden de faydalanmaktadır (Al-Sultan, 1995: 1444).

Tabu aramada hafızadan yararlanmanın temel yolu, bir komşuluktaki hareketlerin alt kümesini yasak (veya tabu) olarak sınıflandırmaktır. Sınıflandırma, araştırmanın geçmişine ve özellikle de öznitelikler adı verilen belirli hareket veya çözüm bileşenlerinin geçmiş çözümlerin üretilmesinde katıldığı yineleme veya sıklığa bağlıdır (Glover vd.,

1995: 114). Bu yasaklama sayesinde daha çeşitli bir arama yapılmasına izin verilir (Kulturel-Konak vd., 2003: 516). Tabu görev süresi denilen belirli yineleme sayısı süresince yasaklanan hareketler tersine çevrilemez (Fescioglu-Unver ve Kokar, 2011: 311). Arama süresince ziyaret edilen bütün çözümleri kaydederek döngülerden kurtulmak fikri, ortaya çıkan bütün bilgiyi kontrol etmeyi gerektirdiği için maliyetli bir işlemdir. Bu nedenle genellikle tabu listelerinde sabit ve uygun miktarda kısıtlı bilgi tutulur. Tabu listesi, aramanın kısa dönemli hafızasını ifade eder (Gendreau, 2003: 43). Tabu listesinin uzunluğu, sabit kalabilir veya arama süresince dinamik olarak değiştirilebilir (Corman vd., 2010: 180). Arama sürecinde ortaya çıkan yeni hareketler, tabu listesindeki en eski hareketlerin yerine yazıldığından tabu listelerinde ilk giren ilk çıkar prosedürü söz konusudur (Pham ve Karaboga, 2000: 9).

Bazen tabular çok güçlü olabilir. Çekici hareketleri yasaklayabilir, hatta döngü tehlikesi yokken bile arama sürecinin bütün olarak durmasına neden olabilir (Gendreau ve Potvin, 2010: 47). Bu tehlike karşısında bazı durumlarda tabu kurallarına aykırı hareket edilebilir. Tabu kurallarına aykırı şekilde hareket edilmesine izin veren bir koşul, tabu yıkma (aspirasyon) kriteri olarak adlandırılır (Glover vd., 1995: 115). Tabu listesindeki bir çözüm, yalnızca bu kriteri sağladığında dikkate alınır.

Tabu arama, sadece yakın zamanda gerçekleştirilmiş hareketleri yasakladığında yerel optimuma takılıp kalınabilir ya da farklı bir bölgede aramaya geçilse bile elde edilecek çözümlerin kalitesi yeterince iyi olmayabilir (Dréo vd., 2006: 69). Tabu listesine ek olarak, çözümlerle ilgili öncel bilgiler ve uzun dönemli hafıza arama sürecinin yoğunlaşma ve/veya çeşitlendirmesini iyileştirmek için kullanılabilmektedir (Zhang ve Sun, 2002: 703). *"Orta ve uzun dönemli hafızalar, tabu listelerinin yerine getirdiği kısa dönemli hafıza fonksiyonu ile kullanıldığında 'öğrenme' ve 'öğrenmeme' arasında bir etkileşim sağlar"* (Glover, 1989: 199).

Tabu arama süreci, en iyi çözümün elde edilmesinden sonra değişiklik olmayan yineleme sayısının önceden belirlenen sayıdan büyük olması, izin verilen maksimum yineleme sayısına ulaşılması gibi diğer meta sezgisel yöntemlerde de kullanılan durdurma kriterleri sağlanıncaya kadar devam eder. Gendreau ve Potvin (2010) tarafından verilen bilgiler ışığında uyarlanan tabu arama algoritmasına ait sözde-kod, Şekil 9'da verilmiştir.

```
prosedür Tabu Arama
    s0 = BaşlangıçÇözümOluştur
    EnİyiÇözüm = s0
    TabuListesi = {}
    tekrar et
        KomşuÇözüm = KomşuBul(EnİyiÇözüm)
        Eğer (KomşuÇözüm ∉ TabuListesi) ve (f(KomşuÇözüm) < f(EnİyiÇözüm))
            EnİyiÇözüm = KomşuÇözüm
        sonlandır
        Güncelle(TabuListesi)
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 9. Tabu Arama Algoritmasına Ait Sözde-Kod

2.5.1.4. Değişken komşuluk arama

Değişken komşuluk arama (variable neighborhood search), kombinatoryal optimizasyon problemlerinde uygulanan meta sezgisel yöntemlerden biridir. Yöntem, Mladenović ve Hansen (1997) tarafından geliştirilmiştir. Yerel arama yöntemlerinde bir başlangıç çözümden başlanarak yerel optimum çözüm elde edilinceye kadar bu çözüm üzerinde değişiklikler yapılır. Yapılan değişiklikler ile çözümde iyileşme sağlanır ve s çözümünün $N(s)$ komşuluğunda s' çözümüne ulaşılır. Bu yöntemlerde genel olarak tek bir komşuluk yapısı bulunmaktadır (Mladenović ve Hansen, 1997: 1097). Değişken komşuluk arama yöntemi ise yerel aramayı iki yönden genelleştiren bir stratejiye sahiptir (Caporossi ve Hansen, 2000: 33):

i. Bir yerine birkaç komşuluk yapısı göz önünde bulundurulur.

ii. İlk önce sıra numarası küçük olan komşuluklar ayrıntılı olarak araştırılır ve daha sonra sıra numarası büyük olan komşuluklarda rastgele bir arama yapılır.

Değişken komşuluk arama, yerel aramaya dayalı diğer meta sezgisellerden (örneğin tabu arama) farklı olarak bir yörünge izlemez; ancak geçerli çözümün artan uzaklıktaki komşuluklarını keşfeder ve yalnızca amaç fonksiyonu değerinde bir gelişme sağlanıyorsa bu çözümden yeni çözüme geçer (Hansen ve Mladenović, 2001: 450). Değişken komşuluk arama yönteminin temel fikri, yerel optimumdan kaçmaya izin vermek için basit bir yerel arama yöntemini zenginleştirmektir. Bu, geçerli çözümün

rastgele seçilen bir komşuluğunda tekrar yerel aramaya başlanarak yapılır. Yeniden başlama adımı, sarsma (shaking) adımı olarak isimlendirilir ve artan boyutlarda farklı komşulukları kullanarak gerçekleştirilir (Driessel ve Mönch, 2011: 338). Değişken komşuluk arama, arama sürecinde komşuluk yapılarını sistematik olarak aşağıdaki bilgilere dayanarak değiştirmektedir (Hansen vd., 2017: 424):

i. Farklı komşulukların yerel optimumları farklı olabilir.

ii. Global optimum, bütün komşuluk yapıları için bir yerel optimumdur.

iii. Yerel optimumların büyük bir kısmı, birbirine yakındır.

Değişken komşuluk arama meta sezgiseline ait bu özellikler değerlendirildiğinde sırası ile incelenen komşuluklarda amaç fonksiyon değeri açısından düşük kalitede çözümlerle karşılaşıldığında birden fazla komşuluk kullanmanın getirdiği avantaj ile sonraki komşuluklara geçilerek daha yüksek kalitede çözümlerin elde edilmesi mümkün hale gelmektedir. Bu strateji ile yapılan arama sonucunda bütün komşuluk yapılarına göre en iyi olan çözüm (en düşük maliyetli ya da en kaliteli çözüm), problem için optimal çözüm ya da optimale yakın çözüm olacaktır. Yerel optimum değerlerinin yakın olması ise umut verici bölgelere yoğunlaşarak daha kısa sürede benzer çözüm değerlerinin elde edilebileceği anlamına gelmektedir.

Değişken komşuluk arama yöntemi üç basit adımı bir araya getirmektedir (Jarboui vd., 2013: 48):

i. Stokastik şekilde geçerli çözümün rastgele komşu çözümünü bulan sarsma adımı

ii. Herhangi bir yerel arama algoritmasının uygulandığı deterministik adım

iii. Yalnızca amaç fonksiyonunu geliştiren çözümleri kabul eden değerlendirme adımı

Yönteme ait üç adım değerlendirildiğinde, değişken komşuluk arama meta sezgiselinin hem stokastik hem de deterministik bileşenlere sahip olduğu görülmektedir. Yöntemin ilk iki adımı, komşuluk yapılarını kullanmayı gerektirdiğinden bu adımlara ilişkin komşulukların karıştırılmaması için farklı gösterimler kullanılmaktadır. Burada,

komşuluk terimi ile değişik sayıda yerel arama algoritması kastedilir (Marinaki ve Marinakis, 2015: 185).

Bu tarzda bir arama süreci; izin verilen maksimum işlemci süresi, iki iyileştiren yineleme arasındaki geçen süre, maksimum yineleme sayısı gibi durdurma koşulları arasından seçilen kriter sağlanıncaya kadar devam eder (Hansen ve Mladenović, 2003: 148). Driessel ve Mönch'e göre (2011: 338) arama sürecinin başarısı; başlangıç çözümün kalitesi, seçilen komşuluklar, kullanılan yerel arama yöntemleri ve komşulukların uygulanma sırası gibi faktörlere bağlıdır. Buna ek olarak, ele alınan probleme bağlı olarak şu problemlerin de yanıtlanması gerekmektedir (Hansen ve Mladenović, 2001: 451-452):

i. *"N_k komşuluk yapısı ne olmalıdır ve bunlardan kaç tanesi kullanılmalıdır?*

ii. *Bu komşulukların arama sırası ne olmalıdır?*

iii. *Komşulukları değiştirmede strateji ne olmalıdır?"*

Söz konusu hususlarda titiz bir şekilde karar alınarak başarılı bir değişken komşuluk arama uygulaması sağlanabilir. Bu konuda literatürdeki benzer problemi ele alan çalışmalar incelenerek ilk etapta fikir sahibi olmak mümkündür. Hansen ve Mladenović (1999) tarafından verilen bilgiler ışığında uyarlanan değişken komşuluk algoritmasına ait sözde-kod Şekil 10'da gösterildiği gibidir.

```
prosedür Değişken Komşuluk Arama
    s0 = BaşlangıçÇözümOluştur
    s = s0
    tekrar et
        for i = 1:kmax
            s' = Sarsma(s,i)
            s'' = YerelArama(s')
            Eğer f(s'') < f(s)
                s = s''
                i = 1
            Değilse
                i = i+1
            sonlandır
        sonlandır
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 10. Değişken Komşuluk Arama Algoritmasına Ait Sözde-Kod

2.5.1.5. Tavlama benzetimi

Tavlama benzetimi (simulated annealing), kombinatoryal optimizasyon problemlerinde yaygın şekilde kullanılan bir meta sezgisel yöntemdir. Söz konusu yöntem, birbirinden bağımsız şekilde Kirkpatrick vd. (1983) ve Černý (1985) tarafından literatüre kazandırılmıştır. Tavlama benzetimi, metallerdeki tavlama (yavaş yavaş ısıtma ve soğutma) süresince atomların tekrar kristalize olmasının benzetimi olan Metropolis algoritmasından gelmektedir (Tavares vd., 2011: 2952). Yoğun madde fiziğinde tavlama, bir katının ısı banyosunda düşük enerji seviyeli durumlarını elde etmek için kullanılan ısıl bir süreçtir. İki adımdan oluşmaktadır (Aarts vd., 1997: 96):

i. Isı banyosunun sıcaklığı katının erime noktasına kadar arttırılır.

ii. Sıcaklık, erimiş parçacıklar tekrar katı hale dönene kadar kontrollü olarak azaltılır.

Kirkpatrick ve arkadaşları 1983 yılında yayımladıkları makaleleri ile katıların tavlama benzetimi ile kombinatoryal minimizasyon arasında bu bağlantıyı ilk kez kurarak sistem enerjisi yerine maliyet fonksiyonunu, fiziksel sistemin durumları yerine ise

minimizasyon probleminin çözümlerini kullanmışlardır (Dekkers ve Aarts, 1991: 370). Tavlama benzetimi her ne kadar kombinatoryal optimizasyon problemleri ile ilişkilendirilse de sürekli optimizasyon problemleri için genişletilmiş uzantıları mevcuttur (Crama ve Schyns, 2003: 551).

Tavlama benzetimi, iniş yerel arama algoritması gibi basit yerel arama prosedürlerini kullanarak başlangıç çözüm elde eder ve bazı hareketlere dayalı olarak da bir komşu çözüm elde eder (Wang vd., 2011: 763). Tavlama benzetimi, ilk uyan (first fit) stratejisi ile amaç fonksiyonunu değiştiren (daha düşük değere sahip olan) bütün çözümlerin derhal kabul edildiği, diğer çözümlerin ise belirli bir olasılıkla kabul ya da reddedildiği, yerel arama algoritmalarının özel bir durumu olarak görülebilir (Józefowska vd., 2001: 141). Tavlama benzetimi, çözüm uzayında mevcut çözüm ya da komşu çözüm arasında seçim yaparak ilerlediği için yörüngeye dayalı bir yöntemdir ve yerel optimumdan kaçabilmek için geçerli çözümü kötüleştiren çözümleri kabul edebilmektedir (Rodríguez vd., 2013: 8579). Geçerli çözümleri kötüleştiren çözümlerin kabul edilebilmesi, yöntemin olasılıksal yönünü ortaya koymaktadır. Buna benzer s' komşu çözümlerin kabul edilebilmesi için gerekli olasılık ise Boltzmann dağılımı yardımı ile Eşitlik (2.2)'de gösterildiği gibi hesaplanır (Wang vd., 2011: 763).

$$P(s') = e^{\left(\frac{-(f(s')-f(s))}{T}\right)} \tag{2.2}$$

Eşitlik (2.2)'de yer alan T parametresi, sıcaklığı kontrol etmektedir. Bu eşitlikte, T parametresinin değeri yükseldikçe s' komşu çözümünün kabul edilme olasılığı artarken; benzer şekilde parametrenin değeri azaldığında ise çözümün kabul edilme olasılığı azalmaktadır. Çözüm kalitesi daha düşük olan çözümlerin kabul edilmesini zorlaştırmak için sıcaklık düşürülerek daha iyi çözümlerin kabul edilmesi sağlanmaktadır. Tavlama benzetimi yöntemi ile yüksek kalitede çözümler elde edebilmek için sıcaklığın kontrol edildiği soğutma planının uygulanışı önem taşımaktadır.

Amaç fonksiyonu değerini kötüleştiren bir komşu çözüm söz konusu olduğunda bu çözümün kabul edilmesi ya da reddedilmesi şu şekilde belirlenir: Öncelikle Eşitlik (2.2) yardımıyla söz konusu komşu çözüm için bir olasılık değeri hesaplanır. Daha sonrasında ise bu çözümün kabulü ya da reddi için $[0,1]$ aralığından tekdüze dağılıma

uyan rastgele bir sayı üretilir. Eğer, üretilen bu rastgele sayının değeri daha küçük ise söz konusu komşu çözüm kabul edilir (Osman ve Potts, 1989: 553).

Eglese (1990) tarafından verilen bilgiler ışığında uyarlanan tavlama benzetimi algoritmasına ait sözde-kod, Şekil 11'de gösterildiği gibidir.

```
prosedür Tavlama Benzetimi
    girdi BaşlangıçSıcaklık
    s0 = BaşlangıçÇözümOluştur
    T = BaşlangıçSıcaklık
    tekrar et
        s' = KomşuÇözümBul( )
        Eğer f(s') < f(s0)
            s0 = s'
        Değilse
            P(s') = exp(−(f(s') − f(s0))/T
            r = RastgeleSayıÜret( )
            Eğer r < P(s')
                s0 = s'
            sonlandır
        sonlandır
        T = SıcaklığıDüşür(BaşlangıçSıcaklık)
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 11. Tavlama Benzetimi Algoritmasına Ait Sözde-Kod

2.5.1.6. Yönlendirilmiş yerel arama

Yönlendirilmiş yerel arama (guided local search), Voudouris ve Tsang (1995) tarafından geliştirilen kombinatoryal optimizasyon problemlerine uygun bir meta sezgisel yöntemdir (Voudouris ve Tsang, 1995: 1). Yönlendirilmiş yerel arama, yerel arama aşamasında yerel optimuma takılmamak için ceza uygulayarak yerel arama yöntemlerini yönlendiren bir meta sezgiseldir (Cramer ve Kampouridis, 2015: 802). Yönlendirilmiş yerel arama algoritmasında temel fikir, çözüm uzayını keşfetmek için yerel arama prosedürünü yönlendirecek bir fonksiyon kullanmaktır (Tairan vd., 2015: 57).

Yönlendirilmiş yerel arama, yerel aramayı özellikler (features) kavramı ile genişletir. Burada bahsedilen özellikler kümesi, bir problemin çözümünü doğal şekilde karakterize eder (Faroe vd., 2003: 274). Özellikler, çözümleri karakterize ettiğinden farklı

çözümleri birbirinden ayırmak için tanımlanır. Böylelikle kötü özellikler cezalandırılarak arama sürecine dâhil edilmemiş olur (Mills vd., 2003: 122). Bir özelliğin iyi ya da kötü olarak nitelendirilmesi, optimizasyon probleminin tipine bağlıdır.

Yerel arama, yerel optimuma takıldığında daha önceden belirlenmiş özellikler kümesinden belirli özellikler seçilir ve seçilen özellikler cezalandırılır. Yerel arama, birikimli cezalarla arttırılmış amaç fonksiyonunu (augmented cost function) kullanarak arama uzayını arar (Alsheddy vd., 2016: 3). Arttırılmış amaç fonksiyonunda yer alan kötü özelliklerin cezalandırılması yöntemin tabu arama algoritmasında olduğu gibi bir hafıza yapısı olduğu anlamına gelmektedir (Boussaïd vd., 2013: 88). Her s çözümünü sayısal bir değer ile eşleştiren g amaç fonksiyonu verildiğinde, yöntem tarafından tanımlanan ve yerel arama tarafından kullanılacak (g fonksiyonunu değiştirerek) h fonksiyonu, Eşitlik (2.3)'te gösterildiği gibidir (Voudouris ve Tsang, 2003: 187).

$$h(s) = g(s) + \lambda \sum (p_i * I_i(s)) \tag{2.3}$$

Eşitlik (2.3)'te yer alan λ, yönlendirilmiş yerel arama algoritmasına özgü bir parametredir. Benzer şekilde p_i, i özelliğine ait cezayı; $I_i(s)$ ise Eşitlik (2.4)'te verildiği gibi bu özelliğin s çözümünün i özelliğini sergileyip sergilemediğini göstermektedir (Voudouris ve Tsang, 2003: 187).

$$I_i(s) = \begin{cases} 1 & , eğer\ s\ çözümü\ i\ özelliğini\ sergiliyorsa \\ 0, & eğer\ s\ çözümü\ i\ özelliğini\ sergilemiyorsa \end{cases} \tag{2.4}$$

Amaç fonksiyonundaki λ parametresi, çözüm maliyetini dikkate alarak cezaların göreli önemini temsil eder ve arama sürecindeki bilginin etkisini kontrol eder (Barbucha, 2012: 12033). λ parametresi, yüksek değere sahip ise arama daha çeşitli olmaktadır yani vadi ve düzlükler daha özensiz şekilde aranmaktadır; λ parametresi, düşük değere sahip ise arama daha yoğun şekilde yapılmaktadır yani vadi ve düzlükler daha özenli şekilde aranmaktadır (Mills vd., 2003: 123).

Yönlendirilmiş yerel arama algoritmasının başlangıcında bütün özelliklere ait ceza değerleri 0'dır. Bu nedenle de başlangıçta problemin orijinal amaç fonksiyonu değeri ile arttırılmış amaç fonksiyonu değeri birbirine eşittir. Yönlendirilmiş yerel arama kullanıldığında karşılaşılan yerel minimum, arttırılmış maliyet fonksiyonu ile ilgilidir ve problemin orijinal maliyet fonksiyonu ile ilişkili yerel minimumdan farklı olabilir. Bu

nedenle yerel minimumdan bahsedildiğinde arttırılmış maliyet fonksiyonundan bahsedilmektedir (Voudouris ve Tsang, 1999: 472). Daha iyi çözüm değerleri elde edebilmek için istenmeyen özellikler cezalandırılarak yerel optimumlardan kaçınılmaya çalışılır. İstenmeyen bir özelliğin s^* yerel optimum çözümünde cezalandırılması durumunda elde edilecek fayda, Eşitlik (2.5)'te verildiği gibidir (Alsheddy vd., 2016: 4).

$$fayda_i(s^*) = I_i(s^*) * \frac{c_i}{1 + p_i} \tag{2.5}$$

Eşitlik (2.5)'te yer alan c_i, i özelliğinin maliyetini ve p_i ise bu özelliğin geçerli ceza puanı değerini göstermektedir. Bu ceza puanı 0 değerinden başlamakta ve her seferinde 1 arttırılmaktadır. Cezalandırma sayısı arttıkça, tekrar cezalandırmanın faydası azalmaktadır (Alsheddy vd., 2016: 4). Fayda fonksiyonu bu sayede kötü bir özelliğin her yinelemede cezalandırılma riskini azaltmaktadır (Flores Lucio vd., 2007: 3180). Algoritma, bir yerel optimuma ulaştığında en kötü çözüm özelliği cezalandırılır. En kötü çözüm özelliğini tespit etmek için ise her bir özelliğin fayda fonksiyonu değeri hesaplanır. Birden fazla özellik, aynı en yüksek fayda derecesine sahip ise bunların hepsi cezalandırılır (Steitz ve Rothlauf, 2012: 48).

Tsang ve Voudouris (1997) tarafından belirtilen prosedüre göre uyarlanan yönlendirilmiş yerel arama algoritmasına ait sözde-kod, Şekil 12'de verilmiştir.

```
prosedür Yönlendirilmiş Yerel Arama
    girdi lambda
    s0 = BaşlangıçÇözümOluştur()
    EnİyiÇözüm = s0
    CezaDeğerleri = 0
    tekrar et
        s* = YerelArama(EnİyiÇözüm)
        FaydaDeğerleri = FaydaDeğeriHesapla(s*)
        CezaDeğerleri = Cezalandır(maks(FaydaDeğerleri))
    durdurma koşulu sağlanıncaya kadar
    g fonksiyonuna göre en iyi aday çözümü ver
sonlandır
```

Şekil 12. Yönlendirilmiş Yerel Arama Algoritmasına Ait Sözde-Kod

2.5.1.7. Geniş komşuluk arama

Geniş komşuluk arama (large neighborhood search), büyük boyutlu kombinatoryal problemlerin çözümünde kullanılan ve yerel arama ile kısıt programlamayı bir araya getiren bir meta sezgiseldir (Pacino ve Van Hentenryck, 2011: 1998). Shaw (1998) tarafından geliştirilen ve çok geniş ölçekli komşuluk arama algoritmaları sınıfının bir üyesi olan yöntem (Monçores vd., 2018: 95), "yık" ve "tamir et" prensibine sahiptir (Vergeylen vd., 2019: 12). Geniş komşuluk aramanın temel fikri, başlangıç çözümden başlayıp geçerli çözüme dönüşümlü bir şekilde yıkma ve tamir etme işlemleri uygulanarak aşamalı şekilde gelişme sağlamaktır (Zhang vd., 2018: 318; Bisaillon vd., 2011: 145). Söz konusu bu temel fikir doğrultusunda yöntem, mevcut çözümün geniş bir kısmını değiştirerek daha iyi bir çözüm arar, büyük komşuluğu keşfetmek küçük komşuluğu keşfetmeye göre daha yüksek kaliteli çözümleri açığa çıkarmaya eğilimlidir ve bunun sonucu olarak daha yüksek hesaplama yükünü beraberinde getirir (Kilby ve Urli, 2016: 11).

Geniş komşuluk arama, çalışılan problemin özelliklerine göre uyarlanması gereken genel bir çerçevedir ve bu nedenle geniş komşuluğun tanımı, problemin amacına bağlıdır (Copado-Méndez vd., 2013: 117). Yöntemde bir s çözümünün komşuluğu $N(s)$, yıkma ve tamir etme yöntemleri ilk uygulandığında ulaşılabilecek çözümlerin kümesidir (Pisinger ve Ropke, 2010: 8). Yöntemin işleyişi şu şekildedir (Pacino ve Van Hentenryck, 2011: 1998): Bir başlangıç çözüm elde edildikten sonra değişken atamalarının bir kısmı gevşetilir (yıkım işlemi) ve geri kalan değişkenler sabit tutulur. Yeni çözüm ise serbest bırakılan değişkenlerin yeniden optimize edilmesi (tamir işlemi) ile elde edilir. Bu iki işlem, durdurma kriteri sağlanıncaya kadar devam eder. Arama yapılacak komşuluğun büyüklüğü, yıkım ve tamir etme süreçlerine dayandığı için yöntemin çözümü iyileştirme konusundaki performansı bu iki işleme bağlıdır. Çözümde gevşetilecek ya da başka bir deyişle silinecek değişkenlerin sayısı arama uzayının ne kadar serbest şekilde keşfedildiğini belirlemektedir. Yıkım işlemine dâhil edilen değişken sayısı arttıkça çözüm kalitesinin artması beklenirken (Monçores vd., 2018: 96), benzer şekilde işlem yükü de artacaktır. Bu nedenle, optimize edilecek değişkenler tamamen rastgele ya da amaç fonksiyonunda en büyük potansiyel etkiye sahip ve ilişkili değişkenleri göz önünde bulundurarak yönlendirilmiş şekilde seçilebilmektedir (Copado-Méndez, 2013: 117).

Pisinger ve Ropke (2010)'un çizdiği çerçeveden uyarlanan geniş komşuluk arama algoritmasına ait sözde-kod, Şekil 13'te verildiği gibidir.

prosedür Geniş Komşuluk Arama
    s = *BaşlangıçÇözümOluştur*
    EnİyiÇözüm = s
    tekrar et
        s' = *TamirEt*(*Yık*(s_0))
        Eğer *KabulFonksiyonu*(s, s')
            $s = s'$
        sonlandır
        Eğer $f(s) < f(En\dot{I}yi\dot{C}\ddot{o}z\ddot{u}m)$
            EnİyiÇözüm = s'
        sonlandır
    durdurma koşulu sağlanıncaya **kadar**
    en iyi çözümü **ver**
sonlandır

Şekil 13. Geniş Komşuluk Arama Algoritmasına Ait Sözde-Kod

2.5.2. Popülasyon Temelli Yöntemler

Popülasyon temelli yöntemler, bir çözüm yerine daha fazla sayıda çözümün bir araya gelerek oluşturduğu popülasyonlar ile arama uzayında arama yapar. Bu sınıftaki meta sezgiseller, genel olarak canlıların doğal yaşamdaki davranışlarını taklit ederek optimal ya da optimale yakın çözümleri elde etmeyi amaçlar. İzleyen alt başlıklarda bu sınıfa giren meta sezgisellerin karakteristik özellikleri ele alınmıştır.

2.5.2.1. Genetik algoritma

Genetik algoritma (genetic algorithms), 1975 yılında Holland tarafından literatüre tanıtılmıştır ve en çok bilinen evrimsel algoritmadır (Eiben ve Smith, 2015: 99). Genetik algoritma, biyolojik evrim ve doğal seçilim prensibine dayanan stokastik arama algoritmasıdır (Nastasi vd., 2018: 1546). Genetik algoritma, Darvin'in evrim teorisinden ilham alarak uyumu daha yüksek canlıların hayatta kalmasının ve bunların genlerinin benzetimini yapar (Mirjalili, 2019: 43).

Genetik algoritma; aday çözümlerin, bireylerin, sabit büyüklükte popülasyonunu sürdüren yinelemeli bir prosedürdür (Grefenstette, 1986: 122). Popülasyonu oluşturan çözümler, kromozom olarak ifade edilir (Kumar vd., 2010: 451). Kromozomlar, genlerden oluşur ve genler reel sayı, bit (ikil) veya bit dizisi şeklinde kodlanarak optimizasyon probleminin karar değişkenlerini temsil eder (Nastasi vd., 2018: 1546). Genetik algoritma, ayrı bir arama uzayı ve çözüm uzayı kullanır. Arama uzayı, kodlanmış çözümlerin (genotip) ya da genlerden oluşan kromozomların uzayıdır. Çözüm uzayı ise gerçek çözümlerin (fenotip) uzayıdır (Renner ve Ekárt, 2003: 710). Örneğin, herhangi bir probleme ait bir aday çözüm, bitlerle Şekil 14'te gösterildiği gibi kodlanmış olabilir.

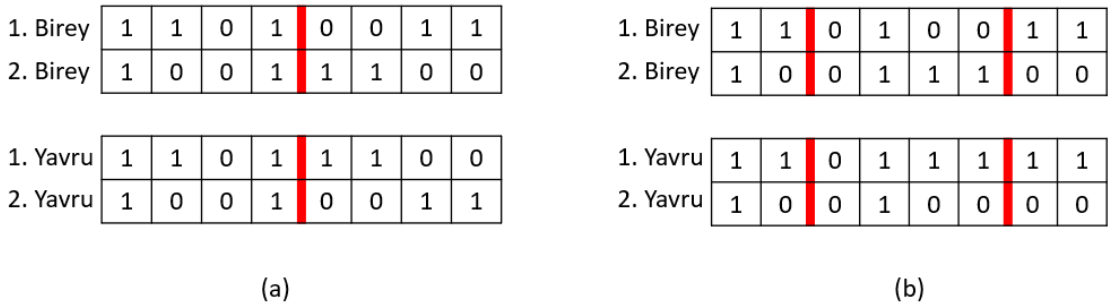
1	**1**	**0**	**0**	**1**	**0**	**1**

Şekil 14. Bitlerle Oluşturulmuş Bir Kodlama Örneği

Kodlama, bir problemin genetik algoritma ile çözülebilmesi için gerekli ilk ve kritik öneme sahip bir adımdır (Taşkın ve Emel, 2009: 33; Cheng vd.,1996: 985). Kodlama işleminin ardından kromozomların uyum değerlerini belirleyebilmek için uyum fonksiyonu tanımlanır ve kromozomların bu fonksiyona göre değerleri kullanılarak olasılıksal yaklaşımla bir sonraki nesil (generation) oluşturulur (Kumar vd., 2010: 451).

Doğal seçilim, genetik algoritma için ana ilham kaynağıdır. Doğada uyumu en yüksek olan bireyler, daha fazla beslenme ve çiftleşme şansına sahiptir. Bu da aynı türün bir sonraki neslinin üretiminde genlerinin daha fazla katkı sağlamasına neden olur (Mirjalili, 2019: 44). Bir popülasyondan gelen çözümler, turnuva veya rulet tekerleği seçimi gibi bir seçim operatörü aracılığıyla seçilir ve birleştirilir; burada seçilme olasılığı, bir şekilde kendi uyum değeri ile orantılıdır (Nastasi vd., 2018: 1547). Bu sebeple her nesilde nispeten iyi kromozomlar ürerken, nispeten kötü kromozomlar ölür. Bu aşama, aramayı en iyiye doğru yönlendirir (Liu, 2008: 3747). Turnuva seçimi, turnuva boyunca seçim baskısını (daha iyi bireylerin ayrıcalık dereceleri) *s* rakip arasında tutarak seçimi sağlar, burada *s* turnuva büyüklüğüdür. Turnuvanın kazananı *s* turnuva rakibi arasından en yüksek uyum değerine sahip bireydir. Kazanan birey, daha sonra eşleşme havuzuna konulur (Miller ve Goldberg, 1995: 195). Rulet tekeri seçimi ise, verilen bir dizinin seçilme olasılığının direkt olarak uyum fonksiyonu ile orantılı olduğu bir olasılık dağılımı kullanarak seçim yapar (Reeves ve Rowe, 2002: 31).

Genetik algoritmada her yinelemede daha iyi çözümler elde ederek optimal veya optimale yakın çözüme ulaşabilmek için kullanılan operatörlerden birisi, çaprazlamadır (crossover). Çaprazlamada daha iyi çözüm elde etmek amacıyla kromozomların parçaları bir araya getirilerek yeni yavrular oluşturulur (Liu, 2008: 3747). Çoğu çaprazlama operatöründe iki birey rastgele seçilir ve p_c çaprazlama operatörü ile bir araya getirilir. Yani, bir tek düze r sayısı oluşturulur ve eğer $r \leq p_c$ ise, seçilen iki birey çaprazlamaya maruz kalır (Sastry vd., 2014: 96). Çaprazlama operatörü içermiş olduğu bu rastgelelik ile arama uzayının yeni çözüm bölgelerinin keşfedilmesini sağlamaktadır (Huang ve Wang, 2006: 234). Şekil 15'te bir ve iki noktalı çaprazlama operatörlerinin kullanımına ilişkin örnek verilmiştir.



Şekil 15. Çaprazlama Operatörünün Kullanıma İlişkin Örnek

Şekil 15'te rastgele oluşturulmuş 2 birey, şeklin (a) kısmı seçilen bir noktalı operatör ile, şeklin (b) kısmı ise seçilen iki noktalı operatör ile çaprazlanmıştır. Öncelikle, şeklin (a) kısmında bir çaprazlama noktası olduğundan dördüncü genden sonraki genler değiştirilerek ebeveynlerden farklı iki yeni yavru elde edilmiştir. Şeklin (b) kısmında ise çaprazlama için iki nokta bulunmaktadır. Bu nedenle üçüncü genden başlayarak altıncı gene kadar olan genler değiştirilerek yeni yavrular oluşturulmuştur.

Genetik algoritmanın bir diğer operatörü ise mutasyondur. Mutasyon, yeni durumları keşfetmek için kullanılır ve algoritmanın yerel optimumdan kaçınmasını sağlar (Larrañaga vd., 1999: 131). Mutasyon, bireysel bir dizideki bir biti tam tersi değerle değiştirir (Grady vd., 2005: 263). Rastgele bir gene uygulanmış mutasyon örneği, Şekil 16'da gösterildiği gibidir.

Mutasyondan Önce	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
Mutasyondan Sonra	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**1**

Şekil 16. Bir Kromozomda Mutasyon Operatörünün Kullanımına Örnek

Çaprazlama ve mutasyon operatörleri kullanılarak elde edilen yavruların bir sonraki nesilde üreyebilmeleri için popülasyona girmeleri gerekmektedir. Elde edilen yavrular, önceki neslin tamamının yerini alarak (nesilsel yaklaşım) ya da mevcut popülasyonda yer alan uyum değeri düşük bireylerin yerini alarak (durağan durum yaklaşımı) popülasyona girebilir (Chu ve Beasley, 1998: 70). Bir başka yaklaşımla arama süresince elde edilen en iyi çözümlerin çaprazlama ya da mutasyon operatörlerinin kullanımına bağlı olarak kötüleşmemesi için bir sonraki nesle değişiklik yapılmadan aktarılmasına elitizm denilmektedir (Mirjalili, 2019: 47).

Kumar vd. (2010) tarafından çizilen çerçeveden uyarlanan genetik algoritmaya ait sözde-kod, Şekil 17'de verilmiştir.

```
prosedür Genetik Algoritma
    p0 = BaşlangıçPopülasyonOluştur()
    s0 = UyumDeğeriniHesapla(p0)
    tekrar et
        p = EnİyiBireyleriSeç(p0, s0)
        YeniBireyler = Çaprazlama(p)
        YeniBireyler = Mutasyon(YeniBireyler)
        s = UyumDeğeriniHesapla(YeniBireyler)
        p0 = YeniPopülasyonOluştur(p0, s0, YeniBireyler, s)
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 17. Genetik Algoritmaya Ait Sözde-Kod

2.5.2.2. Dağınık arama

Dağınık arama (scatter search), Glover (1977) tarafından tam sayılı programlama problemlerini çözebilmek için önerilmiş bir meta sezgiseldir. Dağınık arama, daha iyi çözümlere ulaşmak için çözümlerin konveks ya da konveks olmayan doğrusal kombinasyonlarını kullanan bir evrimsel/popülasyon temelli yöntemdir (Debels vd., 2006: 642). Dağınık arama, tabu arama ile yakın ilişki içindedir ve uyarlanabilir hafızanın özel biçimlerini birleştirerek aramada fayda sağlayabileceği ilkesini benimsemektedir (Saravanan vd., 2008: 1202).

Dağınık arama, yeni ve daha iyi çözümler oluşturmak için mevcut çözümlerin kombinasyonunu kullanarak problem bilgisinden faydalanmaktadır (Alaei ve Setak, 2017: 3300). Arama süresince hem amaç fonksiyonu değeri açısından hem de çözüm çeşitliliği açısından ziyaret edilen en iyi çözümler, referans kümesi (RefKüm) adı verilen bir kümede tutulur (Hanafi ve Wilbaut, 2008: 143-144). Dağınık arama, kaydedilen bu çözümleri kullandığı için yoğunlaştırma ile çeşitlendirme arasında denge kurmaktadır (González vd., 2017: 85). Dağınık arama metodolojisinin temeli olarak aşağıdaki prensipler görülebilir (Martí vd., 2018: 719):

i. "Optimal çözümlerin özellikleri hakkında faydalı bilgiler tipik olarak elit çözümlerin uygun bir koleksiyonunda bulunur.
ii. Bu bilgi, elit çözümlerin kombinasyonunda kullanılmaktadır.
iii. Kombinasyon işleminin amacı elit kümesine çeşitliliği (çözüm özellikleri açısından) ve kaliteyi (amaç fonksiyonu değeri açısından) dâhil etmektir.
iv. Arama mekanizmaları ve stratejileri kombinasyon ile sınırlı değildir, çözüm oluşturma ve iyileştirme kombinasyon yöntemlerini de içermektedir."

Literatürde yer alan dağınık arama uygulamaları, işleyiş olarak Glover (1998) tarafından önerilen 5 aşamalı şablona göre hareket etmektedir. Bu tezde de dağınık arama, bu şablon ışığında ele alınmıştır. Söz konusu 5 aşamanın detayları sırasıyla aşağıda verildiği gibidir:

i. *Çeşitlendirme Üreticisi:* Aramayı başlatmak için temel olan bir dizi farklı çözüm üretmek amacıyla kullanılır (Laguna, 2014: 119). Oluşturulan P popülasyonunun alt kümelerinden b büyüklüğünde $RefKüm$ oluşturulur (Pantrigo vd., 2012: 287).

ii. *İyileştirme Aşaması:* İyileştirme yönteminin amacı, çözümleri amaç fonksiyonuna göre iyileştirmeye çalışmaktır (Burke vd., 2010: 1672).

iii. *Referans Kümesi Güncelleme:* İlk referans kümesinin yapısı, P'den en iyi b_1 çözümün seçilmesi ile başlar. Bu çözümler, $RefKüm$'e eklenir ve P'den silinir. Daha sonra çeşitlilik ekleyebilmek için $RefKüm$'e olan uzaklıklara bağlık olarak b_2 çözüm eklenir. Burada $b_2 = b - b_1$'dir (Egea vd., 2007: 487).

iv. *Alt Küme Oluşturma Yöntemi:* En basit anlamda kombinasyon işlemine girmeleri için $RefKüm$'de verilen sırada çözümler seçilerek alt kümeler oluşturulmasıdır (González vd., 2017: 88).

v. *Çözüm Kombinasyon Yöntemi:* Yeni çözümler elde etmek için alt küme oluşturma yöntemi tarafından oluşturulan alt kümeleri kullanır (Laguna, 2014: 120). Rota birleştirme dağınık aramanın kombinasyon yöntemidir. İki veya daha fazla çözüm kombinasyona girdiğinde doğrudan yeni bir çözüm üretmek yerine, rota birleştirme komşuluk uzayında seçilen çözümler arasında ve dışında yollar oluşturur (Resende vd., 2010: 88). Rota birleştirme, yoğunlaştırma ve çeşitlendirme stratejilerini bir arama prosedürüne entegre etmek için bir çözüm birleştirme yaklaşımı olarak önerilmiştir (Xu ve Qu, 2012: 232). Bu yaklaşım, yüksek kaliteli çözümleri birbirine bağlayan yörüngeleri araştırır.

Glover (1998) tarafından verilen bilgilerden uyarlanan dağınık arama algoritmasına ait sözde-kod, Şekil 18'de gösterildiği gibidir.

```
prosedür Dağınık Arama
    ÇeşitlendirmeÜreticisi()
    İyileştir()
    RefKümOluştur()
    tekrar et
        AltKümeOluştur()
        KombinasyonÜret()
        İyileştir()
        RefKümGüncelle()
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 18. Dağınık Arama Algoritmasına Ait Sözde-Kod

2.5.2.3. Karınca kolonisi optimizasyonu

Karınca kolonisi optimizasyonu (ant colony optimization), kombinatoryal optimizasyon problemlerinde kullanılan meta sezgisellerden birisidir. Karınca kolonisi optimizasyonu algoritmalarının kökleri, 1990'lı yılların başlarında Dorigo ve çalışma arkadaşları tarafından yapılmış çalışmalara dayanmaktadır[2]. Karıncalar, hareket ettiklerinde zemine değişen miktarlarda feromon bırakarak gittikleri yolları işaretler (Dorigo vd., 1991: 1). Karınca kolonisi optimizasyonu, feromon isimli bu kimyasal

[2] Tarihçesi http://iridia.ulb.ac.be/~mdorigo/ACO/publications.html#SIXTH adresinde "ACO history" kısmında bulunabilir.

madde yoluyla birbirleriyle dolaylı yoldan iletişim kuran karınca, termit ya da arı gibi basit canlıların bir araya geldiklerinde bütünsel bir şekilde ortaya çıkardıkları kolektif zekâyı konu alan sürü zekâsının bir dalıdır (Boryczka vd., 2014: 587).

Karınca kolonisi optimizasyonunda yapay feromon izlerine ve çözülecek problemin girdi verisine dayanan sezgisel bilginin fonksiyonu olarak olasılıksal kararlar veren karıncalar, rastgeleleştirilmiş bir kurucu sezgisel kullanırlar. Algoritmanın uygulanması esnasında arama deneyimine göre feromon izlerinin uyarlanması, diğer kurucu sezgisellerden önemli bir farklılıktır (Dorigo ve Stützle, 2019: 312). Karınca kolonisi optimizasyonunda kullanılan yapay karıncalar, aynı zamanda doğal karıncalarda olmayan bazı ekstra özelliklere sahiptir. Her bir karınca, daha önceki hareketleri depolayan dahili bir hafızaya sahiptir ve hesaplanan yolların kalitesini arttırmak için yerel arama gibi bazı karakterlere sahip olabilir (Kashef ve Nezamabadi-pour, 2015: 272).

Karıncaların çözüm arama çabası esnasında yol seçimi, iki parametreye dayanan sezgisel bir prosedürdür: (i) Feromon değeri, karıncaların son zamanlarda izi seçme sayısının bir göstergesi iken; (ii) Sezgisel değer, probleme bağlı kalite ölçüsüdür (Martens vd., 2007: 651-652). Çözüm kurma başladıktan sonra her karınca, her bir çözüm bileşeni üzerinde feromon bırakarak (iz yoğunluğunu güncelleyerek) bir geri bildirim verir (Rajendran ve Ziegler, 2004: 428). Bu güncelleme, arama uzayının yüksek kaliteli çözümler içeren bölgelerine yoğunlaşmayı amaçlamaktadır. Özellikle, çözüm kalitesine bağlı olarak çözüm bileşenlerinin takviyesi, karınca kolonisi optimizasyon algoritmalarının önemli bir bileşenidir. Bu dolaylı olarak iyi çözümlerin, iyi çözüm bileşenlerinden oluştuğunu varsaymaktadır (Dorigo ve Blum, 2005: 244). Çözümü oluşturan bileşenler ise her çözüm oluşturma adımında olasılıksal bir şekilde feromon modeli olarak isimlendirilen bir modele göre seçilmektedir. Bu olasılıklar (geçiş olasılıkları olarak da isimlendirilir), Eşitlik (2.6)'da gösterildiği gibi hesaplanmaktadır (Blum, 2005: 360).

$$P(c_i|s) = \frac{[\tau_i]^{\alpha} * [\eta(c_i)]^{\beta}}{\sum_{c_j \in N(s)} [\tau_j]^{\alpha} * [\eta(c_j)]^{\beta}}, \forall c_i \in N(s) \tag{2.6}$$

Eşitlikte c_i çözüm bileşenini, τ_i çözüm bileşeni ile ilgili feromon değerini, $\eta(c_i)$ bileşenin sezgisel değerini, α ile β ise feromon bilgisi ve sezgisel bilgi ile ilgili pozitif parametreleri ve $N(s)$ ise eklenebilecek çözüm bileşenleri kümesini ifade eder. Başlangıç

feromon değeri, çoğu karınca kolonisi optimizasyon algoritması tarafından Eşitlik (2.7)'de gösterildiği gibi ayarlanmaktadır (Korytkowski vd., 2013: 223).

$$\tau_0 = \frac{1}{n * Z} \tag{2.7}$$

Burada Z, rastgele ya da bir sezgisel algoritmayla elde edilen amaç fonksiyonu değerini ifade ederken, n ise erken yakınsamadan kaçınmak için bazen paydadan silinen bir parametredir. Kısmi çözümlere uygun çözüm bileşenleri eklenerek tam bir aday çözüm elde edildikten sonra, isteğe bağlı şekilde bu çözüme yerel arama uygulanarak iyileştirme yapılabilir (Dorigo ve Blum, 2005: 250).

Çözüm elde edildikten sonra iyi çözümlere ait bileşenlerin daha çekici olabilmesi için global anlamda bir feromon güncellenmesine ihtiyaç duyulmaktadır. Bu güncelleme işlemi ile iyi bileşenlerin feromon miktarının artması sağlanırken, aramanın optimal olmayan bölgeye hızla yakınsamasından da feromon buharlaşması ile kaçınılmaktadır (Dorigo ve Stützle, 2019: 318-319). Bu işlemde seçilen iyi çözümle ilişkili unsurların feromon değeri $\Delta\tau$ kadar arttırılırken; feromon buharlaştırma ile bütün feromon değerleri Eşitlik (2.8)'de gösterildiği gibi azaltılır (Socha ve Dorigo, 2008: 1157).

$$\tau_i = \begin{cases} (1-\rho)\tau_i + \rho\Delta\tau\,, eğer\ \tau_i \in s_{ch} \\ (1-\rho)\tau_i, \quad diğer\ durumlarda \end{cases} \tag{2.8}$$

Eşitlik (2.8)'de $\rho \in (0,1]$ buharlaşma oranını, s_{ch} elde edilmiş en iyi çözümü ifade etmektedir. Feromon güncellemesi yapıldıktan sonra arka plan faaliyetleri (daemon actions) ile tek bir karınca tarafından gerçekleştirilemeyen merkezi eylemleri uygulamak için kullanılabilir (Blum, 2005: 361). Arka plan faaliyetleri, karıncaların davranışlarını gözlemlemek ve ilave feromon bilgilerini biriktirmek, bu şekilde karınca arama sürecini yerel olmayan bir perspektifle yönlendirmek için faydalı küresel bilgiler toplamak için kullanılabilir (Dorigo vd., 1999: 144). Dorigo vd. (1999) tarafından verilen bilgilerden uyarlanan karınca kolonisi optimizasyon algoritmasına ait sözde-kod, Şekil 19'da verildiği gibidir.

```
prosedür Karınca Koloni Optimizasyonu
    Tanımla()
    tekrar et
        KarıncaÇözümOluştur()
        YerelAramaUygula()  #Opsiyonel
        GlobalFeromonGüncelle()
        ArkaPlanİşlemler() #Opsiyonel
    durdurma koşulu sağlanıncaya kadar
    en iyi çözümü ver
sonlandır
```

Şekil 19. Karınca Kolonisi Optimizasyon Algoritmasına Ait Sözde-Kod

2.5.2.4. Parçacık sürü optimizasyonu

Parçacık sürü optimizasyonu (particle swarm optimization), Kennedy ve Eberhart (1995) tarafından sürekli optimizasyon problemleri için önerilmiş bir meta sezgiseldir. Parçacık sürü optimizasyonu, kuşların ya da balıkların avcılardan kaçınma, yiyecek ve eş arama gibi olaylarda sürü halinde hareket etmelerinden ilham almaktadır (Kennedy ve Eberhart, 1995: 1943). Geleneksel parçacık sürü optimizasyonu algoritmasında yiyecek arayan kuşların sosyal davranışlarının benzetimi yapılır. Algoritmada sürünün bireyleri, *parçacık* olarak isimlendirilmektedir ve her bir parçacık arama uzayındaki bir çözümü ifade etmektedir (Wang ve Liu, 2010: 216).

Parçacık sürü optimizasyonu çok boyutlu bir uzayda hareket eden noktaların yörüngelerini ayarlayarak arama yapar (Park vd., 2005: 34). Arama uzayında arama yapan parçacıklar, optimizasyon sürecinde hız ve konum şeklinde iki bileşene sahiptir. Yinelemeli sürecin her adımında parçacığın hızı, bireysel olarak parçacığın hızına, parçacık tarafından şimdiye kadar bulunulan en iyi pozisyona (kişisel etki) ve sürüdeki herhangi bir parçacığın bulunduğu en iyi pozisyona göre (sosyal etki) güncellenir (Roberge vd., 2013: 136). D boyutlu bir uzayda parçacığın t anındaki hızını $v_{id}(t)$, konumunu ise $x_{id}(t)$ göstermek üzere parçacık sürü optimizasyonu algoritmasının temel modeline ait hız ve konum güncelleme formülleri, Eşitlik (2.9) ve (2.10)'da verildiği gibidir (Wang vd., 2019: 5).

$$v_{id}(t+1) = v_{id}(t) + c_1 * rastgelesayı() * \left(p_{idEniyi}(t) - x_{id}(t)\right) + c_2 * Rastgelesayı() * \left(g_{idEniyi}(t) - x_{id}(t)\right) \quad (2.9)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t) \quad (2.10)$$

Eşitlik (2.9)'da eşitliğin sağ tarafında yer alan terimlerden $v_{id}(t)$ parçacığın önceki hızını, $\left(p_{idEniyi}(t) - x_{id}(t)\right)$ parçacığın önceki en iyi pozisyonunu yani kişisel etkiyi, $\left(g_{idEniyi}(t) - x_{id}(t)\right)$ ise bütün parçacıkların önceki en iyi pozisyonunu yani sosyal etkiyi ifade etmektedir. c_1, c_2 sabit sayıları (hızlandırma katsayıları da denilmektedir), $rastgelesayı()\ ve\ Rastgelesayı()$ ise [0,1] aralığından üretilen tek düze sayıları ifade etmektedir. Her parçacığın pozisyonu arama sürecinin başlangıcında rastgele olarak belirlenmektedir (Roberge vd., 2013: 136). v_{id}, parçacığın arama uzayının dışına çıkmaması için $[-V_{max}, V_{max}]$ aralığında sınırlanmıştır.

Hız sınırlaması ihtiyacının kaldırılması için Shi ve Eberhart (1998), Eşitlik (2.9)'da yer alan temel parçacık sürü optimizasyonu algoritmasına ω eylemsizlik katsayısını eklemiştir. Eylemsizlik ağırlığı ω, global ve yerel arama arasında denge kurma rolüne sahiptir. Bu ağırlık pozitif sabit ya da zamanın pozitif doğrusal ya da doğrusal olmayan fonksiyonu olabilir (Shi ve Eberhart, 1998: 70). Eylemsizlik ağırlığına benzer şekilde arama esnasında birkaç yineleme sonrasında çok büyük değerlere ulaşmayı (patlamayı) önlemek, yakınsamayı kontrol etmek için Clerc ve Kennedy (2002) tarafından sıkıştırma katsayısı önerilmiştir.

Eşitlik (2.9)'da ele alınan temel model global olarak bütün parçacıkların en iyi konumunu dikkate aldığı için global en iyi topolojiye (g_{eniyi}) sahiptir. Buna alternatif olarak Eberhart ve Kennedy (1995) tarafından sadece komşuluktaki en iyi çözümleri dikkate alan yerel en iyi topolojiye (l_{eniyi}) sahip algoritma da önerilmiştir. Yerel en iyi topolojiye sahip algoritmada sosyal bilgi oldukça yavaş fakat daha etkili bir şekilde yayılır (Pampara vd., 2005: 91).

Parçacık sürü optimizasyonu, her ne kadar sürekli optimizasyon problemi için önerilmiş olsa da algoritmanın literatürde kesikli problemlerde de kullanılabilmesi için çalışmalar yapılmıştır. Kennedy ve Eberhart (1997), olasılık değişiklikleri açısından yörünge, hız vb. tanımlayarak parçacık sürü optimizasyonu algoritmasını kesikli olarak

ifade etmişlerdir. Bu yaklaşımda v_{id}, x_{id} bitinin 1 değerini almasının olasılığını ifade eder. Bu şekilde p_{id} ve x_{id} {0,1} tamsayıları şeklinde ifade edilebilmektedir. Bunun için $s(v_{id})$ lojistik dönüşümü uygulanır (Kennedy ve Eberhart, 1997: 4105). [0, 1] aralığında üretilen rastgele bir sayı $s(v_{id})$ ile karşılaştırılır ve bu rastgele sayı daha küçük ise x_{id}, 1 değerine; aksi takdirde 0 değerine eşit olmaktadır. $s(v_{id})$ değeri ise Eşitlik (2.11)'de gösterildiği gibi hesaplanmaktadır (del Valle vd., 2008: 176).

$$s(v_{id}) = \frac{1}{1 + \exp(-v_{id})} \tag{2.11}$$

Jarboui vd. (2007) de kukla değişkenler ve en iyi çözümlere olan uzaklıklar kullanarak parçacık sürü optimizasyonu algoritmasının kesikli problemler için kullanılabilmesini sağlayan bir yaklaşım önermiştir. Ai ve Kachitvichyanukul (2009) ise çözümleri önce çözüm gösterimleri ismini verdikleri yaklaşımla reel sayı olarak kodlayıp daha sonra bu kodlar yardımıyla EZTDARP'yi çözmüştür. Eberhart ve Kennedy (1995) tarafından verilen bilgilerden uyarlanan parçacık sürü optimizasyonu algoritmasına ait sözde-kod, Şekil 20'de verilmiştir.

prosedür Parçacık Sürü Optimizasyonu
    RastgeleKonumBelirle()
    RastgeleHızBelirle()
    UyumDeğerlendir()
    tekrar et
        HızGüncelle()
        KonumGüncelle()
        UyumDeğerlendir()
    durdurma koşulu sağlananıncaya **kadar**
    en iyi çözümü **ver**
sonlandır

Şekil 20. Parçacık Sürü Optimizasyonu Algoritmasına Ait Sözde-Kod

ÜÇÜNCÜ BÖLÜM
HİBRİT YİNELEMELİ YEREL ARAMA ALGORİTMASI İLE EŞ ZAMANLI TOPLA DAĞIT ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜLMESİ

Bu tezde, EZTDARP için Yinelemeli Yerel Arama-Değişken Komşuluk İniş-Eşik Kabul meta sezgisellerinin İngilizce kısaltmaları kullanılarak ILS-RVND-TA ismi verilen bir hibrit meta sezgisel algoritma önerilmiştir. Önerilen algoritma, yinelemeli yerel arama çerçevesi altında değişken komşuluk iniş (variable neighborhood descent) ve eşik kabul (threshold acceptance) meta sezgisellerini birlikte kullanmaktadır. Değişken komşuluk iniş, Bölüm 2.5.1.4'te ele alınan değişken komşuluk arama meta sezgiselinin bir versiyonudur ve yerel arama aşamasında kullanılmıştır. Eşik kabul ise Bölüm 2.5.1.5'te ele alınan tavlama benzetimine benzeyen deterministik bir meta sezgiseldir ve algoritmanın kabul kriterinde kullanılmıştır.

Önerilen algoritma, Julia 1.6 programlama dili kullanılarak kodlanmıştır. Julia programlama dili, 2012 yılında kullanıma sunulmuş açık kaynak bir programlama dilidir. Julia programlama dilinin temel amacı, üretkenliği ve performansı bir araya getirmektir (Bezanson vd., 2015: 3). Bunun yanı sıra dilin sözdizimi (syntax), Python, R, MATLAB gibi popüler programlama dillerinde olduğu gibi kolaydır ve temel bilgi düzeyi ile C programlama diline yakın bir performans elde etmek mümkündür (The Julia Language, 2021: 3). Bu tezde ele alınan araç rotalama probleminin çözümü için yukarıda belirtilen avantajları sebebiyle Julia programlama dili tercih edilmiştir. Yazılan kodlar, Intel i7-7500U 2.70GHz işlemci ve 12 GB RAM kullanan ve Windows 10 işletim sistemine sahip bir bilgisayarda çalıştırılmıştır. Algoritmanın bileşenlerine ilişkin bilgiler, izleyen başlıklarda detaylı bir şekilde açıklanmıştır.

3.1. Önerilen Algoritma

Önerilen algoritma, yinelemeli yerel arama çatısı altında rastgele sıralamalı değişken komşuluk iniş ve eşik kabul meta sezgisel algoritmalarını kullanmaktadır. Yinelemeli yerel arama; başlangıç çözüm oluşturma, yerel arama, sarsım ve kabul kriteri bileşenlerinden oluşmaktadır. Bu doğrultuda algoritmanın ihtiyaç duyduğu başlangıç çözüm, en yakın komşuluk sezgiseli ile oluşturulmuştur. Yerel arama aşamasında rastgele sıralamalı değişken komşuluk iniş meta sezgiseli kullanılmıştır. Sarsım aşamasında

arama uzayının farklı bölgelerinde arama yapabilmek için çeşitli operatörler kullanılmıştır. Eşik kabul meta sezgiseli ise kabul kriteri bileşeninde algoritmaya dâhil edilmiştir. Şekil 21'de önerilen algoritmaya ait sözde-kod verilmiştir.

```
prosedür ILS-RVND-TA
    girdi MaxIt
    BaşlangıçÇözüm = BaşlangıçÇözümOluştur()
    İyileştirmeyen_Çözüm = 0
    Kabuledilmeyen_Çözüm = 0
    İyileştiren_Çözüm = 0
    Durdurma_Kriteri = MaxIt/2
    a1 = 1
    a2 = 1
    t = 1 + (a1 * a2)
    GeçerliÇözüm, EnİyiÇözüm = RVND(BaşlangıçÇözüm)
    for i = 1:MaxIt
        SÇözüm = Sarsım(GeçerliÇözüm)
        YeniÇözüm = RVND(SÇözüm)
        Eğer f(YeniÇözüm) < f(EnİyiÇözüm)
            EniyiÇözüm = YeniÇözüm
            İyileştirmeyen_Çözüm = 0
            Kabuledilmeyen_Çözüm = 0
            İyileştiren_Çözüm = İyileştiren_Çözüm + 1
        Değilse
            İyileştirmeyen_Çözüm = İyileştirmeyen_Çözüm + 1
            Kabuledilmeyen_Çözüm = Kabuledilmeyen_Çözüm + 1
        sonlandır
        Eğer f(YeniÇözüm) ≤ t * f(EnİyiÇözüm)
            GeçerliÇözüm = YeniÇözüm
            a1 = f(EnİyiÇözüm) / f(BaşlangıçÇözüm)
            a2 = İyileştiren_Çözüm / i
            t = 1 + (a1 * a2)
        sonlandır
        Eğer Kabuledilmeyen_Çözüm % 6 == 0
            t = t + (a1 * a2)
            Kabuledilmeyen_Çözüm = 0
        sonlandır
        Eğer İyileştirmeyen_Çözüm % 50 == 0
            GeçerliÇözüm = EnİyiÇözüm
        sonlandır
        Eğer İyileştirmeyen_Çözüm == Durdurma_Kriteri
                EnİyiÇözüm ver
        sonlandır
    sonlandır
    EnİyiÇözüm ver
sonlandır
```

Rastgele Sıralamalı Değişken Komşuluk İniş

Eşik Kabul

Yinelemeli Yerel Arama

Şekil 21. ILS-RVND-TA Algoritmasına Ait Sözde-Kod

Algoritma, *MaxIt* değerine ulaşılması ya da birbirini izleyen $MaxIt/2$ yineleme boyunca bulunan en iyi çözümün iyileştirilememesi durumlarında durdurulmaktadır. Buna ek olarak, birbirini izleyen her 50 iyileştirmeyen yinelemede geçerli çözümün değeri, o ana kadar bulunan en iyi çözüm değerine eşitlenir. Bunun amacı, aramanın bulunan en iyi çözümden aşırı uzaklaşmasını engellemektir.

3.1.1. Başlangıç Çözümün Oluşturulması

Başlangıç çözüm, meta sezgisellerle arama uzayında arama yapılırken kaliteli sonuçlara ulaşılmasında önemli bir rol oynar. Optimal ya da optimal çözüme yakın bir noktadan aramaya başlamanın, kullanılan algoritmanın ihtiyaç duyacağı hesaplama zamanını azaltması beklenmektedir. Bu noktadan hareketle literatürde başlangıç çözüm oluşturmak için Bölüm 1.3.2'de de bahsedildiği gibi geliştirilen çeşitli sezgisel yöntemler çalışmalarda kullanılmaktadır. Bazı çalışmalarda ise problemin ve kullanılan meta sezgiselin özelliklerine göre rastgele oluşturulan başlangıç çözümler de kullanılabilmektedir.

Bu tezde başlangıç çözüm, literatürde de sıklıkla kullanılan en yakın komşuluk sezgiseli ile oluşturulmuştur. En yakın komşuluk sezgiseli, başlangıç çözümü şu şekilde oluşturmaktadır (Gajpal ve Abad, 2009: 3217): Sezgisel, çözüm oluştururken öncelikle depoya en yakın müşteriyi seçer. Daha sonra bu müşteriye en yakın müşteri, rotaya eklenir. Rotaya eklemeler, benzer şekilde çözümün uygunluğunu bozmayacak şekilde devam eder. Bir rotaya eklenecek müşteri, rotanın uygunluğunu bozuyorsa işlem sonlandırılarak yeni bir rota oluşturulmaya başlanır. Bu işlemler, bütün müşteriler rotalara eklenınceye kadar devam eder.

3.1.2. Yerel Arama Aşaması

Yerel arama aşamasında mevcut çözümden daha iyi çözümler bulma umuduyla bu çözümün komşulukları taranır. Yerel arama, bir meta sezgisel algoritmanın performansını etkileyen belki de en önemli bileşendir. Arama uzayının ümit veren bölgelerini yoğun bir şekilde arama becerisine sahip algoritmalar, kısa sürelerde kaliteli sonuçlara ulaşabilmektedir. Bu noktadan hareketle, literatürde yerel arama için geliştirilmiş çeşitli meta sezgiseller bulunmaktadır.

Değişken komşuluk iniş, Bölüm 2.5.1.4'te bahsedilen değişken komşuluk arama meta sezgiselinin bir türüdür (Hansen ve Mladenović, 1999: 435). Bu türler, komşuluk yapısının kullanılma şekline göre ortaya çıkmaktadır. Değişken komşuluk inişte birden fazla komşuluk yapısı kullanılır ve bu komşuluk yapılarının sırası, deterministik bir sırada değiştirilir (Hansen ve Mladenović, 2003: 147). Değişken komşuluk iniş, hem kullanımının kolay olması hem de arama uzayında detaylı bir şekilde arama yapabilmesi sebebiyle literatürde yaygın bir şekilde kullanılmaktadır.

Bu tezde de benzer sebeplerle rotalar arası ve rota içi komşuluk yapıları aranırken rastgele sıralamalı değişken komşuluk iniş kullanılmıştır. Algoritmanın çalışması esnasında yerel arama fonksiyonu her çağrıldığında, detayları izleyen başlıklarda verilen komşuluk yapıları için rastgele bir sıralama belirlenmektedir ve fonksiyon durana kadar bu sıralama değişmemektedir. Komşuluklarda arama yapılırken en iyi kabul edilebilir komşu çözümler dikkate alınmıştır. Bu bilgilere göre yerel arama aşamasında kullanılan rastgele sıralamalı değişken komşuluk iniş algoritmasının sözde-kodu, Şekil 22'de verildiği gibidir.

```
prosedür RVND
    girdi GeçerliÇözüm
    kl = RotalarArasıKomşulukYapılarınıRastgeleSırala(1…kmax)
    s = GeçerliÇözüm
    for i = 1:kmax
        s' = N_{i∈kl}(s)'deEnİyiÇözümüBul
        Eğer f(s') < f(s)
            s = s'
            s = RotaİçiKomşulukYapılarınıAra(s)
            i = 1
        Değilse
            i = i+1
        sonlandır
    sonlandır
    en iyi çözümü ver
sonlandır
```

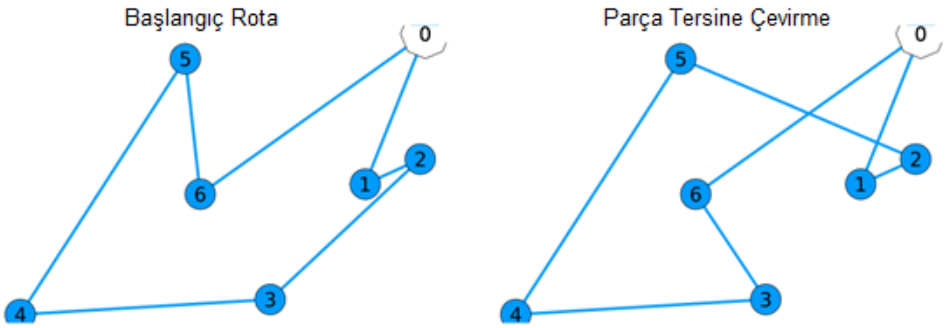
Şekil 22. Yerel Arama Aşamasına Ait Sözde-Kod

Oluşturulan başlangıç çözümü iyileştirmek için yerel arama sürecinde kullanılacak prosedürler, iki gruba ayrılmaktadır: (i) Rota içi komşuluk yapıları, (ii) rotalar arası komşuluk yapıları. Rota içi komşuluk yapıları, çeşitli şekillerde bir rotadaki müşterilerin pozisyonlarını değiştirerek mevcut tura ait komşu çözümleri elde etmeyi amaçlar. Rotalar arası komşuluk yapıları ise, farklı iki rota arasında müşteri değişimi yaparak komşu çözümleri elde etmeyi amaçlar. Bu iki komşuluk yapısı, genel olarak daha iyi çözümleri elde etmeyi ve arama uzayının umut veren bölgelerinde daha yoğun bir arama yapmayı hedeflemektedir. Bu tezde, rota içi komşuluk yapıları için parça tersine çevirme, Or-opt, 2-opt, değişim, ekleme yapıları kullanılmıştır. Rotalar arası komşuluk yapıları için ise değiş tokuş (1-1), öteleme (1-0), değiş tokuş (2-2), öteleme (2-0), değiş tokuş (2-1) ve çaprazlama operatörleri kullanılmıştır. Komşu çözümler arasında arama yapılırken yalnızca uygun çözümler dikkate alınmıştır. Bu nedenle yerel arama süreci, Subramanian vd. (2010: 1902)'nin belirttikleri yaklaşımla depodan ayrılırken ya da depoya dönerken araç kapasitesini aşan rotalar elimine edilerek arama hızlandırılmıştır. Buna ek olarak, Nagy ve Salhi (2005: 131)'nin belirttiği yaklaşımla yük dalgalanması sebebiyle uygun olmayan rotalar ters çevrilerek, ortaya çıkan yeni rotaların uygun olup olmadıkları incelenmiştir. Tersine çevirme işlemi sonucunda uygun hale gelen rotalar dikkate alınmıştır.

3.1.2.1. Rota içi komşuluk yapıları

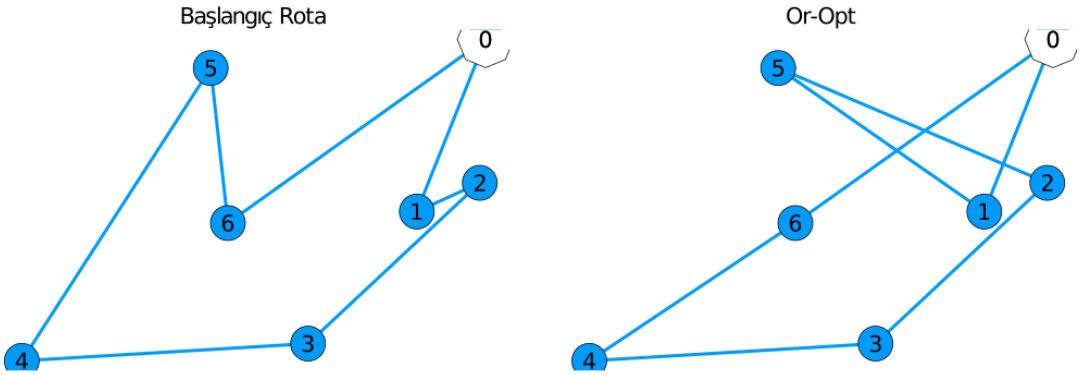
Tezde kullanılan rota içi komşuluk yapılarının çalışma mantığı, aşağıda verildiği gibidir. Rastgele altı müşteriden oluşturulan bir rota üzerinde bu operatörlerin yaptığı değişiklikler şekiller yardımıyla örneklendirilmiştir.

i. ***Parça tersine çevirme (Segment reverse):*** Bir rota içerisinde iki müşteri seçilir. Sonrasında bu iki müşteri ve arasında kalan parça tersine çevrilir (Li, vd., 2015: 3555). Şekil 23'te rotada 3 ve 5 numaralı müşteriler arasında kalan parçanın tersine çevrilmesiyle elde edilen komşu çözüm gösterilmiştir. Başlangıç rota 0-1-2-3-4-5-6-0 iken operatörün uygulanması sonucunda yeni rota 0-1-2-5-4-3-6-0 olarak elde edilmiştir.



Şekil 23. Parça Tersine Çevirme Komşuluk Yapısının Uygulama Örneği

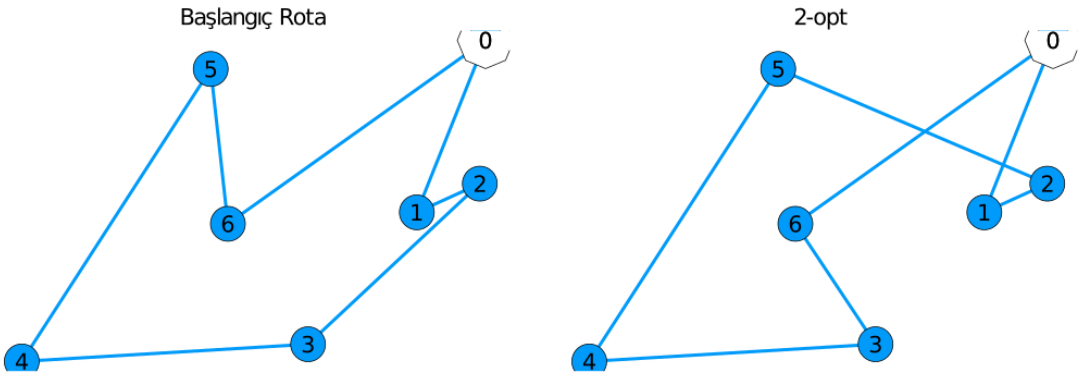
ii. ***Or-opt:*** Or tarafından 1976 yılında önerilen algoritmada bir, iki ya da üç ardışık müşteri, farklı iki müşteri arasına ötelenir (Savelsbergh, 1992: 147). Rota içerisindeki farklı uzunluktaki (bir, iki ya da üç) müşteri alt kümesi, rotanın ilerleyen kısımlarındaki noktalar arasına ötelenerek rotadaki değişim incelenir. Daha iyi bir rotanın ortaya çıkması durumunda bu rota üzerinden işlemlere devam edilir. Daha fazla iyileşme mümkün olmadığında algoritma durur. Şekil 24'te rotada yer alan 2,3 ve 4 numaralı müşterilerin, 5 numaralı müşterinin ardına ötelendiği bir komşu çözümün elde edilişi gösterilmiştir. Başlangıç rota 0-1-2-3-4-5-6-0 iken operatörün uygulanması sonucunda yeni rota 0-1-5-2-3-4-6-0 olarak elde edilmiştir.



Şekil 24. Or-Opt Komşuluk Yapısının Uygulama Örneği

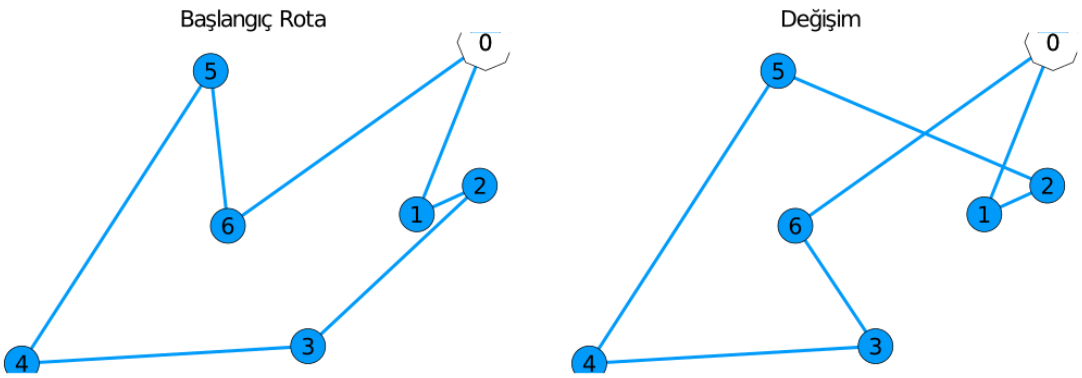
iii. ***2-opt:*** Croes (1958) tarafından gezgin satıcı problemleri için önerilmiş olsa da araç rotalama problemlerinde de rota içi komşuluk yapılarında sıklıkla kullanılan bir algoritmadır. Rota içerisinden 2 müşteri seçilerek, bu iki müşteri arasında kalan müşteriler (seçilen müşteriler dâhil değildir) tersine çevrilerek rota mesafesinde meydana gelen değişiklikler incelenir. Bir değişiklik sonucunda daha iyi bir rota ortaya çıkarsa bu rota üzerinden devam edilir. Süreç daha fazla iyileşme olmayıncaya kadar devam eder. Şekil 25'te 2 ve 6 numaralı müşterilerin

arasında kalan müşterilerin ters çevrilmesi ile elde edilen komşu çözüm gösterilmiştir. Başlangıç rota 0-1-2-3-4-5-6-0 iken operatörün uygulanması sonucunda yeni rota 0-1-2-5-4-3-6-0 olarak elde edilmiştir.



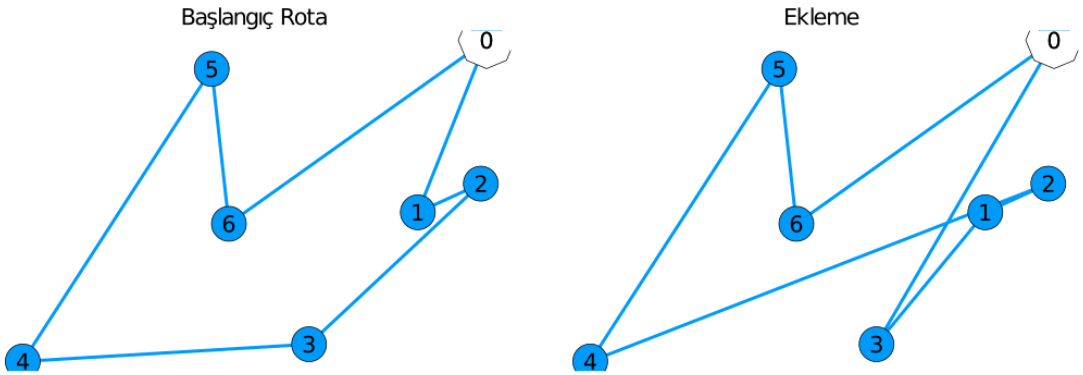
Şekil 25. 2-Opt Komşuluk Yapısının Uygulama Örneği

iv. ***Değişim (Exchange):*** İki nokta seçilerek bunların rota içerisindeki pozisyonları değiştirilir. Şekil 26'da 3 ve 5 numaralı müşterilerin rotadaki pozisyonlarının değiştirilmesi ile elde edilen komşu çözüm gösterilmiştir. Başlangıç rota 0-1-2-3-4-5-6-0 iken operatörün uygulanması sonucunda yeni rota 0-1-2-5-4-3-6-0 olarak elde edilmiştir.



Şekil 26. Değişim Komşuluk Yapısının Uygulama Örneği

v. ***Ekleme (Insertion):*** Bir nokta seçilerek rota içerisinde farklı bir pozisyona taşınır. Şekil 27'de 3 numaralı müşterinin rotada yeni bir pozisyona (birinci pozisyon) eklenmesi ile elde edilen komşu çözüm gösterilmiştir. Başlangıç rota 0-1-2-3-4-5-6-0 iken operatörün uygulanması sonucunda yeni rota 0-3-1-2-4-5-6-0 olarak elde edilmiştir.

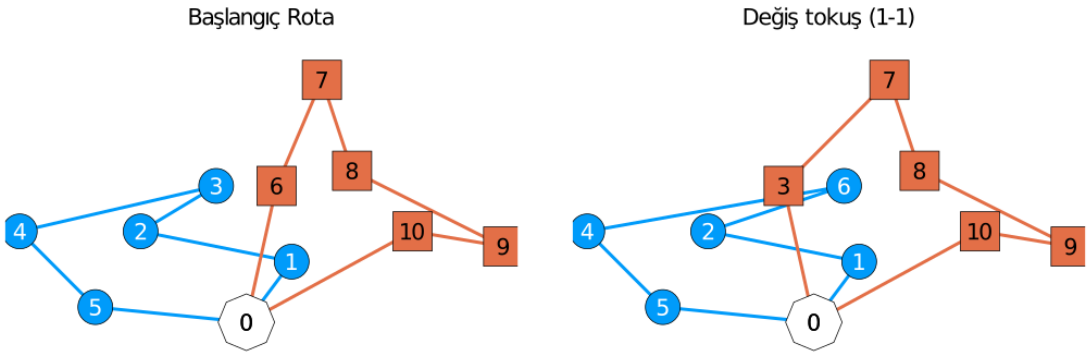


Şekil 27. Ekleme Komşuluk Yapısının Uygulama Örneği

3.1.2.2. Rotalar arası komşuluk yapıları

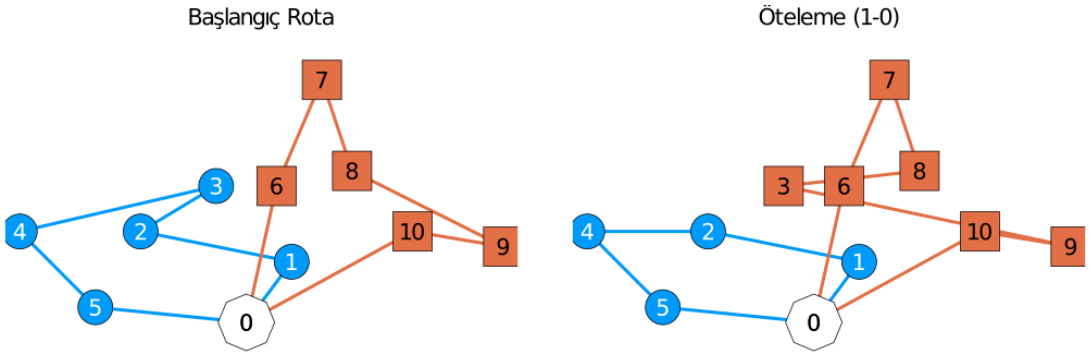
Tezde kullanılan rotalar arası komşuluk yapılarının çalışma mantığı, aşağıda verildiği gibidir. Bu operatörlerden çaprazlama dışında kalan diğer değiş tokuş ve öteleme operatörleri, Osman (1993) tarafından önerilen λ yer değiştirmeye dayanmaktadır. Söz konusu operatörlerde değişimi sınırlayabilmek için λ değeri, genellikle 1 ya da 2 olarak seçilmektedir. $\lambda_1 \leq \lambda$ ve $\lambda_2 \leq \lambda$ olmak üzere λ_1, birinci rotadan ikinci rotaya geçen müşteriyi, λ_2 ise ikinci rotadan birinci rotaya geçen müşteriyi ifade etmektedir (Cordeau ve Laporte, 2005: 147-148). Bu yapıların uygulanması sonucunda rotalardaki değişimler, rastgele beşer müşteriden oluşturulan iki rota üzerinden aşağıdaki şekillerde örneklendirilmiştir.

i. ***Değiş tokuş (Swap) (1-1):*** İki farklı rotadan seçilen birer müşterinin bulundukları rotalar değiştirilir. Şekil 28'de birinci rotadan 3 numaralı müşteri ile ikinci rotadan 6 numaralı müşteri karşılıklı olarak yer değiştirmiştir. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-2-6-4-5-0 ve 0-3-7-8-9-10-0 olarak elde edilmiştir.



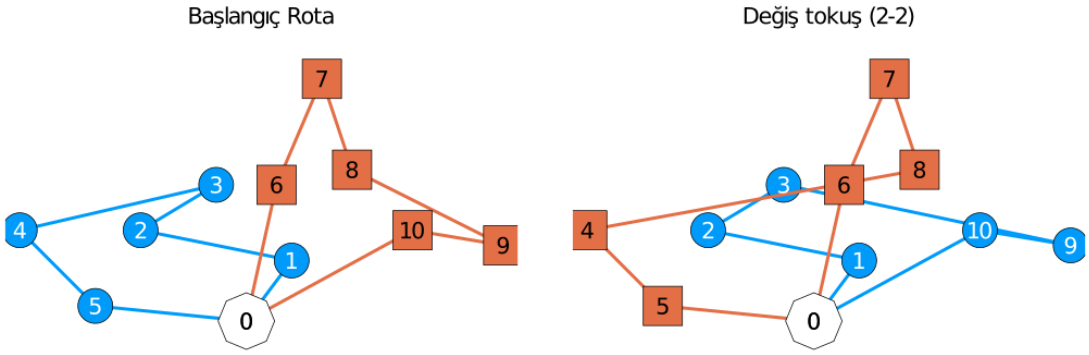
Şekil 28. Değiş Tokuş (1-1) Komşuluk Yapısının Uygulama Örneği

ii. ***Öteleme (Shift) (1-0):*** Birinci rotadan seçilen bir müşteri, ikinci rotaya geçirilir. Şekil 29'da birinci rotadan 3 numaralı müşteri ikinci rotaya aktarılmıştır. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-2-4-5-0 ve 0-6-7-8-3-9-10-0 olarak elde edilmiştir.



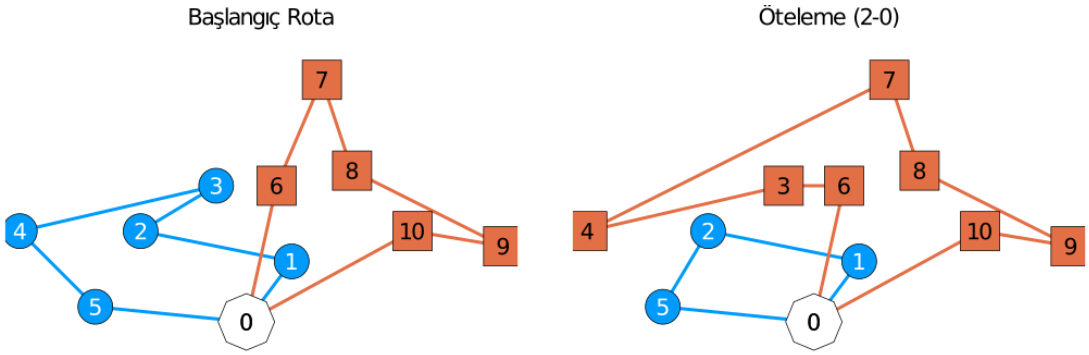
Şekil 29. Öteleme (1-0) Komşuluk Yapısının Uygulama Örneği

iii. ***Değiş tokuş (Swap) (2-2):*** İki farklı rotadan seçilen ardışık iki müşterinin yerleri değiştirilir. Şekil 30'da birinci rotada ardışık olan müşterilerden 4 ve 5 numaralı müşteriler ikinci rotaya, ikinci rotada ardışık olan müşterilerden 9 ve 10 numaralı müşteriler ise birinci rotaya geçirilmiştir. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-2-3-9-10-0 ve 0-6-7-8-4-5-0 olarak elde edilmiştir.



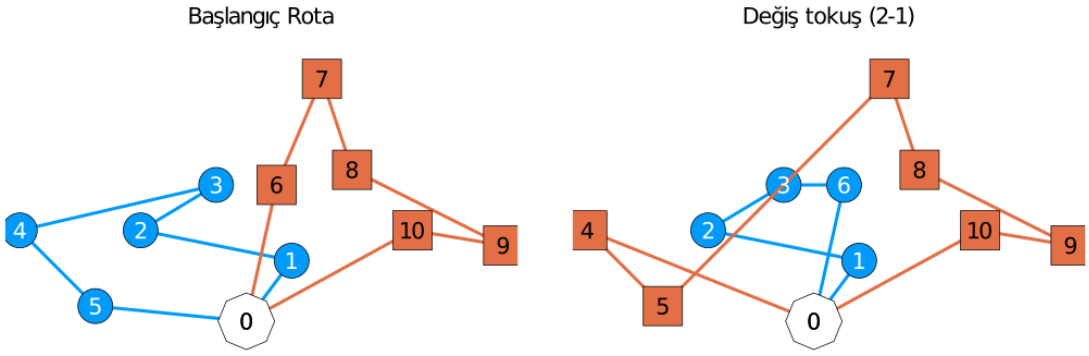
Şekil 30. Değiş Tokuş (2-2) Komşuluk Yapısının Uygulama Örneği

iv. ***Öteleme (Shift) (2-0):*** Birinci rotadan seçilen ardışık iki müşteri ikinci rotaya geçirilir. Şekil 31'de birinci rotada yan yana olan 3 ve 4 numaralı müşteriler ikinci rotaya geçirilmiştir. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-2-5-0 ve 0-6-3-4-7-8-9-10-0 olarak elde edilmiştir.



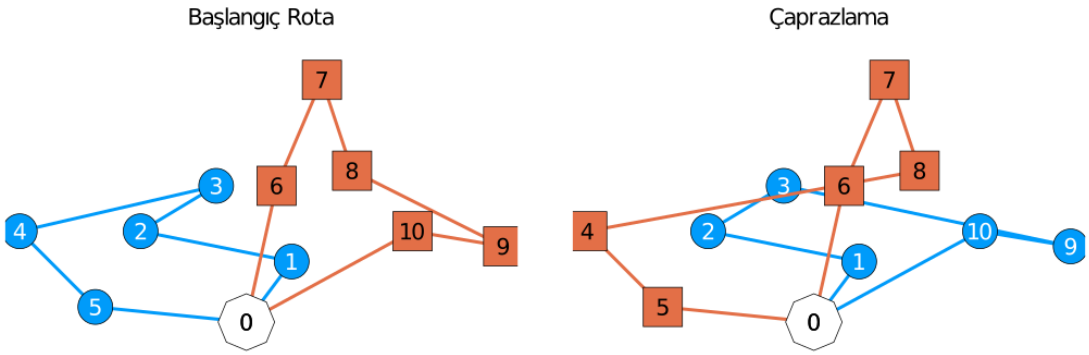
Şekil 31. Öteleme (2-0) Komşuluk Yapısının Uygulama Örneği

v. ***Değiş tokuş (Swap) (2-1):*** Birinci rotadan seçilen ardışık iki müşteri ile ikinci rotadan seçilen bir müşterinin yerleri değiştirilir. Şekil 32'de birinci rotada ardışık olan 4 ve 5 numaralı müşteriler ile ikinci rotadan 6 numaralı müşterilerin rotaları değiştirilmiştir. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-2-3-6-0 ve 0-4-5-7-8-9-10-0 olarak elde edilmiştir.



Şekil 32. Değiş Tokuş (2-1) Komşuluk Yapısının Uygulama Örneği

vi. ***Çaprazlama (Crossover):*** Seçilen her iki rota, iki parçaya bölünür. Birinci rotanın ilk parçası ile ikinci rotanın ikinci kısmı, ikinci rotanın birinci kısmı ile birinci rotanın ikinci kısmı birleştirilerek yeni iki rota oluşturulur. Şekil 33'te birinci rota, [1,2,3] ve [4,5] şeklinde iki parçaya bölünmüş, ikinci rota da ise [6,7,8] ve [9,10] şeklinde iki parçaya bölünmüştür ve bu parçaların birleştirilmesi ile yeni rotalar oluşturulmuştur. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-2-3-9-10-0 ve 0-6-7-8-4-5-0 olarak elde edilmiştir.



Şekil 33. Çaprazlama Komşuluk Yapısının Uygulama Örneği

3.1.3. Sarsım

Sarsım, Bölüm 2.5.1'de de bahsedildiği gibi arama sürecinin yerel optimale takılıp kalmasını engellemek için kullanılan bir mekanizmadır. Yerel arama aşamasında komşuluk yapıları ile umut verici bölgeler daha yoğun bir şekilde incelenirken sarsım aşamasında ise arama uzayının başka bir bölgesine geçilerek arama sürecine çeşitlilik katılır. Sayıca birden fazla ve yerel arama aşamasında kullanılanlardan farklı tipte operatörlerin rastgele seçilerek kullanılması, sarsım aşamasının faydalı olabilmesi için gerekli prensiplerdendir (Brandão, 2018: 150).

Bir minimizasyon probleminde temel amaç, toplam maliyetin en küçük olmasıdır. Bu nedenle çözümde yüksek maliyetli rotaların olması, istenilmeyen bir özelliktir. Buna ek olarak bir rotanın uygunluğu, rotanın maksimum yük miktarına bağlıdır. Maksimum yük miktarı yüksek olan rotalara yeni müşteri eklenmesi durumunda, büyük olasılıkla araç kapasitesi aşılacağı için uygun olmayan rotalar ortaya çıkacaktır. Yerel arama aşamasında bu durum daha iyi çözümlere ulaşmada bir engel oluşturabilir. Bu nedenle sarsım aşamasında rotalar seçilirken bu iki özelliğe bağlı olarak rulet tekeri ile seçim yapılmaktadır.

Rulet tekeri ile seçim yapabilmek için gerekli olasılık değerleri şu şekilde hesaplanmaktadır: Maliyet açısından öncelikle rotalarda iki nokta arasındaki ortalama uzaklık (depo-müşteri ve müşteri-müşteri) hesaplanmaktadır. Daha sonrasında ise rotalar arasındaki farklılıkları ortadan kaldırmak adına ortalama uzaklıklar, Eşitlik (3.1) kullanılarak normalize edilmektedir. Bu işlem sonucunda bütün değerler $(0, 1)$ aralığındaki değerlere dönüştürülmektedir. Maksimum yük açısından ise benzer olarak öncelikle her bir rotanın maksimum yük miktarı hesaplanarak normalize edilmektedir.

Normalize edilmiş ortalama uzaklıklar ve normalize edilmiş maksimum yük değerlerini ağırlıklandırarak rotalar için nihai bir skor elde etmek gerekmektedir. Söz konusu iki kritere ait ağırlık değerlerinin ne olması gerektiği ise bilinmemektedir. Bu nedenle rotaların içerdikleri bilgi durumuna göre objektif bir şekilde ağırlık belirleyebilmek için Shannon (1948) tarafından geliştirilen entropi kavramından faydalanılmıştır. Entropi kullanılarak ağırlık belirleme özellikle çok kriterli karar verme alanında yaygın bir şekilde kullanılmaktadır (Lotfi ve Fallahnejad, 2010; Shemshadi vd., 2011; Wu vd., 2011; Hafezalkotob ve Hafezalkotob, 2016; Yazdani vd., 2020; Gupta ve Kumar, 2021). Ağırlıklar, Eşitlik (3.2) – (3.3)'te gösterildiği gibi elde edilir (Yazdani vd., 2020: 37):

$$p_{ij} = \frac{x_{ij}}{\sum x_{ij}} \tag{3.1}$$

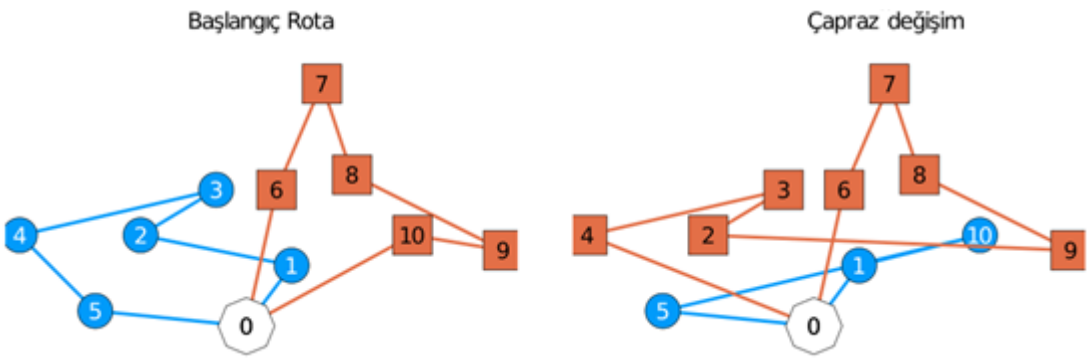
$$E_j = -k \sum_{i=1}^{m} p_{ij} * \ln p_{ij}\,, j = 1,2 \tag{3.2}$$

$$w_j = \frac{1 - E_j}{\sum_{j=1}^{2}(1 - E_j)} \tag{3.3}$$

Eşitlik (3.2)'de yer alan k bir sabittir ve $k = 1/\ln m$ ile hesaplanır, m ise rota sayısını göstermektedir. Ağırlıklandırılmış skorların kümülatif değeri hesaplanarak her bir rotanın seçilme olasılığı elde edilmektedir. Polat vd. (2015), yalnızca rotaların ağırlık merkezini kullanarak çalışan bir rulet tekeri tekniği kullanmıştır; ancak bu tezde kullanılan teknik, farklı kriterler kullanması ve kriterlerin önem derecesini rota bilgisine göre belirlemesi nedeniyle farklılaşmaktadır.

Bu tezde sarsım aşamasında kullanılan operatörler aşağıda verildiği gibidir. Her bir operatör, uygun çözüm elde edebilmek için maksimum 30 deneme yapmaktadır. Bu deneme sayısı, operatörlerin çeşitli deneme sayıları kullanıldığında uygun çözüm üretebilme performanslarına göre belirlenmiştir. Yapılan incelemelerde daha düşük sayıda deneme yapıldığında genel olarak elde edilen uygun çözüm sayının azaldığı; daha yüksek sayıda deneme yapıldığında ise genel olarak elde edilen uygun çözüm sayısının arttığı ve buna bağlı olarak işlem süresinin de arttığı görülmüştür. Önerilen algoritma, sarsım fonksiyonunu her çağırdığında rastgele bir operatör seçilir. Bu operatör, seçilen rotalar üzerinde bir değişiklik yaptıysa süreç sonlandırılmaktadır; aksi taktirde rastgele seçilen farklı bir operatörle bütün operatörler kullanılıncaya kadar tekrar denenir. Ortaya çıkan yeni rota, sadece uygunluk açısından değerlendirilir.

i. ***Çapraz değişim (Cross exchange):*** Taillard vd. (1997) tarafından önerilen bu operatör, daha çok yerel arama için kullanılsa da Chen vd. (2010) tarafından sarsım için kullanılmıştır. Çapraz değişim, şu şekilde çalışmaktadır: Rastgele iki rota ve her rotadan birer rastgele başlangıç noktası seçilir. Daha sonra, [2,4] aralığından rastgele üretilen tam sayı ile müşteri bölüm uzunluğu belirlenir. Başlangıç noktasından rotanın sonuna kadar olan müşteri sayısı, bölüm uzunluğundan az ise bu aralıktaki bütün müşteriler sarsım sürecine dâhil edilir (Chen vd., 2010: 1622). Şekil 34'te birinci rotadan 2, 3 ve 4 numaralı müşterilerden oluşan bir bölümün, ikinci rotadan 10 numaralı müşteriden oluşan bir bölümle çaprazlandığı bir mekanizmanın uygulama örneği verilmiştir. Başlangıçta birinci rota 0-1-2-3-4-5-0, ikinci rota 0-6-7-8-9-10-0 iken operatörün uygulanması sonucunda yeni rotalar 0-1-10-5-0 ve 0-6-7-8-9-2-3-4-0 olarak elde edilmiştir.



Şekil 34. Çapraz Değişim Sarsım Mekanizmasının Uygulama Örneği

ii. ***Üçlü çapraz değişim (Triple cross exchange):*** Çapraz değişim operatörünün rastgele seçilen üç farklı rotaya uygulanmasıdır. Öncelikle birinci rota ile ikinci rota çapraz değişim operatörü ile sarsım işlemine tabi tutulur. Daha sonrasında değiştirilen birinci rota ile üçüncü rota tekrar aynı operatör ile sarsım işlemine tabi tutulur. Polat vd. (2015)'nin geliştirdiği üçlü çapraz (triple cross) operatörü ile benzer bir mantığa sahiptir.

iii. ***k-Öteleme (k-Shift):*** İki rota arasında k müşterinin yer değiştirmesidir (Penna vd., 2013: 213). Rastgele iki rota seçilir. Öncelikle ilk rota olan r_1'de ötelenecek ilk müşterinin pozisyonu rastgele seçilir. Daha sonrasında ise ötelenecek ardışık müşteri sayısı $k \in [5,7]$ seçilir. Seçilen müşteriler, r_2'de rastgele seçilen bir pozisyona ilk müşteriden başlanarak sırayla eklenir.

iv. ***Rota gevşetme (Route relaxion):*** Rota gevşetme, bu tez için geliştirilen yeni bir operatördür. Operatörün temel amacı, noktalar arası ortalama uzaklık ve maksimum yüke göre rastgele seçilen rotalardan yine rastgele müşteriler seçilerek oluşturulan boş bir rotaya öteleme yapmaktır. Böylelikle, müşteri sayısı azalan rotaların yerel arama aşamasında iyileştirilebileceği düşünülmektedir. Oluşturulan boş rotaya eklemeler, rotanın maksimum yükünün araç kapasitesinin %40'ını aşana kadar yapılmaktadır. Bu oran aşıldığında operatör durarak rotaların en son halini vermektedir. Ekleme yapılabilecek kapasite miktarı çok düşük olduğunda (%5, %10 gibi) algoritmanın yerel optimumdan kaçamadığı, çok yüksek olduğunda (%90, %95) ise sarsım işleminden önceki rotalardan çok da farklı olmayan yeni bir rota ortaya çıktığı görüldüğü için dengeli bir oran seçilmiştir.

v. ***Değişim (Exchange) (m, n):*** Rastgele seçilen iki rota arasında ardışık (m, n) müşterinin karşılıklı olarak yer değiştirmesidir (Jun ve Kim, 2012: 5643; Kalayci ve Kaya, 2016: 169). Burada $m, n \in [0,3]$'tür ancak, m değişkeni 0 değerini alması durumunda n değişkeninin de 0 değerini alması engellenmektedir. Sarsım aşamasına ait sözde-kod, Şekil 35'te gösterildiği gibidir.

```
prosedür Sarsım
    girdi Çözüm
    s = Çözüm
    i = 1
    tekrar et
        po = RastgeleSarsımOperatörüSeç
        r1...rn = RotaSeçici()
        s' = SarsımUygula(s, po, r1...rn)
        Eğer s' ≠ s
            s = s'
            sonlandır
        Değilse
            i = i + 1
        sonlandır
    i > |Operatör| oluncaya kadar
    s'i ver
sonlandır
```

Şekil 35. Sarsım Aşamasına Ait Sözde-Kod

3.1.4. Kabul Kriteri

Yinelemeli yerel arama algoritmasının bileşenlerinden bir diğeri de kabul kriteridir. Bir yinelemede sarsım ve yerel arama aşamalarından sonra elde edilen çözümün bir sonraki yineleme için geçerli çözüm olup olmayacağı, kabul kriteri tarafından belirlenmektedir. Kabul kriteri seçilirken iki alternatif söz konusudur. Birinci alternatifte, elde edilen çözüm ile geçerli çözüm karşılaştırılır. Yalnızca çözüm değerinde bir iyileşme söz konusu ise yeni çözüm, bir sonraki yineleme için geçerli çözüm olarak kabul edilir. İkinci alternatifte ise daha esnek bir yaklaşım söz konusudur. Elde edilen yeni çözüm, geçerli çözümden daha kötü olsa dahi kabul edilebilmektedir. Daha kötü çözümün kabul edilmesinin temel fikri, çözüm daha kötü bir amaç fonksiyonu değerine sahip olsa bile arama uzayı hakkında taşıdığı bilgiden faydalanmaktır. Bu bakış açısı ile tavlama benzetimi, kabul kriterinde sıklıkla kullanılmaktadır.

Eşik kabul, iyileştirmeyen çözümleri kabul edebilme yönüyle tavlama benzetimine benzeyen, fakat söz konusu iyileştirmeyen çözümleri deterministik bir şekilde kabul ettiği için tavlama benzetiminden farklılaşan bir meta sezgiseldir (Dueck ve Scheuer, 1990: 163). Arama sürecinde bütün kötü çözümleri kabul etmek, arama uzayında rastgele dolaşmaya neden olacağından, kabul edilebilecek kötü çözümler için bir eşik/sınır değer belirlemek gerektiği açıktır. Bu tezde, Alabas-Uslu ve Dengiz (2011) tarafından geliştirilen, Avci ve Topaloglu (2015; 2016) tarafından EZTDARP için kullanılan yaklaşım ile mevcut çözümü iyileştirmeyen çözümlerden de faydalanılmıştır. Söz konusu yaklaşım, uyarlamalı olduğu için herhangi bir parametre değerine ihtiyaç duymamaktadır ve arama sürecindeki durumlara göre eşik değeri üzerinde değişiklik yapabilmektedir. Yöntem bu yönüyle başlangıç sıcaklık, soğutma oranı gibi parametre ayarlamalarına ihtiyaç duyan tavlama benzetimine göre kullanım kolaylığı sunmaktadır. Gokalp ve Ugur (2020) da benzer bir yaklaşımla MA_ILS-RVND adını verdikleri algoritmada eşik kabulü kullanarak kapasite kısıtlı araç rotalama problemlerini ele almışlardır.

Geçerli çözüm s, söz konusu yinelemede elde edilen çözüm s' ve başlangıç çözüm de s_0 ile ifade edilsin. Eşik kabul algoritmasında s' çözümünün bir sonraki yineleme için geçerli çözüm olarak kabul edilebilmesi için $f(s') \leq t * f(s)$ şartını sağlaması gerekmektedir. Buradaki t, eşik değerini ifade etmektedir ve Eşitlik (3.4) – (3.6)'da gösterildiği gibi hesaplanmaktadır (Alabas-Uslu ve Dengiz 2011: 8991-8993):

$$t = 1 + \ a_1 * a_2 \tag{3.4}$$

$$a_1 = f(s_{En\ iyi}/s_0) \tag{3.5}$$

$$a_2 = C(L_i)/i \tag{3.6}$$

Burada a_1, elde edilen en iyi çözümün başlangıç çözüm değerine oranını, a_2 ise en iyi çözümün iyileştirilme sayısının yineleme sayısına oranını ifade etmektedir. Her iki parametrenin değeri de algoritmanın başında 1 olarak tanımlanır ve her yeni çözüm, kabul edildiğinde güncellenir. Kabul edilmeyen çözüm sayısı, mevcut çözümün komşuluk sayısına ulaştığında, eşik değeri Eşitlik (3.7)'de gösterildiği gibi arttırılarak çözümlerin kabul edilmesi kolaylaştırılmaktadır.

$$t = t + \ a_1 * a_2 \tag{3.7}$$

Avci ve Topaloglu (2015; 2016), kabul şartını $f(s') \leq t * f(s_{En\ iyi})$ olarak, eşik değeri t'nin arttırılma koşulunu ise reddedilen çözümlerin sayısının komşuluk yapısı sayısına ulaşması olarak uyarlayarak kendi çalışmalarında kullanmışlardır. Bu tezde de kabul şartı olarak Avci ve Topaloglu (2015; 2016)'nun yaklaşımı kullanılmıştır.

3.2. Sayısal Analiz

Bu tez kapsamında önerilen algoritma, girdi parametresi olarak yalnızca maksimum yineleme sayısını (*MaxIt*) istemektedir. Algoritmanın yalnızca bir girdi parametresine sahip olması, uzun zaman gerektiren parametre ayarı (parameter tuning) işlemlerine ihtiyaç duyulmamasını sağlamaktadır ve bu durum, algoritma için önemli bir avantajdır. Bu tezde, *MaxIt* değeri için {1000, 5000, 10000, 20000} değerleri incelenmiştir. Bu değerlerle yapılan test sonuçlarına göre hem çözüm süresi hem de çözüm değerleri açısından, *MaxIt* = 5000 en uygun maksimum yineleme sayısı olarak belirlenmiştir. Bu durum, CMT3X problemi kullanılarak yapılan test sonuçlarını içeren Tablo 5'te gösterilmiştir.

Tablo 5. MaxIt Parametresinin Ortalama Çözüm Değeri ve Süre Üzerindeki Etkisi

MaxIt	**Ortalama Çözüm**	**Ortalama Süre***
1000	724,18	15,11
5000	722,27	59,64
10000	721,83	103,99
20000	722,33	191,21

*İşlemci süresi (saniye)

Tablo 5'te yer alan CMT3X problemi için 30 deneme yapılarak elde edilen sonuçlar incelendiğinde, en iyi ortalama çözüm değerinin $MaxIt = 10000$ olduğu durumda 721,83 olarak elde edildiği görülmektedir. Bu çözüm değerinin ortalama çözüm süresi ise 103,99 saniye olarak ölçülmüştür. Buna karşın, $MaxIt = 5000$ olan durumun ortalama çözüm değeri yaklaşık 0,44 kadar (yaklaşık %0,06) daha kötü olmasına rağmen süre açısından bakıldığında yaklaşık 44,35 saniye (yaklaşık %43) daha hızlı çalıştığı görülmektedir. Bu nedenle çözüm süresi ve çözüm kalitesi açısından denge sağlayabilmek adına $MaxIt = 5000$ olarak seçilmiştir.

Bölüm 3.5.1'de detaylarından bahsedilen test problemleri, her bir problem için $[1,1000]$ aralığından rastgele seçilen farklı çekirdek değerleri (seed) ile 30 kere çalıştırılmıştır. Farklı çekirdek değerlerinin seçilmesi ile deneylerin birbirinden bağımsız olması ve deneyin tekrarlanabilir olması sağlanmaktadır. Her bir problem için rastgele belirlenen çekirdek değerleri Ek 1 – 3'te yer almaktadır. Deneme sayısının fazla olması ise algoritmanın farklı sonuçlar üreterek daha güvenilir sonuçlar elde edilmesini sağlamaktadır. İlgili tablolarda raporlanan süreler, saniye cinsinden işlemci süreleridir. Her bir problem için elde edilen en iyi çözüm değeri (EÇ) ile literatürde bilinen en iyi çözüm değeri (BEÇ) değerleri Eşitlik (3.8)'de verilen sapma (%) formülü kullanılarak karşılaştırılmıştır.

$$Sapma(\%) = 100 * \frac{EÇ - BEÇ}{BEÇ} \tag{3.8}$$

3.2.1. Test Problemleri

Önerilen hibrit meta sezgisel algoritmanın performansını test etmek için literatürde en sık kullanılan test problemlerinden faydalanılmıştır. Bu test problemlerinden birincisi, Salhi ve Nagy (1999) tarafından oluşturulmuştur. Söz konusu problem kümesi, Christofides vd. (1979) tarafından kapasite kısıtlı araç rotalama problemleri için oluşturulmuştur ve literatürde CMT olarak bilinen veri setindeki talep miktarının dağıtım ve toplama şeklinde ayrıştırılması ile elde edilmiştir. Söz konusu ayrıştırma i müşterisinin koordinatlarından elde edilen $r_i = \min((x_i/y_i), (y_i/x_i))$ oranı, orijinal talep miktarı q_i ile çarpıldığında $D_i = r_i * q_i$ dağıtım miktarını vermektedir. Aynı müşteriden yapılacak toplama miktarı ise $P_i = (1 - r_i) * q_i$ şeklinde belirlenmektedir. Bu şekilde elde edilen problemlere X tipi

problemler denilmektedir. Y tipi problemlerin elde edilişinde ise literatürde iki tür yaklaşım vardır. Y tipi problemler, ilk yaklaşımda müşterilerin toplama ve dağıtım talepleri kendi içinde değiş tokuş edilerek elde edilirken; ikinci yaklaşımda ise toplama ve dağıtım miktarları diğer müşterilerle değiştirilerek elde edilmektedir. Örneğin, Chen ve Wu (2006), Wassan vd. (2008) gibi çalışmalar toplama ve dağıtım miktarını diğer müşterilerle değiştirerek Y tipi problemleri elde etmiştir. Bu durum, müşterilerin toplama ve dağıtım verisinde değişikliğe yol açacağı için problemin farklı bir karakter kazanmasına neden olmaktadır. Literatürde toplama ve dağıtım miktarını yuvarlayan Montané ve Galvão (2006) gibi çalışmalar da bulunmaktadır. Konu hakkında detaylı bilgiler Zachariadis vd. (2009; 2010), Polat vd. (2015), Kalayci ve Kaya (2016) ve Polat (2017) tarafından ele alınmıştır.

Bahsedilen hususlar sebebiyle literatürde belirli bir problem için birbirinden oldukça farklı olan çözüm değerlerinin raporlandığı görülebilmektedir. Bu nedenle sonuçları karşılaştırırken aynı yaklaşımlara sahip algoritmaların dikkate alınması gerekmektedir. Bu tezde problem verisinde herhangi bir yuvarlama işlemi yapılmamıştır. Y tipi problemler ise her bir müşterinin toplama ve dağıtım miktarlarının değiş tokuş edilmesi ile elde edilmiştir. Elde edilen sonuçlar, benzer yaklaşıma sahip algoritmaların sonuçlarıyla karşılaştırılmıştır. Salhi ve Nagy (1999) problem kümesinde toplam 28 problem örneği yer almaktadır. Problemlerin müşteri sayıları, 50 ile 199 arasında değişmektedir. Son olarak, bu test problemlerinden CMT6-10, CMT13 ve CMT14 problemlerinin X ve Y tiplerinde zaman limiti içerdiğini belirtmek gerekmektedir. Geriye kalan 14 test probleminde ise zaman limiti yoktur.

Dethloff (2001) tarafından oluşturulan test problemleri de EZTDARP literatüründe yaygın şekilde kullanılmaktadır. Problemlerin her biri 50 müşteriden oluşmaktadır. Bu problemler, iki senaryo altında gruplanmaktadır. SCA olarak isimlendirilen ilk senaryo, $[0{,}100]$ aralığında tekdüze dağılan müşterilerden oluşmaktadır. CON olarak isimlendirilen ikinci senaryoda ise müşterilerin yarısı, SCA senaryosunda olduğu gibi dağılırken müşterilerin diğer yarısı ise $[100/3\,,200/3]$ aralığında tekdüze biçimde (daha yoğun) dağılmaktadır. Müşterilere yapılacak teslimat miktarı, $[0{,}100]$ aralığında tekdüze dağılırken toplama miktarı ise teslimat talebi D_j'e bağlı olarak $P_j = \left(0{,}5 + r_j\right) * D_j$ şeklinde hesaplanmaktadır. Burada r_j terimi, [0,1]

aralığında tekdüze dağılan bir rastgele sayıdır. Bu problem kümesinde, minimum araç sayısının 3 ya da 8 olduğu toplam 40 problem örneği bulunmaktadır.

3.2.2. Hesaplama Sonuçları

Önerilen algoritma kullanılarak elde edilen sonuçlar, test problemleri özelinde iki grup halinde incelenmiştir. İlk inceleme grubunda algoritmanın elde ettiği en iyi çözüm değerleri ile literatürde bilinen en iyi sonuç değerleri karşılaştırılmıştır. Bu grupta bahsedilen süre ile en kısa sürede elde edilen çözüm (deney 30 kere tekrarlandığı için aynı sonuç değeri birden fazla kez elde edilebilmektedir) kastedilmektir. İkinci inceleme grubunda ise algoritmanın elde ettiği sonuçlar, kendi içinde karşılaştırılmıştır. İkinci grupta bahsedilen standart sapma, 30 deneme sonucunda elde edilen çözüm değerleri için hesaplanan standart sapma anlamına gelirken; sapma (%) ise ortalama çözüm değerinin bilinen en iyi çözüm değerinden ne kadar saptığı anlamına gelmektedir ve Eşitlik (3.8) yardımıyla hesaplanmaktadır. Referans alınan diğer algoritmaların farklı bilgisayar ortamlarında kullanılması, paralel hesaplama yapması gibi sebepler nedeniyle hesaplama süreleri üzerinde doğrudan etkiye sahiptir. Bu nedenle hesaplama süreleri kullanılarak doğrudan bir karşılaştırma yapmanın yanıltıcı olabileceğini belirtmek gerekmektedir. Referans alınan algoritmaların hesaplama sürelerine ilişkin bilgilere Ek 4 – 6'da yer verilmiştir.

3.2.2.1. Dethloff (2001) test problemlerine ilişkin sonuçlar

Dethloff (2001) tarafından geliştirilen test problemleri kullanılarak elde edilen en iyi çözüm değerleri ile literatürde bilinen en iyi çözüm değerleri, Tablo 6 ve Tablo 7'de gösterildiği gibi karşılaştırılmıştır. SCA problemlerinde elde edilen rotalar SCA problemleri için Ek 7 – 8'de, CON problemleri için Ek 9 – 10'da verilmiştir.

Tablo 6. SCA Problemlerine İlişkin En İyi Çözüm Değerleri

Test Problemi*	**Bilinen En İyi Çözüm**		**ILS-RVND-TA**			
	Referans	Çözüm	En İyi Çözüm	Sapma (%)	Süre**	Araç Sayısı
SCA3-0	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	635,62	**635,62**	0,00	9,80	4
SCA3-1	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	697,84	**697,84**	0,00	9,19	4
SCA3-2	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	659,34	**659,34**	0,00	9,14	4
SCA3-3	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	680,04	**680,04**	0,00	8,80	4
SCA3-4	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	690,50	**690,50**	0,00	8,41	4
SCA3-5	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	659,90	**659,90**	0,00	8,61	4
SCA3-6	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	651,09	**651,09**	0,00	9,39	4
SCA3-7	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	659,17	**659,17**	0,00	8,05	4
SCA3-8	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	719,47	**719,47**	0,00	9,61	4
SCA3-9	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	681,00	**681,00**	0,00	8,75	4
SCA8-0	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	961,50	**961,50**	0,00	6,80	9
SCA8-1	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1049,65	**1049,65**	0,00	7,28	9
SCA8-2	VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1039,64	**1039,64**	0,00	10,24	9
SCA8-3	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	983,34	**983,34**	0,00	6,72	9
SCA8-4	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1065,49	**1065,49**	0,00	7,06	9
SCA8-5	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1027,08	**1027,08**	0,00	9,17	9
SCA8-6	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	971,82	**971,82**	0,00	8,38	9
SCA8-7	VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1051,28	**1051,28**	0,00	9,24	9
SCA8-8	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1071,18	**1071,18**	0,00	7,13	9
SCA8-9	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	1060,50	**1060,50**	0,00	7,31	9
Bulunan En İyi Çözüm Sayısı			**20**			
Ortalama			850,77	0,00	8,45	6,50

ACS: Gajpal ve Abad (2009)**, VLBR:** Zachariadis vd. (2010)**, PILS:** Subramanian vd. (2010)**, HPSO:** Goksal vd. (2013), **ACSEVNS:** Kalayci ve Kaya (2016), **ALNS-PR:** Hof ve Schneider (2019)

*Müşteri sayısı bütün test problemleri için 50'dir. **İşlemci süresi (saniye)

Tablo 6 incelendiğinde önerilen algoritmanın bütün SCA problemleri için literatürde bilinen en iyi çözüm değerlerini elde ettiği görülmektedir. Bu nedenle ortalama sapma değeri, 0 olarak hesaplanmıştır. Ortalama çözüm değeri ise 850,77 olarak

hesaplanmıştır. Tablodaki süre sütunu incelendiğinde çözümlerin [6,72, 10,24] saniye aralığında çözüldüğü görülmektedir. Bu süre, ortalama olarak ise 8,45 saniye olarak ölçülmüştür. Kullanılan araç sayı ise 4 ile 9 arasında değişmekte olup, ortalama olarak 6,50 değerine sahiptir.

Tablo 7. CON Problemlerine İlişkin En İyi Çözüm Değerleri

Test Problemi*	**Bilinen En İyi Çözüm**		**ILS-RVND-TA**			
	Referans	Çözüm	En İyi Çözüm	Sapma (%)	Süre**	Araç Sayısı
CON3-0	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	616,52	**616,52**	0,00	10,03	4
CON3-1	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	554,47	**554,47**	0,00	8,84	4
CON3-2	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	518,00	**518,00**	0,00	14,16	4
CON3-3	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	591,19	**591,19**	0,00	8,89	4
CON3-4	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	588,79	**588,79**	0,00	8,41	4
CON3-5	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	563,70	**563,70**	0,00	9,64	4
CON3-6	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	499,05	**499,05**	0,00	9,30	4
CON3-7	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	576,48	**576,48**	0,00	8,05	4
CON3-8	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	523,05	**523,05**	0,00	10,47	4
CON3-9	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	578,25	**578,25**	0,00	8,53	4
CON8-0	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	857,17	**857,17**	0,00	7,34	9
CON8-1	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	740,85	**740,85**	0,00	7,64	9
CON8-2	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	712,89	**712,89**	0,00	9,06	9
CON8-3	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	811,07	**811,07**	0,00	9,06	10
CON8-4	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	772,25	**772,25**	0,00	7,77	9
CON8-5	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	754,88	**754,88**	0,00	8,23	9
CON8-6	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	678,92	**678,92**	0,00	8,06	9
CON8-7	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	811,96	**811,96**	0,00	7,19	9
CON8-8	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	767,53	**767,53**	0,00	7,78	9
CON8-9	ACS, VLBR, PILS, HPSO, ACSEVNS, ALNS-PR	809,00	**809,00**	0,00	8,97	9
Bulunan En İyi Çözüm Sayısı			**20**			
Ortalama			666,30	0,00	8,87	6,55

ACS: Gajpal ve Abad (2009)**, VLBR:** Zachariadis vd. (2010)**, PILS:** Subramanian vd. (2010)**, HPSO:** Goksal vd. (2013), **ACSEVNS:** Kalayci ve Kaya (2016), **ALNS-PR:** Hof ve Schneider (2019)

*Müşteri sayısı bütün test problemleri için 50'dir. **İşlemci süresi (saniye)

Tablo 7 incelendiğinde önerilen algoritmanın bütün CON problemleri için literatürde bilinen en iyi çözüm değerlerini elde ettiği görülmektedir. Bu nedenle ortalama sapma değeri, 0 olarak hesaplanmıştır. Ortalama çözüm değeri ise 666,30 olarak hesaplanmıştır. Tablodaki süre sütunu incelendiğinde çözümlerin [7,19, 14,16] saniye aralığında çözüldüğü görülmektedir. Bu süre, ortalama olarak 8,87 saniye olarak ölçülmüştür. Kullanılan araç sayı ise 4 ile 10 arasında değişmekte olup ortalama olarak 6,55 değerine sahiptir.

Tablo 6 ve Tablo 7 birlikte incelendiğinde önerilen algoritmanın, 40 problemin tamamı için literatürde bilinen en iyi çözüm değerlerini elde edebildiği görülmüştür. SCA problemleri düzlemde daha dağınık olduğu için ortalama çözüm değerlerinin, düzlemde daha sık dağılan CON problemlerine göre daha yüksek olduğu gözlemlenmiştir. Her iki problem için 30 tekrar sonucunda elde edilen sonuçların detayları ise Tablo 8 ve Tablo 9'da verildiği gibidir.

Tablo 8. SCA Problemleri İçin Detaylı Sonuçlar

Test Problemi*	**Ortalama Çözüm Maliyeti**	**En Kötü Çözüm Değeri**	**Standart Sapma**	**Sapma (%)**	**Ortalama Hesaplama Süresi****
SCA3-0	637,06	640,55	1,96	0,23	10,56
SCA3-1	697,84	697,84	0,00	0,00	9,75
SCA3-2	659,34	659,34	0,00	0,00	9,75
SCA3-3	680,04	680,04	0,00	0,00	10,35
SCA3-4	690,50	690,50	0,00	0,00	9,24
SCA3-5	659,90	659,90	0,00	0,00	9,35
SCA3-6	651,09	651,09	0,00	0,00	10,57
SCA3-7	663,37	669,89	3,85	0,64	9,44
SCA3-8	719,47	719,47	0,00	0,00	10,14
SCA3-9	681,00	681,00	0,00	0,00	9,15
SCA8-0	961,50	961,50	0,00	0,00	7,50
SCA8-1	1050,47	1067,45	3,23	0,08	8,73
SCA8-2	1044,31	1050,37	5,00	0,45	14,48
SCA8-3	983,34	983,34	0,00	0,00	8,34
SCA8-4	1066,19	1068,97	1,41	0,07	9,01
SCA8-5	1028,10	1042,30	3,86	0,10	10,82
SCA8-6	971,82	971,82	0,00	0,00	11,40
SCA8-7	1061,73	1063,22	3,62	0,99	9,96
SCA8-8	1071,18	1071,18	0,00	0,00	7,77
SCA8-9	1061,35	1063,68	1,43	0,08	9,82
Ortalama	851,98	854,67	1,22	0,13	9,81

*Müşteri sayısı bütün test problemleri için 50'dir. **İşlemci süresi (saniye)

Tablo 8'de yer alan değerler incelendiğinde SCA3-0, SCA3-7, SCA8-1, SCA8-2, SCA8-4, SCA8-5, SCA8-7, SCA8-9 problemleri haricindeki problemlerde bütün denemelerde bilinen en iyi çözüm değerine ulaşıldığı görülmektedir. SCA8-2, standart sapmasının en yüksek (5) olması nedeniyle sonuçları en değişken problem olarak değerlendirilmiştir. SCA8-7 ise ortalama olarak bilinen en iyi çözüm değerinden en çok sapan problem (%0,99) olmuştur. Ortalama olarak bu sapma değeri %0,13 olarak hesaplanmış olup, problemin genel olarak bilinen en iyi çözüm değerine yakın sonuçlar verdiği görülmüştür. Ortalama hesaplama süresi, 9,81 olarak saniye ölçülmüştür.

Tablo 9. CON Problemleri İçin Detaylı Sonuçlar

Test Problemi*	Ortalama Çözüm Maliyeti	En Kötü Çözüm Değeri	Standart Sapma	Sapma (%)	Ortalama Hesaplama Süresi**
CON3-0	616,52	616,52	0,00	0,00	11,23
CON3-1	554,73	556,04	0,59	0,05	11,70
CON3-2	520,79	521,38	1,13	0,54	12,03
CON3-3	591,19	591,19	0,00	0,00	9,62
CON3-4	588,79	588,79	0,00	0,00	9,54
CON3-5	563,70	563,70	0,00	0,00	10,41
CON3-6	499,43	502,16	0,87	0,08	12,48
CON3-7	576,48	576,48	0,00	0,00	9,21
CON3-8	523,05	523,05	0,00	0,00	11,08
CON3-9	578,25	578,25	0,00	0,00	9,86
CON8-0	857,17	857,17	0,00	0,00	9,86
CON8-1	740,85	740,85	0,00	0,00	9,11
CON8-2	712,89	712,89	0,00	0,00	11,10
CON8-3	811,07	811,07	0,00	0,00	10,99
CON8-4	772,25	772,25	0,00	0,00	9,21
CON8-5	754,88	754,88	0,00	0,00	9,55
CON8-6	679,14	685,49	1,20	0,03	9,81
CON8-7	811,96	811,96	0,00	0,00	7,97
CON8-8	767,53	767,53	0,00	0,00	9,17
CON8-9	809,41	811,16	0,83	0,05	12,44
Ortalama	666,50	667,14	0,23	0,04	10,32

*Müşteri sayısı bütün test problemleri için 50'dir. **İşlemci süresi (saniye)

Tablo 9'da yer alan değerler incelendiğinde CON3-1, CON3-2, CON3-6, CON8-6 ve CON8-9 problemleri haricindeki problemlerde bütün denemelerde bilinen en iyi çözüm değerine ulaşıldığı görülmektedir. CON8-6, standart sapmasının en yüksek (1,20) olması nedeniyle sonuçları en değişken problem olarak değerlendirilmiştir. CON3-2 ise ortalama olarak bilinen en iyi çözüm değerinden en çok sapan (%0,54) problem olmuştur. Ortalama olarak bu sapma değeri ise %0,04 hesaplanmış olup, algoritmanın SCA

problemlerine göre bilinen en iyi çözüm değerlerine daha yakın sonuçlar verdiği tespit edilmiştir. Ortalama hesaplama süresi, 10,32 saniye olarak ölçülmüştür.

3.2.2.2. Salhi ve Nagy (1999) test problemlerine ilişkin sonuçlar

Salhi ve Nagy (1999) test problemleri, literatürde yaygın şekilde kullanılmaktadır. Bu problem kümesi, farklı sayıda müşteri sayılarından oluşan problemler içerdiği için algoritmaların performansını daha detaylı ölçebilmeye imkân tanımaktadır. Bölüm 3.5.1'de bahsedildiği gibi toplama ve dağıtım miktarları hesaplanırken yuvarlama işlemi yapılmadığından, en iyi sonuçları karşılaştırmak için yine yuvarlama yapmayan çalışmalar referans alınmıştır. Söz konusu sonuçlar, Tablo 10'da verildiği gibidir. Bu problemlerin çözümüne ait rotalar Ek 11 – 12'de verilmiştir.

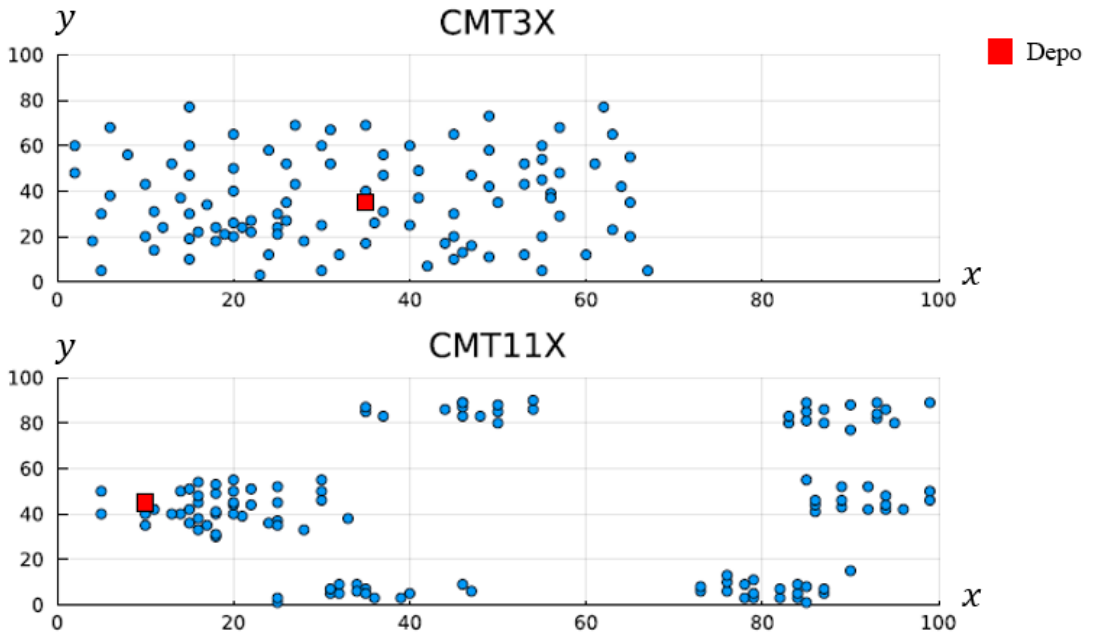
Tablo 10 incelendiğinde, CMT problemlerinde müşteri sayılarının 50 ile 199 arasında değiştiği görülmektedir. Bunun doğal bir sonucu olarak çözüm sürelerinin 7,27 saniye ile 238,31 saniye arasında değiştiği, ortalama çözüm süresinin ise 88,10 saniye olduğu gözlemlenmiştir. Problem kümesinde yer alan 14 problemden CMT4X, CMT4Y, CMT11X ve CMT11Y haricindeki 10 problemde literatürde bilinen en iyi çözüm değerlerine ulaşılmıştır. Bilinen en iyi çözüm değerlerine ulaşılamayan problemlerin sapma değerlerine bakıldığında CMT4X, CMT4Y için %0,04 gibi küçük bir sapma değeri gözlemlenmesine rağmen, CMT11X ve CMT11Y için ise sapma değeri %1,55 olarak hesaplanmıştır. Algoritma, CMT11 problemleri ile benzer müşteri sayısına sahip CMT3 problemleri için bilinen en iyi çözüm değerlerini elde edebilmiştir. Benzer şekilde, CMT4 problemleri ile CMT5 problemleri karşılaştırıldığında, daha çok müşteri sayısına sahip CMT5 problemlerinde bilinen en iyi çözüm değerlerinin elde edilebildiği görülmüştür. Bu durum, söz konusu sapmaların müşteri sayısından çok müşterilerin düzlemdeki konumları nedeniyle algoritmanın en iyi çözüm değerine ulaşamamış olabileceği düşüncesini akla getirmektedir. CMT3X ile CMT11X problemlerinde müşterilerin konumlarını gösteren Şekil 36, bu düşünceyi doğrular niteliktedir. Problemlerin bilinen en iyi çözümden sapması ise %0,23 olarak hesaplanmıştır. Kullanılan araç sayısının 3 ile 10 arasında değiştiği görülürken, kullanılan ortalama araç sayısının ise 5,71 olduğu tespit edilmiştir.

Tablo 10. Salhi ve Nagy (1999) Problemlerine İlişkin En İyi Çözüm Değerleri

Test Problemi		**Bilinen En İyi Çözüm**			**ILS-RVND-TA**			
Test Problemi	**Müşteri Sayısı**	**Referans**	**Çözüm**	**Araç Sayısı**	**En İyi Çözüm**	**Sapma (%)**	**Süre***	**Araç Sayısı**
CMT1X	50	ACS, PILS, HPSO, PVNS, ILS_ANS, ACSEVNS, ALNS-PR	466,77	3	**466,77**	0,00	7,27	3
CMT1Y	50	ACS, PILS, HPSO, PVNS, ILS_ANS, ACSEVNS, ALNS-PR	466,77	3	**466,77**	0,00	7,36	3
CMT2X	75	ACS, GTS, VLBR, PILS, HPSO, PVNS, ACSEVNS, ALNS-PR	684,21	6	**684,21**	0,00	20,50	6
CMT2Y	75	GTS, VLBR, PILS, HPSO, PVNS, ACSEVNS, ALNS-PR	684,21	6	**684,21**	0,00	14,17	6
CMT3X	100	GTS, PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	721,27	5	**721,27**	0,00	70,72	5
CMT3Y	100	GTS, PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	721,27	5	**721,27**	0,00	69,95	5
CMT4X	150	GTS, PILS, VLBR, HPSO, PVNS, ILS_ANS, ACSEVNS, ALNS-PR	852,46	7	852,83	0,04	85,56	7
CMT4Y	150	GTS, PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	852,46	7	852,83	0,04	100,89	7
CMT5X	199	PILS	1029,25	10	**1029,25**	0,00	230,03	10
CMT5Y	199	PILS, ALNS-PR	1029,25	10	**1029,25**	0,00	238,31	10
CMT11X	120	PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	833,92	4	846,86	1,55	139,89	4
CMT11Y	120	PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	833,92	4	846,86	1,55	151,25	4
CMT12X	100	GTS, PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	662,22	5	**662,22**	0,00	43,28	5
CMT12Y	100	GTS, PILS, VLBR, HPSO, PVNS, ACSEVNS, ALNS-PR	662,22	5	**662,22**	0,00	54,28	5
Bulunan En İyi Çözüm Sayısı					**10**			
Ortalama					751,92	0,23	88,10	5,71

ACS: Gajpal ve Abad (2009), **GTS**: Zachariadis vd. (2009), **VLBR**: Zachariadis vd. (2010), **PILS**: Subramanian vd. (2010), **HPSO**: Goksal vd. (2013), **PVNS**: Polat vd. (2015), **ILS_ANS**: Li vd. (2015), **ACSEVNS**: Kalayci ve Kaya (2016), **ALNS-PR:** Hof ve Schneider (2019)

*****İşlemci süresi (saniye)



Şekil 36. CMT3X ve CMT11X Problemlerinde Müşterilerin Konumları

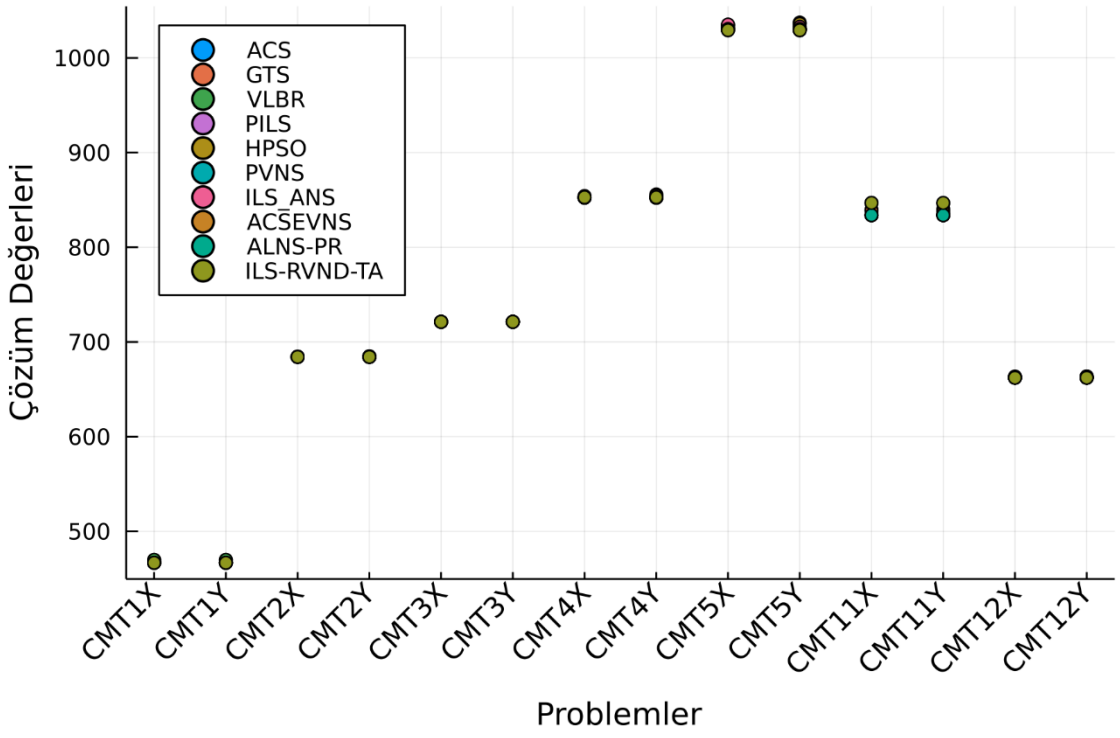
Salhi ve Nagy (1999) problemleri kullanılarak elde edilen sonuçların detaylı analizi ise Tablo 11'de verildiği gibidir. Tablo 11 incelendiğinde yalnızca CMT1X ve CMT1Y problemlerinde tüm denemelerde bilinen en iyi sonucun elde edilebildiği görülmektedir. Bu durumun temel nedeni, EZTDARP'nin kombinatoryal bir problem olması sebebiyle problem boyutunun (müşteri sayısının) arttıkça problemin çözümünün zorlaşmasıdır. CMT11X en yüksek standart sapma değeri (7,06) ile çözümleri en değişken problem türü olmuştur. Ortalama olarak bilinen en iyi çözüm değerinden en çok sapan (%3,17) problem ise benzer şekilde CMT11Y olmuştur. Problemlerin genel olarak bilinen en iyi çözümden sapmaları ise ortalama %0,75 olarak hesaplanmıştır. Ortalama çözüm süresi, 92,48 saniye olarak ölçülmüştür.

Önerilen algoritmanın performansını, diğer algoritmalarla görsel olarak karşılaştırmak için grafiklerden faydalanılabilir. Dethloff (2001) problemleri, küçük boyutlu olduğu için Tablo 6 – 7'de de gösterildiği gibi referans alınan çalışmalar tarafından çok büyük sapmalar olmaksızın çözüldüğü açıkça görülmektedir. Bu nedenle söz konusu problemler için grafik çizilmemiştir. Salhi ve Nagy (1999) problemleri için çizilen grafik ise Şekil 37'de verilmiştir.

Tablo 11. Salhi ve Nagy (1999) Problemlerine İlişkin Detaylı Sonuçlar

İsim	Müşteri Sayısı	Ortalama Çözüm Maliyeti	En Kötü Çözüm Değeri	Standart Sapma	Sapma (%)	Ortalama Hesaplama Süresi*
CMT1X	50	466,77	466,77	0,00	0,00	9.03
CMT1Y	50	466,77	466,77	0,00	0,00	8.33
CMT2X	75	686,53	689,39	1,78	0,34	15.55
CMT2Y	75	686,42	692,52	2,17	0,32	14.39
CMT3X	100	722,39	725,72	1,42	0,16	58.56
CMT3Y	100	722,50	725,72	1,47	0,17	57.51
CMT4X	150	854,11	856,88	0,96	0,15	128.79
CMT4Y	150	854,55	862,47	1,89	0,20	127.78
CMT5X	199	1034,77	1049,56	4,84	0,54	241.95
CMT5Y	199	1034,62	1042,53	2,86	0,52	246.83
CMT11X	120	872,87	876,92	7,06	3,07	135.17
CMT11Y	120	873,67	876,92	5,13	3,17	138.58
CMT12X	100	668,13	673,72	4,01	0,89	59.83
CMT12Y	100	668,96	673,72	3,82	1,02	52.35
Ortalama		758.08	762,83	2,67	0,75	92,48

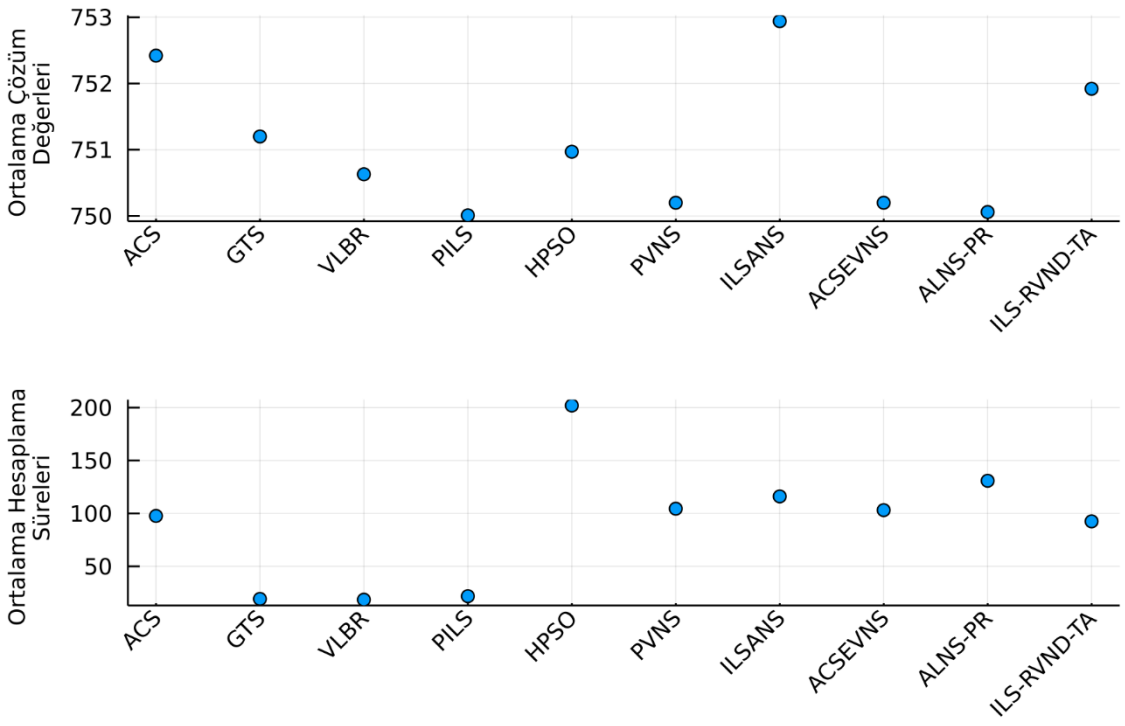
***İşlemci süresi (saniye)



Şekil 37. Algoritmaların Sonuçlarının Karşılaştırılması

Şekil 37 incelendiğinde önerilen algoritmanın, Tablo 10'da da ifade edildiği gibi CMT4X, CMT4Y, CMT11X ve CMT11Y problemleri dışında kalan diğer problemlerde referans alınan çalışmalarla aynı sonuçları elde ettiği görülmektedir. CMT4X ve CMT4Y

problemlerinde sapma çok küçük iken, CMT11X ve CMT11Y'de sapmanın biraz daha fazla olduğu göze çarpmaktadır. Ayrıca CMT5X ve CMT5Y problemlerinde ise referans alınan çalışmaların çoğundan daha iyi sonuçlar elde edildiği de dikkat çekici bir diğer önemli husustur. Söz konusu problemler için algoritmaların elde ettiği en iyi çözümlerin ortalama değerleri ve çalışmalar tarafından raporlanan ortalama hesaplama süreleri ise Şekil 38'de verilmiştir. Söz konusu grafiğe ait veriler ekler bölümünde yer almaktadır.



Şekil 38. Ortalama Çözüm Değerleri ve Hesaplama Süreleri Açısından Algoritmaların Karşılaştırılması

Şekil 38 incelendiğinde önerilen algoritmanın, en iyi çözümlerin ortalama değerleri açısından kabul edilebilir bir performans sergilediği görülmektedir. Bunun nedeni, bazı problemler için literatürde bilinen en iyi çözüm değerlerine ulaşılamaması olarak düşünülebilir. Bu problemlerdeki sapmaların azaltılması ile daha iyi performans sergileyen algoritmalara daha yakın sonuçlar elde edileceği açıktır. Benzer şekilde ortalama hesaplama süreleri incelendiğinde, önerilen algoritmanın iyi bir performans sergilediği görülmektedir. Ancak, bu grafiğin kullanılan bilgisayar mimarilerinin farklılığı, algoritmaların çalışma şekilleri gibi sebepler nedeniyle yanıltıcı olabileceğini belirtmek de gerekmektedir. Örneğin, GTS ve VLBR isimli algoritmalar yalnızca en iyi çözümlere ait hesaplama sürelerini raporlarken, PILS isimli algoritma ise paralel bir

yapıya sahiptir. Bu farklılıklara rağmen önerilen algoritmanın, süre açısından benzer yapıya sahip algoritmalar arasında iyi bir performans sergilediği görülmektedir.

SONUÇ

Önerilen ILS-RVND-TA isimli algoritma, yinelemeli yerel arama çerçevesi altında rastgele sıralamalı değişken komşuluk iniş ve eşik kabul meta sezgisellerini kapsayan bir hibrit meta sezgisel algoritmadır. Yinelemeli yerel arama, yerel arama ve sarsım mekanizmalarını dengeli bir şekilde kullanarak başarılı sonuçlara ulaşabilen bir meta sezgiseldir. Önerilen algoritmanın ihtiyaç duyduğu başlangıç çözüm, en yakın komşuluk sezgiseli ile elde edilmiştir. Kullanılan bu sezgisel, literatürde de yaygın bir şekilde tercih edilmektedir. En yakın komşuluk sezgiseli kullanım kolaylığı sebebiyle kısa sürede başlangıç çözüm oluşturabilmektedir, ancak çözüm oluştururken yalnızca uzaklıkları dikkate almaktadır. Tez kapsamında ele alınan problemde müşterilerin hem toplama hem de dağıtım talepleri karşılandığından rotaların seyahat boyunca yük değişimi oluşturulan çözümlerin uygunluğunu etkilemektedir. Bu nedenle ilerleyen çalışmalarda uzaklık dışında farklı kriterleri de dikkate alan sezgiseller kullanılarak başlangıç çözümler üretmek mümkündür.

Önerilen algoritmanın yerel arama aşamasında yoğunlaştırma faaliyetini etkin bir şekilde yerine getirebilmek için kullanılabilecek bazı yaklaşımlar vardır. Bu yaklaşımlar; tek bir komşuluk yapısının kullanılması, birden fazla komşuluk yapısı arasından rastgele seçilen bir komşuluk yapısının kullanılması ve birden fazla komşuluk yapısının hepsinin birden kullanılması şeklinde farklılaşabilmektedir. Bu yaklaşımlardan sonuncusu bu tezde benimsenmiştir. Birden fazla komşulukta optimal çözümün bulunma olasılığı, tek bir komşulukta bulunma olasılığına göre daha yüksek olduğundan literatürde bu yaklaşım sıklıkla kullanılmaktadır. Rastgele sıralamalı değişken komşuluk iniş de bu yaklaşımda en yaygın kullanılan meta sezgisellerden biridir. Rastgele sıralamalı değişken komşuluk iniş, uygulamasının kolay olması ve başarılı sonuçlar vermesi nedeniyle önerilen algoritmanın yerel arama bileşeninde kullanılmıştır. Altı farklı rotalar arası komşuluk yapısı ve beş farklı rota içi komşuluk yapısı, literatürden edinilen bilgiler ışığında seçilerek kullanılmıştır. Hem rotalar arası komşuluk yapılarında hem de rota içi komşuluk yapılarında en iyi çözümler araştırılarak arama uzayında yoğun şekilde arama yapılmıştır. Yapılan analizlerde yerel arama bileşeninin, algoritmanın en çok zaman tüketen kısmı olduğu tespit edilmiştir. Bu nedenle aramayı hızlandırabilmek için yerel arama sürecinde yardımcı fonksiyonlar kullanılarak uygun olmayan rotalar dikkate alınmamıştır. Bu

bileşende yapılacak geliştirmelerle önerilen algoritmanın performansında potansiyel iyileştirmeler elde etmek söz konusu olabilir.

Yinelemeli yerel aramanın bir diğer önemli bileşeni ise sarsımdır. Sarsım bileşeni sayesinde arama uzayının farklı bölgelerini keşfetmek mümkün hale gelmektedir. Özellikle arama, yerel optimuma takılıp kaldığında sarsım sayesinde farklı bir bölgeye geçerek daha iyi çözümlere ulaşabilmektedir. Sarsım, çözümler üzerinde rastgele değişiklikler yaparak arama uzayının farklı bölgelerine geçmeyi mümkün kılmaktadır. Mevcut çözümün taşıdığı özelliklerin tamamını değiştirecek büyüklükte rastgele değişikliklerin yapılması ise arama süreci açısından fayda sağlamayacaktır. Bu nedenle sarsım aşamasında kullanılacak operatörlerin seçilmesi büyük bir önem arz etmektedir. Çözümler üzerinde çok az değişiklik yapabilen operatörler, arama uzayına ait sınırlı bilgi sunduğu için arama süreci, bulunduğu bölgeye (yerel optimuma) geri dönme eğiliminde olmaktadır. Çözümler üzerinde çok fazla değişiklik yapabilen operatörler ise arama uzayına ait büyük miktarda bilgi sunarak, mevcut bilginin kaybedilmesine yol açar ve bunun sonucunda rastgele arama yapılmasına neden olur. Dolayısıyla arama uzayında doğru noktalarda arama yapabilmek için sarsım aşamasında kullanılacak operatörlerin dengeli bir şekilde değişiklik yapabilmesi gerekmektedir. Bu tezde, sarsım aşamasında kullanılan operatörler bu denge gözetilerek belirlenmiştir. Seçilen operatörler, yerel arama aşamasında kullanılan operatörlerden farklı tiplerde olacak şekilde seçilmiştir. Bunun sebebi sarsım aşamasında yapılacak değişikliklerin, yerel arama aşamasında geriye döndürülmesini engellemektir. Bu tez kapsamında sarsım aşamasında hem literatürden faydalanılarak operatörler belirlenmiş hem de özgün bir operatör geliştirilerek literatürde katkıda bulunulmuştur. Geliştirilen operatör ile rotaların ortalama maliyetleri ve maksimum yük miktarları birlikte dikkate alınarak, rotalar üzerinde değişiklik yapılması ve yerel arama aşamasında iyileştirebilecek yeni rotaların elde edilmesi amaçlanmıştır.

Sarsım aşamasının en önemli özelliği çözümler üzerinde rastgele değişiklikler yaparak arama uzayının farklı bölgelerini keşfetmeyi sağlamaktır. Bu nedenle değişiklik yapılacak çözüm bileşenlerinin seçimini tamamen rastgele yerine yanlı bir şekilde yapılarak arama sürecini yönlendirmek mümkün olabilir. Değişikliğe uğratılacak çözüm bileşenlerini problemin amacına bağlı olarak seçmek, zaten iyi durumda rotaların gereksiz yere değişikliğe uğratılmasını engelleyerek daha hızlı bir şekilde daha iyi

çözümlere ulaşılmasını sağlayabilir. Bu noktadan hareketle önerilen algoritmada bu süreci yönetebilmek adına rota seçici adı verilen özgün bir yaklaşımla sarsıma tabi tutulacak rotaların seçiminde ortalama uzaklık, maksimum yük şeklinde iki kritere dayanan ve Shannon entropisi yardımıyla olasılık değerleri hesaplanan bir rulet tekeri tekniği kullanılmıştır. Yüksek ortalama uzaklık, bir rotadaki müşterilerin birbirinden uzakta olduğuna işaret etmektedir. Bu nedenle yüksek ortalama uzaklığa sahip rotalar, yüksek amaç fonksiyonu değerine sahip çözümlere neden olduğundan yüksek ortalama uzaklık, istenilmeyen bir özellik olarak nitelendirilmektedir. Benzer şekilde yüksek maksimum yük, rotaya yeni müşterilerin eklenmesine engel olarak muhtemel iyi çözümlerin elde edilmesine engel olabilmektedir. Söz konusu iki kriter dikkate alınarak sarsım işlemine tabi tutulacak rotalar belirlenirken, hangi kriterin seçim üzerinde ne kadar ağırlığa (önem derecesi) sahip olduğu ise bilinememektedir. Kriterlerin ağırlıkları belirlenirken rotaların sahip olduğu bilgiden faydalanarak, objektif bir şekilde önem derecelerini belirleyebilmek için Shannon entropisi kullanılmıştır. Söz konusu yaklaşımla arama süresince rotalarda ortaya çıkan değişikliklere göre ağırlık değerlerinin değiştirilmesi de mümkün hale gelmiştir. Bu sayede hem dinamik bir ağırlık belirleme sürecine sahip olunmaktadır hem de hiçbir bilgiyi kullanmadan rastgele şekilde sarsım işlemini uygulayan yaklaşımlara göre bir avantaja sahip olunmaktadır. Söz konusu kriter değerlerine göre istenilmeyen özellikleri daha fazla olan rotaların sarsım işlemi için seçilme olasılıklarının da daha yüksek olduğu bir seçim süreci yürütülerek yerel arama aşamasında daha iyi çözümlere ulaşılması hedeflenmiştir. Bahsedilen bu özelliklere sahip sarsım aşaması, beş farklı operatör kullanarak çözümler üzerinde değişiklikler yapmaktadır. Geliştirilen rota seçici ve rota gevşetme operatörü problemlerin çözümünde başarıyla kullanılmıştır.

Bir yinelemede sarsım ve yerel arama aşamalarından sonra ortaya çıkan çözüm, mevcut çözümden daha kötü bir amaç fonksiyonu değerine sahip olabilmektedir. Bu nedenle sonraki yinelemede bu çözümden faydalanabilmek için kabul kriterinde eşik kabul meta sezgiseli kullanılmıştır. Kullanılan meta sezgisel, uyarlamalı olduğu için eşik değerini arama sürecinin başarı ya da başarısızlık durumlarına göre arttırıp azaltabilmektedir. Bu sayede algoritma için ek bir parametre ayarlamasına ihtiyaç duyulmamaktadır. Eşik kabul, bu yönüyle benzer amaçla kullanılabilecek tavlama benzetimi ile karşılaştırıldığında önemli bir avantaja sahiptir. Tavlama benzetimi; başlangıç sıcaklığı, soğutma planı gibi parametre ayarlamalarına ihtiyaç duymaktadır. Bu

nedenle uygun parametre değerlerinin belirlenebilmesi için zaman alan analizlerin yapılması gerekmektedir. Kullanılan algoritma halihazırda çok sayıda parametreye sahip ise bu durum, algoritmayı geliştiren ve/veya uygulayan kişiler için büyük zorluklara yol açabilmektedir. Gelecekte benzer uyarlamalı algoritmaların, kullanım kolaylığı nedeniyle kabul kriterlerinde ve algoritmaların farklı bileşenlerinde daha yaygın olarak tercih edilmesi beklenmektedir.

Geliştirilen algoritma, Dethloff (2001)'un geliştirdiği 40, Salhi ve Nagy (1999)'nin geliştirdiği 14 problem üzerinde test edilmiştir. Elde edilen sonuçlara göre Dethloff (2001) problemlerinin tamamında, Salhi ve Nagy (1999) problemlerinin ise on tanesinde literatürde bilinen en iyi çözüm değerleri elde edilerek 54 problemin 50'sinde yaklaşık olarak %93 başarıya ulaşılmıştır. Yapılan analizlerde önerilen algoritmanın küçük boyutlu problemlerde iyi bir performans sergilediği görülmüştür. Ele alınan EZTDARP, *NP-Zor* sınıfından bir problem olduğu için problem boyutunun artmasıyla birlikte çözülmesi zorlaşmaktadır. Bu durum hem çözüm değerlerine hem de hesaplama sürelerine yansımaktadır. Bütün problemlerin 50 müşteriye sahip olduğu SCA ve CON problemleri ile yine 50 müşteriye sahip CMT1X ve CMT1Y problemleri incelendiğinde literatürde bilinen çözüm değerlerine kısa sürelerde ulaştığı görülürken, problem boyutu büyüdükçe çözüm sürelerinin de arttığı ve az sayıda büyük boyutlu problem türlerinde literatürde bilinen en iyi çözüm değerlerinden sapan sonuçların elde edildiği görülmektedir. Bazı problem türlerinin çözümündeki sapmaya karşın, özellikle CMT5X, CMT5Y gibi büyük boyutlu ve literatürde az sayıda çalışma tarafından bilinen çözüm değerine ulaşılabilen problemlerin çözülebilmesi, önerilen algoritmanın başarısına işaret etmektedir.

Önerilen algoritma, Subramanian vd. (2008) tarafından geliştirilen ILS-VND isimli algoritma ile benzerlikler taşımaktadır. Her iki algoritma da yinelemeli yerel arama çerçevesi altında değişken komşuluk iniş meta sezgiselinden faydalanmaktadır. Söz konusu algoritmalar, yerel arama aşamasında kullanılan komşuluk yapıları açısından benzerlik taşırken; başlangıç çözüm oluşturmak için kullanılan sezgiseller (en yakın komşuluk sezgiseli – açgözlü ekleme sezgiseli), komşuluk yapılarının kullanılma şekli (rastgele sıralamalı – sabit sıralamalı), sarsım aşamasında kullanılan operatörler ve kabul kriteri (eşik değere bağlı olarak daha kötü çözümleri kabul edebilme – yalnızca daha iyi çözümleri kabul etme) açısından ise farklılaşmaktadır. Bu farklılıkların, elde edilen

sonuçlara da yansıdığı görülmektedir. Bazı problemlerde önerilen algoritma daha iyi sonuçlar verirken, bazı problemlerde ise ILS-VND daha iyi sonuçlar vermektedir.

Önerilen algoritma, Aydoğdu ve Özyörük (2020) tarafından önerilen rassal iteratif yerel arama- değişken komşu iniş algoritması ile başlangıç çözüm ve yerel arama aşamaları ile benzerlik taşırken; sarsım ve kabul kriteri gibi hususlarda farklılaşmaktadır. Souza vd. (2011) tarafından önerilen GENILS isimli algoritma, benzer şekilde yinelemeli yerel arama çerçevesi altında değişken komşuluk iniş ile arama yaparken; üç farklı sezgisel ile başlangıç çözüm oluşturma, yerel arama aşamasında bazı farklı komşuluk yapılarının kullanılması, sarsım aşamasında farklı operatörlerin kullanılması ve kabul kriteri açısından farklılaşmaktadır. Olgun vd. (2021) tarafından geliştirilen HH-ILS isimli algoritma başlangıç çözüm oluşturma, yerel arama bileşenlerinde önerilen algoritma ile benzerlik taşırken; sarsım aşamasında kullanılan operatörler, sarsıma uğratılacak rotaların seçilmesi ve kabul kriteri açısından farklılaşmaktadır.

Gokalp ve Ugur (2020) tarafından geliştirilen MA_ILS-RVND isimli algoritma, farklı türde bir araç rotalama problemini ele alsa da bu tezde önerilen algoritmanın bileşenleri ile benzerlikler taşımaktadır. Algoritmaların genel yapısındaki benzerliğe karşın, başlangıç çözümleri elde eden sezgiseller (en yakın komşuluk sezgiseli – paralel tasarruf sezgiseli), yerel arama aşamasında benimsenen stratejiler (en iyi komşu çözüm – hem en iyi komşu çözüm hem de ilk iyileştiren komşu çözüm yaklaşımları), sarsım aşamasında kullanılan operatörler gibi hususlarda farklılıklar da söz konusudur.

Meta sezgisel algoritmalar, stokastik süreçlere sahip olduğundan her yinelemede farklı sonuçlar üretebilmektedir. Önerilen algoritmanın da özellikle orta büyüklükteki problemlerle yapılan bağımsız denemelerde farklı sonuçlar elde ettiği görülmüştür. Bu, doğal bir sonuç olmakla birlikte algoritmanın geliştirilecek versiyonlarında bu farklılığın azaltılması hedeflenmektedir. Önerilen algoritmanın elde ettiği sonuçlar hakkında şeffaflık sağlayabilmek adına çözümlerin elde edildiği çekirdek değerleri ve çözüm sonucunda ortaya çıkan en iyi rotalar ekler bölümünde paylaşılmıştır. Verilen çekirdek değerleri ile deneyleri tekrarlamak mümkün hale gelirken, verilen rotalar ile de konu üzerinde çalışan ilgililere faydalı bilgiler verebileceği düşünülmektedir.

Algoritmanın az sayıda parametreye sahip olması, önemli bir avantajdır. Bu sayede parametre sayısı fazla olan algoritmalara göre daha kısa sürede en uygun

parametre değeri belirlenebilmektedir. Önerilen algoritmada kullanılan meta sezgisellerin kullanımının kolay olması, hem algoritma davranışının anlaşılabilir olması açısından hem de algoritmanın kolayca kodlanabilir yapıda olması açısından bir diğer önemli avantajdır.

Algoritmanın yeni ve her geçen gün daha da popüler hale gelen Julia programlama dilinde yazılmış olması da tezin öne çıkan bir özelliğidir. Julia programlama dili, kullanıcılarına hem kullanım kolaylığı hem de performans sağladığı için özellikle bilimsel hesaplamalar yapan araştırmacılar arasında ilgi görmektedir. Bu nedenle çok sayıda Julia programlama dilinde yazılmış çeşitli paketler araştırmacıların kullanımına sunulmaktadır. Önerilen algoritma için yazılan kod, paket haline getirilerek benzer şekilde EZTDARP üzerinde çalışmalar yapan kişilerin kullanımına sunulması düşünülmektedir.

Literatürde bilinen en iyi çözüm değerine ulaşılamayan problemler ele alındığında belirgin şekilde kümelenmiş müşterilerin söz konusu olduğu durumlarda algoritmanın rotaları iyileştirme konusunda bir dezavantaja sahip olabileceği görülmektedir. İlerleyen çalışmalarda bu durumu aşabilmek adına önerilen algoritmaya farklı bileşenler eklenebilir. Öncelikle, yerel arama aşamasına eklenecek yeni rotalar arası komşuluk yapıları ile mevcut çözümlerin komşulukları daha farklı yaklaşımlar ile araştırılabilir. Bu amaçla, problem verisindeki müşterilerin kümelenmesi ile ilgili bilgileri dikkate alacak uygun kümeleme yöntemlerinden faydalanılması gerekmektedir. Benzer şekilde, sarsım aşamasında kullanılan operatörlere ek olarak kümelenmeleri dikkate alan yeni operatörlerin kullanılması/geliştirilmesi gerekmektedir. Önerilen algoritmanın sonraki versiyonlarında yapılacak bu iyileştirmeler ile belirgin kümelenmelere sahip problemlerin arama uzaylarında hem daha çeşitli hem de daha yoğun aramalar yapılarak daha iyi çözümlere ulaşılabilir.

Tezin bu bölümünde şimdiye kadar önerilen algoritmanın performansını arttırabilmek için ilerleyen çalışmalara konu olabilecek muhtemel iyileştirme alanlarından bahsedilmiştir. Ancak bu çalışmaların tek konusu performans değildir. Önerilen algoritma üzerinde yapılacak uygun değişiklikler ile algoritmanın ilerleyen çalışmalarda ele alınan problemden farklı olarak çok depolu, heterojen filoya sahip, mesafe, zaman penceresi gibi daha farklı kısıtlamalara sahip benzer problemlerin ya da farklı sınıflardaki araç rotalama problemlerinin çözümünde de kullanılabileceğini söylemek gerekmektedir. Literatürde yer alan farklı meta sezgisellerden faydalanılarak

daha başarılı ya da daha farklı amaçlara hizmet edebilecek hibrit algoritmalar da geliştirilebilir. Ayrıca teknolojideki gelişmelere bağlı olarak bu tezde ele alınan problem için gelecekte kuantum optimizasyon algoritmaları gibi farklı yaklaşımlar da kullanılarak çözüm aranabilir.

KAYNAKLAR

Aarts, E. H. L., Korst, J. H. M. ve Laarhoven, van, P. J. M. (1997). "Simulated Annealing". İçinde E. H. L. Aarts ve J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimization*, s. 91-120, Wiley-Interscience, Chichester.

Abido, M. A. (2002). "Optimal Power Flow Using Tabu Search Algorithm", *Electric Power Components and Systems*, 30/5, 469-483.

Ai, T. J. ve Kachitvichyanukul, V. (2009). "A Particle Swarm Optimization for The Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Computers & Operations Research*, 36/5, 1693–1702.

Alabas-Uslu, C. ve Dengiz, B. (2011). "A Self-Adaptive Local Search Algorithm for The Classical Vehicle Routing Problem", *Expert Systems with Applications*, 38/7, 8990–8998.

Alaei, R. ve Setak, M. (2017). "Selecting Unique Suppliers Through Winner Determination in Combinatorial Reverse Auction: Scatter Search Algorithm", *Scientia Iranica Transaction E: Industrial Engineering*, 24/6, 3297–3307.

Alonso, F., Alvarez, M. J. ve Beasley, J. E. (2008). "A Tabu Search Algorithm for The Periodic Vehicle Routing Problem with Multiple Vehicle Trips and Accessibility Restrictions", *Journal of the Operational Research Society*, 59/7, 963-976.

Alsheddy, A., Voudouris, C., Tsang, E. P. K. ve Alhindi, A. (2016). "Guided Local Search". İçinde R. Martí, P. Panos ve M. G. C. Resende (Eds.), *Handbook of Heuristics*, s. 1-37, Springer International Publishing, Cham.

Al-Sultan, K. S. (1995). "A Tabu Search Approach to The Clustering Problem", *Pattern Recognition*, 28/9, 1443-1451.

Anagnostopoulos, K. P., Chatzoglou, P. D. ve Katsavounis, S. (2010). "A Reactive Greedy Randomized Adaptive Search Procedure for A Mixed Integer Portfolio Optimization Problem", *Managerial Finance*, 36/12, 1057-1065.

Arora, R. K. (2015). *Optimization: Algorithms and Applications*, Chapman & Hall/CRC, Florida.

Arora, S. ve Barak, B. (2009). *Computational Complexity: A Modern Approach*, Cambridge University Press, Cambridge.

Atmaca, E. (2012). "Bir Kargo Şirketinde Araç Rotalama Problemi ve Uygulaması", *TÜBAV Bilim Dergisi*, 5/2, 12-27.

Avci, M. ve Topaloglu, S. (2015). "An Adaptive Local Search Algorithm for Vehicle Routing Problem with Simultaneous and Mixed Pickups and Deliveries", *Computers & Industrial Engineering*, 83, 15–29.

Avci, M. ve Topaloglu, S. (2016). "A Hybrid Metaheuristic Algorithm for Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Expert Systems with Applications*, 53, 160–171.

Aydoğdu, B. ve Özyörük, B. (2020). "Dinamik Eş Zamanlı Topla Dağıt Araç Rotalama Probleminin Çözümü İçin Matematiksel Model ve Sezgisel Yaklaşım: Rassal İteratif Yerel Arama Değişken Komşu İniş Algoritması", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 35/2, 563–580.

Barbucha, D. (2012). "Agent-Based Guided Local Search", *Expert Systems with Applications*, 39/15, 12032-12045.

Belgin, O., Karaoglan, I. ve Altiparmak, F. (2018). "Two-Echelon Vehicle Routing Problem with Simultaneous Pickup and Delivery: Mathematical Model and Heuristic Approach", *Computers & Industrial Engineering*, 115, 1–16.

Bezanson, J., Edelman, A., Karpinski, S. ve Shah, V. B. (2015). "Julia: A Fresh Approach to Numerical Computing", *ArXiv:1411.1607[cs.MS]*. http://arxiv.org/abs/1411.1607.

Binato, S., De Oliveira, G. C. ve De Araújo, J. L. (2001). "A Greedy Randomized Adaptive Search Procedure for Transmission Expansion Planning", *IEEE Transactions on Power Systems*, 16/2, 247-253.

Bisaillon, S., Cordeau, J. F., Laporte, G. ve Pasin, F. (2011). "A Large Neighbourhood Search Heuristic for The Aircraft and Passenger Recovery Problem", *4OR*, 9/2, 139–157.

Blum, C. (2005). "Ant Colony Optimization: Introduction and Recent Trends", *Physics of Life Reviews*, 2/4, 353–373.

Blum, C. ve Roli, A. (2003). "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", *ACM Computing Surveys*, 35/3, 268-308.

Boryczka, U., Probierz, B. ve Kozak, J. (2014). "An Ant Colony Optimization Algorithm for an Automatic Categorization of Emails". İçinde D. Hwang, J. J. Jung ve N. T. Nguyen (Eds.), *Computational Collective Intelligence. Technologies and Applications, ICCI 2014*, s. 583-592, Springer, Cham.

Boussaïd, I., Lepagnot, J. ve Siarry, P. (2013). "A Survey on Optimization Metaheuristics", *Information Sciences*, 237, 82-117.

Braekers, K., Ramaekers, K. ve Van Nieuwenhuyse, I. (2016). "The Vehicle Routing Problem: State of The Art Classification and Review", *Computers & Industrial Engineering*, 99, 300-313.

Brandão, J. (2018). "Iterated Local Search Algorithm with Ejection Chains for The Open Vehicle Routing Problem with Time Windows", *Computers & Industrial Engineering*, 120, 146–159.

Bräysy, O. ve Gendreau, M. (2005). "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms", *Transportation Science*, 39/1, 104-118.

Burke, E. K., Curtois, T., Qu, R. ve Vanden Berghe, G. (2010). "A Scatter Search Methodology for The Nurse Rostering Problem", *Journal of the Operational Research Society,* 61/11, 1667–1679.

Campbell, A., Clarke, L., Kleywegt, A. ve Savelsbergh, M. (1998). "The Inventory Routing Problem". İçinde T. G. Crainic ve G. Laporte (Eds.), *Fleet Management and Logistics*, s. 95-113, Kluwer, Boston.

Caporossi, G. ve Hansen, P. (2000). "Variable Neighborhood Search for Extremal Graphs: 1 The Autographix System", *Discrete Mathematics*, 212/1-2, 29-44.

Çatay, B. (2010). "A New Saving-Based Ant Algorithm for The Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Expert Systems with Applications*, 37/10, 6809–6817.

Černý, V. (1985). "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *Journal of Optimization Theory and Applications*, 45/1, 41-51.

Çetin, O. (2013). *Akaryakıt Dağıtımında Araç Rotalama Problemi*, (Basılmamış Doktora Tezi), İstanbul Üniversitesi Sosyal Bilimler Enstitüsü, İstanbul.

Chen, J. F. ve Wu, T. H. (2006). "Vehicle Routing Problem with Simultaneous Deliveries and Pickups", *Journal of the Operational Research Society,* 57/5, 579–587.

Chen, P., Huang, H. ve Dong, X. Y. (2010). "Iterated Variable Neighborhood Descent Algorithm for The Capacitated Vehicle Routing Problem", *Expert Systems with Applications*, 37/2, 1620–1627.

Cheng, R., Gen, M. ve Tsujimura, Y. (1996). "A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms—I. Representation", *Computers & Industrial Engineering*, 30/4, 983–997.

Chopard, B. ve Tomassini, M. (2018). *An Introduction to Metaheuristics for Optimization,* Springer, Cham.

Christofides, N., Mingozzi, A. ve Toth, P. (1979). "The Vehicle Routing Problem". İçinde N. Christofides, A. Mingozzi, P. Toth ve C. Sandi (Eds.). *Combinatorial Optimization,* s. 315-338, John Wiley & Sons, New York.

Chu, P. C. ve Beasley, J. E. (1998). "A Genetic Algorithm for the Multidimensional Knapsack Problem", *Journal of Heuristics,* 4/1, 63–86.

Clarke, G. ve Wright, J. W. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Operations Research*, 12/4, 568-581.

Clerc, M. ve Kennedy, J. (2002). "The Particle Swarm—Explosion, Stability, And Convergence in A Multidimensional Complex Space", *IEEE Transactions on Evolutionary Computation*, 6/1, 58–73.

Consoli, S., Darby-Dowman, K., Mladenović, N. ve Moreno Pérez, J. A. (2009). "Greedy Randomized Adaptive Search and Variable Neighbourhood Search for The Minimum Labelling Spanning Tree Problem", *European Journal of Operational Research*, 196/2, 440-449.

Copado-Méndez, P. J., Blum, C., Guillén-Gosálbez, G. ve Jiménez, L. (2013). "Large Neighbourhood Search Applied to The Efficient Solution of Spatially Explicit Strategic Supply Chain Management Problems", *Computers & Chemical Engineering*, 49, 114–126.

Cordeau, J. F. ve Laporte, G. (2005). "Tabu Search Heuristics for the Vehicle Routing Problem". İçinde R. Sharda, S. Voß, C. Rego ve B. Alidaee (Eds.), *Metaheuristic Optimization via Memory and Evolution*, vol. 30, s. 145-163, Kluwer Academic Publishers, Boston.

Cordeau, J. F. ve Laporte, G. (2007). "The Dial-A-Ride Problem: Models and Algorithms", *Annals of Operations Research*, 153/1, 29-46.

Cordeau, J. F., Laporte, G., Savelsbergh, M. W. P. ve Vigo, D. (2007). "Vehicle Routing". İçinde C. Barnhart ve G. Laporte (Eds.), *Handbooks in Operations Research and Management Science*, s. 367-428, Elsevier.

Corman, F., D'Ariano, A., Pacciarelli, D. ve Pranzo, M. (2010). "A Tabu Search Algorithm for Rerouting Trains During Rail Operations", *Transportation Research Part B: Methodological*, 44/1, 175-192.

Crama, Y. ve Schyns, M. (2003). "Simulated Annealing for Complex Portfolio Selection Problems", *European Journal of Operational Research*, 150/3, 546-571.

Cramer, S. ve Kampouridis, M. (2015). "Optimising the Deployment of Fibre Optics Using Guided Local Search", *2015 IEEE Congress on Evolutionary Computation (CEC)*, Sendai, Japan, 799-806.

Croes, G. A. (1958). "A Method for Solving Traveling-Salesman Problems", *Operations Research*, 6/6, 791–812.

Dantzig, G. B. ve Ramser, J. H. (1959). "The Truck Dispatching Problem", *Management Science*, 6/1, 80-91.

Datta, S., Roy, S. ve Davim, J. P. (2019). "Optimization Techniques: An Overview". İçinde S. Datta ve J. P. Davim (Eds.) *Optimization in industry: Present Practices and Future Scopes*, s. 1-11, Springer, Cham.

Debels, D., De Reyck, B., Leus, R. ve Vanhoucke, M. (2006). "A Hybrid Scatter Search/Electromagnetism Meta-Heuristic for Project Scheduling", *European Journal of Operational Research*, 169/2, 638–653.

Dekkers, A. ve Aarts, E. (1991). "Global Optimization and Simulated Annealing", *Mathematical Programming*, 50, 367-393.

del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C. ve Harley, R. G. (2008). "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems", *IEEE Transactions on Evolutionary Computation*, 12/2, 171–195.

Dethloff, J. (2001). "Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-Up", *OR-Spektrum*, 23/1, 79–96.

Devi, S. G., Selvam, K. ve Rajagopalan, S. P. (2011). "An Abstract to Calculate Big O Factors of Time and Space Complexity of Machine Code", *International Conference on Sustainable Energy and Intelligent Systems* (SEISCON 2011), Chennai, India, 844-847.

Dorigo, M. ve Blum, C. (2005). "Ant Colony Optimization Theory: A Survey", *Theoretical Computer Science*, 344/2-3, 243–278.

Dorigo, M. ve Stützle, T. (2019). "Ant Colony Optimization: Overview and Recent Advances". İçinde M. Gendreau ve J. Y. Potvin (Eds.), *Handbook of Metaheuristics*, s. 311-351, Springer, Cham.

Dorigo, M., Caro, G. D. ve Gambardella, L. M. (1999). "Ant Algorithms for Discrete Optimization", *Artificial Life*, 5/2, 137–172.

Dorigo, M., Maniezzo, V. ve Colorni, A. (1991). "Positive Feedback as a Search Strategy", Dipartimento Di Elettronica, Politecnico di Milano, Teknik Rapor No. 91–016, Milano, Italy, 1-20. https://www.researchgate.net/publication/2573263_Positive_Feedback_as_a_Search_Strategy (26.11.2019).

Downey, A. B. (2012). *Think Complexity,* Version 1.1, Green Tea Press, Massachusetts.

Dréo, J., Siarry, P., Pétrowski, A. ve Taillard, E. (2006). *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer, Berlin, Heidelberg.

Driessel, R. ve Mönch, L. (2011). "Variable Neighborhood Search Approaches for Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times, Precedence Constraints, And Ready Times", *Computers & Industrial Engineering*, 61/2, 336-345.

Dror, M. ve Trudeau, P. (1990). "Split Delivery Routing", *Naval Research Logistics (NRL)*, 37/3, 383-402.

Dueck, G. ve Scheuer, T. (1990). "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior To Simulated Annealing", *Journal of Computational Physics*, 90/1, 161–175.

Eberhart, R. ve Kennedy, J. (1995). "A New Optimizer Using Particle Swarm Theory", *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 39–43.

Egea, J. A., Rodríguez-Fernández, M., Banga, J. R. ve Martí, R. (2007). "Scatter Search for Chemical and Bio-Process Optimization", *Journal of Global Optimization*, 37/3, 481–503.

Eglese, R. W. (1990). "Simulated Annealing: A Tool for Operational Research", *European Journal of Operational Research*, 46/3, 271-281.

Eiben, A. E. ve Smith, J. E. (2015). *Introduction to Evolutionary Computing*, Second Edition, Springer, Berlin, Heidelberg.

Eksioglu, B., Vural, A. V. ve Reisman, A. (2009). "The Vehicle Routing Problem: A Taxonomic Review", *Computers & Industrial Engineering*, 57/4, 1472-1483.

Eryavuz, M. ve Gencer, C. (2001). "Araç Rotalama Problemlerine Ait Bir Uygulama", *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 6/1, 139-155.

Faroe, O., Pisinger, D. ve Zachariasen, M. (2003). "Guided Local Search for Final Placement in VLSI Design", *Journal of Heuristics*, 9/3, 269-295.

Feo, T. A. ve Resende, M. G. C. (1989). "A Probabilistic Heuristic for A Computationally Difficult Set Covering Problem", *Operations Research Letters*, 8/2, 67-71.

Feo, T. A. ve Resende, M. G. C. (1995). "Greedy Randomized Adaptive Search Procedures", *Journal of Global Optimization*, 6/2, 109-133.

Fescioglu-Unver, N. ve Kokar, M. M. (2011). "Self Controlling Tabu Search Algorithm for the Quadratic Assignment Problem", *Computers & Industrial Engineering*, 60/2, 310-319.

Festa, P. (2002). "Greedy Randomized Adaptive Search Procedures", *AIROnews*, 2/4, 7–11.

Festa, P. ve Resende, M. G. C. (2009). "An Annotated Bibliography of GRASP - Part I: Algorithms", *International Transactions in Operational Research*, 16/1, 1-24.

Fisher, M. (1995). "Vehicle routing". İçinde M. O. Ball, T. L. Magnanti, C. L. Monma ve G. L. Nemhauser (Eds.), *Handbooks in Operations Research and Management Science, Network Routing*, Volume 8, s. 1-33, Elsevier.

Fletcher, R. (2000). *Practical Methods of Optimization*, Second Edition, John Wiley & Sons, New York.

Flood, M. F. (1956). "The Traveling-Salesman Problem", *Operations Research*, 4/1, 61-75.

Flores Lucio, G., Reed, M. J. ve Henning, I. D. (2007). "Guided Local Search as A Network Planning Algorithm That Incorporates Uncertain Traffic Demands", *Computer Networks*, 51/11, 3172-3196.

Fortnow, L. (2009). "The Status of The P Versus NP Problem", *Communications of the ACM*, 52/9, 78-86.

Gajpal, Y. ve Abad, P. (2009). "An Ant Colony System (ACS) For Vehicle Routing Problem with Simultaneous Delivery and Pickup", *Computers & Operations Research*, 36/12, 3215–3223.

Gansterer, M., Hartl, R. F. ve Salzmann, P. E. H. (2018). "Exact Solutions for The Collaborative Pickup and Delivery Problem", *Central European Journal of Operations Research*, 26/2, 357-371.

Gendreau, M. (2003). "An Introduction to Tabu Search". İçinde F. Glover ve G. A. Kochenberger (Eds.), *Handbook of Metaheuristics*, s. 37-54, Kluwer Academic Publishers, Boston.

Gendreau, M. ve Potvin, J. Y. (2010). "Tabu Search". İçinde M. Gendreau ve J. Y. Potvin (Eds), *Handbook of Metaheuristics*, Second Edition, s. 41-59, Springer, Boston.

Gillett, B. E. ve Miller, L. R. (1974). "A Heuristic Algorithm for the Vehicle-Dispatch Problem", *Operations Research*, 22/2, 340-349.

Glover, F. (1977). "Heuristics for Integer Programming Using Surrogate Constraints", *Decision Sciences*, 8/1, 156–166.

Glover, F. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers & Operations Research*, 13/5, 533-549.

Glover, F. (1989). "Tabu Search—Part I", *ORSA Journal on Computing*, 1/3, 190-206.

Glover, F. (1998). "A Template for Scatter Search and Path Relinking". İçinde J. Hao, E. Lutton, E. Ronald, M. Schoenauer ve D. Snyers (Eds.), *Artificial Evolution, AE 1997*, s. 1-51, Springer, Berlin, Heidelberg.

Glover, F. ve Laguna, M. (1998). "Tabu Search". İçinde D. Z. Du ve P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, s. 621-757, Kluwer Academic Publishers, Dordrecht.

Glover, F., Kelly, J. P. ve Laguna, M. (1995). "Genetic Algorithms and Tabu Search: Hybrids for Optimization", *Computers & Operations Research*, 22/1, 111-134.

Gokalp, O. ve Ugur, A. (2020). "A Multi-Start ILS–RVND Algorithm with Adaptive Solution Acceptance for the CVRP", *Soft Computing*, 24/4, 2941–2953.

Goksal, F. P., Karaoglan, I. ve Altiparmak, F. (2013). "A Hybrid Discrete Particle Swarm Optimization for Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Computers & Industrial Engineering*, 65/1, 39–53.

Goldreich, O. (2002). *Introduction to Complexity Theory*, Notes for a One Semester Course, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.5606&rep=rep1&type=pdf (22.04.2019).

Goldreich, O. (2010). *P, NP, and NP-Completeness: The Basics of Computational Complexity*, Cambridge University Press, Cambridge.

Gong, G., Deng, Q., Gong, X., Zhang, L., Wang, H. ve Xie, H. (2018). "A Bee Evolutionary Algorithm for Multiobjective Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Mathematical Problems in Engineering*, 2018, 1–21.

González, M. A., Palacios, J. J., Vela, C. R. ve Hernández-Arauzo, A. (2017). "Scatter Search for Minimizing Weighted Tardiness in A Single Machine Scheduling with Setups", *Journal of Heuristics*, 23/2-3, 81–110.

Grady, S. A., Hussaini, M. Y. ve Abdullah, M. M. (2005). "Placement of Wind Turbines Using Genetic Algorithms", *Renewable Energy*, 30/2, 259–270.

Grefenstette, J. J. (1986). "Optimization of Control Parameters for Genetic Algorithms", *IEEE Transactions on Systems, Man, and Cybernetics*, 16/1, 122–128.

Gupta, R. ve Kumar, S. (2021). "Intuitionistic Fuzzy Scale-Invariant Entropy with Correlation Coefficients-Based VIKOR Approach for Multi-Criteria Decision-Making", *Granular Computing*, https://doi.org/10.1007/s41066-020-00252-0 (06.04.2021).

Hafezalkotob, A. ve Hafezalkotob, A. (2016). "Extended MULTIMOORA Method Based on Shannon Entropy Weight for Materials Selection", *Journal of Industrial Engineering International*, 12/1, 1–13.

Hanafi, S. ve Wilbaut, C. (2008). "Scatter Search for the 0–1 Multidimensional Knapsack Problem", *Journal of Mathematical Modelling and Algorithms*, 7/2, 143–159.

Hansen, P. ve Mladenović, N. (1999). "An Introduction to Variable Neighborhood Search". İçinde S. Voß, S. Martello, I. H. Osman ve C. Roucairol (Eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, s. 433–458, Springer, Boston.

Hansen, P. ve Mladenović, N. (2001). "Variable Neighborhood Search: Principles and Applications", *European Journal of Operational Research*, 130/3, 449-467.

Hansen, P. ve Mladenović, N. (2003). "Variable Neighborhood Search". İçinde F. Glover ve G. A. Kochenberger (Eds.), *Handbook of Metaheuristics*, s. 145-184, Kluwer Academic Publishers, Boston.

Hansen, P., Mladenović, N., Todosijević, R. ve Hanafi, S. (2017). "Variable Neighborhood Search: Basics and Variants", *EURO Journal on Computational Optimization*, 5/3, 423-454.

Hezer, S. ve Kara, Y. (2013). "Eşzamanlı Dağıtımlı ve Toplamalı Araç Rotalama Problemlerinin Çözümü İçin Bakteriyel Besin Arama Optimizasyonu Tabanlı Bir Algoritma", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 28/2, 373–382.

Hillier, F. S. ve Lieberman, G. J. (2015). *Introduction to Operations Research*, Tenth Edition, McGraw Hill, New York.

Hof, J. ve Schneider, M. (2019). "An Adaptive Large Neighborhood Search with Path Relinking for A Class of Vehicle-Routing Problems with Simultaneous Pickup and Delivery", *Networks*, 74/3, 207–250.

Homer, S. ve Selman, A. L. (2011). *Computability and Complexity Theory*, Second Edition, Springer, Boston.

Huang, C. L. ve Wang, C. J. (2006). "A GA-based Feature Selection and Parameters Optimization for Support Vector Machines*", Expert Systems with Applications*, 31/2, 231–240.

Irnich, S., Toth, P. ve Vigo, D. (2014). "The Family of Vehicle Routing Problems". İçinde Toth, P. ve Vigo, D. (Eds.), *Vehicle Routing: Problems, Methods, and Applications,* Second Edition, s. 1-33, SIAM Society for Industrial and Applied Mathematics, Philadelphia.

Jaffe, A. M. (2006). "The Millennium Grand Challenge in Mathematics", *Notices of the AMS*, 53/6, 652-660.

Jarboui, B., Cheikh, M., Siarry, P. ve Rebai, A. (2007). "Combinatorial Particle Swarm Optimization (CPSO) for Partitional Clustering Problem", *Applied Mathematics and Computation*, 192/2, 337–345.

Jarboui, B., Derbel, H., Hanafi, S. ve Mladenović, N. (2013). "Variable Neighborhood Search for Location Routing", *Computers & Operations Research*, 40/1, 47-57.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G. ve Węglarz, J. (2001). "Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling", *Annals of Operations Research*, 102/1-4, 137-155.

Jun, Y. ve Kim, B. I. (2012). "New Best Solutions to VRPSPD Benchmark Problems by A Perturbation Based Algorithm", *Expert Systems with Applications*, 39/5, 5641–5648.

Kalayci, C. B. ve Kaya, C. (2016). "An Ant Colony System Empowered Variable Neighborhood Search Algorithm for The Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Expert Systems with Applications*, 66, 163–175.

Karlaftis, M. G., Kepaptsoglou, K. ve Sambracos, E. (2009). "Containership Routing with Time Deadlines and Simultaneous Deliveries and Pick-Ups", *Transportation Research Part E: Logistics and Transportation Review*, 45/1, 210–221.

Kashef, S. ve Nezamabadi-pour, H. (2015). "An Advanced ACO Algorithm for Feature Subset Selection", *Neurocomputing*, 147, 271–279.

Kassem, S. ve Chen, M. (2013). "Solving Reverse Logistics Vehicle Routing Problems with Time Windows", *The International Journal of Advanced Manufacturing Technology*, 68/1-4, 57–68.

Kennedy, J. ve Eberhart, R. (1995). "Particle Swarm Optimization", *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Perth, Australia, 1942–1948.

Kennedy, J. ve Eberhart, R. C. (1997). "A Discrete Binary Version of The Particle Swarm Algorithm", *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, Orlando, USA, 4104–4108.

Keskintürk, T., Topuk, N. ve Özyeşil, O. (2015). "Araç Rotalama Problemleri ve Çözüm Yöntemleri", *İşletme Bilimi Dergisi*, 3/2, 77-107.

Kilby, P. ve Urli, T. (2016). "Fleet Design Optimisation from Historical Data Using Constraint Programming and Large Neighbourhood Search", *Constraints*, 21/1, 2–21.

Kirkpatrick, S., Gelatt, C. D. ve Vecchi, M. P. (1983). "Optimization by Simulated Annealing", *Science*, 220/4598, 671-680.

Koç, Ç., Laporte, G. ve Tükenmez, İ. (2020). "A Review on Vehicle Routing with Simultaneous Pickup and Delivery", *Computers & Operations Research*, 104987. https://doi.org/10.1016/j.cor.2020.104987

Koch, H., Bortfeldt, A. ve Wäscher, G. (2018). "A Hybrid Algorithm for The Vehicle Routing Problem with Backhauls, Time Windows and Three-Dimensional Loading Constraints", *OR Spectrum*, 40/4, 1029–1075.

Korytkowski, P., Rymaszewski, S. ve Wiśniewski, T. (2013). "Ant Colony Optimization for Job Shop Scheduling Using Multi-Attribute Dispatching Rules", *The International Journal of Advanced Manufacturing Technology*, 67/1-4, 231–241.

Kramer, O. (2014). "Iterated Local Search", İçinde O. Kramer (Ed.), *A Brief Introduction to Continuous Evolutionary Optimization*, s. 45-54, Springer, Cham.

Kulturel-Konak, S., Smith, A. E. ve Coit, D. W. (2003). "Efficiently Solving the Redundancy Allocation Problem Using Tabu Search", *IIE Transactions*, 35/6, 515-526.

Kumar, M., Husian, M., Upreti, N. ve Gupta, D. (2010). "Genetic Algorithm: Review and Application", *International Journal of Information Technology and Knowledge Management*, 2/2, 451–454.

Laguna, M. (2014). "Scatter Search". İçinde E. K. Burke ve G. Kendall (Eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, s. 119-141, Springer, Boston.

Lange, K. (2013). *Optimization*, Second Edition. Springer, New York.

Laporte, G. (2007). "What You Should Know About the Vehicle Routing Problem", *Naval Research Logistics (NRL)*, 54/8, 811-819.

Laporte, G. (2009). "Fifty Years of Vehicle Routing", *Transportation Science*, 43/4, 408-416.

Laporte, G. ve Osman, I. H. (1995). "Routing problems: A Bibliography", *Annals of Operations Research*, 61/1, 227-262.

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*, 7/4-5, 285-300.

Laporte, G., Mercure, H. ve Nobert, Y. (1992). "A Branch and Bound Algorithm for a Class of Asymmetrical Vehicle Routeing Problems", *The Journal of the Operational Research Society*, 43/5, 469-481.

Laporte, G., Nobert, Y. ve Desrochers, M. (1985). "Optimal Routing under Capacity and Distance Restrictions", *Operations Research*, 33/5, 1050-1073.

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I. ve Dizdarevic, S. (1999). "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators", *Artificial Intelligence Review*, 13/2, 129–170.

Leivadeas, A., Papagianni, C. ve Papavassiliou, S. (2013). "Efficient Resource Mapping Framework over Networked Clouds via Iterated Local Search-Based Request Partitioning", *IEEE Transactions on Parallel and Distributed Systems*, 24/6, 1077-1086.

Lenstra, J. K. ve Kan, A. H. G. R. (1981). "Complexity of Vehicle Routing and Scheduling Problems", *Networks*, 11/2, 221-227.

Li, J., Pardalos, P. M., Sun, H., Pei, J. ve Zhang, Y. (2015). "Iterated Local Search Embedded Adaptive Neighborhood Selection Approach for The Multi-Depot Vehicle Routing Problem with Simultaneous Deliveries and Pickups", *Expert Systems with Applications*, 42/7, 3551–3561.

Lim, A. ve Wang, F. (2005). "Multi-Depot Vehicle Routing Problem: A One-Stage Approach", *IEEE Transactions on Automation Science and Engineering*, 2/4, 397-402.

Lin, S. (1965). "Computer Solutions of The Traveling Salesman Problem", *The Bell System Technical Journal*, 44/10, 2245-2269.

Liu, F. B. (2008). "A Modified Genetic Algorithm for Solving the Inverse Heat Transfer Problem of Estimating Plan Heat Source", *International Journal of Heat and Mass Transfer*, 51/15-16, 3745–3752.

Liu, R., Xie, X., Augusto, V. ve Rodriguez, C. (2013). "Heuristic Algorithms for A Vehicle Routing Problem with Simultaneous Delivery and Pickup and Time Windows in Home Health Care", *European Journal of Operational Research*, 230/3, 475–486.

López-Sánchez, A. D., Sánchez-Oro, J. ve Hernández-Díaz, A. G. (2019). "GRASP and VNS for Solving The P-Next Center Problem", *Computers & Operations Research*, 104, 295-303.

Lotfi, F. H. ve Fallahnejad, R. (2010). "Imprecise Shannon's Entropy and Multi Attribute Decision Making", *Entropy*, 12/1, 53–62.

Lourenço, H. R., Martin, O. C. ve Stützle, T. (2001). "A Beginner's Introduction to Iterated Local Search", *Proceedings of the 4th Metaheuristics International Conference*, Porto, Portugal, 1-6.

Lourenço, H. R., Martin, O. C. ve Stützle, T. (2019). "Iterated Local Search: Framework and Applications". İçinde Gendreau, M. ve Potvin, J. Y. (Eds), *Handbook of Metaheuristics*, Third Edition, s. 129-168, Springer, Cham.

Lucay, F. A., Gálvez, E. D. ve Cisternas, L. A. (2019). "Design of Flotation Circuits Using Tabu-Search Algorithms: Multispecies, Equipment Design, and Profitability Parameters", *Minerals*, 9/3, 181.

Majidi, S., Hosseini-Motlagh, S. M. ve Ignatius, J. (2018). "Adaptive Large Neighborhood Search Heuristic for Pollution-Routing Problem with Simultaneous Pickup and Delivery", *Soft Computing*, 22/9, 2851–2865.

Makhadmeh, S. N., Khader, A. T., Al-Betar, M. A., Naim, S., Abasi, A. K. ve Alyasseri, Z. A. A. (2019). "Optimization Methods for Power Scheduling Problems in Smart Home: Survey", *Renewable and Sustainable Energy Reviews*, 115, 109362.

Marinaki, M. ve Marinakis, Y. (2015). "A Hybridization of Clonal Selection Algorithm with Iterated Local Search and Variable Neighborhood Search for The Feature Selection Problem", *Memetic Computing*, 7/3, 181-201.

Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M. ve Baesens, B. (2007). "Classification with Ant Colony Optimization", *IEEE Transactions on Evolutionary Computation*, 11/5, 651–665.

Martí, R., Corberán, Á. ve Peiró, J. (2018). "Scatter Search". In R. Martí, P. M. Pardalos ve M. G. C. Resende (Eds.), *Handbook of Heuristics*, s. 717-740, Springer, Cham.

Miller, B. L. ve Goldberg, D. E. (1995). "Genetic Algorithms, Tournament Selection, and the Effects of Noise", *Complex Systems*, 9/3, 193–212.

Mills, P., Tsang, E. ve Ford, J. (2003). "Applying an Extended Guided Local Search to the Quadratic Assignment Problem", *Annals of Operations Research*, 118/1-4, 121-135.

Min, H. (1989). "The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points", *Transportation Research Part A: General*, 23/5, 377–386.

Mingyong, L. ve Erbao, C. (2010). "An Improved Differential Evolution Algorithm for Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows", *Engineering Applications of Artificial Intelligence*, 23/2, 188–195.

Mirjalili, S. (2019). "Genetic Algorithm". İçinde *Evolutionary Algorithms and Neural Networks: Theory and Applications*, vol. 780, s. 43-55, Springer, Cham.

Mladenović, N. ve Hansen, P. (1997). "Variable Neighborhood Search", *Computers & Operations Research*, 24/11, 1097-1100.

Monçores, M. C., Alvim, A. C. F. ve Barros, M. O. (2018). "Large Neighborhood Search Applied to The Software Module Clustering Problem", *Computers & Operations Research*, 91, 92–111.

Montané, F. A. T. ve Galvão, R. D. (2006). "A Tabu Search Algorithm for The Vehicle Routing Problem with Simultaneous Pick-Up and Delivery Service", *Computers & Operations Research*, 33/3, 595–619.

Moore, C. ve Mertens, S. (2011). *The Nature of Computation*. Oxford University Press, Oxford.

Mu, D., Wang, C., Zhao, F. ve Sutherland, J. W. (2016). "Solving Vehicle Routing Problem with Simultaneous Pickup and Delivery Using Parallel Simulated Annealing Algorithm", *International Journal of Shipping and Transport Logistics*, 8/1, 81-106.

Mucuk, İ. (2016). *Modern İşletmecilik*, Gözden Geçirilmiş 20. Basım, Türkmen Kitabevi, İstanbul.

Naddef, D. ve Rinaldi, G. (2002). "Branch-and-Cut Algorithms for the Capacitated VRP". İçinde Toth, P. ve Vigo, D. (Eds.), *The Vehicle Routing Problem*, s. 53-84, SIAM Society for Industrial and Applied Mathematics, Philadelphia.

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*, 162/1, 126–141.

Nash, J. C. (2000). "The (Dantzig) Simplex Method for Linear Programming", *Computing in Science & Engineering*, 2/1, 29–31.

Nastasi, G., Colla, V., Cateni, S. ve Campigli, S. (2018). "Implementation and Comparison of Algorithms for Multi-Objective Optimization Based on Genetic Algorithms Applied to The Management of An Automated Warehouse", *Journal of Intelligent Manufacturing*, 29/7, 1545–1557.

Nilsson, N. J. (2010). *Yapay Zekâ: Geçmişi ve Geleceği*, (çev. Mehmet Doğan), Boğaziçi Üniversitesi Yayınevi, İstanbul.

Ólafsson, S. (2006). "Metaheuristics". İçinde S. G. Henderson ve B. L. Nelson (Eds.), *Handbooks in Operations Research and Management Science*, vol. 13, s. 633-654, Elsevier.

Olgun, B., Koç, Ç. ve Altıparmak, F. (2021). "A Hyper Heuristic for The Green Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Computers & Industrial Engineering*, 153, 107010.

Osman, I. H. (1993). "Metastrategy Simulated Annealing and Tabu Search Algorithms for The Vehicle Routing Problem", *Annals of Operations Research*, 41/4, 421–451.

Osman, I. H. ve Potts, C. N. (1989). "Simulated Annealing for Permutation Flow-Shop Scheduling", *Omega*, 17/6, 551-557.

Pacino, D. ve Van Hentenryck, P. (2011). "Large Neighborhood Search and Adaptive Randomized Decompositions for Flexible Jobshop Scheduling", *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, s. 1997-2002, AAAI Press, Barcelona, Spain.

Pampara, G., Franken, N. ve Engelbrecht, A. P. (2005). "Combining Particle Swarm Optimisation with Angle Modulation to Solve Binary Problems", *2005 IEEE Congress on Evolutionary Computation*, vol.1, s. 89-96, Edinburgh, Scotland, UK.

Pan, Q. K. ve Ruiz, R. (2012). "Local Search Methods for the Flowshop Scheduling Problem with Flowtime Minimization", *European Journal of Operational Research*, 222/1, 31-43.

Pantrigo, J. J., Martí, R., Duarte, A. ve Pardo, E. G. (2012). "Scatter Search for The Cutwidth Minimization Problem", *Annals of Operations Research*, 199/1, 285–304.

Papadimitriou, C. H. ve Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, New York.

Pardalos, P. M., Qian, T. ve Resende, M. G. C. (1998). "A Greedy Randomized Adaptive Search Procedure for the Feedback Vertex Set Problem", *Journal of Combinatorial Optimization*, 2/4, 399-412.

Park, J. B., Lee, K. S., Shin, J. R. ve Lee, K. Y. (2005). "A Particle Swarm Optimization for Economic Dispatch with Nonsmooth Cost Functions", *IEEE Transactions on Power Systems*, 20/1, 34–42.

Penna, P. H. V., Subramanian, A. ve Ochi, L. S. (2013). "An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem", *Journal of Heuristics*, 19/2, 201-232.

Pham, D. T. ve Karaboga, D. (2000). *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, Springer, London.

Pillac, V., Gendreau, M., Guéret, C. ve Medaglia, A. L. (2013). "A Review of Dynamic Vehicle Routing Problems", *European Journal of Operational Research*, 225/1, 1-11.

Pisinger, D. ve Ropke, S. (2010). "Large Neighborhood Search". İçinde M. Gendreau ve S. Ropke (Eds.), *Handbook of Metaheuristics*, Second Edition, s. 399-419, Springer, Boston.

Polat, O. (2017). "A Parallel Variable Neighborhood Search for The Vehicle Routing Problem with Divisible Deliveries and Pickups", *Computers & Operations Research*, 85, 71–86.

Polat, O., Kalayci, C. B., Kulak, O. ve Günther, H. O. (2015). "A Perturbation Based Variable Neighborhood Search Heuristic for Solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery with Time Limit", *European Journal of Operational Research,* 242/2, 369–382.

Psaraftis, H. N. (1980). "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem", *Transportation Science,* 14/2, 130-154.

Qin, G., Tao, F., Li, L. ve Chen, Z. (2019). "Optimization of The Simultaneous Pickup and Delivery Vehicle Routing Problem Based on Carbon Tax", *Industrial Management & Data Systems,* 119/9, 2055–2071.

Rajendran, C. ve Ziegler, H. (2004). "Ant-Colony Algorithms for Permutation Flowshop Scheduling to Minimize Makespan/Total Flowtime of Jobs", *European Journal of Operational Research,* 155/2, 426–438.

Rardin, R. L. (2017). *Optimization in Operations Research,* Second Edition, Pearson, New Jersey.

Reeves, C. R. ve Rowe, J. E. (2002). *Genetic Algorithms: Principles and Perspectives A Guide to GA Theory,* Kluwer Academic Publishers, Boston.

Renner, G. ve Ekárt, A. (2003). "Genetic Algorithms in Computer Aided Design", *Computer-Aided Design,* 35/8, 709–726.

Resende, M. G. C. (2008). "Metaheuristic Hybridization with Greedy Randomized Adaptive Search Procedures". İçinde Z. L. Chen, S. Raghavan, P. Gray ve H. J. Greenberg (Eds), *State-of-the-Art Decision-Making Tools in the Information-Intensive Age,* s. 295-319, Institute for Operations Research and the Management Sciences (INFORMS), Maryland.

Resende, M. G. C. Ribeiro, C. C., Glover, F. ve Martí, R. (2010). "Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications". İçinde M. Gendreau ve J.Y. Potvin (Eds.), *Handbook of Metaheuristics,* Second Edition, s. 87-107, Springer, Boston.

Resende, M. G. C. ve Ribeiro, C. C. (2014). "GRASP: Greedy Randomized Adaptive Search Procedures". İçinde E. K. Burke ve G. Kendall (Eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques,* s. 287-312, Springer, Boston.

Roberge, V., Tarbouchi, M. ve Labonté, G. (2013). "Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning", *IEEE Transactions on Industrial Informatics,* 9/1, 132–141.

Rodríguez, D. A., Oteiza, P. P. ve Brignole, N. B. (2013). "Simulated Annealing Optimization for Hydrocarbon Pipeline Networks", *Industrial & Engineering Chemistry Research,* 52/25, 8579-8588.

Ropke, S. ve Pisinger, D. (2006). "A Unified Heuristic for A Large Class of Vehicle Routing Problems with Backhauls", *European Journal of Operational Research*, 171/3, 750–775.

Salhi, S. ve Nagy, G. (1999). "A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling", *Journal of the Operational Research Society,* 50/10, 1034–1042.

Saravanan, M., Noorul Haq, A., Vivekraj, A. R. ve Prasad, T. (2008). "Performance Evaluation of The Scatter Search Method for Permutation Flowshop Sequencing Problems", *The International Journal of Advanced Manufacturing Technology*, 37/11-12, 1200-1208.

Sastry, K., Goldberg, D. E. ve Kendall, G. (2014). "Genetic Algorithms". İçinde E. K. Burke & G. Kendall (Eds.), *Search Methodologies,* s. 93-117, Springer, Boston.

Savelsbergh, M. W. P. (1992). "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration", *ORSA Journal on Computing*, 4/2, 146–154.

Say, C. (2018). *50 Soruda Yapay Zekâ*, Yedinci Baskı, Bilim ve Gelecek Kitaplığı, İstanbul.

Sezen, H. K. (2007). *Yöneylem Araştırması*, Genişletilmiş İkinci Baskı, Ekin Yayınevi, Bursa.

Shannon, C. E. (1948). "A Mathematical Theory of Communication", *Bell System Technical Journal*, 27/3, 379–423.

Shaw, P. (1998). "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems". İçinde M. Maher ve J. F. Puget (Eds.), *Principles and Practice of Constraint Programming—CP98*. Lecture Notes in Computer Science, vol. 1520, s. 417-431, Springer, Berlin, Heidelberg.

Shemshadi, A., Shirazi, H., Toreihi, M. ve Tarokh, M. J. (2011). "A Fuzzy VIKOR Method for Supplier Selection Based on Entropy Measure for Objective Weighting", *Expert Systems with Applications*, 38/10, 12160–12167.

Shen, Q., Chen, H., Chu, F. ve Zhou, M. (2011). "Multi-Mode Transportation Planning of Crude Oil Via Greedy Randomized Adaptive Search and Path Relinking", *Transactions of the Institute of Measurement and Control*, 33/3-4, 456-475.

Shi, Y. ve Eberhart, R. (1998). "A Modified Particle Swarm Optimizer", *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence* (Cat. No.98TH8360), s. 69–73, Anchorage, USA.

Socha, K. ve Dorigo, M. (2008). "Ant Colony Optimization for Continuous Domains", *European Journal of Operational Research*, 185/3, 1155–1173.

Sörensen, K. (2015). "Metaheuristics—The Metaphor Exposed", *International Transactions in Operational Research,* 22/1, 3-18.

Souza, M. J. F., Mine, M. T., Silva, M. de S. A., Ochi, L. S. ve Subramanian, A. (2011). "A Hybrid Heuristic, Based on Iterated Local Search and GENIUS, for The Vehicle Routing Problem with Simultaneous Pickup and Delivery", *International Journal of Logistics Systems and Management*, 10/2, 142-157.

Steitz, W. ve Rothlauf, F. (2012). "Using Penalties Instead of Rewards: Solving OCST Problems with Guided Local Search", *Swarm and Evolutionary Computation*, 3, 46-53.

Stützle, T. ve Ruiz, R. (2018). "Iterated Local Search". İçinde R. Martí, P. M. Pardalos ve M. G. C. Resende (Eds.), *Handbook of Heuristics*, s. 579-605, Springer, Cham.

Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S. ve Farias, R. (2010). "A Parallel Heuristic for The Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Computers & Operations Research*, 37/11, 1899–1911.

Subramanian, A., Ochi, L. S. ve Cabral, L. A. F. (2008). "An efficient ILS heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery", Instituto de Computação - Universidade Federal Fluminense, Teknik Rapor, RT 07/08, Brazil, 1-17. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.491.1192&rep=rep1&type=pdf (26.07.2021).

Taillard, É. D., Laporte, G. ve Gendreau, M. (1996). "Vehicle Routeing with Multiple Use of Vehicles", *The Journal of the Operational Research Society*, 47/8, 1065-1070.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F. ve Potvin, J. Y. (1997). "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", *Transportation Science*, 31/2, 170–186.

Tairan, N., Algarni, A., Varghese, J. ve Jan, M. A. (2015). "Population-Based Guided Local Search for Multidimensional Knapsack Problem", *2015 Fourth International Conference on Future Generation Communication Technology (FGCT)*, s. 1-5, Luton, UK.

Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation*, John Wiley & Sons, New Jersey.

Tasan, A. S. ve Gen, M. (2012). "A Genetic Algorithm Based Approach to Vehicle Routing Problem with Simultaneous Pick-Up and Deliveries", *Computers & Industrial Engineering*, 62/3, 755–761.

Taşkın, Ç. ve Emel, G. G. (2009). *Sayısal Yöntemlerde Genetik Algoritmalar*, Alfa Akademi, Bursa.

Tavares, R. S., Martins, T. C. ve Tsuzuki, M. S. G. (2011). "Simulated Annealing with Adaptive Neighborhood: A Case Study in Off-Line Robot Path Planning", *Expert Systems with Applications*, 38/4, 2951-2965.

The Julia Language. (2021). *The Julia Project v1.6*, https://raw.githubusercontent.com/JuliaLang/docs.julialang.org/assets/julia-1.6.1.pdf (19.05.2021).

Toksari, M. D. (2016). "A Hybrid Algorithm of Ant Colony Optimization (ACO) and Iterated Local Search (ILS) for Estimating Electricity Domestic Consumption: Case of Turkey", *International Journal of Electrical Power & Energy Systems*, 78, 776-782.

Toth, P. ve Vigo, D. (1998). "Exact Solution of the Vehicle Routing Problem". İçinde Crainic, T. G. ve Laporte, G. (Eds.), *Fleet Management and Logistics*, s. 1-31, Kluwer Academic Publishers, Boston.

Toth, P. ve Vigo, D. (2002). "Branch-and-Bound Algorithms for the Capacitated VRP". İçinde Toth, P. ve Vigo, D. (Eds.), *The vehicle routing problem*, s. 29- 51, SIAM Society for Industrial and Applied Mathematics, Philadelphia.

Tsang, E. ve Voudouris, C. (1997). "Fast Local Search and Guided Local Search and Their Application to British Telecom's Workforce Scheduling Problem", *Operations Research Letters*, 20/3, 119-127.

Türk Dil Kurumu (2006). *Güncel Türkçe Sözlük,* http://tdk.gov.tr/index.php?option=com_gts&arama=gts&guid=TDK.GTS.5cb46c8fcb2619.39477237 (15.04.2019).

Vergeylen, N., Sörensen, K. ve Vansteenwegen, P. (2019). "Large Neighborhood Search for The Bike Request Scheduling Problem", *International Transactions in Operational Research*. https://doi.org/10.1111/itor.12688

Voudouris, C. ve Tsang, E. (1995). "Guided Local Search", University of Essex, Teknik Rapor, CSM-247, Colchester, United Kingdom, http://www.bracil.net/CSP/papers/VouTsa-GlsTsp-CSM-247-1995.pdf (09.09.2019).

Voudouris, C. ve Tsang, E. (1999). "Guided Local Search and Its Application to The Traveling Salesman Problem", *European Journal of Operational Research*, 113/2, 469-499.

Voudouris, C. ve Tsang, E. P. K. (2003). "Guided Local Search". İçinde F. Glover ve G. A. Kochenberger (Eds.), *Handbook of Metaheuristics*, Second Edition, s. 185-218, Springer, Boston.

Wang, C., Mu, D., Zhao, F. ve Sutherland, J. W. (2015). "A Parallel Simulated Annealing Method for The Vehicle Routing Problem with Simultaneous Pickup–Delivery and Time Windows", *Computers & Industrial Engineering*, 83, 111–122.

Wang, H. F. ve Chen, Y. Y. (2012). "A Genetic Algorithm for The Simultaneous Delivery and Pickup Problems with Time Window", *Computers & Industrial Engineering*, 62/1, 84–95.

Wang, H. M., Chou, F. D. ve Wu, F. C. (2011). "A Simulated Annealing for Hybrid Flow Shop Scheduling with Multiprocessor Tasks to Minimize Makespan", *The International Journal of Advanced Manufacturing Technology*, 53/5-8, 761-776.

Wang, Y. ve Liu, J. H. (2010). "Chaotic Particle Swarm Optimization for Assembly Sequence Planning", *Robotics and Computer-Integrated Manufacturing*, 26/2, 212–222.

Wang, Z., Zhang, J. ve Yang, S. (2019). "An Improved Particle Swarm Optimization Algorithm for Dynamic Job Shop Scheduling Problems with Random Job Arrivals", *Swarm and Evolutionary Computation*, 51, 100594.

Wassan, N. A., Wassan, A. H. ve Nagy, G. (2008). "A Reactive Tabu Search Algorithm for The Vehicle Routing Problem with Simultaneous Pickups and Deliveries", *Journal of Combinatorial Optimization*, 15/4, 368–386.

Winston, W. L. (2004). *Operations Research: Applications and Algorithms*, Fourth Edition, Thomson Brooks/Cole, California.

Wu, J., Sun, J., Liang, L. ve Zha, Y. (2011). "Determination of Weights for Ultimate Cross Efficiency Using Shannon Entropy", *Expert Systems with Applications*, 38/5, 5162–5165.

Xu, Y. ve Qu, R. (2012). "A Hybrid Scatter Search Meta-Heuristic for Delay-Constrained Multicast Routing Problems", *Applied Intelligence*, 36/1, 229–241.

Yang, X. S. (2018). *Optimization Techniques and Applications with Examples*, John Wiley & Sons, New Jersey.

Yazdani, M., Torkayesh, A. E., Santibanez-Gonzalez, E. D. ve Otaghsara, S. K. (2020). "Evaluation of Renewable Energy Resources Using Integrated Shannon Entropy—EDAS Model", *Sustainable Operations and Computers*, 1, 35–42.

Yousefikhoshbakht, M., Didehvar, F. ve Rahmati, F. (2014). "A Combination of Modified Tabu Search and Elite Ant System to Solve the Vehicle Routing Problem with Simultaneous Pickup and Delivery", *Journal of Industrial and Production Engineering*, 31/2, 65–75.

Zachariadis, E. E., Tarantilis, C. D. ve Kiranoudis, C. T. (2009). "A Hybrid Metaheuristic Algorithm for The Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Service", *Expert Systems with Applications*, 36/2, Part 1, 1070–1081.

Zachariadis, E. E., Tarantilis, C. D. ve Kiranoudis, C. T. (2010). "An Adaptive Memory Methodology for The Vehicle Routing Problem with Simultaneous Pick-Ups and Deliveries", *European Journal of Operational Research*, 202/2, 401–411.

Zachariadis, E. E., Tarantilis, C. D. ve Kiranoudis, C. T. (2016). "The Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries and Two-Dimensional Loading Constraints", *European Journal of Operational Research*, 251/2, 369–386.

Zachariadis, E. E., Tarantilis, C. D. ve Kiranoudis, C. T. (2017). "Vehicle Routing Strategies for Pick-Up and Delivery Service Under Two Dimensional Loading Constraints", *Operational Research,* 17/1, 115–143.

Zhang, H. ve Sun, G. (2002). "Feature Selection Using Tabu Search Method", *Pattern Recognition,* 35/3, 701-711.

Zhang, X., Gao, L., Wen, L. ve Huang, Z. (2018). "A Hybrid Algorithm Based on Tabu Search and Large Neighbourhood Search for Car Sequencing Problem*", Journal of Central South University,* 25/2, 315–330.

Zhang, Z., Cheang, B., Li, C. ve Lim, A. (2019). "Multi-Commodity Demand Fulfillment Via Simultaneous Pickup and Delivery for A Fast Fashion Retailer", *Computers & Operations Research,* 103, 81–96.

EKLER

Ek 1. SCA Problemleri İçin Belirlenen Rastgele Çekirdek Değerleri

Problem	Kullanılan Çekirdek Değerleri																														
SCA3-0	838	658	434	173	927	912	82	291	789	309	288	899	116	188	203	256	192	468	551	187	988	480	45	711	493	81	503	465	604	641
SCA3-1	506	932	45	767	951	551	546	668	941	552	378	471	44	688	209	967	413	398	971	919	813	838	262	711	489	84	976	87	450	282
SCA3-2	860	52	310	234	505	383	941	730	478	863	556	724	862	192	351	566	543	875	823	50	93	891	981	66	404	92	649	80	634	264
SCA3-3	237	407	344	715	41	288	593	531	467	405	890	904	165	787	893	2	6	17	341	709	681	585	219	275	496	182	617	873	779	372
SCA3-4	665	703	595	251	475	78	53	928	623	16	471	959	472	108	518	531	660	930	323	984	771	877	698	787	534	798	722	796	247	664
SCA3-5	185	19	211	77	48	454	245	613	448	462	559	82	877	302	651	723	736	95	190	599	932	129	299	321	611	853	84	849	325	94
SCA3-6	176	841	669	592	96	789	862	901	479	257	729	129	443	206	408	598	492	118	86	870	84	347	287	647	610	391	454	142	366	402
SCA3-7	663	683	729	135	112	723	748	188	806	127	785	503	915	151	519	154	395	891	153	769	515	240	524	552	243	994	337	894	604	450
SCA3-8	87	880	473	385	2	353	513	786	531	292	365	9	738	979	364	253	589	962	848	335	444	37	633	325	693	992	644	33	505	425
SCA3-9	367	261	487	156	847	144	272	401	856	985	125	906	389	560	779	798	384	296	963	812	633	300	523	924	823	719	231	894	29	653
SCA8-0	145	968	866	480	271	844	299	616	552	947	198	520	150	295	986	998	644	162	348	255	93	553	416	321	285	830	81	637	666	4
SCA8-1	268	194	397	955	93	780	508	981	775	215	874	964	919	383	439	526	522	971	493	455	692	729	240	237	46	765	681	770	285	922
SCA8-2	776	791	493	662	886	599	423	600	265	75	861	69	405	974	684	997	908	301	272	66	616	227	972	288	909	934	706	390	525	962
SCA8-3	776	429	879	324	229	226	607	441	903	613	517	687	266	317	11	793	528	146	999	784	794	854	591	474	870	20	869	343	205	254
SCA8-4	451	555	316	595	763	557	927	377	376	733	937	132	793	114	512	673	393	799	263	744	391	695	517	272	494	836	384	903	467	92
SCA8-5	729	786	309	675	36	932	664	584	668	655	433	717	81	387	488	628	405	617	624	914	58	634	298	853	153	368	245	801	927	192
SCA8-6	986	775	815	878	255	890	661	564	392	246	189	285	722	441	433	978	210	533	599	713	224	494	882	242	920	711	689	746	877	56
SCA8-7	812	344	830	397	270	754	682	926	440	205	127	981	8	960	445	530	472	693	861	961	351	844	335	201	97	262	744	188	579	858
SCA8-8	838	326	347	474	702	533	501	448	54	315	29	921	887	532	905	536	126	640	810	296	882	826	781	756	915	895	373	561	441	769
SCA8-9	492	231	814	102	145	990	846	174	794	960	65	746	522	817	740	562	109	885	564	718	593	352	457	643	579	609	763	766	373	247

Ek 2. CON Problemleri İçin Belirlenen Rastgele Çekirdek Değerleri

Problem	Kullanılan Çekirdek Değerleri																														
CON3-0	859	65	422	530	745	670	440	824	86	492	771	336	52	965	146	210	621	457	625	26	130	725	95	581	850	881	53	56	119	96
CON3-1	342	19	868	623	262	360	621	329	140	537	97	771	977	309	690	931	954	229	492	341	908	337	389	499	945	796	211	999	497	885
CON3-2	323	962	308	409	588	924	643	73	209	139	131	193	392	395	913	679	717	112	76	424	352	319	633	766	493	915	421	169	238	32
CON3-3	213	151	317	372	416	362	740	831	874	800	406	875	570	95	499	686	401	725	92	467	468	991	233	482	893	555	850	222	496	735
CON3-4	65	164	294	590	842	899	216	231	419	923	72	839	8	936	737	698	864	55	30	130	837	927	111	38	648	660	859	581	401	451
CON3-5	424	822	998	643	130	264	620	945	661	881	709	916	432	419	979	766	867	341	282	449	174	654	636	92	324	850	109	386	43	41
CON3-6	348	196	82	474	678	893	379	192	455	889	278	357	358	301	85	43	422	509	888	682	695	461	484	271	954	745	41	694	132	782
CON3-7	760	332	806	99	256	836	777	727	696	250	891	846	668	216	373	34	781	137	450	926	578	809	662	279	705	263	168	609	179	125
CON3-8	3	27	810	799	880	364	202	216	414	314	831	348	346	75	4	105	995	312	962	22	526	948	169	543	703	435	906	561	214	784
CON3-9	541	133	814	727	982	558	166	546	24	216	867	531	957	142	589	201	92	130	737	853	928	559	348	809	901	985	131	180	277	22
CON8-0	611	827	70	299	164	919	677	193	459	786	234	904	240	668	5	787	118	952	473	462	417	841	207	185	253	280	596	716	267	607
CON8-1	27	846	743	735	524	288	495	134	952	387	163	636	199	727	261	335	574	326	84	599	233	106	532	90	552	498	648	762	167	484
CON8-2	891	683	524	54	49	301	201	596	551	291	57	350	642	568	853	880	824	77	197	915	216	332	281	474	849	527	952	622	986	770
CON8-3	630	307	891	632	639	43	958	815	90	205	97	77	292	804	959	592	697	484	499	94	553	754	558	629	831	403	515	511	316	297
CON8-4	285	171	778	769	472	32	400	362	926	631	776	138	265	418	626	838	564	681	237	839	429	976	427	967	947	391	249	49	884	42
CON8-5	75	110	284	397	206	606	333	929	176	619	898	861	999	483	240	86	842	756	768	116	598	468	196	164	398	143	913	720	967	906
CON8-6	574	124	482	495	58	797	657	730	588	61	158	220	663	310	59	998	119	79	932	309	333	894	78	217	195	771	975	24	222	830
CON8-7	202	675	609	791	953	908	990	448	364	189	706	896	401	184	709	758	283	685	181	541	19	688	819	871	108	9	240	862	330	408
CON8-8	721	259	607	112	819	790	176	797	188	340	669	553	961	541	372	267	873	273	248	710	630	243	491	516	611	151	686	4	120	973
CON8-9	96	154	667	581	529	951	786	659	570	488	791	618	32	742	78	373	761	483	848	165	103	151	157	850	348	452	359	969	613	690

Ek 3. CMT Problemleri İçin Belirlenen Rastgele Çekirdek Değerleri

Problem	Kullanılan Çekirdek Değerleri																													
CMT1X	62	328	816	903	506	253	799	15	376	282	728	594	9	39	377	175	293	788	255	637	869	18	944	196	808	724	230	364	674	669
CMT1Y	200	500	448	436	237	101	635	961	970	711	112	520	842	659	716	449	579	252	672	817	493	277	102	786	362	737	859	637	192	451
CMT2X	31	398	603	300	45	346	647	67	165	204	690	800	413	517	234	283	524	251	629	466	348	778	254	386	438	482	324	152	514	868
CMT2Y	952	307	285	898	429	527	588	312	714	136	555	733	57	657	828	752	594	890	823	822	123	847	883	879	984	380	455	899	234	184
CMT3X	20	376	450	832	332	892	733	38	594	140	689	250	538	65	357	793	666	109	660	453	399	414	128	937	862	447	86	349	478	6
CMT3Y	64	204	659	907	942	127	981	612	649	895	461	692	327	531	731	298	576	289	391	147	383	264	561	657	549	324	867	389	194	740
CMT4X	478	418	291	904	645	721	268	660	960	285	670	921	730	980	290	736	814	355	84	220	555	21	739	377	358	634	28	275	769	409
CMT4Y	634	565	741	295	318	70	861	386	167	105	782	756	575	174	432	219	815	571	559	372	958	843	856	868	228	717	920	141	233	623
CMT5X	948	278	895	753	434	415	112	172	947	835	226	527	944	648	749	560	626	842	246	833	946	722	345	766	595	120	132	776	485	633
CMT5Y	567	714	745	129	177	787	658	238	886	515	160	380	347	399	555	694	377	927	68	51	600	783	237	512	301	443	119	66	733	83
CMT11X	179	447	313	889	863	739	881	535	478	551	81	217	140	241	790	971	818	886	908	667	556	663	857	391	719	465	578	920	815	963
CMT11Y	668	466	585	487	506	851	17	787	561	602	329	823	43	48	133	637	36	464	820	578	216	451	149	139	611	385	7	99	723	353
CMT12X	387	12	246	378	669	611	569	247	455	976	775	762	453	760	859	752	697	873	232	719	875	385	495	293	270	830	31	111	732	519
CMT12Y	271	139	248	272	25	258	442	463	768	67	466	715	504	799	510	632	283	615	827	216	243	319	882	513	55	226	589	796	633	974

Ek 4. SCA Problemleri İçin Algoritmaların En İyi Çözüm Değerleri ve Ortalama Hesaplama Süreleri

	ACS		VLBR		PILS		HPSO		ACSEVNS		ALNS-PR		ILS-RVND-TA	
Problem	Çözüm	Süre	Çözüm	Süre*	Çözüm	Süre	Çözüm	Süre	Çözüm	Süre	Çözüm	Süre	Çözüm	Süre
SCA3-0	**635,62**	6,00	**635,62**	2,50	**635,62**	2,31	**635,62**	4,90	**635,62**	4,77	**635,62**	19,37	**635,62**	10,56
SCA3-1	**697,84**	6,00	**697,84**	2,50	**697,84**	2,28	**697,84**	0,80	**697,84**	5,24	**697,84**	14,59	**697,84**	9,75
SCA3-2	**659,34**	6,00	**659,34**	2,90	**659,34**	2,14	**659,34**	0,40	**659,34**	7,47	**659,34**	22,38	**659,34**	9,75
SCA3-3	**680,04**	6,10	**680,04**	2,30	**680,04**	2,49	**680,04**	1,00	**680,04**	5,20	**680,04**	11,74	**680,04**	10,35
SCA3-4	**690,50**	5,70	**690,50**	2,90	**690,50**	2,18	**690,50**	0,30	**690,50**	4,96	**690,50**	11,95	**690,50**	9,24
SCA3-5	**659,90**	5,10	**659,90**	3,00	**659,90**	2,23	**659,90**	2,00	**659,91**	5,18	**659,90**	16,41	**659,90**	9,35
SCA3-6	**651,09**	6,10	**651,09**	3,10	**651,09**	2,51	**651,09**	0,80	**651,09**	4,68	**651,09**	17,31	**651,09**	10,57
SCA3-7	**659,17**	6,80	**659,17**	2,80	**659,17**	2,49	**659,17**	1,60	**659,17**	6,06	**659,17**	12,38	**659,17**	9,44
SCA3-8	**719,47**	5,40	**719,47**	3,50	**719,48**	2,26	**719,47**	0,50	**719,48**	4,51	**719,47**	12,73	**719,47**	10,14
SCA3-9	**681,00**	6,00	**681,00**	4,70	**681,00**	1,90	**681,00**	0,80	**681,00**	7,08	**681,00**	15,45	**681,00**	9,15
SCA8-0	**961,50**	11,00	**961,50**	2,70	**961,50**	3,37	**961,50**	4,80	**961,50**	5,33	**961,50**	11,69	**961,50**	7,50
SCA8-1	**1049,65**	11,50	**1049,65**	3,80	**1049,65**	2,89	**1049,65**	6,80	**1049,65**	5,62	**1049,65**	7,33	**1049,65**	8,73
SCA8-2	1042,69	11,90	**1039,64**	3,90	**1039,64**	2,38	**1039,64**	10,20	**1039,64**	6,05	**1039,64**	9,42	**1039,64**	14,48
SCA8-3	**983,34**	11,30	**983,34**	2,60	**983,34**	2,98	**983,34**	13,00	**983,34**	8,39	**983,34**	9,24	**983,34**	8,34
SCA8-4	**1065,49**	11,10	**1065,49**	2,40	**1065,49**	2,81	**1065,49**	3,00	**1065,49**	6,07	**1065,49**	7,96	**1065,49**	9,01
SCA8-5	**1027,08**	11,30	**1027,08**	3,40	**1027,08**	3,31	**1027,08**	4,10	**1027,08**	6,96	**1027,08**	13,51	**1027,08**	10,82
SCA8-6	**971,82**	12,00	**971,82**	2,70	**971,82**	3,51	**971,82**	1,60	**971,82**	7,76	**971,82**	10,09	**971,82**	11,40
SCA8-7	1052,17	12,50	**1051,28**	5,10	**1051,28**	3,12	**1051,28**	3,40	**1051,28**	8,14	**1051,28**	10,38	**1051,28**	9,96
SCA8-8	**1071,18**	11,00	**1071,18**	3,60	**1071,18**	2,92	**1071,18**	0,80	**1071,18**	7,06	**1071,18**	10,21	**1071,18**	7,77
SCA8-9	**1060,50**	11,50	**1060,50**	4,80	**1060,50**	2,18	**1060,50**	7,30	**1060,50**	5,29	**1060,50**	9,09	**1060,50**	9,82

***** En iyi çözümün süresidir

ACS: Gajpal ve Abad (2009)**, VLBR:** Zachariadis vd. (2010)**, PILS:** Subramanian vd. (2010)**, HPSO:** Goksal vd. (2013), **ACSEVNS:** Kalayci ve Kaya (2016), **ALNS-PR:** Hof ve Schneider (2019)

Ek 5. CON Problemleri İçin Algoritmaların En İyi Çözüm Değerleri ve Ortalama Hesaplama Süreleri

	ACS		VLBR		PILS		HPSO		ACSEVNS		ALNS-PR		ILS-RVND-TA	
Problem	Çözüm	Süre	Çözüm	Süre*	Çözüm	Süre	Çözüm	Süre	Çözüm	Süre	Çözüm	Süre	Çözüm	Süre
CON3-0	**616,52**	8,30	**616,52**	4,70	**616,52**	3,12	**616,52**	2,10	**616,52**	6,80	**616,52**	8,21	**616,52**	11,23
CON3-1	**554,47**	7,10	**554,47**	2,20	**554,47**	2,83	**554,47**	1,30	**554,47**	5,01	**554,47**	14,49	**554,47**	11,70
CON3-2	**518,00**	6,90	**518,00**	3,10	**518,00**	2,77	**518,00**	1,30	**518,00**	7,55	**518,00**	10,42	**518,00**	12,03
CON3-3	**591,19**	7,20	**591,19**	3,20	**591,19**	2,34	**591,19**	0,50	**591,19**	5,75	**591,19**	22,12	**591,19**	9,62
CON3-4	**588,79**	6,00	**588,79**	2,30	**588,79**	2,63	**588,79**	3,20	**588,79**	3,90	**588,79**	9,67	**588,79**	9,54
CON3-5	**563,70**	6,90	**563,70**	3,70	**563,70**	2,69	**563,70**	0,40	**563,70**	6,86	**563,70**	8,95	**563,70**	10,41
CON3-6	**499,05**	7,30	**499,05**	3,70	**499,05**	2,75	**499,05**	2,30	**499,05**	8,54	**499,05**	10,77	**499,05**	12,48
CON3-7	**576,48**	7,00	**576,48**	1,90	**576,48**	2,75	**576,48**	2,60	**576,48**	4,26	**576,48**	11,05	**576,48**	9,21
CON3-8	**523,05**	7,40	**523,05**	3,80	**523,05**	2,46	**523,05**	1,00	**523,05**	3,89	**523,05**	8,03	**523,05**	11,08
CON3-9	**578,25**	6,80	**578,25**	2,20	**578,25**	3,37	**578,25**	2,90	**578,25**	6,33	**578,25**	8,02	**578,25**	9,86
CON8-0	**857,17**	12,30	**857,17**	4,40	**857,17**	3,65	**857,17**	5,20	**857,17**	5,40	**857,17**	5,21	**857,17**	9,86
CON8-1	**740,85**	12,00	**740,85**	3,30	**740,85**	3,02	**740,85**	2,90	**740,85**	8,46	**740,85**	5,69	**740,85**	9,11
CON8-2	**712,89**	13,00	**712,89**	2,70	**712,89**	3,08	**712,89**	2,10	**712,89**	4,79	**712,89**	9,50	**712,89**	11,10
CON8-3	**811,07**	13,90	**811,07**	2,80	**811,07**	3,99	**811,07**	2,80	**811,07**	7,21	**811,07**	4,69	**811,07**	10,99
CON8-4	**772,25**	11,90	**772,25**	2,80	**772,25**	3,69	**772,25**	3,60	**772,25**	6,70	**772,25**	7,40	**772,25**	9,21
CON8-5	**754,88**	12,40	**754,88**	5,70	**754,88**	4,18	**754,88**	3,40	**754,88**	5,74	**754,88**	5,84	**754,88**	9,55
CON8-6	**678,92**	12,40	**678,92**	3,40	**678,92**	4,09	**678,92**	7,90	**678,92**	4,36	**678,92**	5,33	**678,92**	9,81
CON8-7	**811,96**	13,00	**811,96**	2,50	**811,96**	4,03	**811,96**	3,00	**811,96**	8,38	**811,96**	4,12	**811,96**	7,97
CON8-8	**767,53**	12,50	**767,53**	3,20	**767,53**	3,42	**767,53**	3,20	**767,53**	6,16	**767,53**	4,98	**767,53**	9,17
CON8-9	**809,00**	12,90	**809,00**	3,80	**809,00**	3,48	**809,00**	2,40	**809,00**	7,19	**809,00**	4,81	**809,00**	12,44
Ortalama**	758,64	9,29	758,54	3,27	758,54	2,92	758,54	3,06	758,54	6,13	758,54	10,56	758,54	10,06

* En iyi çözümün değeridir.

** Ortalama değerleri Dethloff (2001) test problemlerinde yer alan 40 problemin tamamı için hesaplanmıştır.

ACS: Gajpal ve Abad (2009)**, VLBR:** Zachariadis vd. (2010)**, PILS:** Subramanian vd. (2010)**, HPSO:** Goksal vd. (2013), **ACSEVNS:** Kalayci ve Kaya (2016), **ALNS-PR:** Hof ve Schneider (2019)

Ek 6. CMT Problemleri İçin Algoritmaların En İyi Çözüm Değerleri ve Ortalama Hesaplama Süreleri

	ACS		GTS		VLBR		PILS		HPSO		PVNS		ILS_ANS		ACSEVNS		ALNS-PR		ILS-RVND-TA	
Problem	**Çözüm**	**Süre**	**Çözüm**	**Süre***	**Çözüm**	**Süre***	**Çözüm**	**Süre**	**Çözüm**	**Süre**	**Çözüm**	**Süre**	**Çözüm**	**Süre**	**Çözüm**	**Süre**	**Çözüm**	**Süre**	**Çözüm**	**Süre**
CMT1X	**466,77**	5,00	469,80	2,89	469,80	2,10	**466,77**	2,28	**466,77**	1,30	**466,77**	16,52	**466,77**	23,00	**466,77**	8,50	**466,77**	13,31	**466,77**	9,03
CMT1Y	**466,77**	5,00	469,80	3,85	469,80	3,80	**466,77**	2,27	**466,77**	1,40	**466,77**	8,26	**466,77**	17,00	**466,77**	8,50	**466,77**	12,20	**466,77**	8,33
CMT2X	**684,21**	20,75	**684,21**	7,42	**684,21**	5,40	**684,21**	6,44	**684,21**	35,60	**684,21**	44,92	684,61	26,00	**684,21**	32,50	**684,21**	32,03	**684,21**	15,55
CMT2Y	684,94	22,25	**684,21**	8,02	**684,21**	6,80	**684,21**	6,41	**684,21**	36,80	**684,21**	46,73	684,75	23,00	**684,21**	36,50	**684,21**	34,90	**684,21**	14,39
CMT3X	721,40	41,25	**721,27**	11,62	**721,27**	11,90	**721,27**	12,10	**721,27**	41,70	**721,27**	52,18	721,40	71,00	**721,27**	45,20	**721,27**	72,55	**721,27**	58,56
CMT3Y	721,40	43,75	**721,27**	13,53	**721,27**	11,00	**721,27**	12,28	**721,27**	55,50	**721,27**	46,09	721,40	78,00	**721,27**	40,30	**721,27**	127,43	**721,27**	57,51
CMT4X	854,12	131,75	**852,46**	27,75	**852,46**	29,60	**852,46**	30,89	852,83	380,20	**852,46**	118,97	**852,46**	195,00	**852,46**	142,10	**852,46**	187,30	852,83	128,79
CMT4Y	855,76	140,25	**852,46**	31,20	**852,46**	27,40	**852,46**	31,61	**852,46**	414,60	**852,46**	136,37	854,38	154,00	**852,46**	136,35	**852,46**	216,23	852,83	127,78
CMT5X	1034,87	377,50	1030,55	51,67	1030,55	62,80	**1029,25**	71,50	1033,50	500,00	1030,55	554,39	1035,22	315,00	1030,55	420,15	1029,87	407,47	**1029,25**	241,95
CMT5Y	1037,34	393,50	1030,55	58,81	1030,55	47,70	**1029,25**	69,58	1036,00	500,00	1030,55	287,05	1032,89	302,00	1030,55	410,50	**1029,25**	448,52	**1029,25**	246,83
CMT11X	839,66	57,25	838,66	17,78	**833,92**	21,20	**833,92**	18,87	**833,92**	244,90	**833,92**	33,91	846,86	149,00	**833,92**	42,45	**833,92**	99,29	846,86	135,17
CMT11Y	840,19	52,75	837,08	14,26	**833,92**	14,40	**833,92**	19,03	**833,92**	368,90	**833,92**	49,64	846,57	110,00	**833,92**	40,50	**833,92**	98,83	846,86	138,58
CMT12X	663,01	36,25	**662,22**	11,80	**662,22**	9,30	**662,22**	10,29	662,94	141,40	**662,22**	33,91	663,54	98,00	**662,22**	38,25	**662,22**	39,97	**662,22**	59,83
CMT12Y	663,50	39,25	**662,22**	7,59	**662,22**	4,80	**662,22**	10,76	663,50	105,40	**662,22**	33,34	663,59	64,00	**662,22**	41,50	**662,22**	41,82	**662,22**	52,35
Ortalama	752,42	97,61	751,20	19,16	750,63	18,40	750,01	21,74	750,97	201,98	750,20	104,45	752,94	116,07	750,20	103,09	750,06	130,85	751,92	92,48

***** En iyi çözümün süresidir.

ACS: Gajpal ve Abad (2009), **GTS**: Zachariadis vd. (2009), **VLBR**: Zachariadis vd. (2010), **PILS**: Subramanian vd. (2010), **HPSO**: Goksal vd. (2013), **PVNS**: Polat vd. (2015), **ILS_ANS**: Li vd. (2015), **ACSEVNS**: Kalayci ve Kaya (2016), **ALNS-PR:** Hof ve Schneider (2019)

Ek 7. SCA Problemleri İçin Elde Edilen Rotalar (1. Kısım)

Problem	Rotalar	Problem	Rotalar
SCA3-0	[0, 35, 22, 29, 34, 10, 25, 38, 37, 32, 15, 24, 16, 44, 5, 47, 11, 50, 20, 1, 26, 0] [0, 6, 33, 41, 7, 2, 40, 30, 14, 39, 9, 4, 12, 17, 27, 8, 36, 48, 0] [0, 43, 28, 42, 49, 19, 46, 31, 23, 3, 45, 21, 18, 0] [0, 13, 0]	SCA3-5	[0, 6, 42, 37, 20, 0] [0, 10, 4, 15, 48, 43, 32, 31, 30, 11, 17, 14, 1, 28, 33, 24, 0] [0, 22, 13, 38, 9, 26, 47, 19, 23, 44, 40, 39, 8, 46, 3, 27, 36, 21, 18, 0] [0, 7, 41, 12, 34, 16, 5, 25, 45, 29, 49, 35, 50, 2, 0]
SCA3-1	[0, 12, 33, 16, 36, 42, 29, 45, 15, 46, 27, 5, 47, 20, 32, 34, 0] [0, 28, 41, 13, 26, 48, 4, 38, 35, 10, 19, 0] [0, 44, 22, 24, 17, 39, 50, 31, 0] [0, 40, 8, 23, 21, 25, 18, 1, 30, 6, 7, 43, 14, 11, 3, 9, 49, 2, 37, 0]	SCA3-6	[0, 31, 14, 36, 27, 3, 41, 29, 22, 49, 0] [0, 37, 7, 4, 16, 42, 40, 38, 25, 11, 46, 17, 0] [0, 35, 26, 39, 34, 21, 13, 20, 32, 6, 2, 23, 30, 9, 33, 43, 19, 28, 50, 0] [0, 15, 24, 1, 47, 12, 44, 48, 18, 10, 5, 45, 8, 0]
SCA3-2	[0, 35, 1, 37, 16, 41, 26, 46, 3, 17, 0] [0, 11, 42, 40, 7, 15, 49, 30, 34, 21, 5, 48, 18, 23, 0] [0, 14, 45, 8, 31, 19, 27, 32, 12, 10, 25, 9, 47, 43, 2, 0] [0, 50, 44, 39, 4, 6, 13, 20, 29, 22, 24, 28, 38, 36, 33, 0]	SCA3-7	[0, 18, 50, 23, 10, 39, 35, 21, 33, 31, 46, 1, 29, 9, 36, 6, 32, 11, 25, 0] [0, 8, 27, 22, 44, 30, 37, 42, 2, 49, 43, 17, 26, 20, 0] [0, 28, 5, 16, 40, 0] [0, 45, 24, 13, 15, 38, 3, 4, 14, 12, 48, 41, 7, 19, 34, 47, 0]
SCA3-3	[0, 16, 28, 49, 44, 13, 29, 48, 40, 38, 15, 32, 39, 3, 20, 33, 0] [0, 6, 1, 18, 25, 10, 27, 50, 23, 14, 5, 34, 2, 37, 26, 12, 41, 8, 0] [0, 19, 17, 11, 9, 42, 21, 35, 31, 7, 36, 4, 46, 0] [0, 24, 30, 45, 43, 22, 47, 0]	SCA3-8	[0, 14, 9, 33, 2, 47, 4, 37, 22, 29, 25, 11, 15, 27, 0] [0, 23, 49, 26, 39, 1, 17, 36, 38, 42, 7, 24, 28, 0] [0, 35, 18, 21, 46, 6, 31, 8, 50, 13, 45, 5, 0] [0, 12, 48, 44, 3, 43, 20, 32, 34, 30, 16, 40, 10, 19, 41, 0]
SCA3-4	[0, 25, 9, 19, 12, 35, 38, 50, 14, 3, 17, 32, 15, 34, 37, 23, 0] [0, 33, 44, 26, 1, 0] [0, 11, 24, 6, 29, 22, 36, 47, 5, 49, 43, 7, 4, 27, 2, 41, 0] [0, 10, 30, 20, 45, 8, 39, 28, 13, 31, 16, 48, 18, 40, 42, 46, 21, 0]	SCA3-9	[0, 20, 45, 35, 6, 7, 33, 0] [0, 26, 28, 43, 27, 30, 36, 19, 11, 34, 48, 8, 4, 29, 0] [0, 49, 47, 16, 3, 14, 5, 22, 23, 15, 46, 21, 24, 39, 42, 31, 0] [0, 44, 37, 38, 1, 50, 32, 12, 9, 40, 41, 13, 10, 25, 17, 2, 18, 0]

Ek 8. SCA Problemleri İçin Elde Edilen Rotalar (2. Kısım)

Problem	Rotalar	Problem	Rotalar	Problem	Rotalar
SCA8-0	[0, 43, 28, 26, 0] [0, 40, 2, 41, 3, 0] [0, 30, 14, 39, 9, 4, 12, 17, 27, 8, 0] [0, 6, 21, 33, 0] [0, 1, 50, 11, 5, 47, 42, 49, 19, 0] [0, 29, 48, 34, 13, 0] [0, 18, 46, 31, 23, 45, 7, 0] [0, 35, 22, 38, 24, 16, 44, 20, 0] [0, 36, 10, 25, 15, 32, 37, 0]	SCA8-4	[0, 17, 34, 37, 23, 0] [0, 3, 14, 50, 38, 35, 21, 12, 0] [0, 46, 42, 40, 18, 48, 16, 0] [0, 26, 1, 39, 8, 45, 20, 30, 0] [0, 15, 32, 6, 24, 11, 0] [0, 33, 19, 9, 25, 0] [0, 31, 13, 28, 44, 0] [0, 22, 29, 5, 43, 49, 47, 36, 0] [0, 7, 4, 27, 2, 41, 10, 0]	SCA8-8	[0, 13, 50, 6, 31, 8, 45, 19, 10, 41, 0] [0, 23, 48, 39, 26, 49, 0] [0, 40, 16, 30, 34, 32, 20, 0] [0, 35, 46, 21, 18, 0] [0, 1, 17, 36, 38, 42, 7, 47, 24, 28, 0] [0, 14, 11, 33, 9, 0] [0, 27, 15, 5, 0] [0, 12, 44, 3, 43, 0] [0, 25, 2, 4, 37, 29, 22, 0]
SCA8-1	[0, 25, 18, 1, 6, 7, 30, 0] [0, 31, 24, 22, 0] [0, 34, 32, 20, 44, 0] [0, 33, 36, 15, 46, 27, 5, 47, 0] [0, 37, 21, 23, 8, 10, 19, 28, 0] [0, 49, 9, 3, 11, 14, 43, 2, 0] [0, 40, 35, 38, 4, 48, 26, 13, 41, 0] [0, 17, 39, 50, 0] [0, 45, 29, 42, 16, 12, 0]	SCA8-5	[0, 20, 42, 6, 10, 0] [0, 37, 49, 29, 50, 35, 0] [0, 46, 36, 18, 21, 2, 0] [0, 44, 40, 39, 8, 3, 27, 0] [0, 12, 34, 31, 30, 11, 17, 14, 1, 0] [0, 4, 15, 33, 28, 24, 0] [0, 48, 32, 43, 7, 0] [0, 23, 19, 47, 26, 9, 38, 13, 22, 0] [0, 16, 5, 45, 25, 41, 0]	SCA8-9	[0, 33, 26, 28, 43, 0] [0, 37, 38, 32, 40, 0] [0, 27, 30, 36, 19, 11, 8, 4, 0] [0, 7, 6, 35, 45, 20, 0] [0, 49, 47, 16, 3, 42, 0] [0, 44, 29, 34, 48, 1, 50, 0] [0, 46, 15, 23, 5, 14, 22, 0] [0, 12, 9, 41, 13, 10, 25, 17, 18, 0] [0, 2, 21, 24, 39, 31, 0]
SCA8-2	[0, 17, 3, 40, 42, 11, 23, 0] [0, 2, 28, 22, 29, 44, 50, 0] [0, 9, 32, 12, 10, 25, 0] [0, 45, 31, 19, 27, 43, 0] [0, 46, 26, 41, 16, 37, 1, 35, 0] [0, 14, 8, 30, 5, 48, 0] [0, 49, 15, 7, 34, 21, 18, 0] [0, 20, 13, 6, 4, 39, 0] [0, 33, 47, 36, 38, 24, 0]	SCA8-6	[0, 49, 41, 29, 22, 31, 0] [0, 17, 46, 19, 28, 50, 0] [0, 14, 36, 27, 3, 24, 15, 0] [0, 13, 20, 32, 6, 2, 35, 0] [0, 38, 16, 42, 40, 4, 7, 0] [0, 26, 39, 21, 34, 8, 0] [0, 5, 10, 18, 48, 45, 0] [0, 37, 11, 25, 43, 33, 9, 30, 23, 0] [0, 44, 12, 47, 1, 0]		
SCA8-3	[0, 42, 31, 35, 9, 11, 17, 19, 0] [0, 33, 20, 3, 0] [0, 16, 21, 7, 36, 4, 0] [0, 28, 49, 44, 13, 29, 38, 0] [0, 6, 1, 18, 25, 10, 27, 50, 23, 14, 0] [0, 5, 34, 2, 37, 26, 12, 0] [0, 47, 22, 30, 45, 43, 0] [0, 39, 32, 15, 40, 48, 46, 0] [0, 24, 41, 8, 0]	SCA8-7	[0, 47, 38, 44, 30, 37, 22, 27, 0] [0, 40, 16, 5, 28, 0] [0, 9, 36, 6, 32, 0] [0, 18, 21, 35, 39, 10, 23, 17, 0] [0, 29, 1, 46, 31, 33, 50, 20, 0] [0, 26, 43, 49, 2, 42, 0] [0, 45, 8, 15, 13, 24, 0] [0, 48, 41, 7, 19, 34, 0] [0, 3, 4, 12, 14, 11, 25, 0]		

Ek 9. CON Problemleri İçin Elde Edilen Rotalar (1. Kısım)

Problem	Rotalar	Problem	Rotalar
CON3-0	[0, 17, 38, 34, 42, 46, 43, 7, 19, 11, 3, 27, 26, 49, 10, 13, 28, 0] [0, 12, 4, 31, 2, 8, 39, 40, 29, 6, 47, 16, 37, 0] [0, 33, 9, 22, 32, 50, 1, 21, 48, 23, 18, 15, 35, 0] [0, 41, 30, 24, 14, 45, 36, 5, 44, 20, 25, 0]	CON3-5	[0, 4, 40, 11, 29, 43, 9, 0] [0, 16, 45, 38, 6, 37, 19, 25, 26, 49, 31, 32, 10, 47, 20, 44, 18, 35, 0] [0, 46, 5, 33, 48, 39, 50, 36, 7, 21, 8, 30, 24, 14, 27, 34, 28, 0] [0, 42, 22, 3, 17, 1, 12, 13, 2, 41, 23, 15, 0]
CON3-1	[0, 2, 16, 45, 39, 19, 4, 24, 10, 25, 46, 30, 0] [0, 26, 22, 14, 0] [0, 17, 12, 47, 21, 23, 18, 28, 35, 50, 7, 49, 43, 9, 3, 20, 8, 15, 32, 42, 1, 0] [0, 33, 6, 37, 48, 40, 44, 41, 5, 11, 36, 31, 34, 38, 13, 27, 29, 0]	CON3-6	[0, 47, 34, 19, 26, 13, 10, 3, 43, 39, 48, 49, 41, 12, 4, 42, 46, 40, 16, 0] [0, 24, 5, 17, 9, 37, 50, 21, 20, 32, 22, 1, 33, 14, 15, 45, 7, 36, 44, 6, 0] [0, 23, 8, 38, 2, 30, 25, 18, 28, 35, 11, 29, 0] [0, 31, 27, 0]
CON3-2	[0, 26, 39, 45, 20, 18, 14, 44, 50, 32, 4, 16, 49, 15, 23, 0] [0, 31, 3, 43, 7, 12, 36, 2, 28, 24, 6, 37, 22, 46, 42, 8, 19, 38, 27, 25, 48, 0] [0, 13, 10, 29, 40, 5, 11, 9, 34, 47, 41, 33, 30, 35, 17, 1, 0] [0, 21, 0]	CON3-7	[0, 40, 15, 44, 10, 17, 31, 38, 18, 12, 21, 50, 25, 5, 48, 6, 28, 27, 0] [0, 30, 23, 7, 14, 9, 1, 29, 47, 8, 3, 22, 35, 11, 43, 20, 37, 36, 34, 0] [0, 26, 0] [0, 2, 41, 24, 19, 49, 32, 16, 13, 46, 39, 45, 4, 33, 42, 0]
CON3-3	[0, 4, 25, 39, 49, 17, 36, 2, 0] [0, 32, 1, 44, 28, 13, 50, 20, 19, 7, 26, 31, 10, 21, 37, 23, 48, 0] [0, 6, 38, 43, 45, 18, 33, 5, 11, 40, 41, 29, 30, 27, 0] [0, 8, 24, 22, 12, 15, 47, 42, 46, 16, 14, 3, 9, 34, 35, 0]	CON3-8	[0, 34, 38, 40, 23, 32, 44, 28, 33, 7, 11, 0] [0, 35, 45, 16, 5, 10, 30, 1, 13, 17, 19, 26, 15, 48, 49, 20, 46, 47, 8, 0] [0, 14, 41, 6, 12, 22, 18, 21, 3, 0] [0, 39, 29, 31, 50, 43, 2, 9, 4, 24, 37, 36, 42, 27, 25, 0]
CON3-4	[0, 36, 49, 45, 12, 14, 4, 38, 50, 41, 33, 24, 42, 27, 3, 1, 6, 19, 22, 0] [0, 46, 23, 29, 11, 40, 37, 13, 35, 44, 8, 20, 31, 32, 0] [0, 17, 5, 0] [0, 43, 48, 10, 2, 25, 9, 28, 15, 18, 39, 7, 16, 30, 34, 47, 21, 26, 0]	CON3-9	[0, 46, 32, 39, 0] [0, 49, 44, 19, 42, 38, 41, 15, 37, 21, 1, 14, 17, 20, 23, 22, 0] [0, 50, 33, 6, 43, 9, 40, 26, 24, 45, 47, 8, 27, 16, 34, 4, 3, 7, 0] [0, 35, 48, 11, 36, 31, 5, 28, 25, 18, 30, 12, 13, 29, 2, 10, 0]

Ek 10. CON Problemleri İçin Elde Edilen Rotalar (2. Kısım)

Problem	Rotalar	Problem	Rotalar	Problem	Rotalar
CON8-0	[0, 34, 38, 17, 12, 33, 0] [0, 28, 13, 10, 49, 26, 46, 42, 0] [0, 9, 32, 15, 35, 0] [0, 45, 14, 30, 24, 0] [0, 4, 7, 31, 50, 0] [0, 25, 20, 44, 5, 36, 41, 0] [0, 18, 23, 39, 8, 21, 48, 0] [0, 22, 1, 2, 19, 11, 3, 27, 43, 0] [0, 40, 29, 6, 47, 16, 37, 0]	CON8-4	[0, 23, 29, 37, 40, 11, 7, 16, 0] [0, 19, 1, 6, 36, 0] [0, 43, 10, 2, 45, 48, 0] [0, 9, 25, 39, 18, 15, 28, 12, 49, 0] [0, 21, 34, 30, 47, 0] [0, 5, 46, 26, 17, 0] [0, 24, 33, 41, 50, 38, 4, 14, 0] [0, 22, 3, 42, 27, 32, 31, 20, 0] [0, 8, 13, 35, 44, 0]	CON8-8	[0, 50, 43, 2, 9, 41, 0] [0, 49, 48, 19, 26, 15, 18, 0] [0, 14, 21, 3, 0] [0, 42, 36, 37, 24, 4, 0] [0, 28, 44, 32, 17, 13, 1, 30, 10, 5, 16, 0] [0, 11, 34, 38, 23, 40, 0] [0, 7, 33, 46, 20, 47, 8, 0] [0, 6, 12, 27, 25, 22, 0] [0, 35, 45, 31, 29, 39, 0]
CON8-1	[0, 1, 32, 15, 42, 14, 0] [0, 2, 16, 3, 8, 20, 0] [0, 23, 18, 28, 35, 50, 7, 49, 43, 9, 0] [0, 30, 46, 4, 19, 0] [0, 48, 36, 11, 5, 41, 22, 0] [0, 24, 13, 38, 34, 31, 27, 29, 0] [0, 39, 21, 47, 12, 45, 17, 0] [0, 33, 6, 37, 44, 40, 0] [0, 26, 10, 25, 0]	CON8-5	[0, 40, 11, 29, 43, 9, 0] [0, 28, 27, 34, 4, 0] [0, 22, 17, 3, 45, 38, 0] [0, 6, 37, 19, 25, 26, 1, 13, 0] [0, 36, 50, 39, 48, 33, 5, 46, 0] [0, 14, 30, 8, 21, 7, 24, 0] [0, 41, 23, 44, 18, 35, 0] [0, 15, 42, 16, 0] [0, 20, 47, 10, 32, 31, 49, 12, 2, 0]	CON8-9	[0, 1, 14, 17, 20, 18, 0] [0, 16, 27, 45, 47, 8, 4, 34, 0] [0, 49, 15, 21, 37, 23, 0] [0, 36, 31, 5, 28, 25, 0] [0, 19, 42, 38, 41, 44, 0] [0, 43, 9, 40, 26, 24, 6, 0] [0, 7, 3, 35, 29, 13, 22, 0] [0, 46, 50, 33, 32, 39, 0] [0, 10, 12, 30, 48, 11, 2, 0]
CON8-2	[0, 49, 7, 12, 36, 2, 28, 24, 0] [0, 25, 27, 38, 19, 17, 1, 0] [0, 50, 32, 4, 16, 13, 0] [0, 48, 8, 6, 37, 22, 46, 42, 35, 0] [0, 15, 3, 43, 31, 23, 0] [0, 34, 47, 41, 33, 30, 0] [0, 39, 10, 45, 20, 26, 0] [0, 18, 14, 44, 29, 0] [0, 21, 40, 5, 11, 9, 0]	CON8-6	[0, 47, 19, 34, 9, 5, 24, 0] [0, 28, 35, 46, 40, 15, 16, 0] [0, 17, 37, 50, 21, 20, 7, 36, 44, 0] [0, 6, 45, 32, 22, 1, 33, 14, 0] [0, 48, 49, 41, 12, 4, 42, 0] [0, 29, 18, 11, 0] [0, 26, 13, 10, 3, 43, 39, 38, 8, 0] [0, 25, 30, 2, 23, 0] [0, 27, 31, 0]		
CON8-3	[0, 27, 2, 0] [0, 13, 50, 20, 19, 7, 26, 31, 10, 37, 48, 0] [0, 36, 8, 6, 0] [0, 25, 42, 46, 16, 14, 3, 9, 34, 0] [0, 24, 29, 41, 40, 11, 0] [0, 4, 39, 49, 17, 0] [0, 30, 22, 12, 47, 15, 0] [0, 38, 33, 5, 18, 45, 43, 0] [0, 44, 28, 21, 23, 0] [0, 32, 1, 35, 0]	CON8-7	[0, 7, 1, 29, 47, 3, 8, 20, 37, 0] [0, 26, 0] [0, 40, 16, 13, 33, 42, 0] [0, 30, 23, 28, 27, 41, 0] [0, 22, 35, 11, 43, 39, 0] [0, 2, 21, 12, 18, 38, 19, 0] [0, 46, 15, 17, 31, 10, 44, 32, 49, 0] [0, 6, 48, 5, 25, 50, 24, 0] [0, 4, 45, 36, 9, 14, 34, 0]		

Ek 11. CMT Problemleri İçin Elde Edilen Rotalar (1. Kısım)

Problem	Rotalar
CMT1X	[0, 46, 11, 38, 5, 49, 9, 50, 16, 29, 21, 34, 30, 10, 39, 33, 45, 15, 44, 37, 17, 0] [0, 12, 47, 18, 4, 42, 19, 40, 41, 13, 25, 14, 24, 43, 7, 23, 6, 0] [0, 32, 1, 22, 2, 20, 35, 36, 3, 28, 31, 26, 8, 48, 27, 0]
CMT1Y	[0, 6, 23, 7, 43, 24, 14, 25, 13, 41, 40, 19, 42, 4, 18, 47, 12, 0] [0, 17, 37, 44, 15, 45, 33, 39, 10, 30, 34, 21, 29, 16, 50, 9, 49, 5, 38, 11, 46, 0] [0, 27, 48, 8, 26, 31, 28, 3, 36, 35, 20, 2, 22, 1, 32, 0]
CMT2X	[0, 51, 33, 1, 73, 62, 28, 74, 2, 68, 75, 0] [0, 53, 11, 66, 65, 38, 31, 55, 25, 50, 18, 24, 49, 3, 0] [0, 6, 16, 63, 23, 56, 41, 43, 42, 64, 22, 61, 21, 47, 48, 30, 0] [0, 26, 12, 58, 10, 72, 39, 9, 32, 44, 40, 17, 0] [0, 67, 34, 46, 8, 52, 27, 45, 4, 0] [0, 7, 35, 14, 59, 19, 54, 13, 57, 15, 37, 20, 70, 60, 71, 69, 36, 5, 29, 0]
CMT2Y	[0, 3, 49, 24, 18, 50, 25, 55, 31, 38, 65, 66, 11, 53, 0] [0, 29, 5, 36, 69, 71, 60, 70, 20, 37, 15, 57, 13, 54, 19, 59, 14, 35, 7, 0] [0, 30, 48, 47, 21, 61, 22, 64, 42, 43, 41, 56, 23, 63, 16, 6, 0] [0, 17, 40, 44, 32, 9, 39, 72, 10, 58, 12, 26, 0] [0, 4, 45, 27, 52, 8, 46, 34, 67, 0] [0, 51, 33, 1, 73, 62, 28, 74, 2, 68, 75, 0]
CMT3X	[0, 28, 76, 68, 80, 29, 24, 54, 4, 55, 25, 39, 67, 23, 56, 75, 41, 22, 74, 72, 73, 21, 40, 0] [0, 58, 2, 57, 15, 43, 42, 14, 44, 38, 86, 16, 61, 85, 91, 100, 37, 98, 93, 99, 96, 6, 0] [0, 26, 12, 77, 3, 79, 78, 34, 35, 71, 65, 66, 32, 90, 63, 64, 49, 36, 46, 8, 45, 17, 84, 5, 60, 83, 18, 89, 0] [0, 53, 13, 87, 97, 92, 59, 95, 94, 0] [0, 27, 69, 1, 50, 33, 81, 9, 51, 20, 30, 70, 10, 62, 11, 19, 47, 48, 82, 7, 88, 31, 52, 0]
CMT3Y	[0, 52, 31, 88, 7, 82, 48, 47, 19, 11, 62, 10, 70, 30, 20, 51, 9, 81, 33, 50, 1, 69, 27, 0] [0, 89, 18, 83, 60, 5, 84, 17, 45, 8, 46, 36, 49, 64, 63, 90, 32, 66, 65, 71, 35, 34, 78, 79, 3, 77, 12, 26, 0] [0, 58, 2, 57, 15, 43, 42, 14, 44, 38, 86, 16, 61, 85, 91, 100, 37, 98, 93, 99, 96, 6, 0] [0, 53, 13, 87, 97, 92, 59, 95, 94, 0] [0, 40, 21, 73, 72, 74, 22, 41, 75, 56, 23, 67, 39, 25, 55, 4, 54, 24, 29, 80, 68, 76, 28, 0]
CMT4X	[0, 13, 117, 97, 92, 37, 98, 100, 91, 85, 61, 16, 141, 86, 113, 17, 45, 125, 84, 5, 118, 60, 89, 0] [0, 1, 51, 9, 103, 71, 135, 35, 136, 65, 66, 20, 128, 131, 32, 90, 126, 63, 64, 49, 143, 36, 47, 124, 46, 8, 114, 83, 18, 52, 146, 0] [0, 138, 12, 109, 80, 150, 68, 121, 29, 24, 134, 54, 130, 55, 25, 139, 39, 67, 23, 56, 4, 110, 149, 26, 0] [0, 28, 111, 132, 69, 101, 70, 122, 30, 108, 10, 62, 11, 107, 19, 123, 48, 82, 106, 7, 148, 88, 31, 127, 27, 0] [0, 112, 94, 95, 59, 93, 99, 104, 96, 6, 147, 0] [0, 53, 58, 137, 87, 144, 42, 142, 14, 119, 44, 140, 38, 43, 15, 57, 2, 115, 145, 41, 22, 133, 75, 74, 72, 73, 21, 40, 105, 0] [0, 50, 102, 33, 81, 120, 34, 78, 129, 79, 3, 77, 116, 76, 0]
CMT4Y	[0, 27, 127, 31, 88, 148, 7, 106, 82, 48, 123, 19, 107, 11, 62, 10, 108, 30, 122, 70, 101, 69, 132, 111, 28, 0] [0, 146, 52, 18, 83, 114, 8, 46, 124, 47, 36, 143, 49, 64, 63, 126, 90, 32, 131, 128, 20, 66, 65, 136, 35, 135, 71, 103, 9, 51, 1, 0] [0, 89, 60, 118, 5, 84, 125, 45, 17, 113, 86, 141, 16, 61, 85, 91, 100, 98, 37, 92, 97, 117, 13, 0] [0, 105, 40, 21, 73, 72, 74, 75, 133, 22, 41, 145, 115, 2, 57, 15, 43, 38, 140, 44, 119, 14, 142, 42, 144, 87, 137, 58, 53, 0] [0, 147, 6, 96, 99, 104, 93, 59, 95, 94, 112, 0] [0, 26, 149, 110, 4, 56, 23, 67, 39, 139, 25, 55, 130, 54, 134, 24, 29, 121, 68, 80, 150, 109, 12, 138, 0] [0, 76, 116, 77, 3, 79, 129, 78, 34, 120, 81, 33, 102, 50, 0]

Ek 12. CMT Problemleri İçin Elde Edilen Rotalar (2. Kısım)

Problem	Rotalar
CMT5X	[0, 50, 51, 9, 103, 161, 71, 135, 35, 136, 65, 66, 181, 64, 49, 143, 36, 47, 168, 124, 46, 174, 8, 114, 18, 0] [0, 26, 149, 195, 179, 130, 54, 134, 163, 68, 150, 80, 177, 109, 12, 138, 154, 28, 0] [0, 176, 1, 122, 30, 20, 188, 128, 160, 131, 32, 90, 63, 126, 11, 175, 107, 19, 123, 48, 82, 7, 194, 106, 153, 52, 146, 0] [0, 89, 60, 118, 5, 84, 173, 61, 16, 141, 191, 44, 119, 192, 14, 38, 140, 86, 113, 17, 45, 125, 199, 83, 166, 0] [0, 184, 76, 196, 116, 77, 158, 3, 129, 79, 185, 157, 102, 111, 0] [0, 156, 112, 183, 94, 95, 59, 151, 92, 97, 117, 13, 0] [0, 132, 69, 162, 101, 70, 108, 10, 189, 159, 62, 148, 182, 88, 31, 190, 127, 167, 27, 0] [0, 33, 81, 120, 164, 34, 78, 169, 121, 29, 24, 165, 55, 25, 170, 67, 23, 186, 56, 39, 187, 139, 155, 4, 110, 198, 40, 0] [0, 53, 152, 58, 2, 178, 115, 145, 41, 22, 133, 75, 197, 72, 74, 171, 73, 21, 180, 105, 0] [0, 147, 6, 96, 99, 104, 93, 85, 91, 193, 100, 98, 37, 172, 42, 142, 43, 15, 57, 144, 87, 137, 0]
CMT5Y	[0, 40, 198, 110, 155, 4, 139, 187, 39, 56, 186, 23, 67, 170, 25, 55, 165, 24, 29, 121, 169, 78, 34, 164, 120, 81, 33, 0] [0, 146, 52, 153, 106, 194, 7, 82, 48, 123, 19, 107, 175, 11, 126, 63, 90, 32, 131, 160, 128, 188,20, 30, 122, 1, 176, 0] [0, 111, 102, 157, 185, 79, 129, 158, 3, 77, 116, 196, 76, 184, 0] [0, 18, 114, 8, 174, 46, 124, 168, 47, 36, 143, 49, 64, 181, 66, 65, 136, 35, 135, 71, 161, 103, 9, 51, 50, 0] [0, 166, 83, 199, 125, 45, 17, 113, 86, 140, 38, 14, 192, 119, 44, 191, 141, 16, 61, 173, 84, 5, 118, 60, 89, 0] [0, 28, 138, 154, 12, 109, 177, 150, 80, 68, 163, 134, 54, 130, 179, 195, 149, 26, 0] [0, 13, 117, 97, 92, 151, 59, 95, 94, 183, 112, 156, 0] [0, 137, 87, 144, 57, 15, 43, 142, 42, 172, 37, 98, 100, 193, 91, 85, 93, 99, 104, 96, 6, 147, 0] [0, 105, 180, 21, 73, 171, 74, 72, 197, 75, 133, 22, 41, 145, 115, 178, 2, 152, 58, 53, 0] [0, 132, 69, 162, 101, 70, 108, 189, 10, 159, 62, 148, 182, 88, 31, 190, 127, 167, 27, 0]
CMT11X	[0, 95, 96, 93, 94, 97, 115, 110, 40, 43, 45, 48, 51, 50, 49, 46, 44, 41, 42, 39, 38, 37, 109, 114, 90, 91, 92, 89, 85, 86, 111, 82, 119, 0] [0, 52, 57, 54, 53, 55, 58, 56, 60, 63, 66, 64, 62, 61, 65, 59, 47, 29, 36, 34, 35, 32, 28, 31, 30, 33, 27, 24, 22, 25, 19, 16, 17, 20, 23, 26, 21, 81, 0] [0, 18, 118, 108, 8, 12, 13, 14, 15, 11, 10, 9, 7, 6, 5, 4, 3, 1, 2, 83, 113, 117, 84, 112, 88, 0] [0, 87, 102, 101, 99, 100, 116, 98, 68, 73, 76, 77, 79,80, 78, 75, 72, 74, 71, 70, 69, 67, 103, 104, 107, 106, 105, 120, 0]
CMT11Y	[0, 88, 112, 84, 117, 113, 83, 2, 1, 3, 4, 5, 6, 7, 9, 10, 11, 15, 14, 13, 12, 8, 108, 118, 18, 0] [0, 120, 105, 106, 107, 104, 103, 67, 69, 70, 71, 74, 72, 75, 78, 80, 79, 77, 76, 73, 68, 98, 116, 100, 99, 101, 102, 87, 0] [0, 119, 82, 111, 86, 85, 89, 92, 91, 90, 114, 109, 37, 38, 39, 42, 41, 44, 46, 49, 50, 51, 48, 45, 43, 40, 110, 115, 97, 94, 93, 96, 95, 0] [0, 81, 21, 26, 23, 20, 17, 16, 19, 25, 22, 24, 27, 33, 30, 31, 28, 32, 35, 34, 36, 29, 47, 59, 65, 61, 62, 64, 66, 63, 60, 56, 58, 55, 53, 54, 57, 52, 0]
CMT12X	[0, 91, 98, 96, 95, 94, 92, 93, 97, 100, 99, 12, 14, 16, 15, 19, 18, 17, 13, 0] [0, 69, 66, 68, 40, 41, 42, 43, 47, 49, 46, 44, 45, 48, 51, 50, 52, 31, 35, 37, 38, 39, 36, 34, 33, 32, 0] [0, 75, 1, 2, 4, 3, 5, 7, 6, 8, 9, 11, 10, 23, 26, 28, 30, 29, 27, 25, 24, 22, 21, 20, 0] [0, 90, 87, 86, 89, 88, 85, 84, 83, 82, 81, 78, 76, 71, 70, 73, 77, 79, 80, 72, 0] [0, 67, 65, 63, 62, 74, 61, 64, 55, 54, 53, 56, 58, 60, 59, 57, 0]
CMT12Y	[0, 13, 17, 18, 19, 15, 16, 14, 12, 99, 100, 97, 93, 92, 94, 95, 96, 98, 91, 0] [0, 20, 21, 22, 24, 25, 27, 29, 30, 28, 26, 23, 10, 11, 9, 8, 6, 7, 5, 3, 4, 2, 1, 75, 0] [0, 57, 59, 60, 58, 56, 53, 54, 55, 64, 61, 74, 62, 63, 65, 67, 0] [0, 72, 80, 79, 77, 73, 70, 71, 76, 78, 81, 82, 83, 84, 85, 88, 89, 86, 87, 90, 0] [0, 32, 33, 34, 36, 39, 38, 37, 35, 31, 52, 50, 51, 48, 45, 44, 46, 49, 47, 43, 42, 41, 40, 68, 66, 69, 0]