KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK BİR MODEL ÖNERİSİ

Zafer BOZYER

KOCAELİ 2013

KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK BİR MODEL ÖNERİSİ

Zafer BOZYER

Prof. Dr. Alpaslan FIĞLALI
Danışman, Kocaeli Üniv. .....

Yrd. Doç. Dr. Alparslan S. DEMİR
Jüri Üyesi, Sakarya Üniv. .....

Yrd. Doç. Dr. Celal ÖZKALE
Jüri Üyesi, Kocaeli Üniv. .....

Tezin Savunulduğu Tarih: 24.06.2013

ÖNSÖZ ve TEŞEKKÜR

Günümüz dünyasında lojistik faaliyetler hayatın vazgeçilmez bir parçası haline gelmiştir. Ürünler, hizmetler ve insanlar sürekli bir noktadan başka bir noktaya hareket etmekte dolayısıyla ciddi miktarda taşıma ve dağıtım maliyetleri ortaya çıkmaktadır. Bu maliyetleri en küçükleyebilen işletmeler pazarda söz sahibi olmaktadırlar. Bu çalışmada, dağıtım faaliyetlerinden dolayı ortaya çıkan maliyetlerin azaltılabilmesi amacıyla araç rotalama problemleri üzerinde çalışılmış ve problemlerin çözümü için yeni bir model önerilmiştir.

Çalışmalarım boyunca, değerli görüş ve katkılarıyla bana her konuda yardımcı olan danışman hocam Prof. Dr. Alpaslan FIĞLALI'ya teşekkürü bir borç bilirim.

Ayrıca bu zorlu süreçte her zaman yanımda olan, varlıklarından güç aldığım sevgili anneme, babama ve kardeşime, verdikleri manevi destek için değerli arkadaşlarım ve meslektaşlarım Atakan ALKAN, Sertan ALKAN ve Serkan PELDEK'e sonsuz teşekkürlerimi sunarım.

Haziran - 2013 Zafer BOZYER

İÇİNDEKİLER

ÖNSÖZ ve TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iv
TABLOLAR DİZİNİ v
SİMGELER DİZİNİ VE KISALTMALAR vi
ÖZET viii
ABSTRACT ix
GİRİŞ 1
1. LOJİSTİK KAVRAMI 3
1.1. Lojistiğin Tanımı 3
1.2. Lojistikte Karar Seviyeleri 4
1.2.1. Stratejik seviye 4
1.2.2. Taktiksel seviye 4
1.2.3. Operasyonel seviye 5
1.3. Temel Lojistik Faaliyetler ve Dağıtım Lojistiği 5
2. ROTALAMA PROBLEMLERİ 7
2.1. Gezgin Satıcı Problemi 7
2.1.1. Gezgin satıcı problemlerinin çözümünde kullanılan yöntemler 11
2.2. Araç Rotalama Problemi 12
2.3. Araç Rotalama Problem Türleri 15
2.3.1. Çevrenin durumuna göre araç rotalama problemleri 16
2.3.1.1. Statik araç rotalama problemi 16
2.3.1.2. Dinamik araç rotalama problemi 16
2.3.2. Rota durumuna göre araç rotalama problemleri 20
2.3.2.1. Açık uçlu araç rotalama problemi 20
2.3.2.2. Kapalı uçlu araç rotalama problemi 21
2.3.3. Kısıt ve amaç fonksiyonuna göre araç rotalama problemleri 21
2.3.3.1. Kapasite kısıtlı araç rotalama problemi 22
2.3.3.2. Zaman pencereli araç rotalama problemi 25
2.3.3.3. Topla-dağıt araç rotalama problemi 28
2.3.3.4. Çok depolu araç rotalama problemi 32
2.3.3.5. Parçalı dağıtım araç rotalama problemi 33
2.3.3.6. Periyodik araç rotalama problemi 35
3. ROTALAMA PROBLEMLERİNDE KULLANILAN ALGORİTMALAR 37
3.1. Kesin Çözüm Yöntemleri 37
3.1.1. Dal sınır yöntemi 37
3.1.2. Dal kesme yöntemi 38
3.2. Klasik Sezgisel Algoritmalar 39
3.2.1. Tur kurucu sezgiseller 40
3.2.1.1. Tasarruf algoritması 40
3.2.1.2. Geliştirilmiş tasarruf algoritması 42
3.2.1.3. Sıralı ekleme algoritması 43
3.2.2. İki aşamalı sezgiseller 45

3.2.2.1. Süpürme algoritması 45
3.2.2.2. Fisher ve Jaikumar algoritması 46
3.2.2.3. Bramel ve Simch-Levi algoritması 48
3.2.2.4. Petal algoritması 49
3.2.3. Tur geliştirici sezgiseller 51
3.2.3.1. Tek rota geliştirme algoritmaları 52
3.2.3.2. Çoklu rota geliştirme algoritmaları 53
3.3. Metasezgiseller 54
3.3.1. Tavlama Benzetimi 56
3.3.2. Genetik Algoritma 58
3.3.3. Karınca Kolonisi 61
3.3.4. Tabu Arama 63
4. ARAÇ ROTALAMA PROBLEMİ İÇİN YENİ BİR MODEL ÖNERİSİ 67
4.1. Önerilen Yöntemin Yapısı 67
4.1.1. Bulanık C-Ortalama Algoritması 70
4.2. Önerilen Yöntemin Algoritması 71
4.2.1. Gruplama algoritması 72
4.2.2. Rotalama ve iyileştirme algoritması 73
4.3. Önerilen Yöntemin Örnek Bir Problem Üzerinde Uygulanması 74
4.3.1. Talep noktalarının kümelere ayrılması 74
4.3.2. Rotaların oluşturulması ve iyileştirilmesi 76
4.4. Yöntemin Uygulanması ve Bulgular 83
5. SONUÇLAR ve ÖNERİLER 92
KAYNAKLAR 94
ÖZGEÇMİŞ 117

ŞEKİLLER DİZİNİ

Şekil 1.1. Lojistik karar seviyelerinin kapsamı ve aralarındaki ilişki 5
Şekil 1.2. Temel lojistik faaliyetleri ve aralarındaki ilişkiler 6
Şekil 2.1. Gezgin satıcı probleminin temsili örneği 7
Şekil 2.2. Alt turlar bulunduran beş noktalı GSP örneği 8
Şekil 2.3. Çözümü Hamilton Turu olmayan GSP örneği 9
Şekil 2.4. Araç rotalama probleminin temsili bir örneği 13
Şekil 2.5. Dinamik araç rotalama örneği 17
Şekil 2.6. Örnek DARP'nde olayların zaman çizelgesi üzerinde gösterimi 17
Şekil 2.7. DARP'de noktaların tamamının rotalanması 18
Şekil 2.8. DARP için geliştirilen stratejiler 19
Şekil 2.9. Açık ve kapalı uçlu araç rotalama problemi 21
Şekil 2.10. ZPARP'nde kullanılan zaman değişkenleri 26
Şekil 2.11. TDARP türleri 29
Şekil 2.12. TDARP türleri arasındaki ilişki 31
Şekil 2.13. ÇDARP'nde noktaların gruplandırılması ve oluşan çözümler 32
Şekil 2.14. PDARP'nde daha iyi sonuçların elde edilmesi 33
Şekil 2.15. PARP örneği (talep periyotları ve rotalar) 36
Şekil 3.1. Tasarruf algoritması örneği 40
Şekil 3.2. Geliştirilmiş tasarruf algoritmasında λ'nın etkisi 42
Şekil 3.3. Sıralı ekleme algoritması örneği 44
Şekil 3.4. Süpürme algoritması örneği 46
Şekil 3.5. 1-petal ve 2-petal sezgiselinde oluşan rotalar 49
Şekil 3.6. Petal sezgiselinde gezgin satıcı turunun oluşturulması 50
Şekil 3.7. 1-petal sezgiseli ile elde edilen rotalar 51
Şekil 3.8. 2-opt ve 3-opt uygulama örneği 52
Şekil 3.9. Isıl işlem sonrası atomların dizilimi 56
Şekil 3.10. Tabu listesinin güncellenmesi 66
Şekil 4.1. Bulanık aitlik değerleri ile komşu çözümlerin elde edilmesi 69
Şekil 4.2. Bulanık c-ortalama algoritması akış şeması 71
Şekil 4.3. Küme 1 'de tasarruf algoritması ve oluşturulan rota 78
Şekil 4.4. Küme 1'de λ-opt algoritması ve geliştirilen rota 80
Şekil 4.5. Komşu çözümün aranması ve rotalar arası nokta takası 82
Şekil 4.6. Küme 1'de rotanın güncellenmesi 82
Şekil 4.7. E-n22-k4 test probleminin çözümü 85
Şekil 4.8. E-n23-k3 test probleminin çözümü 86
Şekil 4.9. E-n30-k4 test probleminin çözümü 87
Şekil 4.10. E-n33-k4 test probleminin çözümü 89
Şekil 4.11. Önerilen yöntemin zayıf noktası 90

TABLOLAR DİZİNİ

Tablo 2.1. Yıllara göre çözülmüş en büyük gezgin satıcı problemleri 11
Tablo 2.2. Araç rotalama problem türleri 15
Tablo 3.1. Klasik sezgisel türleri ve yöntemler 39
Tablo 3.2. Tasarruf algoritmasının adımları 41
Tablo 3.3. Süpürme algoritmasının adımları 45
Tablo 3.4. Fisher ve Jaikumar algoritmasının adımları 47
Tablo 3.5. Araç kapasitesini aşmadan oluşturulan rotalar 50
Tablo 3.6. Rulet tekerleği seçimine bir örnek 59
Tablo 4.1. BCO işleminden sonra küme merkezleri 74
Tablo 4.2. BCO işleminden sonra talep noktalarının aitlik dereceleri 75
Tablo 4.3. Kümelerin talepleri ve talep noktaları 75
Tablo 4.4. Takas sonrası kümelerin talepleri ve talep noktaları 76
Tablo 4.5. Küme 1 için noktaların tasarruf değerleri ve rotalanması 77
Tablo 4.6. Rotalar ve tur uzunlukları 78
Tablo 4.7. λ-opt ile iyileştirilmiş rotalar ve tur uzunlukları 79
Tablo 4.8. Aitlik değerleri $f \leq 0{,}4$ olan talep noktaları 80
Tablo 4.9. Kümelerden çıkan ve dâhil olan noktalar 81
Tablo 4.10. Komşu çözüme ait rotalar ve tur uzunlukları 83
Tablo 4.11. Test problemlerinden elde edilen sonuçlar 83
Tablo 4.12. E-n22-k4'da BCO sonrası kümeler 84
Tablo 4.13. E-n22-k4'da küme taleplerinin dengelenmesi 84
Tablo 4.14. E-n23-k3'da BCO sonrası kümeler 84
Tablo 4.15. E-n23-k3'da küme taleplerinin dengelenmesi 85
Tablo 4.16. E-n30-k4'de BCO sonrası kümeler 86
Tablo 4.17. E-n33-k4'de BCO sonrası kümeler 87
Tablo 4.18. E-n33-k4'de küme taleplerinin dengelenmesi 88

SİMGELER DİZİNİ ve KISALTMALAR

a_i : Hizmetin en erken başlayabileceği zaman
b_i : Hizmetin en geç başlayabileceği zaman
c : BCO'da küme sayısı (cluster)
c_i : i çantasınnı kapasitesi
c_{ij} : (i,j) nokta çifti arasındaki mesafe
D : Depolar kümesi
d_i : i müşterisinin talep miktarı (dağıtım)
d_{ijk} : deponun k grubunun merkezine olan uzaklığı
d_{ik} : BCO'da k noktasının i küme merkezine olan uzaklığı
d_{oi} : deponun i noktasına olan uzaklığı
E_r : Tavlama Benzetimi'nde toplam rota uzunluğu
fi : i noktasına yapılan servisin süresi
L : BCO ile elde edilen gruplar kümesi
m : Araç miktarı / satıcı miktarı
n : Talep noktası sayısı
N : Talep noktaları kümesi
P_i : i. talep noktasının rotada bulunma ihtimali
p_i : i müşterisinin toplama talebi
p_{ij} : i çantasına j paketinin yerleştirilmesi halinde kazanç miktarı
p_{ik} : i müşterisinin k grubuna ait olması halinde oluşan maliyet
Q_i : i deposunun kapasitesi
q_k : k aracının kapasitesi
q_{kl} : l noktasına hizmet veren k aracının kapasitesi
Q(x) : İkinci aşama fonksiyon (recourse function)
r_{ij} : i çantasına yerleştirilen j paketinin ağırlığı veya boyutu
r_k : k aracının rotada kalabileceği maksimum süre
S : N talep noktaları kümesinin bir alt kümesi
S_{ij} : (i,j) nokta çifti arasındaki tasarruf değeri
T : Sıcaklık değeri
t_i : i noktasına ulaşma zamanı
t_{ij} : (i,j) düğümleri arasında yolculuk süresi
u_{ik} : BCO'da k noktasının i kümesine olan aitlik değeri
V : Araçlar kümesi
V_i : i deposunda yer alan araçlar kümesi
v_i : i kümesinin merkez noktası
w_i : i noktasında zaman penceresi açılmadan önce bekleme süresi
w_{ij} : i deposuna j noktasının atanıp atanmadığını belirten ikili değişken

w_{ik} : i noktasına ait talebin k aracı ile karşılanma miktarı
x_{ic} : BCO'da c kümesine aktarılan i noktası
x_{ij} : (i,j) nokta çifti arasında bağlantı olup olmadığını belirten ikili değişken
x_{ijk} : (i,j) yolunun k aracı ile kat edildiğini gösteren ikili değişken
x_{ijkt} : (i,j) yolunun k aracı ile t periyotta kat edildiğini gösteren ikili değişken
x_{ik} : i noktasının talebinin k aracıyla karşılandığını gösterek ikili değişken
y_{ij} : i noktasından j noktasına ulaşana kadar toplanan yük miktarı
z_{ij} : i noktasından j noktasına ulaşana kadar yapılan dağıtım miktarı
ε : Durma ölçütü
λ_j : j noktasına ait talebin karşılanma maliyeti
τ_{ij}^{k} : (i,j) noktaları arasına k adet karınca ile bırakılan feremon miktarı

Kısaltmalar

AGSP : Asimetrik Gezgin Satıcı Problemi
ARP : Araç Rotalama Problemi
AUARP : Açık Uçlu Araç Rotalama Problemi
BCO : Bulanık C-Ortalama
ÇDARP : Çok Depolu Araç Rotalama Problemi
DARP : Dinamik Araç Rotalama Problemi
GA : Genetik Algoritma
GSP : Gezgin Satıcı Problemi
KKARP : Kapasite Kısıtlı Araç Rotalama Problemi
KKO : Karınca Kolonisi Optimizasyonu
KUARP : Kapalı Uçlu Araç Rotalama Problemi
m-GSP : m Araçlı Gezgin Satıcı Problemi
PARP : Periyodik Araç Rotalama Problemi
PDARP : Parçalı Dağıtım Araç Rotalama Problemi
SARP : Statik Araç Rotalama Problemi
SGSP : Simetrik Gezgin Satıcı Problemi
TA : Tabu Arama
TB : Tavlama Benzetimi
TDARP : Topla Dağıt Araç Rotalama Problemi
ZPARP : Zaman Pencereli Araç Rotalama Problemi

ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK BİR MODEL ÖNERİSİ

ÖZET

Bilgi çağının getirdiği yenilikler ile teknolojinin akıl almaz bir hızla gelişmesi, yerel pazarların yerini küresel pazarların almasının ve işletmeler arasında yaşanan rekabetin daha da artmasının nedenlerinden birisidir. Günümüzde piyasada söz sahibi olabilmek için kaynakların en verimli şekilde kullanılması, müşterilere istedikleri ürünün en kısa zamanda en az maliyetle ulaştırılması bir zorunluluk haline gelmiştir. Müşteriye özel ürünler geliştirilmesi ile artan ürün çeşitliliği ve taleplerin hızlı karşılanması yönünde oluşan baskı, dikkatlerin lojistik faaliyetler üzerinde toplanmasına ve lojistik faaliyetlerinin içerisinde yer alan araç rotalama problemlerinin öneminin artmasına neden olmuştur. Son yarım asır içerisinde araç rotalama probleminin çözümüne yönelik birçok çalışma yapılmış olsa bile tam olarak çözülebildiği söylenemez. Hesaplama zorluğu nedeniyle büyük boyutlu araç rotalama problemleri kesin çözüm yöntemleri ile çözülememekte ve çalışmalarda kabul edilebilir sonuçlar veren sezgisel algoritmalar üzerinde durulmaktadır. Bu çalışmada da araç rotalama probleminin çözümü için, talep noktalarının önce gruplandırılması sonra araçlara atanarak rotalanması mantığına dayalı bir sezgisel algoritma önerilmiştir. Kümeleme işlemi altında yatan amaç, problemi çözümü daha kolay olan gezgin satıcı problemine dönüştürerek daha iyi çözümler elde edebilmektir. Kümeleme işleminden sonra talep noktaları rotalanmış ve elde edilen rotalar bir iyileştirme algoritması ile geliştirilmiştir. Bazı örnek problemler üzerinde algoritma test edilerek sonuçlar ve bulgular tartışılmıştır.

Anahtar Kelimeler: Araç Rotalama Problemi, Bulanık C-Ortalama, Önce Kümele Sonra Rotala, Tabu Arama.

A MODEL PROPOSAL FOR THE SOLUTION OF VEHICLE ROUTING PROBLEM

ABSTRACT

The incredible speed of development of technology brought by the information age is one of the consequences of increased competition between enterprises as well as the alteration from local markets to global markets. Nowadays, in order to have a say in the market, utilization of sources in most efficient way, dispatching products to the customers at the shortest time at the lowest cost have become a necessity. The increasing product diversity, resulting from the development of products tailored according to customers and the pressure aimed at meeting demands rapidly gathered attention on logistic activities and the increased importance of vehicle routing problems which is part of the logistic activities. Although many studies have been carried out in the past half-century to find a solution to the vehicle routing problem, it cannot be said that it has been completely solved. Heuristic algorithms producing acceptable results are being emphasized in the studies, due to calculation difficulty large sized vehicle problems cannot be solved by definite solution methods and in the studies. In this study, a heuristic algorithm, which is based on initially clustering the customers and then routing by assigning them to vehicles, has been proposed. The aim behind clustering process is to transform the vehicle routing problem into travelling salesman problem which is easier to solve for obtain better solutions. Subsequent to the clustering process the nodes have been routed and have been improved with an algorithm. The algorithm has been tested on some problems, and the findings have been discussed.

Keywords: Vehicle Routing Problem, Fuzzy C-Means, Cluster First Then Route, Tabu Search.

GİRİŞ

Lojistik kelimesi 1850'lerde hayatımıza girmiş olmasına rağmen, taşıma ve nakliye faaliyetleri şüphesiz tekerleğin icadına dayanır ve dönemin getirdiği ihtiyaçlara göre lojistik faaliyetlerinin amacı ve içeriği farklılık göstermiştir. Mısırlılar piramitleri inşa ederken kilometrelerce uzaklıktaki taşları nakletmiş, atlardan ve fillerden oluşan ordular dağlardan aştırılmış, ülkeleri ticaret adına birbirine bağlayan ipek yolu, baharat yolu gibi denizden ve karadan şehir ağları kurulmuş ve tacirler uğrayacakları şehirler için kat edecekleri mesafeyi o günlerde optimize etmeye başlamıştır.

Lojistik ihtiyacı geçmişten günümüze hep varlığını korumuştur, ancak uygulanması için kullanılan araçlar ve yöntemler oldukça değişmiştir. Şüphesiz değişmeye de devam edecektir. Sanayi devrimi ile ürünlerin ve insanların daha uzak noktalara taşınması mümkün hale gelirken, günümüzde teknolojinin getirdiği yenilikler ve bilgiye kolay erişim sayesinde uzak kavramı anlamını yitirmeye yüz tutmuştur. Artık pazarlar küresel; dolayısıyla pazarlardaki rekabet çok daha çetindir. Dünyanın öteki ucunda olup aynı sektörde hizmet veren bir işletme, müşterilerinize ürün satabilirken, varlığınızı sürdürebilmek için eşdeğer bir cevap verebilme yetisine sahip olmak gerekmektedir. Kısaca, rakip olabilecek beceriye ve güce sahip olmalıdır.

Lojistik faaliyetlerinden bir tanesi olan dağıtım lojistiği rekabetin sürdürülebilmesi veya ileriye taşınabilmesi için üzerinde durulan konulardan birisidir. Müşterilerin istedikleri ürünü, en kısa zamanda en az maliyetle taşımak çözülmesi oldukça zor bir problemdir. Bu bilinçle 1950'lerde araç rotalama problemleri üzerinde çalışılmaya başlanmış, çözümü NP-Zor olan problem için etkili sonuçlar verecek kesin, yaklaşık veya sezgisel algoritmalar önerilmiştir. Dağıtım faaliyetlerinin hayatın her alanında olmasından dolayı problem çok farklı kısıtlar ile tanımlanabilmekte ve içerdiği kısıtlara göre problemin farklılaşmasıyla çözüme ulaşmak gittikçe zorlaşmaktadır. Günümüze kadar araç rotalama problemi için çözüm önerileri sunulsa bile kesin sonuç veren algoritmalarla 100-135 civarında nokta içeren bazı örnekler çözülebilmiştir. Öte yandan daha iyi sonuçlar üretecek sezgisel algoritmaların geliştirilmesi için çalışmalar devam etmektedir.

Bu çalışmanın amacı, araç rotalama problemlerinin çözümünde kullanılacak yeni bir sezgisel algoritmanın önerilmesidir. Birinci bölümde lojistik kavramı hakkında genel bilgiler sunulmuş, karar seviyeleri, lojistik faaliyetler hakkında bilgi verilmiş ve araç rotalama problemlerinin dağıtım faaliyetlerindeki önemine işaret edilmiştir. İkinci bölümde ise rotalama problemleri incelenmiş, araç rotalama probleminin temelini oluşturan gezgin satıcı problemi açıklanmış ve araç rotalama probleminin alt türleri araştırılmıştır. Üçüncü bölümde rotalama problemlerinin çözümünde kullanılan algoritma ve yöntemlere yer verilmiştir. Algoritmalar özelliklerine göre gruplandırılmış, zayıflıkları ve avantajları tartışılmıştır. Son bölümde ise önce kümele sonra rotala mantığına dayanan sezgisel bir algoritma önerilmiş ve bazı örnek veriler ile test edilmiştir. Önerilen algoritmada kümele yöntemi olarak Bulanık c-ortalama algoritması kullanılmış, talep noktaları kapasite kısıtı göz önünde bulundurularak gruplandırılmıştır. Başlangıç turu elde edildikten sonra önerilen bir komşuluk arama sezgiseli ile rotalar iyileştirilmiştir. Son olarak elde edilen bulgular yorumlanmış ve çalışma değerlendirilmiştir.

1. LOJİSTİK KAVRAMI

1.1. Lojistiğin Tanımı

Lojistik kavramı, Yunanca "logistikos" kelimesinden türemiş olup, 1840 yılında Fransız Akademisi tarafından taşımacılık şekillerini birleştiren ve koordine eden anlamına gelen "logistique" olarak tanımlamasıyla literatürde yer almaya başlamıştır [1]. Yıllar içerisinde ürün çeşitliliğinin artması, nüfus artışına paralel olarak yerleşim alanlarının genişlemesi, piyasada rekabet eden işletme sayısındaki artış, yerel pazarların yerini küresel pazarların alması, vb. etkenlerin ortaya çıkmasıyla lojistik birçok teorik ve pratik çalışmanın konusu olmuştur. Öyle ki işletmeler varlığını sürdürebilmek için kendilerine uzak bir coğrafyada konumlanmış olan müşterilere ürün veya hizmetlerini ulaştırmakla birlikte, söz konusu ulaştırma faaliyetlerini sahip olduğu insan gücü, para, teçhizat, vb. kaynakları verimli kullanarak yapmak zorunda kalmıştır. Eski bir kavram olan lojistik, dönemin ihtiyaçları dâhilinde zamanla gelişip değişerek birçok farklı şekilde tanımlanmıştır. Lojistik kavramı, Fiziksel Dağıtım, Malzeme Lojistiği Yönetimi, Malzeme Yönetimi, Fiziksel Tedarik, Dağıtım Lojistiği, Pazarlama Lojistiği, Toplam Dağıtım, vb. terimlerle de ifade edilmiştir [2]. Lojistik, müşteri ihtiyaç ve taleplerinin fark edilip; söz konusu ihtiyaçların karşılanması için para, malzeme, işgücü, teknoloji ve bilginin kullanılmasıyla, üretim ve hizmet sağlayan şebekelerin optimize edilmesi ve optimize edilmiş şebekeler sayesinde taleplerin gerekli zamanda talep noktalarına ulaştırılması işlemlerinin tamamıdır [3]. Başka bir tanımıyla Lojistik, kaynakların doğru zamanda, doğru yerde, uygun maliyetle, istenilen kalitede, performans ölçütlerinin yerine getirilmesi ve kısıtların sağlanmasıyla konumlandırılmasıdır [4].

Üretim ve tüketimin olduğu her noktada lojistik sistemlerinin ve uygulamalarının varlığı kaçınılmazdır. Piyasalardaki acımasız rekabet koşulları nedeniyle müşterilere hızlı bir şekilde ürünlerin ve hizmetlerin ulaştırılmayla birlikte taşıma maliyetlerinin minimize edilmesini gerekmektedir. Başarılı bir lojistik faaliyeti, işleri hızlandırıp müşteri hizmetlerini iyileştirirken maliyetleri düşürmektir [5]. Geçmişten günümüze

kadar dönemin ihtiyaçları dâhilinde çok farklı şekilde ifade edilmesine rağmen lojistik tanımlarında ürünlerin taşınması, stoklanması, paketlenmesi, elleçlenmesi, kontrol edilmesi, bilginin ve hizmetlerin ulaştırılması kavramları yer almaktadır. Güncel lojistik tanımı, sadece ulaşım ve depolama ile sınırlı kalmayıp, teknolojik gelişmeler ve küresel pazarların oluşmasıyla stok yönetimi, ambalajlama, depo yer seçimi gibi kavramları içerir hale gelmiştir.

1.2. Lojistikte Karar Seviyeleri

Lojistik, üretim veya hizmet sektöründe faaliyet gösteren bir işletme için temel faaliyetlerden birisidir. Sunulan hizmet veya ürün ne kadar kaliteli olursa olsun müşteriye uygun zaman ve koşullarda ulaştırılmadığı takdirde işletmeye sağladığı katkı azalacaktır. Bu nedenle lojistik kararlarının doğru ve hızlı alınması gerekmektedir. Lojistik yönetiminde karar seviyeleri kararların hiyerarşik olarak alındığı ve aktarıldığı operasyonel, taktiksel ve stratejik olmak üzere üç seviyeye ayrılmaktadırlar [6].

1.2.1. Stratejik seviye

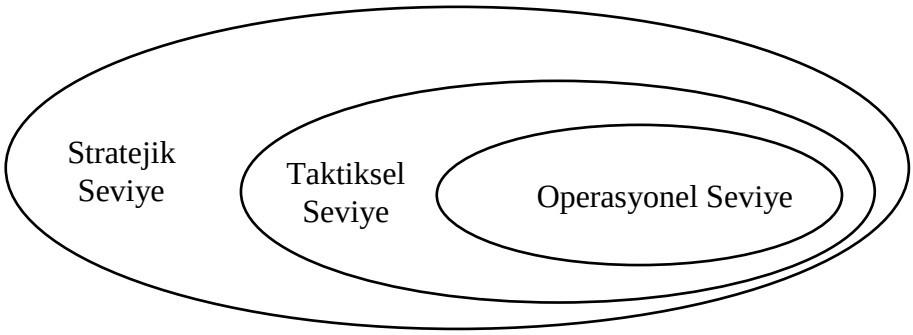
Stratejik seviyede uzun dönemi kapsayan kararlar üst yönetim tarafında alınıp, hedefler, geleceğe dair planlar ve tahminler yapılır. Stratejik seviyede alınan kararlar taktiksel ve operasyonel seviye kararların sınırlarını belirlemekle birlikte işletme hedeflerine yönelik daha genel kararlar olmaları nedeniyle belirsizlik düzeyleri yüksektir. Üretim alanları, depo sayısı, depo kapasiteleri ve yerleri, depolardan taşınacak ürünlerin nasıl bir lojistik ağı ile taşınacağı bu seviyede belirlenir.

1.2.2. Taktiksel seviye

Taktiksel seviyede orta düzey yöneticiler tarafından alınan kararlar, stratejik seviye göre daha kısa vadeli aylık, üç aylık veya yıllık dönemi kapsarlar. Stratejik seviyede alınan kararlar doğrultusunda işletme kaynaklarının etkili ve verimli kullanılması için alınan kararlardır. İşletme amaçlarına ulaşabilmek adına kaynakların nasıl ve kimler tarafından kullanılacağının belirlendiği seviyedir. Kaynakların planlanması, lojistik stratejilerinin, satın alma ve üretim politikalarının belirlenmesi taktiksel seviyede alınan kararlara örnek olarak gösterilebilir.

1.2.3. Operasyonel seviye

Operasyonel seviyede ise anlık ve günlük bazen haftalık olarak, araçların yüklenmesi, sevk edilmesi, çizelgelenmesi, rotalanması, rutin depo akış işlemleri gibi kararlar alınmaktadır. Bu seviyede, taktiksel kararların uygulanabilmesi için yapılacak görevlere dair kararlar alınır. Görevin nasıl yapılacağı detaylandırılır. Operasyonel seviyede kararlar stratejik seviyedeki gibi işletme genelini kapsamazken, stratejik seviyede alınan kararlar gibi belirsizlik seviyeleri yüksek değildir. Lojistik karar seviyelerinin kapsamı ve aralarındaki ilişki Şekil 1.1'de verilmiştir.



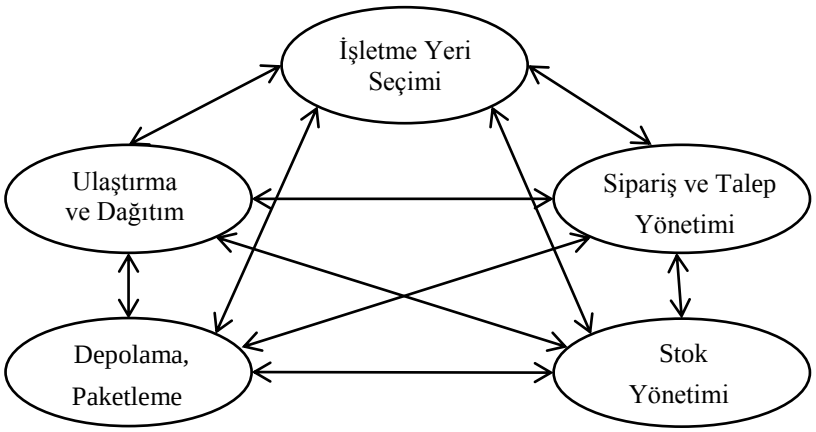
Şekil 1.1. Lojistik karar seviyelerinin kapsamı ve aralarındaki ilişki

1.3. Temel Lojistik Faaliyetler ve Dağıtım Lojistiği

Bir lojistik sistemi, üretilecek olan ürün veya hizmet için gerekli olan kaynakların tedarik edilmesinden başlayarak üretilen ürünün talep tahmin, ambalajlanması, stok kontrolü, dağıtımı, parça ve servis desteğinin sağlanması, bilişim işlemleri ve diğer faaliyetleri içerir. İşletmeler lojistik uygulamalarında, işletmenin yerinin seçilmesi, ürününün depolanması ve paketleme, stok yönetimi, taleplerinin ve siparişlerin yönetimi, nakliye ve dağıtım faaliyetleriyle ilgilenmektedirler [7]. Bu faaliyetler beş temel lojistik faaliyeti olarak kabul edilmişlerdir. Temel lojistik faaliyetleri ve aralarındaki ilişki Şekil 1.2'de verilmiştir [8].

Söz konusu temel lojistik faaliyetlerinin hepsi çok önemli olmalarına rağmen yapılan araştırmalar maliyete en fazla etki eden faaliyetin % 50 oranla ulaşım ve dağıtım faaliyetleri olduğunu göstermiştir. Stok tutma % 20, depolama % 20, yönetim faaliyetleri ise % 10 oranında toplam maliyet içerinde pay sahibidirler [4]. Maliyetlerin neredeyse yarısının ulaşım ve dağıtım faaliyetleri kaynaklı olması

nedeniyle, dağıtım lojistiği üzerinde durulması gereken önemli bir konu haline gelmiştir. Dağıtım lojistiği ürün veya hizmetin üretildiği noktadan tüketiciye verimli bir şekilde ulaştırılabilmesi ile ilgili birçok aktiviteyi bünyesinde barındırır. Bunlar, taşımanın hangi ulaşım kanalı ile yapılacağı (deniz, hava, karayolu, demiryolu), araçların rotalanması, araç kapasitelerinin ve nakliye firmasının belirlenmesi, vb. aktivitelerdir.



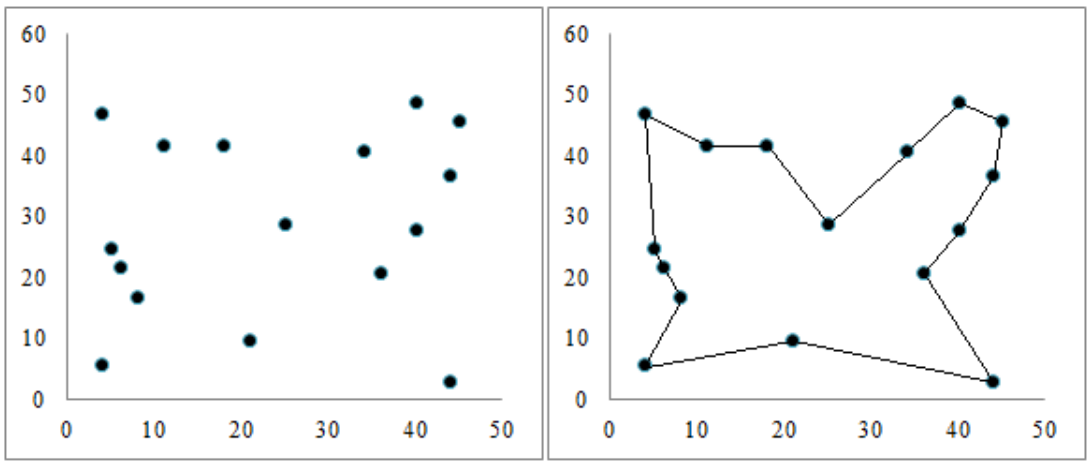
Şekil 1.2. Temel lojistik faaliyetleri ve aralarındaki ilişkiler

Araç rotalama probleminin karmaşıklığı ve dağıtım ağının yapısına bağlı olarak, duruma özel problemlerin ortaya çıkması nedeniyle hakkında birçok araştırma yapılmış ve halen yapılmakta olan bir faaliyettir. Çağın ihtiyaçlarına ve bilgi teknolojilerindeki gelişmelere bağlı olarak rotalama problemlerin kapsamı, zorluk derecesi ve çeşitliliği değişmiştir. En çok bilinen rotalama problemleri, Gezgin Satıcı Problemi (GSP, Travelling Salesman Problem – TSP) ve Araç Rotalama Problemidir (ARP, Vehicle Routing Problem – VRP). Rotalama problemleri, kendi içinde kısıtlarına göre daha alt problem türlerine ayrılmaktadırlar ve çalışmanın ilerleyen bölümlerinde GSP ve ARP detaylı bir şekilde ele alınacaktır.

2. ROTALAMA PROBLEMLERİ

2.1. Gezgin Satıcı Problemi

Ürünlerin, insanların ve hizmetin üretildiği noktadan tüketildiği noktaya en verimli şekilde nasıl taşınacağı rotalama problemlerinin konusudur. Bir satıcının belirli bir noktadan yola çıkarak uğraması gereken tüm noktalara sadece bir defa uğramak ve ilk hareket noktasına tekrar dönmek kaydıyla en kısa sürede veya en kısa yol kat ederek turlaması için uygun rotanın tayini, Gezgin Satıcı Problemi olarak literatürde yerini almıştır. GSP, belirli bir sayıda şehir çiftleri arasındaki uzaklıkların verilerek, başlama noktasına dönmek şartıyla tüm şehirlerin dolaşılarak en uygun yolun bulunmasıdır [9]. Şekil 2.1'de temsili bir gezgin satıcı problemi örneği verilmiştir. GSP anlaşılması oldukça kolay olmakla birlikte, çözüme ulaşılması problemde verilen nokta sayısına bağlı olarak zorlaşan bir problemdir. 1800'lü yılların ortalarından itibaren üzerinde çalışılan GSP, gerçek hayatta birçok uygulama alanı olması nedeniyle hala bilim insanlarının ilgilendiği konulardan birisidir. Elektronik sektöründe kartlar üzerindeki devrelerin yerleşimi ve parçaların ek kısa yollarla bağlanması, araçların ürün dağıtımı için izleyecekleri rotanın belirlenmesi, bir işletmede ürünün sırayla farklı makinalarda işlenmesi halinde ürünün hangi makinada hangi sırayla işleneceğinin çizelgelenmesi, GSP'nin uygulandığı gerçek hayat problemlerine örnek olarak verilebilirler [10].



Şekil 2.1. Gezgin satıcı probleminin temsili örneği

GSP, matematiksel olarak şu şekilde ifade edilebilir. N={1,…,n}'e kadar şehirlerin olduğu küme iken her *(i,j)* nokta çiftinin c_{ij} maliyetli bir bağlantı oluşturduğunu varsayalım. N kümesindeki her nokta çifti A kümesinden bir bağlantı ile bağlanırsa G = (N,A) şeklinde bir graf elde edilmiş olur. Herhangi *(i,j)* nokta çifti arasında bağlantı olup olmadığı x_{ij} olarak ikili bir değişken ile kontrol edilerek maliyetlerin dâhil edilmesiyle problem aşağıdaki gibi modellenmiştir.

$$enk\sum_{i\in V}\sum_{j\in V}c_{ij}x_{ij}, \qquad i\neq j \tag{2.1}$$

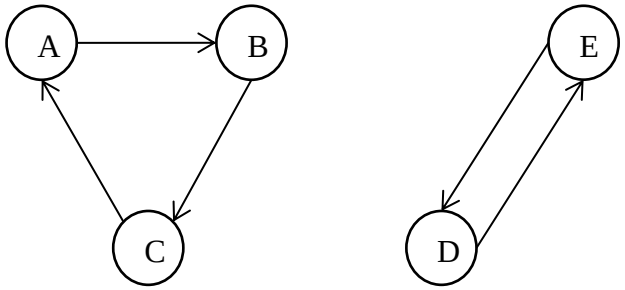
$$\sum_{j\in N}x_{ij}=1, \qquad i\neq j, \quad \forall i\in N \tag{2.2}$$

$$\sum_{i\in N}x_{ij}=1, \qquad i\neq j, \quad \forall j\in N \tag{2.3}$$

$$\sum_{i\in S}\sum_{j\in S}x_{ij}\leq |S|-1, \qquad i\neq j, \quad \forall S\subset N \tag{2.4}$$

$$x_{ij}\in\{0,1\}, \qquad \forall(i,j)\in A \tag{2.5}$$

Denklem (2.1) şehirler arasındaki bağlantı maliyetini minimize eden amaç fonksiyonudur. Denklem (2.2) ve Denklem (2.3) ile her noktaya sadece bir bağlantı gelmesi ve bir bağlantı ayrılması sağlanmış olmaktadır. Oluşabilecek alt turların engellenmesi için kısıtlara Denklem (2.4)'ün eklenmesi gereklidir. N noktalar kümesinin bir alt kümesi olan S kümesinde bulanan noktalar en fazla S kümesinin eleman sayısının bir eksiği kadar bağlantı oluşturmalıdır. Şekil 2.2'de verilen örnekte çözümde iki adet alt turun oluştuğu görülmektedir. A, B ve C noktaları bir alt tur oluşturmuş olup, kümenin eleman sayısı |S|=3'tür.

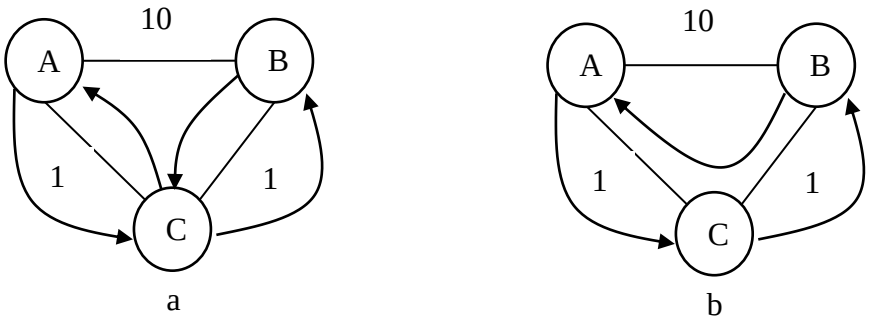


Şekil 2.2. Alt turlar bulunduran beş noktalı GSP örneği

A,B,C noktaları aralarında (a,b), (b,c) ve (c,a) olmak üzere üç adet bağlantı oluşturmuşlardır. Denklem (2.4)'de verilen kısıt test edildiğinde $3 \leq 2$ eşitsizliği elde edilmektedir. Şartın sağlanmaması verilen çözümün geçerli bir GSP çözümü olmadığını gösterir. Bu durum kısıtlarda Denklem (2.4) olmadan çözülmesiyle gerçekleşmiştir.

Modellenmesi kolay olan GSP'nin çözümü ise zordur. n adet şehir içeren bir problemde tüm olası rotalar test edilmekte ve en küçük maliyete sahip rota çözüm olarak sunulmaktadır. n şehirli bir problemde tüm şehir arasında bağlantı olduğu takdirde (n-1)! / 2 adet farklı rota oluşur. Bir noktadan bir defa geçmek kaydıyla başlama noktasına dönülmesiyle gerçekleşen tura Hamilton turu denilmektedir. GSP probleminin amacı en düşük maliyete sahip Hamilton turunu bulmaktır. 5 adet şehir içeren bir örnekte (5-1) ! / 2 = 12 adet yol test edilmektedir. Şehir sayısı 10 olursa test edilecek yol sayısı 181.440 olmaktadır. Eğer şehir sayısı 50 olursa, problemin çözülmesi için 1,5 x 10^{64} adet yolun test edilmesi gerekecek ve süper bilgisayarlar kullanılsa dahi problemin çözülebilmesi yıllar sürecektir.

Ancak, GSP için en iyi çözüm her zaman bir Hamilton turu olmayabilir. Şekil 2.3 (a) da verilen örnekte Hamilton turu olmadığı halde daha kısa bir rotanın varlığı söz konusudur. Satıcı, sırasıyla A-C-B-C-A noktalarına uğrarsa, tüm noktaları 1+1+1+1=4 birim maliyetle dolaşmış olacaktır. Çözüm iki adet alt turun birleşmiş halidir. Denklem (2.1) - (2.5) de verilen model ile bu çözüme ulaşılamayacak ve Şekil 2.3 (b)'deki A-C-B-A rotasını takip eden satıcının maliyeti 1+1+10 = 12 birim olacaktır. Denklem (2.6) problemin çözümünün ne zaman Hamilton turu olabileceğini ifade etmektedir [11].



Şekil 2.3. Çözümü Hamilton turu olmayan GSP örneği

Şekil 2.3'de verilen örnek Denklem (2.6) ile test edilirse, $10 \leq 1 + 1$ eşitsizliği elde edilir ve problemin çözümünün Hamilton turu olmadığı belirlenmiş olur. N adet şehir içeren bir problemde tüm nokta çiftleri için denklemin test edilmesi gerekmektedir.

$$d(x,y) \leq d(x,z) + d(z,y), \qquad x \neq y, \quad x \neq z, \quad y \neq z \tag{2.6}$$

Literatürde GSP'nin birçok farklı türüyle karşılaşmak mümkündür ve hepsi üzerinde yapılmış çalışmalar mevcuttur [12]. Şehirlerin koordinatlarının, uzaklıklarının değişmesi, şehirler kümesine yeni bir şehrin eklenmesi veya mevcut şehirlerin çıkarılması gibi durumlarda problem, Dinamik Gezgin Satıcı Problemine (DGSP) dönüşmektedir. Bu nedenle DGSP, GSP'ye göre çözümü çok daha zor bir problemdir. DGSP, kenar ağırlıklarının ve/veya düğüm sayı ve yerlerinin sürekli değiştiği bir durumda en kısa Hamilton rotasının bulunması problemidir [13]. Bazen i şehrinden j şehrine olan mesafe ile j'den i'ye olan mesafe eşit olmayabilir. Bu tarz problemler Asimetrik Gezgin Satıcı Problemleri (AGSP) olarak anılmaktadır. Genelleştirilmiş bir GSP problemi olan m-GSP, şehirleri bir satıcı yerine m adet satıcıyla dolaşılmasını ile ilgilenir. Tüm şehirlerin bir defa ziyaret edilmesi şartı devam ederken, m adet farklı rotanın varlığı söz konusudur. m-GSP de farklı olarak bir depo noktası belirlenir ve bu depo noktası m kadar çoğaltılır. Yani n nokta içeren bir sistem m + n kadar noktaya sahipmiş gibi problem çözülür. Problemdeki tüm noktaların oluşturduğu kenarlar için Denklem (2.6) sağlanıyorsa m satıcı için elde edilecek çözümün değeri, aynı sistemin klasik GSP olarak çözülmesi halinde elde edilecek değerden eşit veya büyüktür. GSP için verilen tamsayılı programlama modeline iki adet ek kısıt eklenmesiyle model m-GSP içinde geçerli olur. Bu kısıtlar Denklem (2.7) ve Denklem (2.8)'de verilmiştir.

$$\sum_{j=2}^{n} x_{1j} = m, \qquad \forall j \in V \tag{2.7}$$

$$\sum_{j=2}^{n} x_{j1} = m, \qquad \forall j \in V \tag{2.8}$$

Denklem (2.7) ile seçilen depo noktasından satıcı *m* sayısı kadar bağlantı ayrılması sağlanırken, Denklem (2.8) ile seçilen depo noktasına yine satıcı *m* sayısı kadar bağlantı gelmesi sağlanmış olur. m-GSP üzerinde yapılmış çok fazla çalışma yoktur.

GSP'nin etkili çözüm yöntemlerinin hala araştırılması ve çalışmaların m-GSP'ye göre uygulama alanı fazla olan Araç Rotalama Problemleri (ARP) üzerinde durulması, GSP üzerinde az çalışılmış olmasının nedenleri arasında gösterilebilir.

2.1.1. Gezgin satıcı problemlerinin çözümünde kullanılan yöntemler

GSP, modellenmesi kolay bir problem olmakla beraber, ileri teknoloji bilgisayarlarla dahi çözümü uzun sürmektedir. Bu nedenle çözüm süresini azaltmak ve daha büyük boyutlu GSP'ni çözebilmek için farklı çözüm yöntemleri önerilmiştir. Literatürde yer alan yöntemlerin bir kısmı tam sayılı programlama modeline dayanan, çözüm kümesinin tamamının veya bir kesme düzlemine göre çözüm kümesinin bir bölümünün aranmasıyla en iyi sonuçların elde edildiği yöntemlerdir. Bu yöntemler en iyi sonucu bulurken, çözüm süreleri çok uzun olmaktadır. Ayrıca, GSP'nde en iyi sonuca ulaşmak yerine, kabul edilebilir çözümler üretilebilir. Çözüm süresi çok uzun olmayan bu çözüm yöntemleri sezgisel algoritmalar olarak anılmaktadırlar.

Kesin algoritmalar, 1954 yılında Danztig, Fulkerson ve Johnson tarafından önerilen ve Denklem (2.1) – Denklem (2.5) ile verilmiş olan tamsayılı modele dayanır. Bu model üzerinde yapılan çalışmalar neticesinde dal-sınır ve dal-kesme yöntemleri önerilmiş ve bilgisayar teknolojilerinde yaşanan gelişmelerle nispeten daha büyük problemler çözülebilmiştir. 1954 yılında 49 şehirli bir problem çözülmüş iken, 2004 yılında 24978 şehirden oluşan bir GSP problemi için en iyi sonuç bulunmuştur. GSP çözümü için tarihsel süreçte yaşanan önemli gelişmeler Tablo 2.1'de verilmiştir.

Tablo 2.1. Yıllara göre çözülmüş en büyük gezgin satıcı problemleri [14]

Yıl	Araştırma Takımı	Problem Boyutu
1954	G. Dantzig, R. Fulkerson, ve S. Johnson	49
1971	M. Held ve R.M. Karp	64
1975	P.M. Camerini, L. Fratta, ve F. Maffioli	67
1977	M. Grötschel	120
1980	H. Crowder ve M.W. Padberg	318
1987	M. Padberg ve G. Rinaldi	532
1987	M. Grötschel ve O. Holland	666
1987	M. Padberg ve G. Rinaldi	2392
1994	D. Applegate, R. Bixby, V. Chvátal ve W. Cook	7397
1998	D. Applegate, R. Bixby, V. Chvátal ve W. Cook	13509
2001	D. Applegate, R. Bixby, V. Chvátal ve W. Cook	15112
2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook ve K. Helsgaun	24978

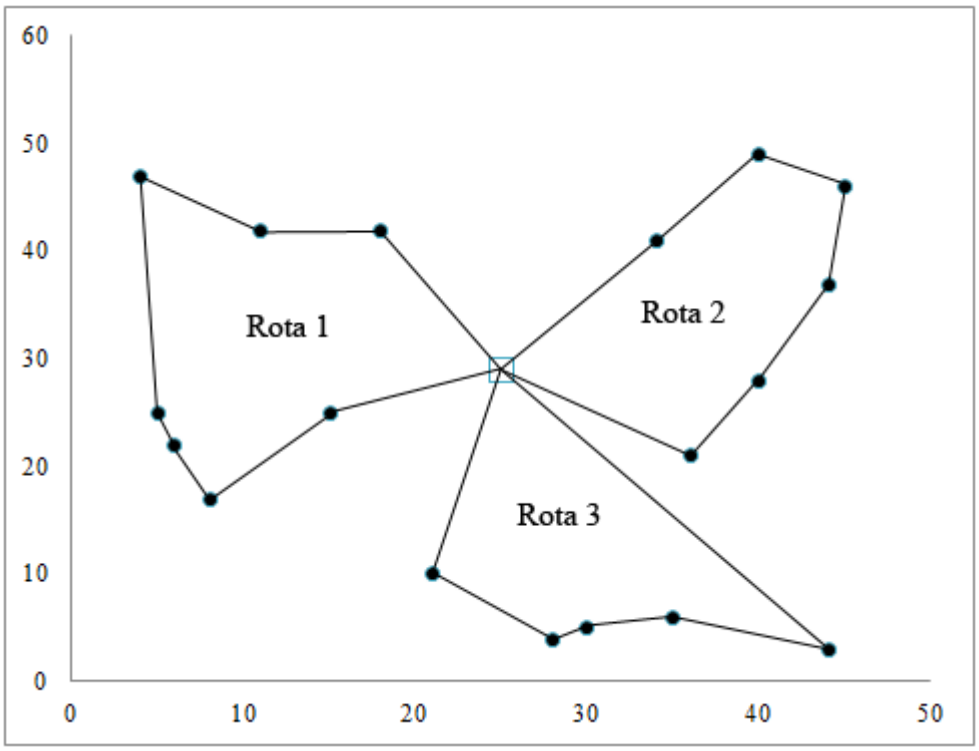
2004 yılında çözülmüş olan 24978 nokta içeren problem için, 96 adet bilgisayardan oluşan bir ağ kurulmuştur. Bu problemin, Intel Xeon 2.8 Ghz işlemcili bir bilgisayar ile çözülebilmesi için 91.9 yıl gerekmektedir. Değerlerden anlaşıldığı üzere, kesin algoritmalar küçük boyutlu problemlerde etkilidirler ve gerçek hayat içerisinde kullanımları sınırlıdır. Kesin algoritmalar hakkında yapılan çalışmalara paralel olarak, büyük boyutlu problemleri kabul edilebilir bir sonuç ile kısa zamanda çözebilen sezgisel algoritmalar geliştirilmiştir. Rotalama problemlerinde kullanılan sezgisel algoritmalar hakkında ilerleyen bölümlerde detaylı bilgi verilmiştir.

2.2. Araç Rotalama Problemi

1959 yılında Dantzig ve Ramser tarafından modellenen Araç Rotalama Problemi (ARP), merkezi bir depodan başlayıp aynı depoda sonlanmak kaydıyla talepleri bilinen müşterilere servis veren araç rotalarının minimum maliyetle belirlenmesi olarak tanımlanmıştır. ARP, m satıcılı Gezgin Satıcı Probleminin (m-GSP) bir türevidir. m-GSP'de bir noktadan hareket eden satıcılar tüm noktalardan sadece bir defa geçecek şekilde turlamak zorunda iken, ARP'de bütün bunlara ek olarak her satıcının belirli bir miktarda yük taşıyabileceği ve her noktanın talebinin farklı olacağı kısıtları eklenmiştir. ARP, üzerinde en çok çalışılan ve algoritma geliştirilen problemlerinden birisidir [15].

ARP, fiziksel dağıtım ve lojistik içerisinde merkezi bir role sahip olan, bir veya birkaç depodan hareket edilerek coğrafi olarak dağılmış müşteriler için en iyi ulaştırma veya toplama rotalarının oluşturulmasıdır [16]. Lojistik maliyetlerinin önemli bir bölümünü taşıma maliyetlerinin oluşturması nedeniyle dağıtım yönetimi ve lojistik alanında önemli bir yere sahiptir. Dağıtım maliyetini azaltmak ve hizmet kalitesini artırmak amacıyla bir dağıtım probleminin, en uygun sonuçları kabul edilebilir bir zamanda verebilecek şekilde çözülmesi üzerinde durulan konulardan birisi olmuştur [17].

Araç rotalama problemleri, içerdiği ek kısıtlara göre farklı türlere ayrılırlar. Literatürde yer alan ARP türleri, zaman, maliyet, talep noktalarının birbirlerine olan uzaklıkları, araçların kapasitesi, kısıtların dinamikliği, vb. kısıtlardan bir tanesinin veya birkaçının bir araya gelmesiyle oluşmuşlardır. Bu nedenle çok fazla ARP türü bulunmaktadır.



Şekil 2.4. Araç rotalama probleminin temsili bir örneği

Şekil 2.4'de araç rotalama problemi için bir örnek verilmiştir. Talep noktalarının ortasında yer alan bir depodan çıkan araçlar üç farklı rotayı takip ederek yine depoya geri dönmektedirler. Bir rotaya atanmış olan araçların kapasitesi rotada yer alan noktaların taleplerini karşılayabilecek büyüklükte olmak zorundadır.

Araç rotalama problemlerinin çözümünde dikkat edilmesi gereken temel kısıtlar aşağıda listelenmiştir [18]:

- Her bir talep noktası sadece bir defa ziyaret edilmelidir.
- Araçlar çizelgelendikleri yolu turlamak için depodan hareket etmeli ve yine depoya dönmelidirler.
- Araçların kapasitesi çizelgelendikleri yolun talebini karşılayacak seviyede olmalıdır.
- Her araç bir seferde sadece bir rotaya atanmalıdır.
- Noktalarının tamamının talepleri karşılanmalıdır.
- Toplam rota sayısı toplam araç sayısından büyük olmamalıdır.
- Araçların rotalanmasının temel amacı kat edecekleri yolun zaman veya maliyet olarak minimize edilmesi olmalıdır.

Araç rotalama problemleri için geçerli olan genel kısıtlarla birlikte, problemin niteliğine göre daha fazla kısıtın dikkate alınması gerekebilmektedir. Filoda yer alan araçların kapasiteleri eşit olmayabilir veya noktaların talepleri sadece belirli bir zaman dilimi içerisinde geçerli olabilir. Problemini türüne bağlı olarak dikkate alınacak ek kısıtlardan bazıları aşağıda verilmiştir:

- Araçlar atandıkları noktalara belirli zaman dilimleri içerisinde erişebilirler.
- Araçların atandıkları rotayı dolaşmaları için bir zaman kısıtı olabilir. Belirlenen zaman aralığında rotayı dolaşarak başlangıç noktasına dönmeleri gerekecektir.
- Araçların kapasitesi rotada yer alan noktaların taleplerini karşılasa bile rotanın toplam uzunluğu belirli bir sınırın üzerinde olabilir.
- Rotalarda yer alan noktalar birer talep noktası oldukları gibi, diğer noktalara veya depoya geri götürülmek ürünler nedeniyle araçların kapasitesini kullanabilirler.
- Talepler, araçlar hareket halinde iken değişebilir. Dinamik bir yapı söz konusu olabilir.

Kısıtlarda olduğu gibi, ele alınan problemin türüne ve ihtiyaçlarına göre araç rotalama problemlerinde rota uzunluğunun en küçüklenmesi dışında farklı amaç fonksiyonları da tanımlanabilmektedir. Bu fonksiyonlardan bazıları şöyledir:

- Kullanılan araç sayısının en küçüklenmesi
- Rotaların dolaşılması için gerekli olan sürenin en küçüklenmesi
- Hizmet verilecek nokta sayısının en büyüklenmesi
- Araçların yüklü kapasiteleri arasındaki farkların en küçüklenmesi
- Araçların rota uzunlukları ve tur süreleri arasındaki farkların en küçüklenmesi

Bahsedilen kısıtlar ve amaç fonksiyonları nedeniyle araç rotalama problemlerinin karmaşıklığı aynı boyuttaki gezgin satıcı problemine göre çok daha fazladır. En iyi kesin çözüm algoritmalarıyla bile ARP için büyüklüğü 135 noktaya kadar olan problemler çözülebilmektedir. Öte yandan sezgisel algoritmalar, araç rotalama problemlerinin her türünde uygulanabilmekte ve daha büyük boyutlu problemler için kabul edilebilir çözümler sunabilmektedirler [19].

2.3. Araç Rotalama Problem Türleri

Araç rotalama dağıtım lojistiğinin merkezinde yer alan en önemli faaliyetlerden birisidir. Günümüz temel lojistik faaliyetlerinde toplam maliyetin yarısı dağıtım ve ulaştırma faaliyetlerinden kaynaklanmaktadır. Pazarların küreselleşmesi ve artan rekabet koşulları işletmeleri taşıma dâhil tüm maliyetleri azaltmaya, müşteriye sunulan hizmetin kalitesini ve müşteri memnuniyetini artırmak için hatasız, hızlı dağıtım yapmaya zorlamaktadır.

Hayatın her alanında lojistiğin varlığı, birbirinden çok farklı dağıtım problemlerinin ortaya çıkmasına neden olmuş; yeni kısıtların eklenmesiyle problemlerin çözümü gittikçe zorlaşmış ve her biri bir araştırma konusu olan ARP türleri ortaya çıkmıştır. ARP türleri farklı başlıklar altında gruplandırılmaktadırlar ve literatürde yer alan çalışmalarda birbirinden çok farklı ve fazla ARP türünden bahsedilmiştir. Araç rotalama problemleri çevre faktörlerinin ve taleplerin zamana bağlı olarak değişip değişmediğine, depodan hareket eden araçların rota durumlarına, kısıt ve amaç fonksiyonlarına ve yol durumuna göre üç şekilde sınıflandırılmaktadırlar [20]. Bu çalışmada, araç rotalama problemleri çevrenin durumuna göre statik ve dinamik, rota durumuna göre açık ve kapalı uçlu, kısıtlarına göre kapasite, zaman, dağıtım türü, depo sayısı, dağıtım miktarı, dağıtım periyodu dikkate alınarak alt türlere ayrılmışlardır. Tablo 2.2'de bu üç sınıfa ve her sınıfın içerdiği alt araç rotalama türleri verilmiştir.

Tablo 2.2. Araç rotalama problem türleri

Çevrenin ve talebin durumuna göre araç rotalama problemleri	1. Statik araç rotalama problemi 2. Dinamik araç rotalama problemi
Rota durumuna göre araç rotalama problemleri	1. Açık uçlu araç rotalama problemi 2. Kapalı uçlu araç rotalama problemi
Kısıt ve amaç fonksiyonlarına göre araç rotalama problemleri	1. Kapasite kısıtlı araç rotalama problemi 2. Zaman pencereli araç rotalama problemi 3. Topla-dağıt araç rotalama problemi 4. Çoklu depo araç rotalama problemi 5. Parçalı (Kısmi) dağıtım araç rotalama problemi 6. Periyodik araç rotalama problemi

2.3.1. Çevrenin ve talebin durumuna göre araç rotalama problemleri

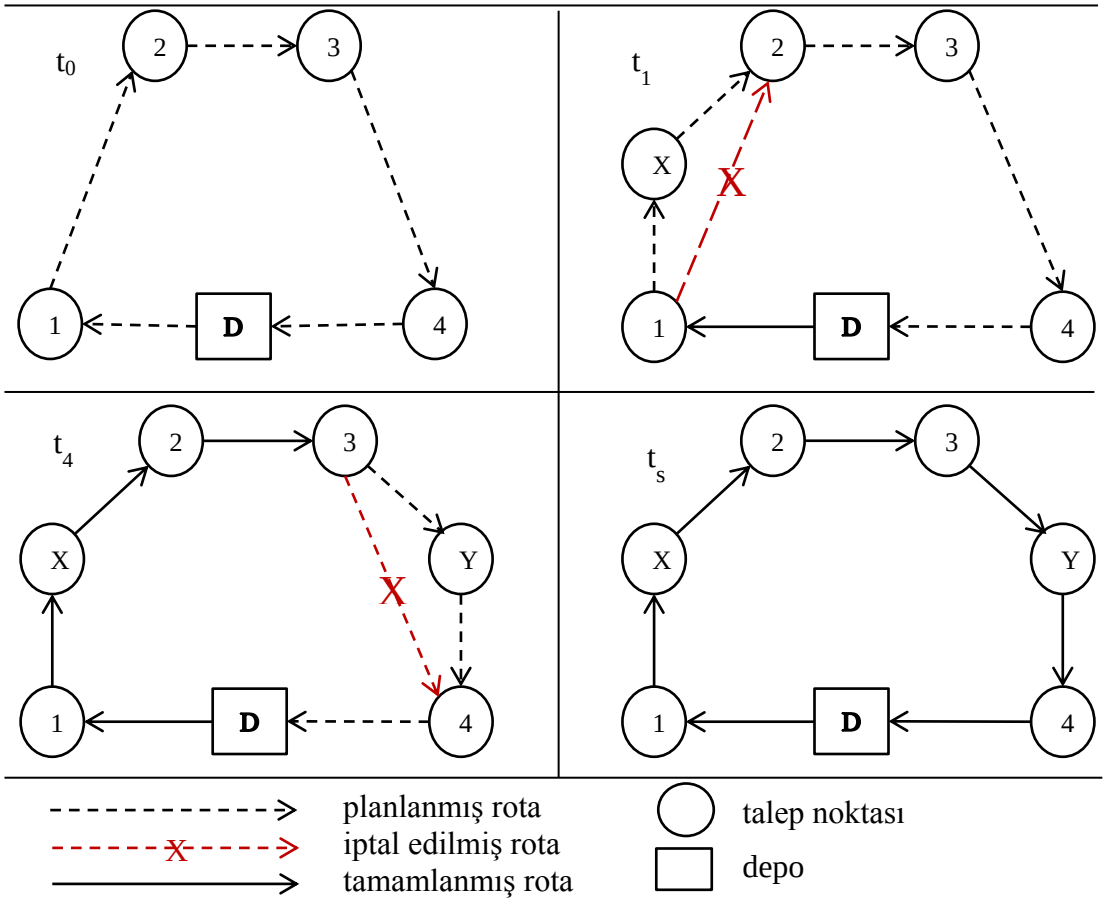
Çevrenin ve talebin durumuna göre araç rotalama problemlerinde talep noktalarından kaynaklanan zaman, talep miktarı, talep noktası sayısı, vb. kısıtların sürekli değişmesi söz konusudur. Bazen müşterilerden gelen talepler anlık olarak değişebilmektedir, hatta rota dolaşılırken yeni talep noktaları oluşabilmektedir. Bu nedenle araç rotalama problemleri bilginin deterministik ve stokastik olduğu durumlara göre statik araç rotalama problemi (SARP) ve dinamik araç rotalama problemi (DARP) olarak ikiye ayrılmışlardır.

2.3.1.1. Statik araç rotalama problemi

SARP'nde rotalama yapılırken, problemin çözümünde kullanılan depo sayısı, nokta sayısı, araç sayısı, araç kapasiteleri, noktaların talepleri, noktalar arası mesafeler, vb. veriler bilinmektedir. Problem çözülüp rotalama işlemi bitmesinin ardından bu verilerde herhangi bir değişiklik olmamaktadır. Modellenmesindeki kolaylık nedeniyle literatürde en fazla üzerinde durulan problem türüdür. Deterministik araç rotalama problemi olarak da bilinir. Talep miktarı, talep noktaları, zaman gibi problem çözümünde kullanılan değişkenlerin değişmediği bilimsel ve gerçek hayat problemleri bu grup altındadırlar.

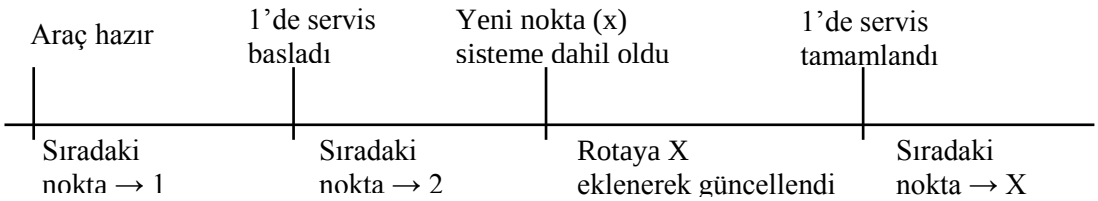
2.3.1.2. Dinamik araç rotalama problemi

DARP'nde ise SARP'den farklı olarak problem çözümünde kullanılan değişkenler/bilgi zamana bağlı olarak değişmektedir. Bir araç depodan ayrıldıktan sonra rota üzerinde yer alan noktaların talepleri artabilir veya hiçbir rotaya dâhil olmayan yeni noktalar oluşabilir. Özellikle müşteri memnuniyetinin ve hizmet kalitesinin ön planda olduğu durumlarda, rotayı anlık güncelleyebilmek gerekmektedir. Talep miktarı, talep noktaları, zaman gibi kısıtların stokastik olduğu bilimsel ve gerçek hayat problemleri bu grup altındadırlar. DARP taşıma esnasında müşteri sayısının değişmesi, müşteri taleplerinin güncellenmesi, müşteriler arasındaki mesafelerin değişmesi gibi beklenmedik durumlarda yeni kararlar alınarak rotanın gözden geçirilmesiyle yeni koşullar altında en iyi rotanın bulunmasıdır [21]. Dinamik araç rotalama probleminin mantığının tam olarak anlaşılabilmesi için Şekil 2.5'deki örnek verilmiştir.



Şekil 2.5. Dinamik araç rotalama örneği

Örnekte aracın her nokta arasını 1 birim zamanda kat ettiği varsayılmıştır. t_0 anında talep noktalarını (D-1-2-3-4-D) içeren başlangıç rotası oluşturulmuş, araç depodan 1 noktasına hareket etmiştir. t_1 anında 1 noktasının talebini karşılanmış ve yeni bir müşteri (x) talepte bulunmuştur. Bu nedenle rota gözden geçirilip güncellenerek, (1-2) bağlantısı çıkartılmış ve (1-X-2) bağlantısı eklenmiştir. t_4 anında 3 noktasının talebi karşılanmış ve sisteme y noktasının dâhil olduğu tespit edilmiştir. Tekrar rota gözden geçirilmiş ve yeni müşteri rotaya eklenmiştir. t_s anında dolaşılan rota (D-1-X-2-3-Y-4-D) olarak gerçekleşmiştir. Şekil 2.6'da olayların zaman çizelgesi üzerinde gösterilmiştir.



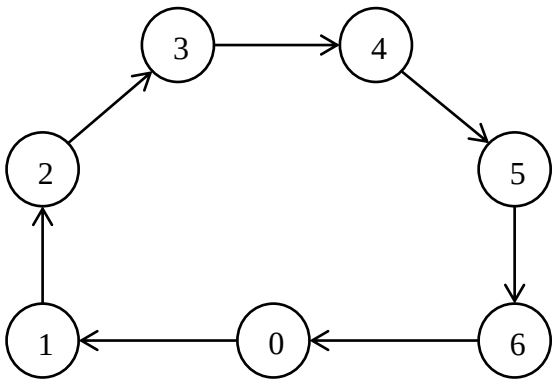
Şekil 2.6. Örnek DARP'nde olayların zaman çizelgesi üzerinde gösterimi

DARP üzerinde yapılan çalışma sayısı oldukça azdır. Öte yandan günümüzde yaşanan yeni teknolojik gelişmelerle birlikte dinamik araç rotalama problemleri üzerinde çalışmaların artacağı düşünülmektedir [22]. GPS (Global Positioning System) olarak bilinen küresel konumlandırma sistemlerinin gelişmesi ile araçlar artık sürekli takip edilebilmektedirler. Böylece gerçek zamanlı rotalama problemleri daha kolay çözülmektedir. Dinamik araç rotalama, işletmelerin operasyonel maliyetlerini düşürmesinde, müşterilere sunulan hizmet kalitesinin artırılmasında, doğaya ve çevreye verilen zararın azaltılması konusunda etkili rol oynamaktadır.

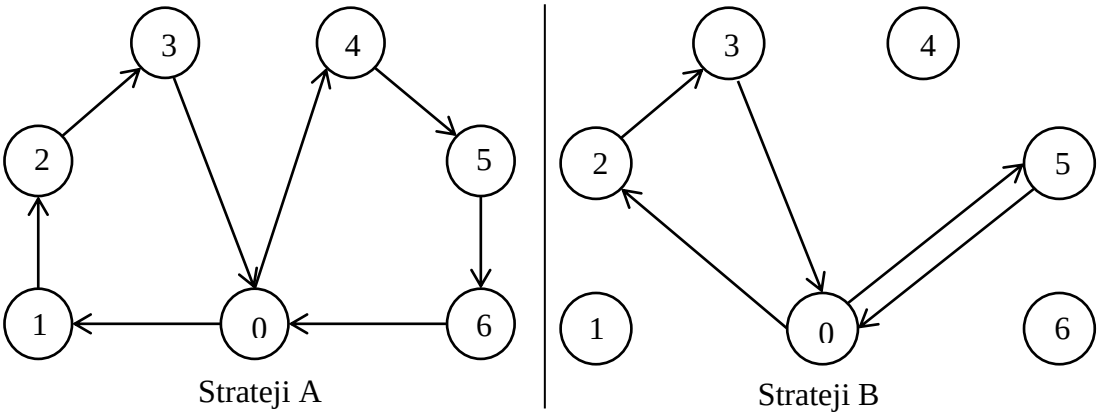
Dinamik rotalama problemlerinde en çok karşılaşılan üç durum aşağıda verilmiştir:

1. Stokastik müşteriler: Her müşteri i belirli bir o_i olasılığı ile rotada yer alır. Rotada yer almama ihtimali ise 1- o_i olacaktır.
2. Stokastik talepler: Her müşteri i için alınan talep bir değişken olan d_i kadardır ve d_i değeri sabit değildir.
3. Stokastik zamanlar: Her müşteri i için var olan f_i servis zamanı ve (i,j) düğümleri arasında yolculuk süresi t_{ij} sabit olmayan birer değişkendirler [23].

DARP çözümünde talep ve müşteri hakkındaki bilgiye bağlı olarak iki adet strateji önerilmiştir [24]. Bu stratejilerle dinamik problemler, statik problemlere benzetilerek çözülmeye çalışılmıştır. Birinci stratejide tüm noktalar talep olup olmadığına bakılmaksızın rotalanmaktadır (Şekil 2.7). Ardından depodan hareket eden araç sırasıyla noktalara uğrayarak talebi olanlara servis vermektedir. Araç kapasitesinin rotanın talebini karşılayamaması durumunda araç depoya dönmekte ve rotanın kalanına talepleri aynı mantıkla taşınmaktadır. Bu stratejiler Şekil 2.8'de verilmiştir.



Şekil 2.7. DARP'de noktaların tamamının rotalanması



Şekil 2.8. DARP için geliştirilen stratejiler

Strateji A'da talepler araç depodan ayrılmadan önce bilinmemektedir. Bu nedenle araç ilk aşamada belirlenmiş rotadaki noktalara sırasıyla uğrayarak talep olan noktalara dağıtım yapmaktadır. Aracın talepleri karşılayamaması halinde depoya dönülür ve ikmal yapılarak kalan noktalara dağıtım yapılır. SARP'den farklı olarak talebin karşılanamaması halinde depoya ikmal için yapılacak seferler ek maliyetlerin ortaya çıkmasına neden olacaktır. Strateji B'de ise araç depodan veya talep noktasından ayrılmadan önce rotadaki hangi noktaların talebi olduğu bilinmektedir. Böylece hiç talebi olmayan noktalar atlanmakta bir sonraki noktaya araç yönlendirilmektedir.

m-GSP için geliştirilen model üzerinde yapılan değişikliklerle tek araçlı dinamik araç rotalama problemi için bir model yazılabilir. Çalışmada daha önce Denklem (2.1) - (2.8)'e kadar verilmiş tüm denklemler m-GSP'ini modellemektedir. DARP problemlerine uygulanabilecek genel iki aşamalı modelin gösterimi için m-TSP modelinin Denklem (2.9)'daki gibi amaç fonksiyonunun güncellenmesi ve Denklem (2.10) ile Denklem (2.11) deki kısıtların eklenmesi gerekmektedir.

$$enk\sum_{i<j} c_{ij}x_{ij} + Q(x) \tag{2.9}$$

$$x_{1j} \in \{0,1\}, \qquad j = (2,...,n) \tag{2.10}$$

$$x_{j1} \in \{0,1\}, \qquad j = (2,...,n) \tag{2.11}$$

$Q(x)$, beklenen ikinci aşama fonksiyonudur *(recourse function)*.

2.3.2. Rota durumuna göre araç rotalama problemleri

Rotalama problemlerinde araçlar talep noktaları arasında dağıtıma bir noktadan başlamalı bir noktada bitirmelidirler. GSP'de başlama noktası rastsal olarak belirlenirken, ARP'de genellikle başlama noktaları depolardır. ARP'de araçlar depodan hareket eder ve kendilerine atanan rotayı dolaştıktan sonra başladıkları depoya dönerler. Fakat bazı durumlarda bir noktadan hareket eden araç başladığı noktaya geri dönmez ve turunu başka bir noktada tamamlar. Araçlar turlarına başlama ve bitiş noktalarına göre kapalı uçlu araç rotalama problemleri ve açık uçlu araç rotalama problemleri olarak sınıflandırılırlar [20]. Şekil 2.9'da kapalı uçlu araç rotalama problemi ve açık uçlu araç rotalama problem örnekleri verilmiştir.

2.3.2.1. Açık uçlu araç rotalama problemi

Depodan başlayarak, bir noktada veya farklı bir depoda araç turunun tamamlandığı problemler açık uçlu araç rotalama problemleridir (AUARP). Başlangıç ve bitiş noktası dışında her nokta bir gelen birde giden olmak üzere komşu noktalarda bağlantı kurmak zorundadır. Başlangıç ve bitiş noktalarının sadece bir kenar ile bağlantılı olması için problem Denklem (2.12) ile sınırlandırılır.

$$\sum_{k\in V}\sum_{j\in N} x_{0jk} + \sum_{k\in V}\sum_{j\in N} x_{i0k} = 1, \qquad i \neq 0, \quad j \neq 0 \tag{2.12}$$

Başlangıç noktasının *0* noktası olduğu bir problemde *k*, V araçlar kümesinin bir elemanı iken x_{0jk}, depodan *k* aracıyla herhangi bir *j* noktasına giden bağlantıyı, x_{i0k} ise herhangi bir *i* noktasından depoya yapılan bağlantıyı temsil etmektedir. Denklem depoya sadece bir bağlantı yapılabilmesine izin vermektedir. Böylece problem açık uçlu bir rotalama problemi olacaktır. AUARP literatürde fazla yer almamakla birlikte gerçek hayatta örnekleri bulunmaktadır. Farklı şehirlerde şubesi olan araç kiralama firmalarında, araçlar bir şehirden kiralanıp başka bir şehirde teslim edilebilmektedir. Kargo taşımacılığında bir şehirden harekete geçen araçlar, hedef şehre rotası üzerindeki diğer şehirlerin kargo paketlerini teslim ederek gitmektedirler. Harekete başladıkları noktayı depo olarak kabul edersek, kargo araçları başladıkları noktaya dağıtım işlemi sonunda geri dönmemektedirler. Nakliye işlerini yapan taşımacılık firmalarında da benzer problemlere rastlanmaktadır.

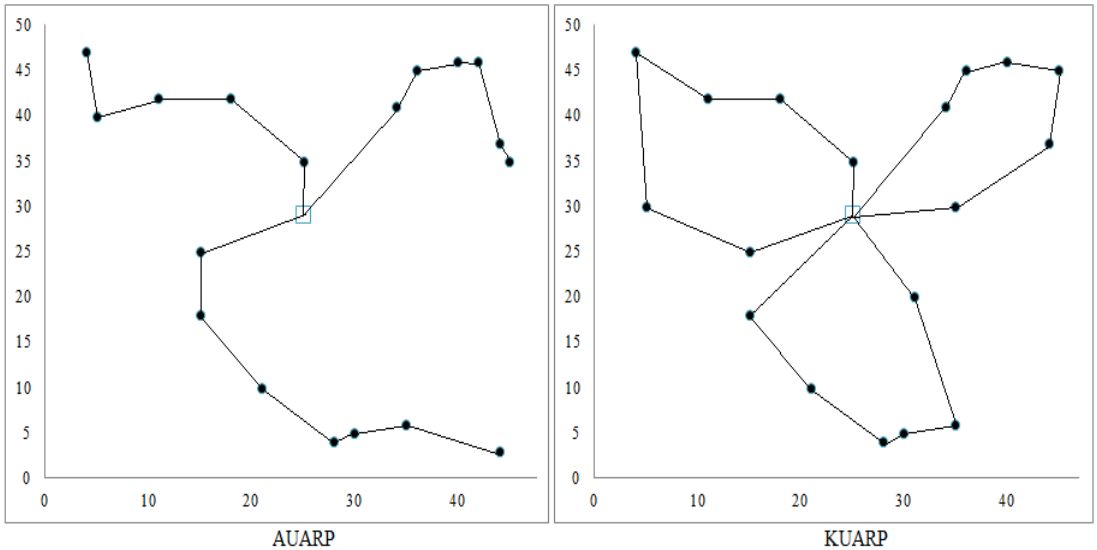
2.3.2.2. Kapalı uçlu araç rotalama problemi

Kapalı uçlu araç rotalama problemlerinde (KUARP) araçlar, atandıkları rotayı dolaştıktan sonra ilk harekete geçtikleri noktaya geri dönmek durumundadırlar. Bir araç ile bir depodan yapılacak rotalama için her nokta bir gelen birde giden olmak üzere iki bağlantıya sahiptir. Deponun iki bağlantı yapması için gerekli olan kısıt Denklem (2.13) ve Denklem (2.14)'de verilmiştir.

$$\sum_{k \in V} \sum_{j \in N} x_{0jk} = 1, \qquad j \neq 0 \tag{2.13}$$

$$\sum_{k \in V} \sum_{i \in N} x_{0ik} = 1, \qquad i \neq 0 \tag{2.14}$$

Denklem (2.13) ve Denklem (2.14) ile problem, her bir *k* aracı için depodan ayrılan ve depoya gelen birer bağlantı yapılmasını sağlamaktadır. Şekil 2.9'da kapalı uçlu araç rotalama problemi için üç adet rota oluşmuş, depo ile toplam altı bağlantı yapılmıştır.



Şekil 2.9. Açık ve kapalı uçlu araç rotalama problemi

2.3.3. Kısıt ve amaç fonksiyonuna göre araç rotalama problemleri

Araç rotalama problemleri, uygulama alanlarının geniş olması ve her uygulamanın diğer araç rotalama çözümlerinden farklı özellikler gösterebilmesi sebebiyle kısıtlarına göre birçok türe ayrılabilir. Araç rotalama problemlerinin en temel ve

üzerinde en çok çalışılan türü taleplerin değişmediği, her talebin sadece bir araç ile karşılandığı ve araç kapasitesi ile maliyetlerinin sabit tutulduğu kapasite kısıtlı araç rotalama problemidir (KKARP). Araç rotalama problemleri, içerdiği kısıtlara göre sınıflandırılırlar. Ancak araç rotalama problemlerinin çözümü için tanımlanmış çok fazla kısıt olması nedeniyle belirlenen sınıfların sınırları tam olarak belirli değildir. KKARP, müşterilere belirlenen zamanlar içerisinde taleplerin ulaştırılması gerektiği durumlarda zaman pencereli bir probleme dönüşürken, talepler birden fazla araç ile karşılanabildiğinde parçalı dağıtım problemi olmaktadır. Hatta problem her iki kısıtı da içerebilmektedir. Bu başlık altında literatürde en çok yer alan kısıt ve amaç fonksiyonlarına göre araç rotalama problemlerine yer verilmiştir.

2.3.3.1. Kapasite kısıtlı araç rotalama problemi

Kapasite kısıtlı araç rotalama problemleri (KKARP) üzerinde en fazla çalışılan ARP türüdür. Öyle ki ARP denilince akla doğrudan KKARP problemi gelmektedir. KKARP müşterilerin ve taleplerin deterministik olduğu halde, q_k kapasitesine sahip en az bir araç yardımıyla, müşteri taleplerinin en fazla araç sayısı kadar rota ile çizelgelenmesidir. Kapasite kısıtlı araç rotalama probleminde yer alan kısıtlar aşağıda detaylı olarak listelenmiştir:

- Sistemde en az bir adet depo bulunmalıdır.
- Her müşterinin talebi d_i ve müşteri sayısı n deterministik olmalıdır.
- Müşterilerin talebini karşılayabilecek en az m adet araç bulundurulmalıdır.
- Her müşteri bir defa ve sadece bir araç ile ziyaret edilmelidir.
- Rota sayısı toplam araç sayısına eşit veya az olmalıdır.
- Her aracın belirli bir taşıma kapasitesi q_k olmalıdır.

Bazı KKARP türlerinde depo sayısı birden fazla olmaktadır. Bu durumda noktaların oluşturduğu rotalar en yakın depoya bağlanarak, problem bir depo içeren birden fazla problemin birleşimi gibi değerlendirilmektedir. Ayrıca araç kapasiteleri de farklılık gösterebilmektedir. Bir araç filosunda yer alan tüm araçlar aynı kapasiteye sahipse problem homojen filolu, aksi halde heterojen filolu kapasite kısıtlı araç rotalama problemi olmaktadır. Herhangi bir kısıta ait verilerin stokastik olması durumunda problemin dinamik bir probleme dönüşmektedir.

KKARP'ne ait tamsayılı model aşağıdaki gibidir [25]:

$$enk \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{2.15}$$

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \qquad \forall i \in N \tag{2.16}$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq q_k, \qquad \forall k \in V \tag{2.17}$$

$$\sum_{j \in N} x_{0jk} = 1, \qquad \forall k \in V \tag{2.18}$$

$$\sum_{i \in N} x_{i0k} = 1, \qquad \forall k \in V \tag{2.19}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \qquad \forall h \in N, \quad \forall k \in V \tag{2.20}$$

$$x_{ijk} \in \{0,1\}, \qquad i \neq j, \qquad \forall i, j \in N, \qquad \forall k \in V \tag{2.21}$$

Aynı kapasiteye sahip (homojen) v adet araç, n noktanın olduğu bir sistemde x_{ijk}, k aracı ile *(i,j)* bağlantısının kurulduğu durumda "1" değerini alan ikili bir değişkendir. d_i, i'nin, C müşteriler kümesinin elemanı olduğu halde noktanın talebini ifade eder. Denklem (2.15) modelin amaç fonksiyonudur. Fonksiyon ile k adet araç ile *(i,j)* arasındaki bağlantıların c_{ij} ağırlığına göre en küçüklenmesine çalışılmaktadır. Denklem (2.16) her noktaya sadece bir araç ile hizmet verilebilmesini sağlayan kısıttır. Rotaların araç kapasitesi olan q_k değerini aşmaması Denklem (2.17) ile modele dâhil edilir. Denklem (2.18) ve Denklem (2.19) her k aracı için depo noktasına bir gelen birde giden bağlantının yapılmasını sağlar. Depoya yapılan bağlantı sayısı araç sayısının iki katı kadar olmalıdır. Denklem (2.20) ise akışı düzenleyen kısıtı ifade eder. Bir noktaya gelen bir araç varsa, araç mutlaka o noktadan ayrılmalıdır.

KKARP problemi ortaya çıktığı günden itibaren üzerinde en fazla çalışılan ARP türü olması nedeniyle literatürde hakkında oldukça fazla çalışmaya rastlanılmaktadır. Kesin çözüm elde etmek amacıyla kullanılmış en popüler yöntem dal-sınır

yöntemidir. En kısa dallanan ağaç ve en kısa yol yöntemlerinin gevşetilmesine dayalı geliştirilen yöntemlerle 50 nokta içeren araç rotalama problemleri ancak çözülebilmiştir [26-28]. Daha sonra "Lagrange" fonksiyonunun gevşetilmesi ile elde edilen dal-kesme yöntemleri ile 75-101 şehirli problemlerin bazılarına %1 sapma ile yaklaşılabilmiştir [29]. Bir tarafta aynı yöntemler kullanılarak çok büyük boyutlu gezgin satıcı problemleri çözülebilmişken, KKARP için ancak 130 civarı şehir içeren problemler, sezgisel bir yöntem olan tabu aramanın kısıtların ayrıştırılmasında kullanılmasıyla tam olarak çözülebilmiştir [30]. Yine de 100 nokta üzerindeki her problem çözülememiş sadece birkaç örnek veri üzerinde optimum sonuca ulaşılabilmiştir. KKARP için büyük boyutlu problemlerin çözülememesinin nedeni, GSP'nden fazla olarak talep ve araç kapasite kısıtı içermesi ve sistemde birden fazla aracın olmasıdır. Kesin sonuç veren algoritmalar üzerine yapılan çalışmalar hakkında detaylı literatür taramaları yapılmıştır [31-33].

Kesin sonuç veren algoritmalar kadar sonuca kesin olarak vermeyen ancak çok daha büyük boyutlu problemler için kabul edilebilir sonuçlar bulabilen sezgisel algoritmalar üzerine de çalışmalar yapılmıştır. Sezgisel algoritmalardan en çok kullanılanı Clarke and Wright tarafından önerilen tasarruf algoritmasıdır [34]. Ancak bu algoritmanın açgözlü hareket etme potansiyeline (depodan uzakta olan noktaların tasarruf değerinin depoya yakın olan noktalara göre çok fazla olması) sahip olması nedeniyle model üzerinde güncellemeler yapılmıştır [35]. Günümüze kadar tasarruf algoritması fonksiyonu üzerinde birçok değişiklik yapılmıştır [36, 37]. Algoritmanın hızlı çözüme ulaşması ancak iyi sonuçları üretme garantisinin olmaması nedeniyle genellikle tur kurucu sezgisel olarak kullanılmıştır. Başlangıç turu elde edildikten sonra iyileştirme algoritmaları ile sonuç geliştirilmiştir. Bütün bu sezgisel algoritmaların yanında özellikle 1990 yıllarından itibaren metasezgisel olarak nitelendirilen ve genellikle doğadan esinlenilerek geliştirilmiş sezgisel algoritmalar KKARP'nin çözümünde kullanılmaya başlanmıştır. Metasezgiseller, problemin ek kısıtlarda özelleştirilmiş alt türlerine uyarlanabilmektedir. Benzetilmiş Tavlama, Tabu Arama ve Karınca kolonisi gibi birçok algoritma KKARP problemlerinin çözümünde kullanılmıştır [38]. Son yıllarda KKARP için yeni metasezgiseller de önerilmiştir. Tabu Arama algoritması geliştirilmiş [39], öncelikle tasarruf sezgiseli ile elde edilen başlangıç rotasının tabu arama mantığına dayalı uyarlanabilir hafızaya

sahip bir algoritma ile iyileştirilmesi sağlanmıştır [40]. Uyarlanabilir hafıza prosedürü ile kılçık olarak isimlendirilen rota parçacıkları birleştirilerek daha iyi sonuçlara ulaşılmaya çalışılmıştır. KKARP çözümü için uyarlanabilir hafıza kullanılan bir diğer çalışmada ise seçkin çözüm parçaları aranmış ve elde edilen sonuçlar depolandıktan sonra bir tur kurucu sezgiselle birleştirilmiş ve tabu arama algoritması ile sonuçlar iyileştirilmiştir [41]. Ayrıca genetik algoritma ile karınca kolonisi algoritmasının birlikte kullanılmasıyla hibrit algoritmalar da elde edilmiştir [42].

2.3.3.2. Zaman pencereli araç rotalama problemi

Zaman pencereli araç rotalama problemi (ZPARP), KKARP'nin talep noktalarına yapılacak servise ait zaman aralığının sınırlandırılmış halidir. KKARP'de verilen kısıtlara ek olarak aşağıda verilen kısıtlar problemde yer almaktadır:

- Bir i müşterisine $[a_i,b_i]$ zaman aralığında hizmet verilebilir. a_i, hizmetin en erken başlayacağı zamanı, b_i ise bitmesi gereken zamanı gösterir.
- Bir araç (i,j) noktası arasındaki c_{ij} mesafesi belirli bir $[a_i,b_i]$ zaman aralığı içerisinde kat edilmelidir.

Her nokta için $[a_i=0, b_i=\infty]$ olması halinde problem KKARP olmaktadır. İlk kısıtta noktalar arası yolun ne kadar zaman aldığının bir önemi yok iken, her noktaya verilecek hizmetin zaman aralığı belirlenmiştir. ZPARP, zaman pencerelerinin kesin ve esnek olmasına göre iki gruba ayrılırlar. Esnek pencereli problemlerde, a_i zamanından önce veya b_i zamanından sonra gelen araçlar talepleri karşılayabilirken, kesin pencereli problemlerde a_i zamanından önce noktaya ulaşan araç, pencerenin açılması anında kadar beklemek zorundadır. Aracın b_i zamanından sonra ulaşması halinde talep karşılanamaz [43]. ZPARP'ne genellikle esnek zaman çizelgelerine bağlı olarak hizmet veren işletmelerde rastlanılmaktadır. Banka para sevkiyatları, servis taşımacılığı, posta ve kurye hizmetleri bu kategoridedir [44]. İkinci kısıtta ise noktalar arası yollardan tanımlı zaman aralıkları içerisinde geçilmelidir. Eğer bir kenar (vertex) üzerinden belirlenen zaman diliminde geçilmezse yol kapanabilir veya uzayabilir. Trafiğin yoğun olduğu saatlerde, bazı araç türlerinin yolları, köprüleri veya tünelleri kullanmalarına izin verilmeyebilir. Rota üzerinde böyle bir kısıt söz konusu ise yolun araç için kullanıma açık olduğu zaman diliminde kat edilmesi

gerekmektedir. Örneğin kamyon, otobüs, vb. büyük araçların gün içerisinde yerleşim birimlerinde bazı yollara girişleri yasaktır. Denklem (2.15) – Denklem (2.21) arasında verilen KKARP modeline aşağıda verilen zaman kısıtları eklenilerek model ZPARP'ne dönüştürülebilir [45].

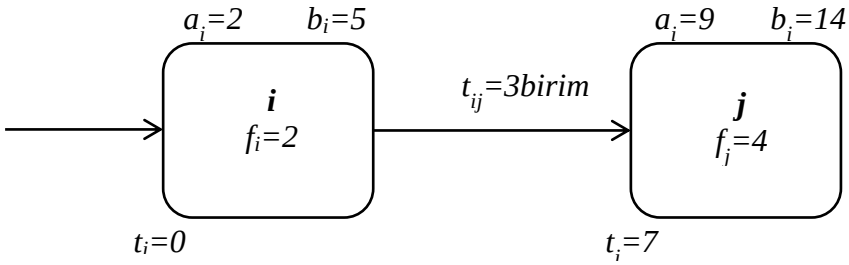
$$\sum_{i\in N}\sum_{j\in N} x_{ijk}(t_{ij} + f_i + w_i) \le r_k, \qquad \forall k \in V \tag{2.22}$$

$$t_0 = w_0 = f_0 = 0 \tag{2.23}$$

$$\sum_{k\in V}\sum_{i\in N} x_{ijk}(t_i + t_{ij} + f_i + w_i) = t_j, \qquad \forall j \in N \tag{2.24}$$

$$a_i \le (t_i + w_i) \le b_i, \qquad \forall i \in N \tag{2.25}$$

Bir problemde *(i,j)* nokta çifti arasındaki yolculuk t_{ij} kadar, *i* noktasına servis f_i kadar, *i* noktasında zaman penceresi açılmadan önceki bekleme w_i kadar zaman sürerken; t_i, *i* noktasına ulaşma zamanı, a_i servise en erken başlanacak zamanı, b_i de en geç başlanacak zamanı ifade etmektedir. Denklem (2.22), *(i,j)* noktasının *k* aracı ile kat edilmesi halinde her nokta çifti arasındaki yolculuk süresi, noktaların servis süresi ve bekleme süreleri toplamının araç için belirlenmiş toplam r_k süresini aşmamasını sağlamaktadır. Denklem (2.23)'de başlama noktası olan depoda geliş zamanı, bekleme süresi ve servis süresi sıfıra eşitlenmektedir. Denklem (2.24) *i* noktasından *j* noktasına hareket ederken, aracın *j* noktasına ulaşma zamanının hesaplanabilmesi için modelde yer almaktadır. Bir araç *i* noktasına t_i anında gelir; *wi* kadar süre bekler (zaman penceresi açık değilse); f_i kadar süre hizmet verir ve t_{ij} kadar sürede *i* noktasından *j* noktasına ulaşırsa; *j* noktasına geliş zamanı t_j olacaktır. Şekil 2.10'da bu ilişki görsel olarak bir örnek ile ifade edilmektedir.



Şekil 2.10. ZPARP'nde kullanılan zaman değişkenleri

Örnekte *i* noktasına t=0 anında (t_j=0) gelen araç a_i=2 olması nedeniyle (w_i=2-0) 2 birim zaman beklemek zorundadır. t=2 anında *i* noktasına hizmet verilmeye başlanır ve servis süresi f_i=2 olması nedeniyle t=4 anında hizmet tamamlanır. t_{ij} arasındaki yol 3 birim zaman sürmektedir. *j* noktası için servis penceresi henüz açılmadığından wj = 9-7 = 2 birim olarak hesaplanır. Denklem (2.24)'e göre t_j =7 olacaktır. Denklem (2.25) ise geliş zamanı ve aracın bekleme süresinin, i noktasında hizmetin başlanabileceği en erken (a_i) ve en geç (b_i) zamanları arasında olmasını sağlayan kısıtı ifade etmektedir. Bu kısıt modelde yer almadığı takdirde araçlar [a_i,b_i] zaman aralıkları içerisinde noktalara servis veremeyecektir.

ZPARP'nde kullanılan araç sayıları sabit olsa bile NP-Zor sınıfında bir problem olmaktadır [46]. Bu nedenle kesin sonuç veren algoritmalar yerine sezgiseller üzerinde daha fazla durulmuş, ancak kesin çözüm yöntemleri de araştırılmaya devam edilmiştir. Problemin mevcut kısıtlarla çözümünün zor olduğunun anlaşılmasının ardından kısıtların gevşetilmesi ile amaç fonksiyonuna bir ceza değer tanımlanmış ve gevşetilmiş model ile çözümler elde edilmiştir. 1990'dan itibaren lagrange gevşetmesi ve k-ağaç arama algoritmasının gevşetilmesi üzerine yapılan çalışma sayısı oldukça fazladır. Madsen ve ark. Lagrange gevşetmesi üzerinde herhangi bir sayısal sonuç olmadan dört farklı yaklaşım sunmuşlardır [47]. Ayrıca dal-sınır yöntemine dayanan yeni bir model oluşturulmuş ve modeldeki kısıtların gevşetilmesi ile problem için iyi sonuçlar elde edilmeye çalışılmıştır [48]. Lagrange gevşetmesi ile bazı test problemlerinin çözümü 1997 yılında yapılabilmiştir ve ZPARP için en iyi sonuçları veren yöntemler bu yöntemi kullanmaktadırlar. Denklem (2.16) da verilen bir noktanın en fazla bir defa ziyaret edilebileceği kısıt modelden kaldırılmış ve amaç fonksiyonu Denklem (2.26)'da verildiği haliyle güncellenmiştir [49].

$$enk\sum_{k\in V}\sum_{i\in N}\sum_{j\in N}(c_{ij}-\lambda_j)x_{ijk}+\sum_{j\in C}\lambda_j \qquad (2.26)$$

λ_j, *j* noktasındaki müşteriye hizmetin maliyetini ifade eder ve *j* noktasına hizmet verilmesi için gerekli olan kısıt ile ilişkilidir. Çarpan vektörünün bir değeri için $\lambda=(\lambda_1+\lambda_2+...+\lambda_c)$ model, her araç için bir alt probleme dönüşmektedir [49]. 1999 yılından itibaren ZPARP'nde kullanılan test problemlerinden büyüklüğü 100 noktaya kadar olanların birçoğu kesin olarak çözülebilmiştir [49-51].

2.3.3.3. Topla-dağıt araç rotalama problemi

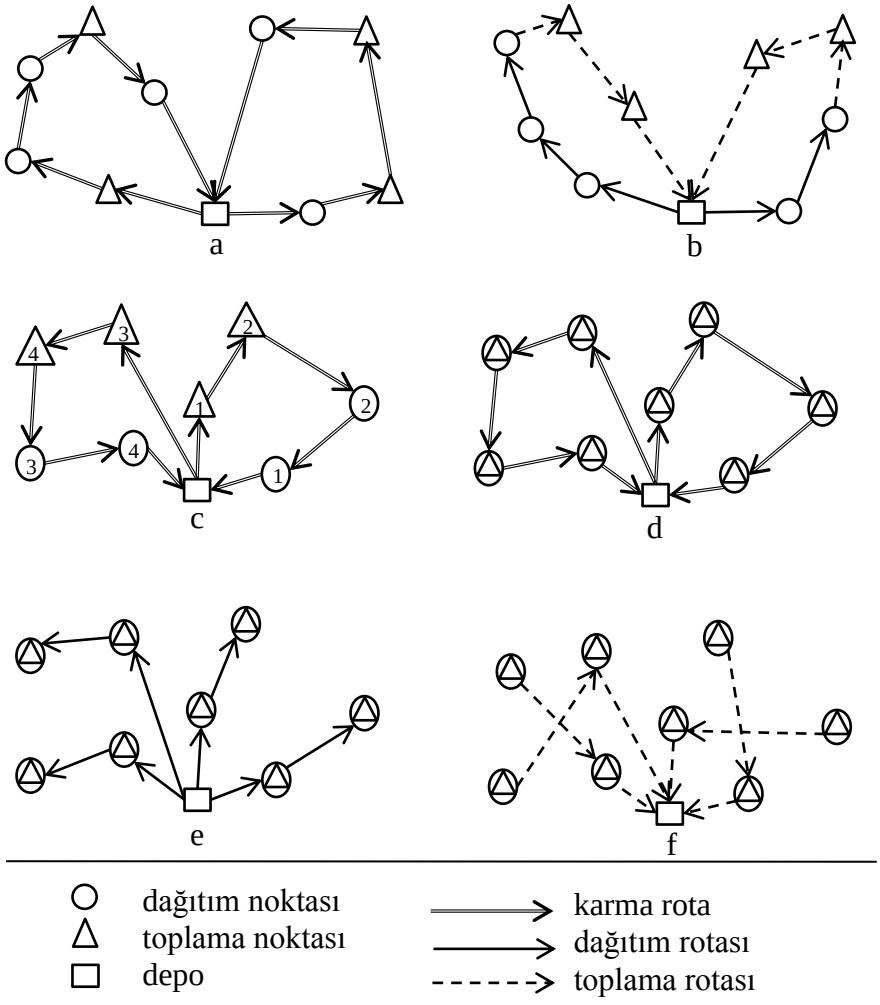
Temel araç rotalama probleminde, noktaların belirli bir talebi olduğu kabul edilir ve bu talepler depolardan karşılanmaktadır. Ancak hem talep eden hem de talepleri karşılayan noktaların olduğu araç rotalama problemleri de mevcuttur. Topla-dağıt araç rotalama problemlerinde (TDARP), araçlar bazı noktalardan ürünleri toplarken diğer noktalara da bu ürünleri taşımaktadırlar. Daima depolarda ürün stoklanması şartı yoktur. Özellikle talep noktalarından arızalı, ömrünü tamamlamış veya depozitolu ürünleri toplanması gibi tersine lojistik uygulamaları ile ilgilenen işletmelerde bu tarz problem uygulamalarına rastlanmaktadır. Tersine lojistik, ürünlerin ve bilginin tüketim noktasından orijin noktasına doğru akışının yönetilmesi olarak tanımlanmaktadır [52]. Ayrıca evsel atıkların, ambalaj atıklarının, pil, atık yağ, ömrünün tamamlamış lastiklerin geri kazanımı için kaynağından yapılan toplamalar TDARP problemlerinin kullanıldığı tersine lojistik uygulamalarına örnek olarak verilebilirler. Topla-dağıt problemlerinde karşılaşılan kısıtlardan bazıları aşağıda verilmiştir:

- i noktasında dağıtım (d_i) veya toplama (p_i) taleplerinden sadece birisi veya ikisi birden eşzamanlı olarak oluşabilir.
- Toplama işlerinin yapılabilmesi için önce dağıtım işlerinin tamamlanması gereklidir. Bazı türlerde karışık olarak toplama ve dağıtım işlemi yapılabilir.
- Rota üzerinde yer alan noktaların (p_i) ve (d_i) değerleri toplamı araç kapasitesini aşamaz.

Kısıtlarında yansıttığı üzere TDARP problemi kendi içerisinde alt türlere ayrılmaktadır. Bu çalışmada dört adet TDARP türü verilmiş ve Şekil 2.11'de eşzamanlı topla-dağıt, önce dağıt sonra topla, talebe bağlı taşıma, hızlı taşıma problemleri gösterilmiştir [53].

Şekil 2.11(a)'da dağıtım ve toplama işleri aynı anda yapılan klasik topla-dağıt türü problem verilmiştir. Bu problem türünde araçlar topladıkları yüklerle sonraki noktalara iletilecek talepleri birlikte taşırlar. Araç kapasitelerinin aşılmadan rotalama yapılması gereklidir. Şekil 2.11(b)'de ise önce dağıt sonra topla türü problem gösterilmiştir. Araçlar ilk toplama noktasına kadar noktaların talepleri sonrasında ise noktaların yüklerini taşırlar. Araçlar rotalanırken k aracına atanan rotada toplanacak

yükler ile dağıtılacak yükler toplamları araç kapasitesini aşmamalıdırlar. Genellikle taşımacılık da yolcuların bir noktadan alınıp başka noktaya götürülmesi problemi olan talebe bağlı taşıma (dial-a-ride problem) Şekil 2.11(c)'de sunulmuştur. Sisteme müşteri toplama ve bırakma noktasını içeren bir nokta çiftiyle giriş yapar. Bu problemler türü gerçek hayatta genellikle zaman kısıtı da içermekte ve zaman pencereli bir probleme dönüşmektedir. Şekil 2.11(d)'de verilen eş zamanlı topla dağıt problemi klasik topla-dağıt probleminin bir türevidir. Bu problemde her nokta mutlaka hem toplama hem bırakma noktasıdır. Eş zamanlı topla dağıt probleminde araç kapasitelerinin dengelenmesine dikkat edilmesi gerekir. Şekil 2.11(e) ve Şekil 2.11(f) 'de ise hızlı taşıma problemi gösterilmiştir. Bu problemde de eş zamanlı topla dağıt problemi gibi noktalar talep ve bırakma noktasıdır. Ancak, önce dağıtım sonra toplama işlemi yapılır. Böylece araç kapasitelerinin daha kolay yönetilmesi sağlanmış olur.



Şekil 2.11. TDARP türleri

TDARP'nin tamsayılı modeli KKARP modeli üzerinde yeni kısıtların eklenmesi ile elde edilebilir. Amaç fonksiyonu ve her aracın bir rotaya atanması, parçalı dağıtım yapılamaması, araçların kapasite sınırı olması kısıtları SDARP için de geçerlidir. Denklem (2.15) – Denklem (2.21) ile verilen KKARP modeline aşağıda verilen denklemlerle ifade edilen kısıtların eklenmesiyle, eşzamanlı topla-dağıt araç rotalama probleminin modeli oluşturulabilir. [53].

$$\sum_{i \in N} y_{ji} - \sum_{i \in N} y_{ij} = p_j, \qquad \forall j \in N, \quad i \neq j \tag{2.27}$$

$$\sum_{i \in N} z_{ij} - \sum_{i \in N} z_{ji} = d_j, \qquad \forall j \in N, \quad i \neq j \tag{2.28}$$

$$y_{ij} + z_{ij} \leq Q \sum_{k \in V} x_{ijk}, \qquad \forall i \in N, \quad \forall j \in N \tag{2.29}$$

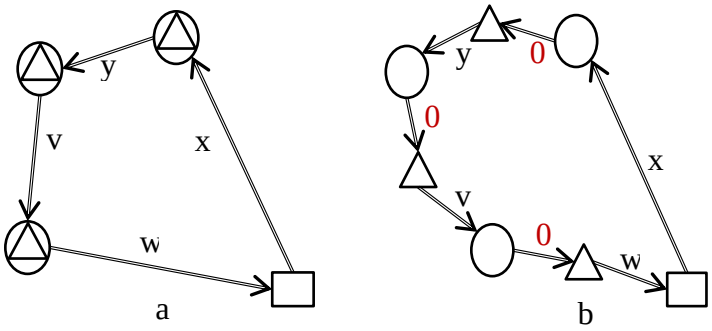
$$y_{ij} \geq 0, \qquad \forall i \in N, \quad \forall j \in N \tag{2.30}$$

$$z_{ij} \geq 0, \qquad \forall i \in N, \quad \forall j \in N \tag{2.31}$$

Denklem (2.27) her j noktasından toplama talebine (p_j) ait akışın dengelenmesi, Denklem (2.28) ise dağıtım talebine (d_j) ait akışın dengelenmesi için kullanılırlar. Aracın j noktasına ulaşana kadar rotada yer alan tüm noktalarda topladığı yükün miktarı y_{ij} değişkeni ile ifade edilir ve j noktasından ayrılan aracın toplama noktalarından aldığı yük ile j noktasına gelen aracın yükü arasındaki fark j noktasının toplama talebine (pick-up) eşittir. Aynı durum Denklem (2.28)'de görüleceği gibi d_j ile ifade edilen dağıtım talebi (demand) için de geçerlidir. Böylece rotada yer alan tüm noktaların hem toplama hem dağıtım talebi olan noktalar olması sağlanmış olur. Denklem (2.29) ise (i,j) nokta çiftleri rotalandığı takdirde j noktasına gelene kadar dağıtım ve toplama taleplerinin kümülatif değerinin araç kapasitesi olan Q değerinden küçük olması gerektiğini ifade eder. Eğer rota üzerinde herhangi bir noktada Q değeri aşılırsa toplama işlemleri yapılamayacaktır.

TDARP problem türleri, eşzamanlı topla-dağıt araç rotalama probleminden (ETDARP) yola çıkılarak elde edilebilir. Her noktanın hem dağıtım hem de toplama talebi olduğu bir problemde, talep ve dağıtım taleplerinin farklı noktalardan geldiğini düşünerek, sanal noktalar oluşturacak olursak, problem eşzamanlı olmaktan çıkıp,

TDARP haline gelecektir [53]. Şekil 2.12'deki örnek bu durumu görsel olarak özetlemektedir.



Şekil 2.12. TDARP türleri arasındaki ilişki

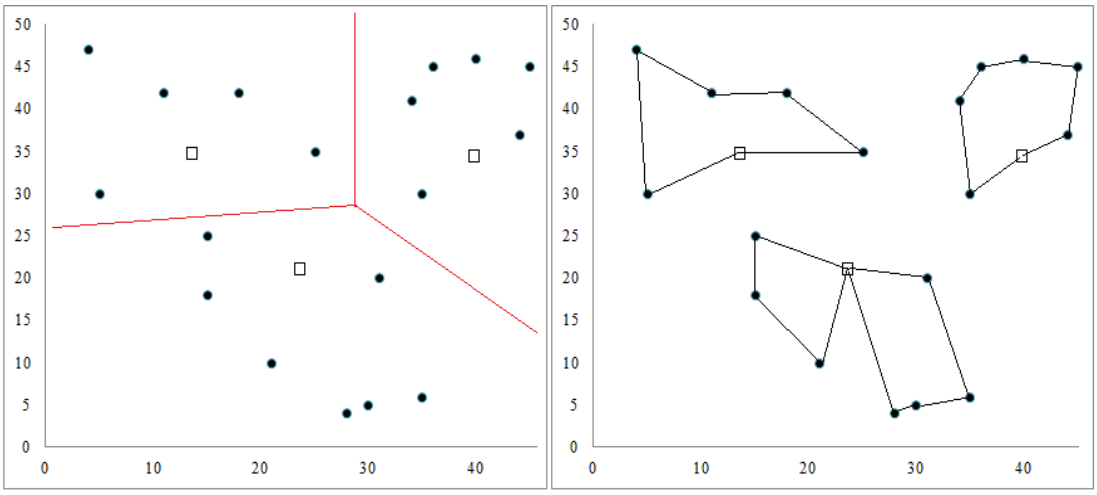
Şekil 2.12a'da her noktada hem toplama hem de dağıtım yapılırken, toplama işi farklı bir noktada yapılıyormuş gibi uzaklığı 0 birim olan sanal noktalar oluşturulmuş ve problem eşzamanlı olmaktan çıkartılmıştır. ETDARP üzerinde bir takım değişiklikler ve varsayımlar yapılarak diğer TDARP türleri elde edilebilmektedir. Bu nedenle problem çözümü için önerilen yöntemler, bazı değişiklikler ve kabullerle tüm problem türlerinde kullanılabilir.

ETDARP problemleri, iki adet araç yardımıyla merkezi bir kütüphane bağlı 22 adet halk kütüphanesine kitapların dağıtılması ve toplanılması probleminin Min tarafından çalışılmasıyla literatürde yer almıştır [54]. Min çalışmada önce kümele, sonra rotala prensibine göre sezgisel bir algoritma önermiştir. NP-Zor bir problem olması nedeniyle yapılan çalışmaların çoğunluğu sezgisel algoritmalarla çözüm elde edilmesi üzerinedir. Dethloff, farklı kriterlere göre ekleme tabanlı bir sezgisel önermiş ve önerisini geliştirdiği 40 ayrı test probleminde test etmiştir [55]. Önce rotaların oluşturulduğu ve sonuçların genetik algoritma tabanlı bir sezgisel ile yerel aramalar yapılarak iyileştirildiği bir yöntem önerilmiş ve Dethloff'un bulduğu sonuçlar geçilmiştir [56]. Komşuluk arama [57], tabu arama [53], sezgisellerine dayanan çözüm yöntemleri de çalışılmıştır. Son yıllarda bilgisayarların hesaplama kabiliyetlerinin artmasıyla kesin çözüm yöntemleri üzerinde de durulmaktadır. Dal-fiyat algoritması ile boyutu 40 noktaya kadar olan problemler için en iyi çözümler bulunmuştur [58]. Son yıllarda önerilen kesin çözüm algoritmaları ile boyutu 100 noktaya kadar olan problemlerden bazıları çözülebilmiştir [59].

2.3.3.4. Çok depolu araç rotalama problemi

Çok depolu araç rotalama problemleri (ÇDARP), taleplerin birden fazla depo ile karşılandığı problemlerdir. Depolar arasında araçların gerçekleştirdiği harekete göre alt türlere ayrılmaktadır. Bir araç harekete başladığı depoya geri dönebilir veya bir depoda başlayan hareket başka bir depoda sonlanabilir. Farklı depolardan hareket edilmesi ve hareketin tamamlanması halinde problem en kısa yolun bulunması problemine dönüşmektedir. Bir döngü söz konusu olmadığı için ARP modellerinde ele alınan alt turların elenmesi, araçların başladıkları depoya dönmesi gibi kısıtlar göz önünde bulundurulmazlar. Araçlar harekete başladıkları depoya geri dönüyorlarsa, noktalar talep ve uzaklık değerleri dikkate alınarak bir depoya atanır.

Bu türdeki problemlerin çözümüne üç aşamalı bir karar verme süreci ile ulaşılmaktadır. İlk aşamada noktalar gruplara ayrılarak depolara olan yakınlıklarına göre atanırlar. İkinci aşamada her depo ve servis vereceği noktalar rotalanır. Son aşamada ise depolardan hareket edecek araçlar çizelgelenir ve sıralanır [60]. Depoların servis vereceği noktaların belirlenmesi ile problem depo sayısı kadar KKARP problemine parçalanmış olacaktır (Şekil 13). Böylece depoların yeterliliği ve araçların kapasitesi göz önünde bulundurularak sanki her parça birbirinden ayrı bir problem gibi ele alınıp çözüme ulaşılabilir.



Şekil 2.13. ÇDARP'nde noktaların gruplandırılması ve oluşan çözümler

Ancak noktalar depolara atanırken deponun belirli bir servis kapasitesi varsa, tüm noktalara hizmet verebileceğinin doğrulanması gerekir. Bir deponun kapasitesinin Q_i, i deposuna atanmış j noktaların w_{ij} ikili değişkeni, j noktasının talebinin ise d_j ile

ifade ediliyor olsun. D, depolar kümesi, N ise noktalar kümesi iken depo kapasitesinin yeterliliği Denklem (2.32) ile kontrol edilir.

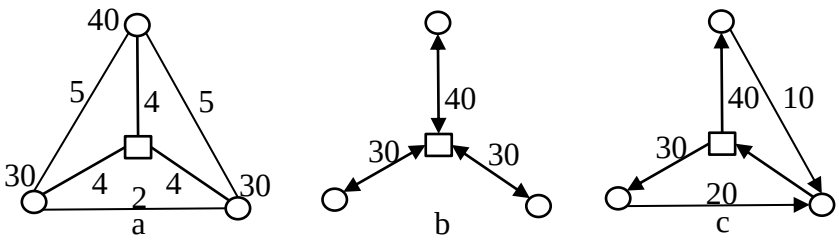
$$d_j \sum_{i \in D} w_{ij} \leq Q_i, \qquad \forall j \in N \tag{2.32}$$

Noktalar, depolara Denklem (2.32)'i sağlayacak şekilde atanırken depolarda yer alan araçların kapasitesinin ve sayısının noktalara hizmet verebilecek düzeyde olması gereklidir. V_i, *i* deposunda yer alan araçlar kümesini, q_k, *k* aracının toplam kapasitesini ifade ederken Denklem (2.33) ile *i* deposuna atanmış noktaların talebini taşıyacak araçların kapasite yeterlilikleri kontrol edilebilir.

$$d_j \sum_{i \in D} w_{ij} \leq \sum_{k \in V_i} q_k, \qquad \forall j \in N \tag{2.33}$$

2.3.3.5. Parçalı dağıtım araç rotalama problemi

Parçalı dağıtım araç rotalama problemi (PDARP) literatüre 1989 yılındaki çalışmalarıyla Dror ve Trudeau tarafından kazandırılmıştır. Araç rotalama problemlerinde genellikle her noktaya sadece bir araç tarafından hizmet verildiği kabul edilir. Ancak parçalı dağıtım araç rotalama problemlerinde, Denklem (2.16) verilen her noktanın bir araç tarafından ziyaret edilebileceği kısıt yer almaz. Böylece bir noktanın talebi kısmi olarak birden fazla araç ile karşılanabilir. PDARP aslında KKARP'nin gevşetilmiş bir türüdür ve bu gevşetme sayesinde bazı durumlarda toplam rota uzunluğunda %50 kadar iyileştirmeler yapılabilmekte ve kullanılan araç sayısı azaltılabilmektedir [61,62]. Şekil 2.14(a)'da verilen örnekte merkezde yer alan depodan üç noktanın talebi karşılanacaktır. Araç kapasitesi 50 birim iken KKARP için çözüm Şekil 2.14(b), PDARP için çözüm Şekil 2.14(c)'de gösterilmiştir [63].



Şekil 2.14. PDARP'nde daha iyi sonuçların elde edilmesi

Noktalara parçalı dağıtım yapılmaması kısıtı varken Şekil 2.14(b)'de görüldüğü gibi depodan ayrılan üç araç bir noktanın talebini karşılayabilir. Toplam kat edilen mesafe ise (4+4)x3 = 24 birim olacaktır. Araçlar birden fazla noktaya hizmet verebildiği takdirde, depodan ayrılan iki adet araç ile Şekil 2.14(c)'deki rotalar üzerinden toplam 4+4+5+4+4+2=23 birim yol kat edilerek talepler karşılanabilir. Böylece araç sayısı azaltıldığı gibi toplam rota uzunluğu da azaltılmış olur.

Ayrıca problemin karmaşıklığı da gevşetmeden etkilenmektedir. KKARP ile PDARP için yapılan karmaşıklık analizinde PDARP'nin daha az karmaşık olduğu ve daha kolay çözülebileceği gösterilmiştir [64]. Dört test problemlerinin üç tanesi araç sayısı sınırı olmadan PDARP kısıtlarıyla polinom (çok terimli) zamanda çözülebilirken, KKARP için sadece bir tanesi çok terimli zamanda çözülebilmektedir. PDARP, KKARP'nin her talep noktasına sadece bir araç ile hizmet verilebilmesi kısıtının gevşetilmiş halidir. Bu nedenle tamsayılı modelinde Denklem (2.16) da verilen kısıt yer almaz. Fakat modelde, kapasitelerin kontrol edilmesi ve her noktaya ait talebinin tamamının karşılanmasını sağlamak amacıyla bazı denklemler tanımlanmalıdır [65].

$$w_{ik} \le d_i x_{ik}, \qquad \forall i \in N, \quad \forall k \in V \tag{2.34}$$

$$\sum_{k \in V} w_{ik} = d_i, \qquad \forall i \in N \tag{2.35}$$

$$\sum_{i \in N} w_{ik} = q_k, \qquad \forall k \in V \tag{2.36}$$

$$x_{ik} \in \{0,1\} \tag{2.37}$$

$$w_{ik} \ge 0 \tag{2.38}$$

x_{ik}, i noktasına k aracı ile yapılan hizmeti belirleyen ikili değişkeni, w_{ik}, i noktasının talebinin k aracı ile karşılanma miktarını, d_i ise i noktasının toplam talebini ifade etmektedir. Her aracın i noktasına taşıdığı yükün i noktasının toplam talebinden düşük olması Denklem (2.34) ile kısıtlandırılmıştır. Denklem (2.35)'de bir noktaya birden fazla araç ile hizmet verilirken araçların bıraktıkları yük miktarlarının noktanın talebine eşit olması, Denklem (2.36)'da ise bir k aracıyla birden fazla noktaya hizmet verildiği halde araç kapasitesinin q_k olması gerektiği verilmiştir.

2.3.3.6. Periyodik araç rotalama problemi

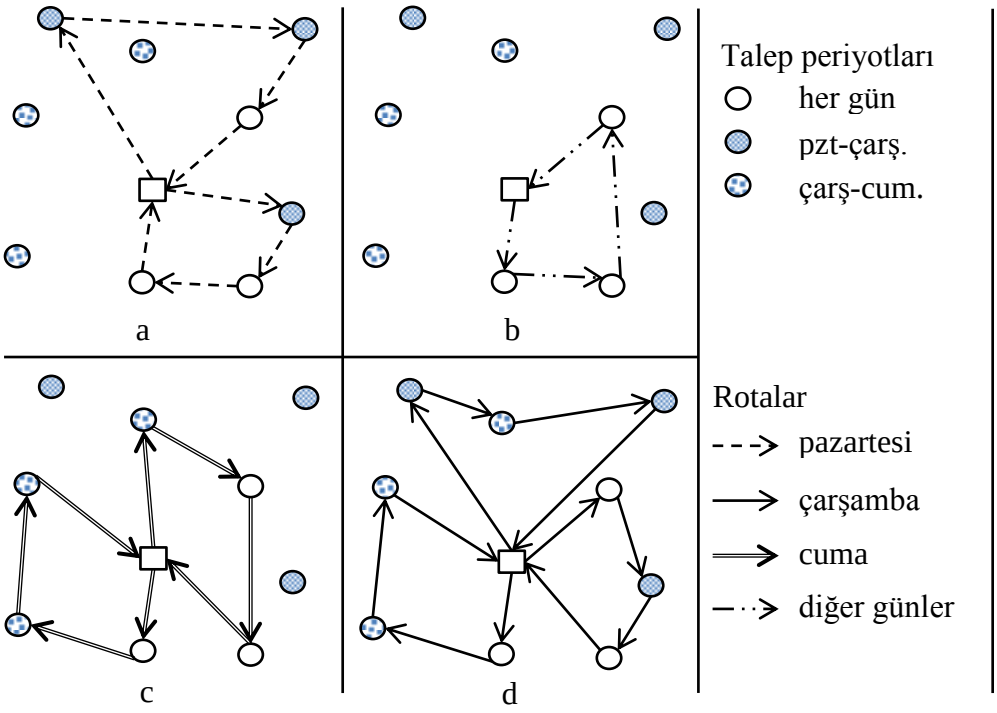
Periyodik araç rotalama problemi (PARP), q_k kapasiteli k araç yardımıyla *n* adet nokta içeren bir dağıtım ağında taleplerin, her noktanın talebinin belirli periyotlarla yapıldığı rotalama problemidir. KKARP'ye ek olarak noktalar periyot olarak ifade edilen zaman dilimi içerisinde belirli miktarda talepte bulunurlar. Talep frekansına göre hangi noktanın ne zaman rotalanacağı belirlenir. Her müşteri planlanan günlerde ziyaret edilmelidir. PARP probleminde amaç aşağıda verilen kısıtlar dâhilinde verilen çizelgeye göre her gün en uygun rotaların tespit edilmesidir [66].

- Her müşterinin talebi zaman ve frekansına göre uygun günler içerisinde karşılanır.
- Her gün için rotalanan araç sayısı toplam araç sayısını aşamaz.
- Araçların kapasitesi, rotalarındaki müşteri taleplerinden fazladır.

PARP'nin tamsayılı modeli, KKARP'ne ait modelin denklemlerine periyot faktörünün eklenmiş halidir. Denklem (2.15)'de verilen amaç fonksiyonunun güncellenmesiyle PARP amaç fonksiyonu elde edilir (Denklem 2.39). Ayrıca *i* noktasında *j* noktasına *k* aracıyla hareket edildiğini temsil eden x_{ijk} ikili değişkenine *t* değeri eklenerek periyotlar kısıtlara da dâhil edilmelidir. x_{ijk} değişkeni denklemlerde x_{ijkt} olarak $\forall t \in T$ olmak üzere güncellenmelidir. Böylece Denklem (2.15) - Denklem (2.21)'de verilen denklemler PARP modeli olarak güncellenmiş olacaktır.

$$enk \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} \sum_{t \in T} c_{ij} x_{ijkt} \tag{2.39}$$

Konunun daha kolay anlaşılabilmesi için Şekil 2.15'de bir örnek bir problem verilmiştir [67]. Müşteriler talepleri doğrultusunda haftanın belirli günlerinde hizmet almaktadırlar. Müşterilerin bir kısmı pazartesi ve çarşamba, bir kısmı çarşamba ve cuma, kalanı ise cuma günleri taleplerinin kendilerine ulaştırılmasını istemektedirler. Depoda yeterince araç olduğu kabulüyle, en az araç kullanarak ve kapasite kısıtlarını da dikkate alarak rotalama yapıldığı takdirde Şekil 2.15(a) pazartesi, Şekil 2.15(d) çarşamba, Şekil 2.15(c) cuma ve son olarak Şekil 2.15(b) ise haftanın diğer günlerinde araçların takip edecekleri rotaları göstermektedir.



Şekil 2.15. PARP örneği (talep periyotları ve rotalar)

3. ROTALAMA PROBLEMLERİNDE KULLANILAN ALGORİTMALAR

Araç rotalama gibi kombinatoryal optimizasyon problemlerinin çözümü için önerilen yöntemler, kesin çözüm yöntemleri, klasik sezgiseller ve metasezgiseller olarak üç gruba ayrılmıştır. Kombinatoryal optimizasyon problemlerinde, nesnelerin bütün olası farklı sıralamalarını içeren kümenin en az bir elemanı en iyi sonucu verir. Bu da en iyi sonuca ulaşmanın, olası tüm farklı sıralamaların belirlenmesi ve değerlendirilmesi ile elde edileceği anlamına gelmektedir. Küçük boyutlu problemler için çözüm bulunabilse de, büyük boyutlu problemler, çözüm kümesinin tamamının taranması en hızlı bilgisayarlarla bile çok uzun zaman alacağından bu yaklaşımla çözülemezler. Kombinatoryal optimizasyon problemlerinin birçoğu NP-Zor problemlerdir. Yani, bu problemlerin çözümü için polinom zamanlı bir algoritma bulunmamaktadır. Gezgin satıcı problemi ve araç rotalama problemi de bu sınıfa dâhildirler. Bu nedenle rotalama problemleri için önerilmiş kesin çözüm yöntemi sayısı fazla değildir.

3.1. Kesin Çözüm Yöntemleri

Kesin çözüm yöntemleri daha çok GSP problemlerinin çözümü için geliştirilmiş olmalarına rağmen ARP'nin, GSP'nin bir türevi olması nedeniyle, ARP'ni çözmek için de kullanılmışlardır. Fakat ARP'nde kullanılan kesin çözüm yöntemleriyle, bilgisayarlarının hesaplama kapasitesine ve geliştirilen algoritmanın çözüm uzayını arama stratejisine bağlı olarak sınırlı büyüklükteki problemler çözülebilmiştir. Araç rotalama problemlerinin kesin çözümlerinin elde edilmesi amacıyla yapılan çalışmalarda çoğunlukla dal sınır, dal kesme ve küme kapsama yöntemleri ve türevleri kullanılmıştır [15, 32].

3.1.1. Dal sınır yöntemi

Dal sınır algoritması tamsayılı problemlerin çözümünde kullanılan yöntemlerden birisidir. Problemler, kısıtlar dikkate alınarak çözüm uzayı içerisinde belirlenmiş alt ve üst sınırlar arasında kalan çözümlerin dallandırılıp daha iyi sonuçların

aranmasıyla çözülmeye çalışılır. Böylece optimum sonuç, sistematik bir şekilde çözüm uzayının içerisinde aramış olur. Dal sınır yönteminde, alt dalların oluşması modelde yer alan kısıtların gevşetilmesi (esnetilmesi) ile meydana gelir. Yani, dallanma aşamasında uygun olmayan çözümler kabul edilir. Mesela tamsayı olarak tanımlanmış bir değişken ondalık değer alabilir. Kısıtları gevşetme işlemi, daha iyi çözümlere ulaşılmasını sağladığı için gereklidir. Eğer elde edilen yeni çözüm daha iyi ise yeni sınır değeri olarak atanır ve güncellenmiş sınır değeri dikkate alınarak tekrar dallanma adımına geçilir. Bu döngü, dallarda uygun bir çözüm oluşmayana kadar devam ettirilir.

Kısıtların gevşetilmesi için kullanılan yöntemlerden biriside lagrange gevşetmesidir. Daha karmaşık ve ileri seviye olan bu dallandırma yöntemiyle daha büyük boyutlu problemler çözülebilmiştir [32, 50, 68].

3.1.2. Dal kesme yöntemi

Dal kesme yöntemi, kesme düzlemi yöntemi ile dal sınır yöntemine birleşiminden ortaya çıkmış bir yöntemdir. Ana tema uygun çözümlerin bulunduğu sonuç kümesinin kesme düzlemi ile daha verimli bir alanının aranmasıdır. Kesme işleminin eklenmesiyle daha iyi sınır değerleri belirlenmekte ve bazen dallanma olmadan alt problemler çözülebilmektedir [69]. Yani, kesme düzlemi eklenmesiyle problemin çözüm uzayında uygun (olurlu) sonuçlar vermeyen çözümler çözüm kümesinden çıkartılmış olmaktadır.

Sadece kesme düzleminin kullanılmasıyla bir problemin verimli olarak çözülebilmesi mümkün değildir, çözümün iyileştirilebilmesi için dallandırma yapmak gereklidir. Kesme düzlemi kısıtı eklendikten sonra dallanma işlemi yapıldığı takdirde en iyi sonucun bulunduğu arama uzayı daraltılmakta ve algoritma çözüme daha kısa sürede ulaşabilmektedir. [70].

Dal kesme yönteminde Gomory tarafından önerilen kesme düzlemi kullanıldığı gibi Chv´atal'ın yuvarlama prosedürü, Schrijver'in yuvarlama prosedürü ve Çokyüzlü kesme düzlemleri de kullanılmaktadırlar.

3.2. Klasik Sezgisel Algoritmalar

Rotalama problemlerinde kesin çözüm yöntemlerinin küçük boyutlu problemlerde uygulanabiliyor olması nedeniyle sezgisel algoritmalar üzerinde yapılan çalışma sayısı oldukça fazladır. Sezgiseller, genel olarak klasik sezgiseller ve metasezgiseller olmak üzere iki kategoriye ayrılmışlardır. 1990'lı yıllara kadar klasik sezgiseller üzerinde yapılmış birçok çalışma mevcut iken daha sonra doğadan esinlenerek geliştirilen metasezgisel algoritmalar rotalama problemlerinde kullanılmışlardır.

Klasik sezgiseller kendi aralarında tur kurucu sezgiseller, iki aşamalı sezgiseller ve tur geliştirici sezgiseller olarak üç gruba ayrılırlar [15]. Tur kurucu sezgiseller, kısıtları göz önünde bulundurarak uygun (olurlu) bir çözüm üretirler. İki aşamalı sezgisellerde genellikle noktalar araç kapasitelerine göre gruplara ayrıldıktan sonra her bir araç için kat edilecek en kısa yolu veren rota hesaplanmaya çalışılır. Ayrıca önce rotalama işleminin yapıldığı, ardından rotanın gruplandırılarak araçlara atandığı algoritmalarda geliştirilmiştir. Tur geliştirici sezgiseller ise, rota içerisinde veya rotalar arasında noktaların değiştirilmesiyle mevcut rotaların geliştirilmeye çalışıldığı algoritmalardır. Tur geliştirme algoritmaları, rotaları geliştirebilmesi için başlangıç rotalarının verilmiş olması gerekir. Klasik sezgisellerin türleri ve geliştirilen yöntemler Tablo 3.1'de verilmiştir.

Tablo 3.1. Klasik sezgisel türleri ve yöntemler

Tur kurucu sezgiseller	1. Tasarruf algoritması 2. Geliştirilmiş tasarruf algoritması 3. Sıralı ekleme algoritması
İki aşamalı sezgiseller	1. Önce grupla sonra rotala • Süpürme algoritması • Fisher ve Jaikumar algoritması • Bramel ve Simch-Levi algoritması • Petal (taç yaprağı) algoritması 2. Önce rotala sonra grupla
Tur geliştirici sezgiseller	1. Tek rota geliştirme algoritmaları 2. Çok rota geliştirme algoritmaları

Klasik algoritmalar üzerine literatürde yapılmış birçok çalışma ve haklarında detaylı bilginin elde edilebileceği birçok kaynak bulunmaktadır [15, 36, 37, 71].

3.2.1. Tur kurucu sezgiseller

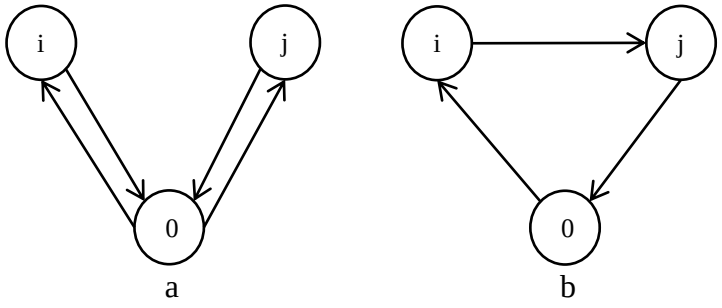
Tur kurucu sezgiseller, olurlu olmayan bir çözüm kümesi ile çözüme başlayan, kapasite kısıtları ve noktalar arasındaki uzaklıkları kontrol ederek olurlu bir çözüme ulaşmaya çalışan algoritmalardır. En çok bilineni Clarke ve Wright tarafından geliştirilen tasarruf algoritmasıdır [34].

3.2.1.1. Tasarruf algoritması

Tasarruf algoritmasının ilk adımında, her noktanın bir araç ile depodan ziyaret edildiği kabul edilir. Yani bir araç, sadece bir noktaya hizmet verip geri depoya dönmektedir. Bu kabule göre müşteri sayısı kadar araca ihtiyaç vardır ve hem gereğinden fazla araç kullanılması hem de kat edilen mesafenin fazla olması nedeniyle, çözüm olurlu bir çözüm olduğu halde iyi bir çözüm değildir. Araç kapasitelerinin verimli kullanmak adına noktalar birleştirilmelidir. Birleştirme işlemi için bağlantı yapabilecek her *(i,j)* nokta çifti arasında tasarruf adı veriler bir değer hesaplanır. Bu değerin en yüksek olduğu nokta çiftler arasında bağlantı kurulması gereken noktalar olarak kabul edilir ve kapasite kısıtı aşılmadığı sürece en büyük tasarruftan en küçüğe doğru alt turlar oluşmayacak şekilde noktalar araçlara atanırlar. Tasarruf algoritmasının denklemi aşağıda verilmiştir. S_{ij}, *(i,j)* çifti arasındaki tasarruf değerini, c_{ij}, *(i,j)* çifti arasındaki mesafeyi, c_{0i} ve c_{0j} ise depodan *i* ve *j* noktalarına olan uzaklığı temsil eder.

$$S_{ij} = c_{0i} + c_{0j} - c_{ij}, \tag{3.1}$$

Şekil 3.1'de verilen örnekte, ilk önce depodan i ve j noktalarına birer adet araç ile hizmet verilmişken, ikinci resimde i ve j noktaları beraber rotalanmıştır.



Şekil 3.1. Tasarruf algoritması örneği

Şekil 3.1 (a)'da rota uzunluğu $R_a = c_{0i} + c_{0j} + c_{i0} + c_{j0}$ iken, Şekil 3.1 (b) 'de ise $R_b = c_{i0} + c_{j0} + c_{ij}$ olacaktır. Tasarruf değeri iki rota arasındaki fark ($S_{ij} = R_a - R_b$) olarak tanımlandığına göre $S_{ij} = c_{0i} + c_{0j} + c_{i0} + c_{j0} - (c_{i0} + c_{j0} + c_{ij})$ olarak hesaplanır. Bu işlemin sonunda Denklem 3.1 de verilen eşitlik elde edilir. Problemde bulunan noktaların hepsinin oluşturacağı ikili çiftler için tasarruf değeri hesaplanır. Sonraki adımda en yüksek tasarruf değerinden başlanarak noktalara araç kapasitesini aşmadıkları sürece birleştirilerek bir tur elde edilir. Turların kurulması iki farklı yöntemle yapılır. Sıralı tasarruf algoritmasında her adımda sadece bir tur kurulur ve ilk tur tamamlanınca ikinci tur kurulmaya başlanır. Paralel tasarruf algoritmasında ise *n* adet tur aynı anda kurulabilir. İlk tur için *(i,j)* çifti rotalanamıyorsa, çift farklı bir tura atanır. Tablo 3.2'de tasarruf algoritmasının adımları verilmiştir.

Tablo 3.2. Tasarruf algoritmasının adımları

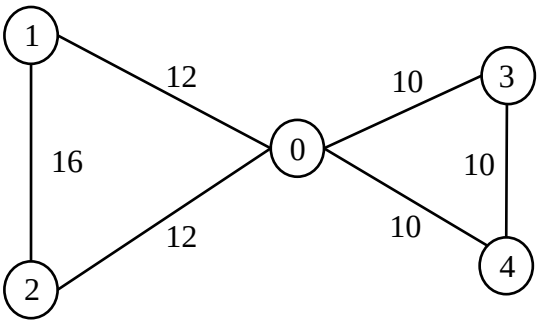
Adım 1	i = 1 den i = N oluncaya kadar (N = toplam nokta sayısı)
Adım 1.1	j = 1 den j = N oluncaya kadar ($j > i$) iken
Adım 1.1.1	Denklem (3.1) kullanarak S_{ij} değerlerini hesapla
Adım 2	S_{ij} değerlerini D[x] boyutlu mesafe matrisine ata ve büyükten küçüğe sırala. (x değeri N'nin ikili kombinasyonuna eşittir)
	Sıralı tasarruf ise;
Adım 3	i = 1 den i = v oluncaya kadar (v = toplam araç sayısı)
Adım 3.1	j = 0 den j = x-1 oluncaya kadar
Adım 3.1.1	$q_i \leq$ D[j] olduğu müddetçe (q_i , i aracının kapasite değeri)
Adım 3.1.2	Eğer $R_i \neq \varnothing$ ise Adım 3.1.4'e git. (R_i boş kümeyse)
Adım 3.1.3	j. sıradaki (i,j) çifti $i \in R_i$, $j \in R_i$ için (i ve j noktalarından biri R_i kümesinin elemanı ise ve R_i kümesi boş değilse)
Adım 3.1.4	R_i rotasına (i,j) çiftini ekle
	Paralel tasarruf ise;
Adım 3	j = 0 den j = x-1 oluncaya kadar
Adım 3.1	i = 1 den i = v oluncaya kadar (v = toplam araç sayısı)
Adım 3.1.1	Eğer $R_i \neq \varnothing$. Adım 3.1.5'e git. (R_i boş kümeyse)
Adım 3.1.2	$q_i \leq$ D[j] olduğu müddetçe (q_i , i aracının kapasite değeri)
Adım 3.1.3	j. sıradaki *(i,j)* çifti $i \in R_i$, $j \in R_i$ için (i ve j noktalarından biri R_i kümesinin elemanı ise) Adım 3.1.5'e git
Adım 3.1.4	j. sıradaki *(i,j)* çifti $i \notin R_i$, $j \notin R_i$ için (i ve j noktaları R_i kümesinin elemanı değilse) Adım 3.1'e git
Adım 3.1.5	R_i rotasına *(i,j)* çiftini ekle

3.2.1.2. Geliştirilmiş tasarruf algoritması

Tasarruf algoritması, kolay ve anlaşılır bir algoritma olmasının yanında bazı durumlarda iyi sonuçlar üretemez. Deneyimler çoğu zaman algoritmanın yeterli çözümler ürettiğini ama bazı özel durumlarda çözümlerin çok kötü olabildiğini göstermiştir [15]. Bu nedenle Yellow 1970 yılında yaptığı çalışmasında tasarruf denkleminde yaptığı bir değişiklikle daha iyi çözümlerin elde edilebileceğini göstermiştir [35]. Geliştirilen tasarruf algoritması Denklem (3.2)'de verilmiştir.

$$S_{ij} = c_{0i} + c_{0j} - \lambda c_{ij}, \tag{3.2}$$

λ, *(i,j)* nokta çifti arasındaki mesafenin tasarruf değerini ne kadar etkileyeceğini belirleyen [0,1] arasında değer alan bir parametredir. λ=1 olursa, problem klasik tasarruf algoritması haline dönüşmektedir. Parametre üzerinde yapılan değişikliklerle farklı rotaların elde edilmesi mümkün olmaktadır (Şekil 3.2).



Şekil 3.2. Geliştirilmiş tasarruf algoritmasında λ'nın etkisi

Şekil 3.2'de verilen örnek problemde tasarruf değerlerini λ=1 ve λ=0,5 alınırsa aşağıda verilen tasarruf değerleri bulunur.

λ= 1 için, $S_{12} = 12 + 12 - 1 \times 16 = 8$ $S_{34} = 10 + 10 - 1 \times 10 = 10$

λ = 0,5 için, $S_{12} = 12 + 12 - 0{,}5 \times 16 = 16$ $S_{34} = 10 + 10 - 0{,}5 \times 10 = 15$

λ=1 olarak alındığında depoya yakın yakın olan 3-4 noktaları önce rotalanırken, λ=0,5 olarak alınırsa depoya daha uzak olan 1-2 noktaları önce rotalanmaktadır. Böylece tasarruf değeri nedeniyle depoya uzak olan noktaların daha erken rotaya dâhil edilmesine imkân tanınmış olunmaktadır ve λ değerini değiştirerek farklı rotalar elde etmek mümkündür.

3.2.1.3. Sıralı ekleme algoritması

Ekleme Sezgisel Algoritması, araç sayısının değişken olarak alındığı fakat yükleme kapasitesi bilinen problemler için geliştirilmiş bir yöntemdir. Burada süreç genel olarak başlangıç rotalarının oluşturulması ve bu rotalarla ilgili talep noktalarının rotaya en az maliyet artışına sebebiyet verecek şekilde eklenmesine dayanmaktadır. Bu konuyla ilgili iki çalışma bulunmaktadır. 1976'da Mole ve Jameson tarafından geliştirilen modelde, bir iterasyonda sadece bir rota ele alınmaktadır. Christofides, Mingozzi ve Toth ise 1979'da bu yöntemi sıralı ve paralel rota oluşturan yordamlar kullanarak uygulamışlardır [20].

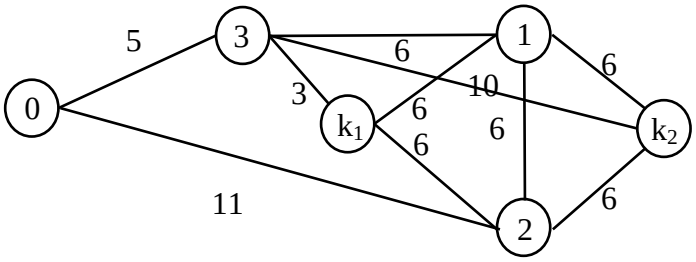
Mole ve Jameson, bir rotanın alt yapısına kadar ulaşmak için λ ve μ parametrelerini içeren bir algoritma önermişlerdir [15].

$$\alpha(i,k,j) = (c_{ik} + c_{jk} - \lambda c_{ij}), \qquad \forall i,k,j \in N \tag{3.3}$$

$$\beta(k) = [\mu c_{0k} - \min(\alpha(i,k,j))], \qquad \forall i,k,j \in N \tag{3.4}$$

Denklem (3.3) ile N noktalar kümesinden seçilen ve henüz rotalanmamış bir nokta, mevcut rotadaki *(i,j)* nokta çiftlerinde hangisinin arasına eklenirse toplam rota uzunluğunun en az artacağı hesaplanmaktadır. Böylece yeni nokta, öyle bir *(i,j)* çifti arasına yerleştirilerek; sıralama *(i,k,j)* şeklinde güncellenmektedir ki yeni nokta rota uzunluğunu minimum artırmış olsun. Akabinde Denklem (3.3), Denklem (3.4) de yerine yazılarak eklenen noktanın depoya olan uzaklığı kontrol edilmekte ve μ parametresinin aldığı değere göre k noktasının depoya olan uzaklığı da hesaba katılmaktadır. Bu denklemlere göre kapasite kısıtları göz önünde bulundurularak rotalar oluşturulduktan sonra rota iyileştirme sezgiselleri ile rota içerisinde arama yapılarak daha bir dizilimin mümkün olup olmadığı araştırılır. Bu adımlar tüm noktalar bir rotaya atanana kadar devam eder.

λ ve μ parametreleri ile noktalar rotalarda farklı bölgelere yerleştirilerek farklı çözümler elde edilebilir. λ parametresi noktanın rotada nereye yerleştirileceğinde, μ parametresi ise depoya olan uzaklığa bağlı olarak hangi noktanın rotaya dâhil edileceğinin karar verilmesinde rol oynar. Şekil 3.3'de verilen örnek ile yöntem daha net olarak anlaşılmaktadır.



Şekil 3.3. Sıralı ekleme algoritması örneği

Mevcut rota (0,3,1,2,0) noktaları olarak verilmiş olsun ve kapasite sınırı nedeniyle k_1 veya k_2 noktalarından birini rotaya ekleyebileceğimizi varsayalım. k_1 ve k_2 noktalarını (3,1) veya (1,2) noktaları arasına eklenebilmektedir ve Denklem (3.3) kullanılarak yeni noktaların hangi nokta çiftinin arasına ekleneceği belirlenmelidir. $(\lambda = 1,\ c_{ok_1} = 8\ ve\ c_{0k_2} = 12)$

$\alpha(3,k_1,1) = (3+6-3) = 3$ $\qquad$ $\alpha(1,k_1,2) = (6+6-6) = 6$

$\alpha(3,k_2,1) = (10+6-6) = 10$ $\qquad$ $\alpha(1,k_2,2) = (6+6-6) = 6$

Hesaplamalara göre k_1 (3,1), k_2 ise (1,2) noktaları arasına eklenmelidir. Çünkü minimum α değerleri k_1 ve k_2 bu noktalar arasına eklendiğinde hesaplanmıştır. Şimdi Denklem (3.4)'ü kullanarak k_1 veya k_2 noktasından hangisini rotaya ekleyeceğimizi belirleyelim.

$\mu = 1$ ise $\qquad \beta(k) = \max[(1\times 8 - \min(3,6)),(1\times 12 - \min(10,6))]$

$\mu = 0,5$ ise $\qquad \beta(k) = \max[(0,5\times 8 - \min(3,6)),(0,5\times 12 - \min(10,6))]$

$\beta(k) = \max[5,6]$ değerini elde ederiz. Böylece rotaya k_2 noktasının ekleneceği belirlenmiş olacaktır. Rota (0,3,1,k_2,2,0) olarak değiştirilir. Fakat μ = 0,5 olarak alınırsa $\beta(k) = \max[1,0]$ olacak ve k_1 noktası rotaya eklenecektir. μ parametresi 0 değerine yaklaştıkça depoya daha yakın noktaların rotaya eklenmesini sağlayacaktır. Hem λ=0 ve hem de μ=0 değerini alırsa, depoya en yakın ve *(i,j)* arasındaki uzaklığa bakılmaksızın *(i,k)* ve *(k,j)* arası uzaklığın toplam değerinin en az olduğu noktalar rotaya eklenirler. Parametrelerde yapılacak değişikliklerle istenilen şartlara uygun rotalar oluşturulabilmek mümkündür. Ancak parametre değerlerini 0 olarak belirlemek depoya uzak olan noktaları dışlayacağı için bir risk oluşturur.

3.2.2. İki aşamalı sezgiseller

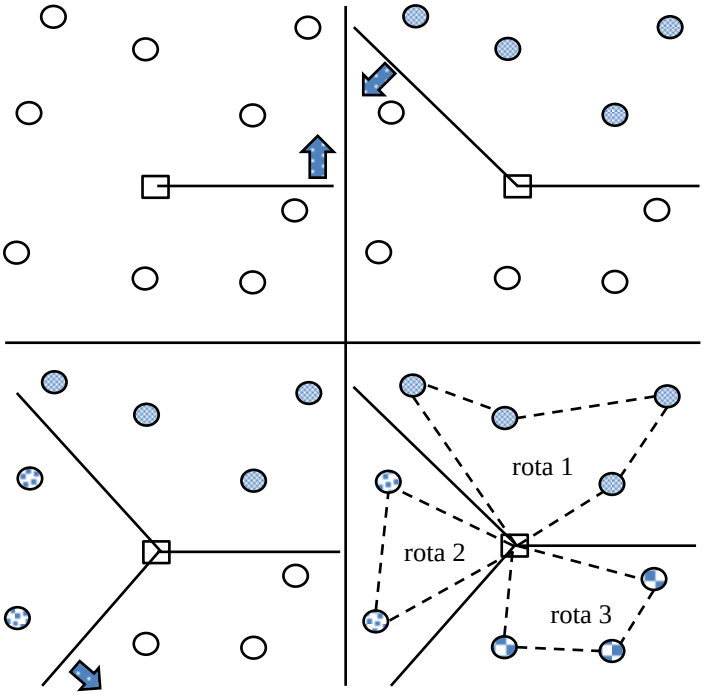
Bu algoritmalar önce grupla sonra rotala ve önce rotala sonra grupla olarak iki kategoride yer alırlar. Birinci kategoride noktalar araç sayısına ve kapasite kısıtlarına göre gruplanır ve her bir grup içerisinde kalan noktalar tek araçlı gezgin satıcı problemi gibi çözülür. Buradaki mantık her bir alt rotanın uzunluğu bağımsız olarak en küçüklenirse amaç fonksiyonu da en küçüklenmiş olacaktır. İkinci kategoride geliştirilmiş algoritma sayısı oldukça azdır. Rotala-grupla algoritmalarında, tüm noktalar tek bir araç ile hizmet verilecekmiş gibi rotalanır ve ardından kapasite kısıtları göz önünde bulundurularak rotalar parçalara ayrılıp araçlara atanırlar. Bu çalışmada yaygın olarak kullanılmış iki aşamalı sezgisellere yer verilmiştir.

3.2.2.1. Süpürme algoritması

Süpürme algoritması, birbirine yakın olan müşterilerin aynı grup içerisinde gruplanarak aynı araç ile servis edilmesinin sağlandığı bir algoritmadır. Algoritma düzlemsel problemler üzerinde kullanılabilmektedir. Araç kapasitesinin aşılmadığı olurlu çözümler depodan belli bir açıyla yapılan tarama ile elde edilir. Ardından noktalara depoda eklenerek rotalama işlemi yapılır. Şekil 3.4'de süpürme algoritmasının adımlarının gösterildiği bir örnek verilmiştir. Algoritma adımları aşağıdaki gibidir [72].

Tablo 3.3. Süpürme algoritmasının adımları

Adım 1	Deponun yeri polar koordinat düzleminin merkezi olarak belirlenir.
Adım 2	Her noktanın depoya göre polar koordinatlar hesaplanır. (Θ,r) Θ açıyı, r ise depoya olan dik uzaklık değerini belirmektedir
Adım 3	Bir yarıçap r değeri tanımlanır ve 0^o açı ile başlanarak saat tersinin yönüne düzlem taranmaya başlanır.
Adım 4	Taranan noktalar kapasite kısıtlarını aşılmadığı takdirde mevcut kümeye dâhil edilir
Adım 5	Eğer karşılaşılan noktanın talebi araç kapasitesini aşıyorsa tarama durdurulur ve saat yönüne tarama yapılmaya başlanır. Saat yönünde de kapasite kısıt aşılırsa küme kapatılır ve yeni küme için tarama yapılır.
Adım 6	Kapasite kısıtı aşılırsa, yeni bir küme oluşturulur ve tarama son noktadan ileriye devam ettirilir.
Adım 7	Adım 4 - Adım 6 tüm noktalar bir kümeye dâhil edilinceye kadar tekrarlanır.
Adım 8	Tüm noktalar bir kümeye dâhil edildikten sonra, kümeler kendi içerisinde GSP gibi çözülür ve uygun bir yöntemle iyileştirilir.



Şekil 3.4. Süpürme algoritması örneği

Süpürme algoritması anlaşılması, uygulanması ve programlama dilleri ile kodlanması kolay bir algoritmadır. Ancak tasarruf algoritması kadar iyi sonuçlar elde edilmesi hayli güçtür. Nokta taramanın başladığı nokta, taranan r yarıçap değeri gibi faktörler sonucu etkilerken, noktalar arası uzaklıklar dikkate alınmamaktadır [73]. Ayrıca, algoritma esnek değildir, her kısıt ifade edilemez ve bu nedenle her ARP türünde kullanılamaz. Örneğin; Depo sayısı birden fazla olan ÇDARP'nde hangi deponun polar düzlemin merkezi olacağını belirlemek bir dizi varsayım yapılmasını gerektirecektir.

3.2.2.2. Fisher ve Jaikumar algoritması

Süpürme algoritması gibi önce grupla sonra rotala mantığına sahip bir diğer algoritma ise Fisher ve Jaikumar tarafından önerilmiştir. Süpürme algoritmasından ayrılan yanı, kümele adımında polar koordinat düzlemini kullanılması yerine bazı matematiksel işlemlere yer verilmiştir. Tüm noktalar kapasite kısıtlarını ve araç sayısını göz önünde bulundurularak depo noktası merkez olacak şekilde dilimlere ayrılırlar. Her dilimde ortada yer alan nokta belirlenir ve talep noktalarının bu merkez noktalarına olan uzaklık maliyetlerine göre atama yapılır. Atama işlemi için genelleştirilmiş atama yöntemi kullanılmaktadır. Genelleştirilmiş atama yöntemi Sırt

Çantası Probleminde (Knapsack Problem) adı geçen, çantaları en kazançlı şekilde nasıl doldurmamız gerektiği sorusuna cevap arayan bir kombinatoryal optimizasyon problemidir. Problemin modeli aşağıda verilmiştir. *i* çantaları *j* paketleri temsil ederken p_{ij}, *i* çantasına *j* paketinin ne kadar kazanç sağladığını belirtmektedir. $r_{ij,}$ *i* çantasına yerleştirilen *j* paketinin ağırlığını veya boyutunu temsil eder. c_i değeri ise *i* çantasının kapasite sınırıdır.

$$enb\sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij} \tag{3.5}$$

$$\sum_{j\in J} r_{ij}x_{ij} \le c_i, \qquad \forall i \in I \tag{3.6}$$

$$\sum_{i\in I} x_{ij} = 1, \qquad \forall j \in J \tag{3.7}$$

$$x_{ij} = [0,1], \qquad \forall i \in I, \quad \forall j \in J \tag{3.8}$$

Bu atama yöntemi x adet işin y kadar kişiye atanması problemine benzer birçok problemin çözümünde kullanılmaktadır. Fisher ve Jaikumar algoritmasında ise elde edilen her dilimin merkez noktası ile atanacak nokta arasındaki ve depoya olan mesafeleri hesaplanarak bir maliyet değeri belirlenir. Noktaların talepleri r_{ij} ile her aracın kapasitesi ise sırt çantası kapasitesinin ifade edildiği c_i değişkeni ile gösterilerek problem çözülür. Algoritmanın adımları Tablo 3.4'de verilmiştir.

Tablo 3.4. Fisher ve Jaikumar algoritmasının adımları

Adım 1	Deponun yeri merkez olacak şekilde N araç sayısı kadar kapasite kısıtını aşmamak kaydıyla dilimlere böl.
Adım 2	Her dilimin ortasındaki noktayı belirle ve merkez noktası olarak ata.
Adım 3	Her noktanın maliyetini Denklem (3.9) ile hesapla.
Adım 4	Her noktanın talebini ve araç kapasitesini dikkate alarak Denklem (3.5) - Denklem (3.8) verilen genelleştirilmiş atama problemini çöz.
Adım 5	Oluşturulan her grup için rotaları belirle.
Adım 6	Rotaları bir iyileştirme yöntemi kullanarak geliştir.

$$p_{ik} = d_{0i} + d_{ij_k} + d_{j_k 0} \tag{3.9}$$

p_{ik}, *i* müşterisinin k grubuna ait olması halinde maliyeti, d_{0i}, deponun *i* noktasına uzaklığını, d_{ij_k} ise i noktasının k grubunun merkezine (j_k) olan uzaklığını ifade eder.

3.2.2.3. Bramel ve Simch-Levi algoritması

Fisher ve Jaikumar algoritmasına benzer bir yöntem olan Bramel ve Simch-Levi algoritmasının tek farkı kümeleme işlemindeki farklılıktır. Kümeler, talep noktalarının birbirlerine olan yakınlığına göre belirlenmektedir. Merkez noktaları, kapasite kısıtlı lokasyon belirleme problemi çözümünde kullanılan yaklaşımla elde edilmektedir. Talep noktalarının N kümesinin bir elemanı olduğunu ve her noktadan gelen talebi karşılamak için Q kapasiteli t adet işletme kuracağımızı düşünürsek; işletmeleri talepler karşılanırken müşteriye olan uzaklığın minimize edildiği lokasyonlara kurmak gerekir. Algoritmanın adımları aşağıda açıklanmıştır [74].

Adım 1. İlk aşamada m adet küme merkezi belirlenir. Küme merkezleri talep noktalarının bir alt kümesidir. Daha sonra her bir talep noktasının küme merkezlerine olan uzaklıkları (maliyetleri) hesaplanır. Bu maliyet c_{ij} ile gösterilir. i talep noktasını j ise küme merkezini temsil eder. c_{ij} maliyetini hesaplamak için kullanılan denklemler Denklem (3.10) ve Denklem (3.11)'de verilmiştir.

Adım 2. Adım 1 'de elde edilen c_{ij} değerleri kapasite kısıtlı lokasyon belirleme algoritması ile işlenir ve her i noktasının hangi j kümesine atanacağı belirlenir. Bu atama esnasında kapasite kısıtları ve c_{ij} maliyeti birlikte değerlendirilir. Böylece her kümeye ait noktaların talepleri Q araç kapasitesinin altında bir değer olur.

Adım 3. Kümeler belirlendikten sonra lokasyon belirleme problemi, rotalama problemine dönüştürülür. Kümelerin merkez noktaları talep noktasına çevrilir ve depo dikkate alınarak problem tek araçlı gezgin satıcı problemi gibi çözülür. Her kümenin merkez noktasının talep noktasına dönüştürülmesinin nedeni bu noktanın da aslında talep noktası olmasından kaynaklanır. Adım 1'de belirlenen merkez noktalarını aslında talep noktalarının bir alt kümesidirler.

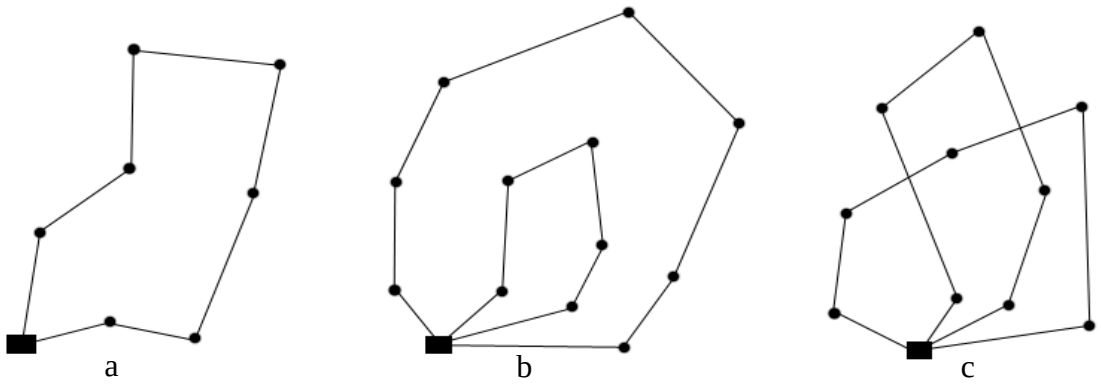
$$c_{ij} = \min\{2d_{ij_l}\}, \qquad \forall i \in N, \quad \forall j_l \in p \tag{3.10}$$

$$c_{ij} = \min\{d_{ij_l} + d_{ij_{l+1}} - d_{j_l j_{l+1}}\}, \qquad \forall i \in N, \quad \forall j_l \in p \tag{3.11}$$

p değeri, merkez noktalarının GSP gibi düşünülerek ardı sıra rotalanması halinde oluşan kümenin boyutudur. ($T_j = \{x_{j0}, x_{j1}, x_{j2}, \ldots, x_{jp}, x_{j0}\}$)

3.2.2.4. Petal algoritması

Süpürme algoritmasından esinlenilerek geliştirilmiş olan petal algoritması, petal (taç yaprağı) adı verilen ve aynı talep noktasının birden fazla rotada yer alabildiği bir yöntemdir [72]. Foster ve Ryan tarafından 1976 yılında ortaya atılmış olmakla birlikte Ryan, Hjorring and Glover [75] oluşan rotaları küme kapsama yöntemi ile çözerek daha iyi sonuçlar elde etmişlerdir. Renaud ve ark. çalışmalarında mevcut yöntemi 1-petal sezgiseli olarak adlandırıp, 2-petal sezgiseli olarak yeni bir yöntem önermişledir [76]. Böylece iç içe ve kesişen rotaların oluşması mümkün hale gelmiştir. 2-petal sezgiseli diğer klasik sezgisel yöntemlerle kıyaslanmış ve daha iyi sonuçların elde edilebildiği görülmüştür. Şekil 3.5'de petal sezgisellerine ait görsellere yer verilmiştir.

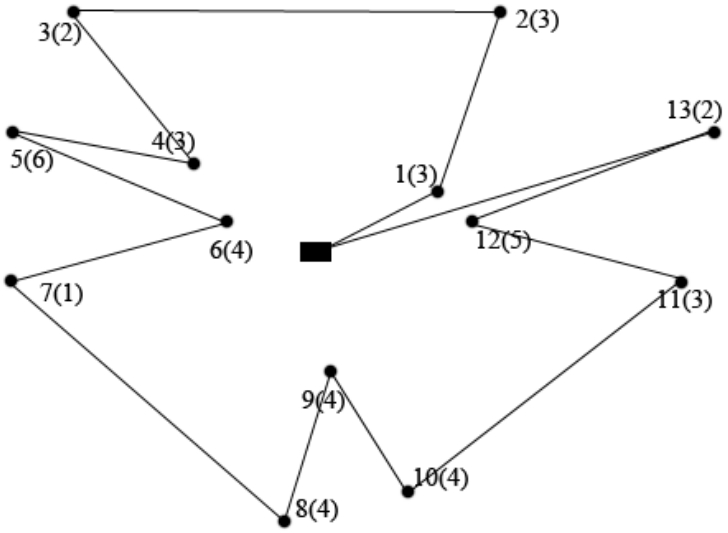


Şekil 3.5. 1-petal ve 2-petal sezgiselinde oluşan rotalar [76]

Şekil 3.5 (a)'da, 1-petal sezgiseli ile, Şekil 3.5 (b) ve (c)'de ise 2-petal sezgiseli ile elde edilen iç içe ve kesişen rotalar gösterilmektedir. Petal algoritmasının adımları ise şöyledir:

- Adım 1. Tüm noktalar GSP problemi olarak tek rotada rotalanır.
- Adım 2. İlk noktadan başlanılarak, Q kapasite sınırı aşılmadığı takdirde alt rotalar tespit edilir.
- Adım 3. Tüm noktalar olası alt rotalara ayrıldıktan sonra ilk noktadan başlanılarak en çok nokta içeren araç rotası seçilir ve çizelgelenir. Fakat bu adımda her noktanın sadece bir rotada olması kısıtına dikkat edilir.
- Adım 4. Rotalanmayan nokta kalmayana kadar Adım 3 tekrarlanır.
- Adım 5. Rotalar iyileştirme sezgiselleri ile iyileştirilir.

Petal sezgiselinin daha net anlaşılabilmesi için her bir aracın kapasitesinin 10 olduğu, bir depo ve 13 adet talep noktasından oluşan örnek problem üzerinde duralım. Şekil 3.6'da problemde verilen noktalar sırasıyla birleştirilmiş ve gezgin satıcı turu üretilmiştir. Gezgin satıcı turunun elde edilmesinden sonra ilk noktadan başlanarak araç kapasitesi aşılmadan olası alt rotalar aranmıştır. İlk nokta için; (D-1-D, D-1-2-D ve D-1-2-3-D), ikinci nokta için (D-2-D, D-2-3-D ve D-2-3-4-D), üçüncü nokta için; (D-3-D, D-3-4-D) elde edilir.

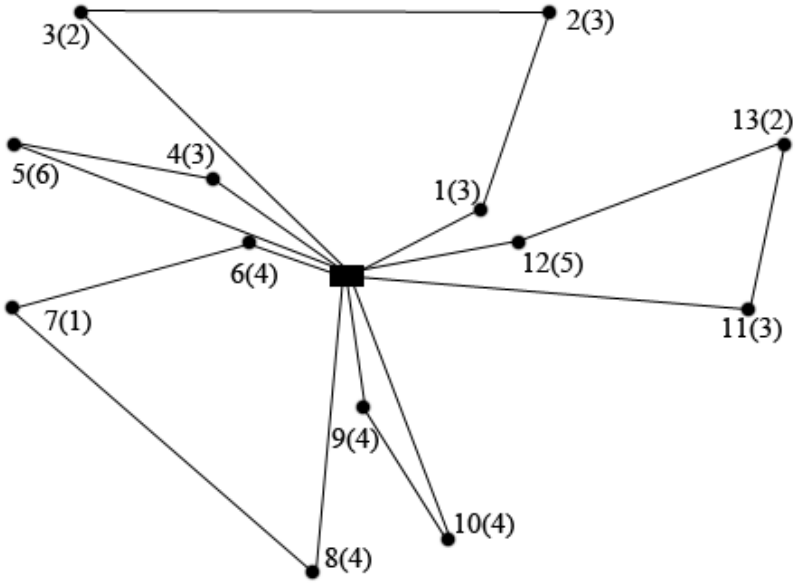


Şekil 3.6. Petal sezgiselinde gezgin satıcı turunun oluşturulması [75]

D-3-4-5-D alt rota olarak kabul edilmez, çünkü (3,4,5) noktalarının toplam talebi araç kapasitesini aşmaktadır. Bu şekilde tüm noktaları için alt rotalar hesaplanır. Bu rotaların her birine taç yaprağı denilmekte algoritmanın adı buradan gelmektedir. Tablo 3.5'de tüm rotalar listelenmiştir.

Tablo 3.5. Araç kapasitesini aşmadan oluşturulan rotalar [75]

(1)	(1,2)	(1,2,3)	
(2)	(2,3)	(2,3,4)	
(3)	(3,4)		
(4)	(4,5)		
(5)	(5,6)		
(6)	(6,7)	(6,7,8)	
(7)	(7,8)	(7,8,9)	
(8)	(8,9)		
(9)	(9,10)		
(10)	(10,11)		
(11)	(11,12)	(11,12,13)	
(12)	(12,13)	(12,13,1)	
(13)	(13,1)	(13,1,2)	(13,1,2,3)



Şekil 3.7. 1-petal sezgiseli ile elde edilen rotalar [75]

Tüm alt rotalar listelendikten sonra, ilk noktadan başlanarak, en çok noktanın talebinin karşılandığı rotalar araçlara atanacaktır. Örnekte, 1.noktanın dâhil olduğu rotalar kontrol edilmiş ve (1,2,3) rotası ilk araca atanmıştır. 2 ve 3. noktaların rotalanmış olması nedeniyle 4. nokta kontrol edilmiş ve (4,5) ikinci aracın rotası olarak belirlenmiştir. Tüm noktalar aynı yöntemle kontrol edilmiş ve son olarak (11,12,13) noktaları 5. araca atanarak rotalama işlemi tamamlanmıştır (Şekil 3.7).

Petal algoritması, özellikle 2-petal algoritması iyi sonuçlar vermektedir ve farklı kısıtların problemle dâhil edilebildiği esnek bir algoritmadır. Ancak diğer klasik yöntemlere göre oldukça karmaşık ve kodlanması oldukça zordur [73].

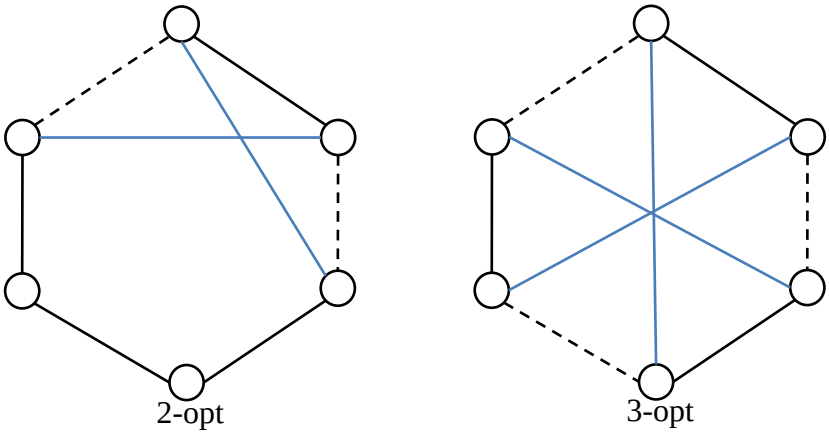
3.2.3. Tur geliştirici sezgiseller

Tur kurucu sezgisellerin, rotalama problemlerine uygulanmasının ardında elde edilen başlangıç çözümünün kalitesini artırmak için kullanılmaktadırlar. Tur geliştirme sezgiselleri ile rotalar kendi içerisinde geliştirebildiği gibi, birden fazla rota arasında noktaların değiştirilmesiyle rotaların tamamının geliştirilmesi de mümkündür. Tek rota geliştirilmesi söz konusu olduğunda 1965 yılında Lin tarafından önerilen λ-opt sezgiseli çoğunlukla kullanılmıştır. Çok rotalı iyileştirme sezgiseller de ise rotalar arasında, noktaların karşılıklı takası, noktaların aktarılması, kenarların takası, vb. yöntemler kullanılarak rotalar iyileştirilmeye çalışılır.

3.2.3.1. Tek rota geliştirme algoritmaları

Bu algoritmaların mantığı rotada λ adet bağlantının kaldırılarak, çaprazlama yoluyla olası farklı bağlantıların denenmesine dayanır. Eğer yeni oluşan rota daha iyi ise mevcut çözüm olarak atanır ve sonraki aramalar bu yeni çözüm üzerinden devam ettirilir. Algoritma belli bir iterasyon sayısına ulaşınca veya tanımlı bir süre boyunca gelişme olmaması halinde durdurulur. λ-opt algoritmasının karmaşıklığı $O(n^{\lambda})$ olarak hesaplanmıştır. Temel mantığa bağlı kalarak geliştirilmiş farklı algoritmalar mevcuttur [15]. λ-opt algoritmasının adımları aşağıda verilmiştir.

- Adım 1. λ değeri kadar kenarı kaldır.
- Adım 2. Kenarları bağlantı noktalarını değiştirerek tekrar kur.
- Adım 3. İyileşme olup olmadığını kontrol et. Olduysa rotayı mevcut rota olarak ata
- Adım 4. Durma kriterini kontrol et, sağlanıyorsa çık aksi halde Adım 1'e git.



Şekil 3.8. 2-opt ve 3-opt uygulama örneği

Şekil 2.8'de 2-opt ve 3-opt sezgiselinde yapılan rota çıkarma ve ekleme işlemine bir örnek verilmiştir. Ancak bir graf planar ise; yani noktalar arasında yapılan bağlantılar birbirini kesmiyorsa en kısa yol planar çizimlerden birisidir. Yani λ-opt sezgiselinde kenarlar değiştirilirken eğer yeni rota birbirini kesen bağlantılardan oluşursa iyi bir rota elde edilmemiş demektir. Böylece bu rotalar elenir ve arama işlemi daha kısa sürede tamamlanabilir. Sezgisel yöntemlerle elde edilen sonuçların iyileştirilmesinde sıklıkla kullanılan bu yöntemle, ilk iterasyonlarda % 60 son iterasyonlarda ise % 2 civarında iyileştirme yapılabildiği görülmüştür [72].

3.2.3.2. Çoklu rota geliştirme algoritmaları

Rota içerisinde yapılacak değişikliklerden ziyade, rotalar arası değişiklikler yapılarak sonuçların iyileştirilmesinin amaçlandığı sezgisel yöntemlerdir. Önerilmiş yöntem sayısı fazla olmakla birlikte birbirlerine oldukça benzemektedirler. Çoklu rota iyileştirme sezgiselleri noktaların takaslanması, noktaların aktarılması, kenarların takaslanması ve karma yöntem olarak dört başlık altında sınıflandırmıştır [77]. Bu yöntemlerin hepsinin ortak amacı, mevcut çözümün daha iyi sonuç içeren bir komşu çözümün varlığının araştırılmasıdır. Şekil 3.9'da yöntemlere örnekler verilmiştir.

Kenar Takası: Bu yöntemde komşu iki rota arasında *(i, i+1)* ile *(j, j+1)* noktalarını bağlayan kenarlar takas edilir ve yeni rotalar *(i-1, i, j+1, j+2…), (j-1, j, i+1, i+2…)* elde edilirler.

Nokta Takası: Bu yöntemde komşu iki rota arasında noktalar değiştirilir. Noktalar komşu rotaya aktarıldıktan sonra hangi *(j, j+1)* nokta çifti arasında yerleştirileceğinin belirlenmesi gerekir. Bu aşamada genellikle tasarruf formülü kullanılır. Ayrıca iki rota arasında takaslanan nokta sayısı eşit olmayabilir. Bir rota diğerine iki nokta verirken diğeri tek nokta verebilir.

Nokta Atama: Bu yöntemde bir rota içerisinde seçilen bir nokta kopartılarak atanacağı rotada uygun bir nokta çifti arasında eklenir. Bu sezgisel uygulanırken rota sayısında azalma görülebilir. İki noktadan oluşan bir rotamız olduğunu farz edelim. Nokta atama yöntemi ile bu noktaları başka rotalara atadığımız takdirde, mevcut rota ortadan kalkmış olacak ve rota sayısı bir eksilecektir.

Karma Yöntem: Bu yöntemde nokta takası ve nokta atama yöntemlerinin beraber kullanmaktadırlar. Ayrıca bu yönteme benzer λ-takası adı altında bir başka yöntem önerilmiştir [15]. Belirlenen λ değerine göre rotalar arasında yapılacak takas işlemi sınırlandırılır. λ=2 için yapılabilecek takaslar (0,1), (1,0), (1,1), (0,2), (2,0), (2,1), (1,2) ve (2,2) şeklinde ifade edilir. Örneğin (1,2) ele alırsak; birinci rotadan 1 adet nokta ikinci rotaya, ikinci rotadan ise 2 adet nokta birinci rotaya atanacak demektir [78]. λ'nın aldığı değere göre iki rota arasında yapılacak takas kombinasyonu değişir ve $[(\lambda-1)^2-1]$ formülü ile hesaplanır. λ=4 iken, 15 farklı takas kombinasyonu oluşur. Öte yandan λ değeri arttıkça olurlu (feasible) çözüm elde etmek zorlaşacaktır.

3.3. Metasezgiseller

Metasezgisel kelimesi ilk defa Glover tarafından 1986 yılında kullanılmasıyla literatüre girmiştir. Klasik sezgisellere göre, daha iyi çözümler üretirler. Kombinatoryal optimizasyon problemlerinin tamamında kullanılabilmeleri nedeniyle özelleştirilmiş ve hatta diğer algoritmalarla birleşerek hibritleşmiş yüzlerce türevleri bulunmaktadır. Ele alınan problemin temel kavramları iyi bir şekilde adapte edilir ve problem net olarak tanımlanabilirse, sezgisel algoritmalar iyi sonuçlar üretebilirler [79].

Metasezgiseller, birden fazla sezgisel algoritmanın bir araya gelmesiyle meydana gelen ve çözüm uzayının umut veren bölgelerinde verimli bir şekilde arama yapılmasını sağlayan iteratif problem çözme yöntemleridirler. Bu yöntemler elde ettikleri çözümleri her iterasyonda iyileştirmeye çalışarak komşu çözümler ararlar. Aramalar, stokastik olduğu gibi belirli bir düzey bilinçte içermektedir. [80].

Metasezgiseller, çözüm uzayının araştırılması için klasik sezgisellerle farklı yöntemlerin zekice bir araya getirildiği ve optimal değere yakın sonuçların verimli bir şekilde elde edilebilmesi için öğrenme stratejilerinin kullanıldığı yinelemeli geliştirme süreçleri olarak tanımlanabilir [81].

Metasezgiselleri, klasik sezgisellerden ayıran en büyük özelliği, çözüm uzayının farklı noktalarında arama yapabilme kabiliyetine sahip olması ve böylece yerel optimum değerlerden uzaklaşabilmesidir. Çoğu metasezgisel yöntemde komşu çözümler aranırken, yerel optimumlardan kaçmak için bazı prosedürler, kısıtlar, bellekler tanımlanmıştır. Bu nedenle daha iyi sonuç üretirler. Ancak klasik sezgisellerde olduğu gibi bir model her kombinatoryal optimizasyon probleminde olduğu gibi kullanılamazlar. Metasezgisel algoritmalarla problemler modellenirken algoritmanın mantığı içerisinde kalarak, algoritmayı probleme özel olarak tasarlamak ve parametrelerini belirlemek gerekir. Bu nedenle literatürde farklı problemlere uyarlanmış yüzlerce çeşit metasezgisel yönteme rastlanmaktadır.

En çok bilinen metasezgiseller bunlarla kısıtlı olmamakla beraber, Adaptif Hafıza Tabanlı Arama (HTA), Tabu Arama (TA), Genetik Algoritma (GA), Benzetilmiş Tavlama (TB) ve Karınca kolonisi Optimizasyonu (KKO)'dur.

Metasezgisel yöntemler hakkında aşağıdaki genellemeler yapılmıştır [80].

- Çözüm uzayında aramanın nasıl yapılacağının sınırlarını çizen stratejilerdir.
- Çözüm uzayının etkili bir şekilde araştırılması ile optimum veya optimuma yakın sonuçlara ulaşmayı amaçlarlar.
- Basit arama prosedürlerinden, karmaşık öğrenme işlemlerine kadar geniş bir alana uygulanabilirler.
- Genellikle deterministik olmayan yaklaşık çözümler üretirler.
- Çözüm uzayında arama yaparken yerel optimum değere takılmayı önleyecek mekanizmalara sahiptirler.
- Temel modelleri basit tanımlar içerirler. Fakat ilgilenilen probleme göre modeller değişmektedir.
- Bir probleme özel geliştirilmemişlerdir.
- Günümüzde arama işlemlerine kullanılmak üzere hafıza tabanlı algoritmalar daha fazla geliştirilmektedir.

Metasezgiselleri tanımlamanın ve sınıflandırmanın birden fazla yolu bulunmaktadır. Çıkış noktasına ve içerdikleri prosedürlere göre farklı sınıflara dâhil edilirler. Prosedürlere ve esin kaynaklarına göre doğadan esinlenenler, popülasyon tabanlı sezgiseller, dinamik sezgiseller ve komşuluk yapılarına göre sezgiseller olarak dört grupta sınıflandırılırlar [81].

Doğadan esinlenmiş sezgiseller: Genlerin çaprazlanması ve kalıtımdan esinlenerek geliştirilen genetik algoritma ve karıncaların yuvalarından yiyeceklere olan en kısa yolu tespit etme sistemlerine dayanan Karınca Kolonisi algoritması bu sınıftadırlar.

Popülasyon tabanlı sezgiseller: Belli sayıda çözümü aynı anda değerlendiren algoritmalardır. Arama alanını daraltmak için tabu adı verilen yasak hareketler listesinin oluşturulduğu Tabu Arama, popülasyondaki komşu çözümleri tarayan Değişken Komşuluk Arama bu sınıfta yer alırlar.

Dinamik sezgiseller: Amaç fonksiyonun arama işlemi esnasında güncellenebildiği sezgisellerdir. Yerel optimumlardan kaçmak için amaç fonksiyonunun güncellendiği Yönlendirilmiş Yerel Arama bu sınıfta yer alır.

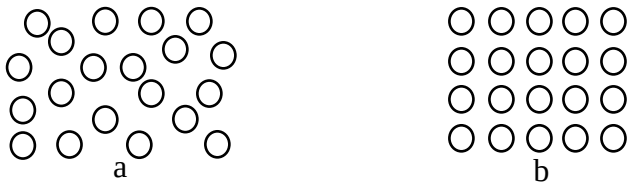
Komşuluk yapılarına göre sezgiseller: Arama işlemi esnasında sezgiseller genellikle bir komşu üzerinde arama yaparlar. Değişken Komşuluk Arama sezgiseli ise birden fazla komşu yapı üzerinde aynı anda arama yapabilmektedir.

3.3.1. Tavlama benzetimi

Tavlama Benzetimi (TB), yerel optimumlardan kurtulabilme özelliğine sahip en eski metasezgisel algoritmalardan birisidir. Adını metallerin yüksek enerjili durumdan daha düşük enerjili duruma geçmeleri için yapılan tavlama işleminden alır. Metal atomları katı bir yapıda iken aralarında boşluklar bulunan düzensiz bir dizilime sahiptirler. Metalin işlenmesiyle hammadde üzerinde çatlak, yırtık gibi deformasyonlar oluşabilir. Bu deformasyonları minimum seviyeye indirmek için metallerin işlenebilirliği artırılmalıdır. Metal erime noktasına kadar ısıtılmakta ve enerji kazanan atomlar hareket etmeye başlamaktadırlar. Bu durum su atomlarının suyu kaynattığımızda hareket etmesiyle aynıdır. Sonra düzenli bir soğutma işlemi uygulanarak atomların daha homojen bir dizilim oluşturmaları sağlanır. Şekil 3.9'da ısıl işlemden önce ve sonrasında metal atomlarının dizilimi gösterilmektedir. Metallerin Şekil 3.9 (a)'da verilen durumdan (b)'deki duruma geçilebilmesi için belli bir ısıya kadar yükseltilen metaller kontrollü olarak soğutulurlar. Bu soğuma işlemi aşağıda verilen Denklem (3.11)'e bağlı olarak belirli bir olasılıkla gerçekleşir [80].

$$P(\Delta E) = e^{\frac{-\Delta E}{kT}} \tag{3.12}$$

$P(\Delta E)$ değeri amaç fonksiyonunda ΔE kadar değişimin gerçekleşme olasılığı vermektedir. T ise o andaki sıcaklık değerini ifade eder. T sıcaklığı yüksek bir değerden başlatılarak belirli bir değere kadar kontrollü bir şekilde azaltılır. T değeri tavlama benzetimi algoritmasında ΔE değişimin kabul edilip edilmeyeceğinin belirlemesinde etkin bir role sahiptir.



Şekil 3.9. Isıl işlem sonrası atomların dizilimi

Bu algoritma, optimizasyon problemlerinde 1983 yılında kullanılmaya başlanmıştır. Model alınan yöntemde bahsedilen katının durumları muhtemel çözümlere, enerji düzeyi/seviyesi amaç fonksiyonunun değerine, düşük enerjili durum optimum çözüme dönüştürülmüştür.

Yüksek enerjili metallerin enerji düzeyi düşük bir duruma indirgenmesi gibi, ARP'nde amaç fonksiyonu bir araç rotasının minimize edilmesidir. Bir başlangıç çözümü oluşturulup, bu çözümden hareketle komşu çözümler aranacak ve bulunana çözümler Denklem (3.11)'de verilen denkleme göre belirli bir ihtimal ile çözüm olarak kabul edilecektir. Çözüm, kabul edildiğinde metalin soğuduğu gibi daha kısa bir rota bulunmuş olacak ve T sıcaklık değeri azaltılacaktır. ARP 'de kullanılacak TB algoritmasının adımları aşağıdaki gibi olacaktır [20].

- Adım 1. Bir başlangıç rotası oluşturulur. Rota uzunluğu E_r değişkenine atanır.
- Adım 2. Bir T sıcaklık değeri belirlenir. Durma kriteri belirlenir. (T sıcaklığı belli bir değerin altına düşünce, algoritma t kadar iterasyon çalışınca, vb.)
- Adım 3. λ-takası, gibi bir komşuluk arama sezgiseli ile komşu çözümler üretilir.
- Adım 4. Üretilen komşu çözüm, mevcut çözüm ile kıyaslanır.
- Adım 4a. Eğer üretilen komşu çözüm, mevcut çözümden iyi ise $(E_{r+1} \geq E_r)$ doğrudan yeni çözüm olarak kabul edilir. Adım 4'e atlanır. (Metal atomları daha düşük enerji seviyesine geçti)
- Adım 4b. Eğer üretilen komşu çözüm, mevcut çözümden kötü ise $(E_r \geq E_{r+1})$ aşağıdaki adımlar yapılır.
  - $\Delta E = E_r\text{-}E_r\text{+}1$ hesaplanır ve Denklem (3.11) uygulanarak belirlenen bir T değeriyle $P(\Delta E)$ olasılığı üretilir.
  - $0 < r < 1$ olacak biçimde bir r değeri üretilir.
  - $P(\Delta E) > r$ ise çözüm kabul edilir (kötü çözüm kabul edildi), değilse Adım 3'e dönülür.
- Adım 5. T sıcaklık değeri düşürülür. Durma kriteri kontrol edilir. Sağlanmıyorsa Adım 3'e gidilir.

Adım 5'de bahsedilen sıcaklığın düşürülmesi işlemi için birçok sıcaklık azaltma yöntemi önerilmiştir. En çok kullanılan geometrik ve logaritmik soğutma yöntemleri Denklem (3.13) ve Denklem (3.14)'de verilmiştir [82].

$$T_k = \alpha T_0 \tag{3.13}$$

$$T_k = \frac{T_0}{\ln k} \tag{3.14}$$

α, değeri soğutma işlemi yavaş yapılmak isteniliyorsa 0.9 ile 0.99 arasında olmalıdır [77]. Aksi halde yeterince komşu çözüm kontrol edilemeyecektir.

Bir diğer önemli nokta ise T sıcaklık değerinin belirlenmesi ile ilgilidir. Eğer sıcaklık değeri ΔE değerine oranla çok yüksek tutulursa, Adım 3b'deki $P(\Delta E)$ çok yüksek olacak ve her durum kabul edilecektir. Çok küçük olduğu takdirde de kötü çözümler hiçbir zaman çözüm olarak kabul edilmeyecek ve yerel optimumlardan kurtulmak mümkün olmayacaktır.

3.3.2. Genetik algoritma

Genetik Algoritmalar (GA), Michigan Üniversitesi'nden psikolog ve bilgisayar bilimci olan J.H.Holland tarafından literatüre kazandırılmıştır. Algoritmanın esin kaynağı Darwin'in Evrim Teorisi'dir. GA'da evrim teorisinde yer alan en iyinin korunumu ve doğal seçilim prensiplerine göre çalışan bir arama yöntemidir. Bu yöntemde çözümlerin her biri kromozon olarak tanımlanırlar. Kromozomların kendi aralarında yaptıkları çaprazlamalar yeni bireyleri, yani optimizasyon problemleri için yeni çözümleri oluştururlar. Kromozomların iyilik (kalite) değerinin ölçülebilmesi için bir fonksiyondan yararlanılır. Kaliteli olan çözümler, diğer bir ifadeyle amaç fonksiyonunu en iyileyen çözümler uygun çözüm olarak kabul edilirler [83].

GA'da yeni kromozomlar bazı parametrelerin ve operatörlerin kullanılmasıyla elde edilirler. Çaprazlama operatörü, mutasyon oranı, popülasyon büyüklüğü, seçilim operatörü, vs. önceden tanımlanması gereklidir. Çaprazlama operatörü iki kromozomun genlerini nasıl değiştireceklerini belirler. Böylece iki ebeveyn kromozomdan yeni bireyler elde edilir. Seçilim operatörü ise ebeveynlerin belirlenmesi için kullanılmaktadır. Popülasyon arasından en uygun ebeveynlerin seçilerek daha iyi bireylerin üretilmesi bu operatörlere doğrudan bağlıdır. En çok kullanılan yöntem rulet tekerleği yöntemidir. Bu yöntemde her kromozom kalitesine göre değerlendirilmekte ve en yüksek değere sahip olanların seçilme şansı daha fazla

olmaktadır [84]. Böylece doğada var olan doğal seleksiyon optimizasyon problemlerinde kullanılabilir bir hale gelmektedir. Tablo 3.6'da Rulet Tekerleği yöntemine bir örnek verilmiştir.

Tablo 3.6. Rulet tekerleği seçimine bir örnek

Kromozom	Dizilim	Kalite	Oran (%)
1	AFGDH	100	27,70083
2	AFHDG	25	6,925208
3	FAGDH	66	18,28255
4	AGHFD	120	33,24100
5	HGFDA	50	13,85042

Örnekte, harflerle ifade edilmiş olan beş adet şehre uğrama dizilimleri verilmiştir. Farklı beş kromozomun toplam kat edilen mesafe dikkate alınarak uygunluk değerleri hesaplanmış ve bu değerler toplama oranlanarak seçilme şansları elde edilmiştir. Yeni bireylerin üretilmesinde kullanılacak ebeveynler oranları en yüksek olan kromozomlar olacaktır. Örnekte bu ebeveynler 4. ve 1. kromozomlardır. Böylece yeni dizilimler elde edilirken uzunluğu en kısa olan (uygun) çözümlerden yola çıkılarak daha kısa şehir dizilimleri elde edilmeye çalışılacaktır.

Popülasyon büyüklüğü ve mutasyon oranı belirlenmesi gereken iki önemli parametredir. Popülasyon çok küçük olursa, çözüm uzayının tam olarak ifade edilmemesi söz konusu olabilir. Yani uygun yeni bireyler verecek kromozomların ebeveynler arasına dâhil edilme ihtimali düşecektir. Popülasyon çok geniş tutulduğu takdirde ise çözüme ulaşmak için uzun süre arama yapmak gerekebilir. Mutasyon, GA'da çaprazlamalar sonucunda daha iyi bireylerin elde edilemediği durumlarda, çözüm uzayının farklı bir bölgesinde arama yapmayı mümkün kılmak için kullanılır. Bir kromozomdaki gen, veya genlerin çıkartılması veya yerlerinin değiştirilmesi ile gerçekleşir. Çaprazlama yapılması gerekmez. Örneğin "AFGDH" sırasının "AFHDG" olarak güncellemesi bir mutasyon işlemidir. Mutasyon oranı çok yüksek belirlenmemelidir. Çünkü belirli bir kurala uymadan yapılan takasların fazlalığı, en iyinin korunumu prensibine ters düşmektedir ve arama işleminin rastgele, bir kurala uymadan yapılması anlamına gelmektedir. Uygun bir kromozom, mutasyonlarla uygun olmaktan çıkabilir. Genetik algoritma uygulamalarında da mutasyon oranı % 0,1 ile % 1 arasında belirlenir.

Genel olarak Genetik Algoritma ile çözüme aşağıda verilen adımların izlenilmesiyle ulaşılır [83]:

- Adım 1. Öncelikle başlangıç çözümlerinin bulunduğu bir ebeveyn kümesi oluşturulur. Bu küme popülasyon her bir ebeveynde kromozom olarak anılmaktadır. Genel olarak popülasyon büyüklüğü 100-300 arasındadır. Fakat standart bir büyüklük söz konusu değildir, popülasyon büyüklüğü problemlere göre farklılık gösterir. Ayrıca bu adımda ebeveynlerin nasıl kodlanacağı belirlenir. Örneğin, *100001111* şeklinde ikili sistem ile, *ABCDEFG* şeklinde harflerle, rakamlarla veya ağaç yapısında kodlanabilir.
- Adım 2. Her kromozomun uygunluk değeri hesaplanır. Uygunluk değerini hesaplamak için bir uygunluk fonksiyonuna ihtiyaç duyulmaktadır. Bu fonksiyon genetik algoritmanın beynini oluşturmaktadır ve GA'da probleme özel çalışan tek kısım bu fonksiyondur. ARP için şehirlerarası mesafe, araç sayısı, harcanan zaman, vb. uygunluğun belirlenmesi için kullanılabilirler. GA'nın başarısı genellikle uygunluk değerinin verimli ve hassas bir şekilde ölçülüp ölçülmediğiyle bağlantılıdır.
- Adım 3. Adım 2'de belirlenen kromozomlar bir seçme yöntemi (rulet tekerleri, turnuva tekniği, vb.) ile seçilerek ebeveynleri oluştururlar. Ardından kromozomlar çaprazlanarak ve mutasyona uğratılarak yeni bireyler elde edilir ve yeni bir popülasyon oluşturulur.
- Adım 4. Popülasyon büyüklüğünü sabit tutabilmek için eski kromozomlar kaldırılır. Bu işlem genellikle elitism (seçkinlik) prensibine göre yapılır. Yani uygunluk değeri en yüksek kromozomlar saklanırken, en düşük uygunluğa sahip kromozomlar popülasyondan çıkartılır. Böylece yeni popülasyonun başarısı artmış olacaktır.
- Adım 5. Eğer durma kriteri sağlanmıyorsa Adım 3'e dönülür ve GA birçok defa çalıştırılarak çok sayıda popülâsyon elde edilir. Durma kriterinin sağlanıp sağlanmadığı her popülasyonun uygunluk değeri kontrol edilerek bulunur. Aksi halde Adım 6'ya geçilir.
- Adım 6. O ana kadar bulunan en iyi kromozom sonuçtur. Çünkü popülâsyonların hesaplanmasında en iyi bireyler saklanmıştır.

3.3.3. Karınca kolonisi

Doğadaki birçok canlı insan zekâsına sahip olmamalarına rağmen, geliştirdikleri sosyal sistemler sayesinde doğada varlıklarını sürdürebilmektedirler. Karıncalar bu durumun en güzel örnekleridir. Bir karıncanın tek başına yuva ile yiyecek arasındaki en kısa mesafeyi bulması imkânsız iken, bir sürünün yaptığı hareket ve tercihlerle rota optimize edilebilmektedir. Karıncalar, yönlerini salgıladıkları feremon adı verilen kimyasal maddeyi takip ederek bulmaktadırlar. Bir karınca yiyeceğe ulaştığı zaman yiyeceğin kalitesi ve miktarı ile orantılı olarak, yuvaya dönüş yolunda geçtiği yerlere feremon bırakır. Yuvadan çıkan karıncalar hangi rotayı takip edeceklerine feremon miktarlarına göre karar verirler. Fazla feremon bulunan rotanın tercih edilme oranı doğal olarak daha fazla olmaktadır. Eğer bir karınca yiyeceğe daha kısa bir yol üzerinden ulaşırsa, yolun daha kısa olması nedeniyle birim mesafeye bırakılan feremon miktarı daha fazla olacağından, bir süre sonra karıncalar daha kısa olan yeni yoldan geçmeye başlayacak ve rota iyileştirilmiş olacaktır [83].

Karınca Kolonisi Optimizasyon (KKO) algoritması Dorigo ve arkadaşları tarafından 1991 yılında karıncaların birlikte oluşturdukları kümülatif zekadan esinlenilerek geliştirilmiştir. Gerçek karınca davranışlarının taklit edilebilmesi için birtakım parametreler ve denklemler yardımıyla yapay karıncalar tanımlanmıştır. KKO algoritması gerçek karınca kolonisi davranışının matematiksel olarak ifade edilmesidir. Fakat yapay karıncaların bir optimizasyon aracı olarak değerlendirilmesinden dolayı önerilen algoritmalar gerçek karınca davranışından biraz farklı yapıdadır [84]. Yapay karıncalar ile doğal karıncalar arasındaki farklar aşağıda verilmiştir [85]:

- Her yapay karınca daha önce geçtiği yolları hatırlayabilecek bir hafızaya sahiptir. Gerçek karıncalarda hafıza yoktur. Sadece feremon izlerini takip ederler. Yapay karıncaya bu özelliğin verilmesinin nedeni arama uzayını daraltma ve arama işlemini kontrollü yapabilme ihtiyacından kaynaklanmıştır.
- Gerçek karıncalar takip ettikleri iz boyunca sürekli feremon salgılarlar. Ancak yapay karıncaların salgıladıkları feremon miktarı bulunan yolun kalitesine bağlı olarak değişir. Ayrıca yapay karıncalar sadece çözüm olarak kabul edilebilecek tam rotalar/yollar üzerinde feremon salgılarlar.

- Yapay karıncalar tamamen ayrık bir dünyada yaşarlar, dış etkiler yoktur. Feremon izleri kontrolsüz bir şekilde azalmaz veya yok olmaz.
- Yapay karıncalarda gerçek karıncalarda olmayan birtakım yetenekler verilebilir. Örneğin ileriyi görme, yerel optimizasyon yapabilme, çözüm için umut verici rotaların bulunduğu bir aday listesinden seçim yapma, vb.
- Yapay karıncaların salgıladıkları feremonlar farklı fonksiyonlara bağlı olarak buharlaşır. Buharlaşma işleminin amacı; bir yolda sürekli feremon birikmesini önlenmesiyle verimsiz yolların buharlaşma neticesinde ortadan kalkmasının ve karıncaların daha kısa yolları tercih etmelerinin sağlanmasıdır.

KKO algoritmasında önemli işlemlerden birisi feremon miktarını güncellenmesidir. Bu adımda farklı yöntemler ve algoritmalar kullanılmasına rağmen en çok kullanılanlar karınca yoğunluk ve karınca miktar algoritmalarıdır. Karınca yoğunluk yönteminde *i* noktasında *j* ye hareket eden bir karınca iki nokta arasındaki yola Q_1 miktarında feremon bırakır. Bu yöntemde yolun uzunluğu önemli değildir, daha çok karınca tarafından tercih edilen yol en iyi yoldur ve çözüm kümesine alınır. Karınca miktarı algoritmasında ise bırakılan feremon *(i,j)* noktası arasındaki uzaklık olan d_{ij} değerine oranla belirlenir. Bu yöntemde karınca sayısı kadar yolun uzunluğu da önemlidir. Karınca yoğunluk yönteminde yol üzerine bırakılan feremon miktarı Denklem (3.15), Karınca miktar yönteminde ise Denklem (3.16) kullanılarak hesaplanmaktadır [84].

$$\Delta\tau_{ij}^{k}(t,t+1)=\begin{cases}Q_1, & \text{k. karınca (i,j) arasında hareket ederse}\\ 0, & \text{aksi takdirde}\end{cases} \tag{3.15}$$

$$\Delta\tau_{ij}^{k}(t,t+1)=\begin{cases}\dfrac{Q_2}{d_{ij}}, & \text{k. karınca (i,j) arasında hareket ederse}\\ 0, & \text{aksi takdirde}\end{cases} \tag{3.16}$$

Karınca kolonisi optimizasyonu rotalama problemlerinin çözümünde kullanılmış ve literatürdeki test problemleri ile denenmiştir. Tavlama benzetimi, yapay sinir ağları yöntemlerine yakın sonuçlar elde edilmiş ancak Tabu arama yöntemi ile daha iyi sonuçlara ulaşılmıştır. Karınca kolonisi optimizasyonunun araç rotalama problemlerinde uygulanışı genel olarak aşağıda verildiği gibidir [86]:

- Adım 1. KKO başlamadan önce karınca miktarı k belirlenir ve bu karıncalardan bir kısmı rota oluşturmak farklı yolları aramak için gezgin olarak atanır. Kalan karıncalar ise feremon yoğunluğuna bağlı olarak oluşan yolları takip edecek olan takipçi karıncalardır.
- Adım 2. k adet karınca n adet nokta içeren bir sistemde rastgele seçilen bir noktadan serbest bırakılır. (Bazen birden fazla noktadan da bırakılabilirler)
- Adım 3. Her bir karınca feromon miktarına bağlı olarak bir olasılıkla takip edeceği yolu seçer.
- Adım 4. Bir yol üzerinden geçen karınca belirlenmiş bir feromon güncelleme yöntemine göre yol üzerine feremon bırakır ve böylece feremon miktarları güncellenir.
- Adım 5. Sistem üzerindeki tüm noktalar ziyaret edilene kadar Adım 2 ve Adım 3 tekrar edilir.
- Adım 6. Bütün karıncalar turu tamamladığı zaman en çok feromon içeren kenarlar en iyi rota olarak belirlenir. Bulunan rota değerleri saklanır.
- Adım 7. Daha iyi bir rota aramak üzere karıncaların hafızaları silinir ve durma kriteri sağlanana kadar önceki adımlar tekrarlanır. Elde edilen rotalar içerisinde en iyi sonuç çözüm olarak kabul edilir.

3.3.4. Tabu arama

1989 yılında Glover tarafından önerilen ve adı ingilizcede yasak anlamındaki "taboo" kelimesinden gelen Tabu Arama (TA) algoritması, Tavlama Benzetimi algoritmasına benzese de elde ettiği çözümleri hafızada saklama ve komşu çözümlerin tamamını araştırması özellikleriyle Tavlama Benzetimi algoritmasından ayrılır. Algoritmanın bu isimle anılmasının nedeni içerdiği yasaklama prosedürüdür. TA'nın, birbirinden farklı ve çözülmesi zor kombinatoryal optimizasyon problemlerine ve uygulamalarına iyi sonuçlar üretebildiğini kanıtlanmıştır [87].

Tabu arama algoritmasında verilen bir başlangıç çözümünün tüm komşularının öncelikle komşuluk dizisine kaydedilir. Komşu çözümlerden en iyi sonuç veren çözümü seçilir ve mevcut çözümden komşu çözüme geçilebilmesi için yapılacak hareketin yasak (tabu) olup olmadığını kontrol ettikten sonra eğer hareket tabu değilse komşu çözümü mevcut çözüm olarak değiştirilir. Hareket tabu olduğu halde,

o ana kadar elde edilen en iyi sonucu ulaşılmasını sağlıyorsa, tabu durumu göz ardı edilir. Bu göz ardı etme olayına tabu yıkma (aspirasyon) adı verilir ve tabu yıkma kriterleri TA algoritmasının başında tanımlanır. Bu döngü iteratif bir şekilde durma kriteri sağlanana kadar devam ettirilir. Paragrafta açıklanan TA adımları aşağıda verilmiştir [84]:

- Adım 1. Bir başlangıç çözümü (S) al. Algoritmada kullanılacak tabu listesi, tabu süresi, aspirasyon kriteri, vb. parametleri belirle.
- Adım 2. Bir komşu çözüm üretme yöntemini kullanarak S'ye ait komşu çözümleri üret ve tabu listesinde olmayan tüm ($S^1 \in N(S)$) komşu çözümler içerden en iyi kabul edilebilir olan (S_{eniyi}) olarak seç.
- Adım 3. Mevcut çözümü (S), S_{eniyi} ile yer değiştir ve S'den S_{eniyi}'ye geçmek için yapılan hareketi tabu listesine ekle.
- Adım 4. Durdurma kriteri sağlanıncaya kadar Adım 2 ve 3'ü tekrar et.

Tabu arama algoritmasında amaç fonksiyonu geliştirilemezse arama sonlandırılmaz. Amaç fonksiyonunu en az etkileyen hareket ile devam edilerek yerel optimumlardan kurtulmaya çalışılır. Ancak, bazen amaç fonksiyonun en az etkileyen hareket daha önce üretilmiş bir komşu çözüme ulaştırabilir. Bu durumda sonsuz bir çevrim içerisinde sıkışılmış olur. Bu durumu aşmak için söz konusu hareketin yapılması yasaklanır ve tabu listesinde belirli bir süre kalmak üzere eklenir. Bu yasaklama stratejisi başka hiçbir metasezgisel yöntemde bulunmamaktadır [84].

Tabu arama algoritmasının diğer algoritmalardan farkının anlaşılabilmesi için algoritma adımları ve kullanılan stratejiler gibi ana bileşenleri iyi kavranmalıdır. Birçok kombinatoryal optimizasyon problemleri doğal olarak *n* elemanın permütasyon araması olarak modellenmektedir. Çözüm problemin elemanlarının farklı dizilimleri arasında aranmaktadır. Her bir dizilim N(s) komşu çözümler kümesinin elemanı ise, (Ns_{yeni}) = *H* (Ns_{mevcut}) arasında geçiş *H* ile verilen bir hareket ile gerçekleşir. Komşuluk çözümlerinin elde edilmesi için yapılacak *H* hareketi için çeşitli yaklaşımlar mevcuttur. Çalışmanın önceki bölümlerinde bu yöntemlerin bazılarından (nokta takası, kenar takası, nokta atama, karma yöntemler) bahsedilmiştir. Seçtiğimiz komşu çözüm oluşturma yöntemi, komşu çözümler kümesini etkileyeceğinden çözümlerden birini rassal seçerek iyileştirme yapan diğer

metasezgisellerden farklı olarak tüm komşu çözümler içerisinde en iyi çözümü arayan tabu arama için çok daha önemlidir [88].

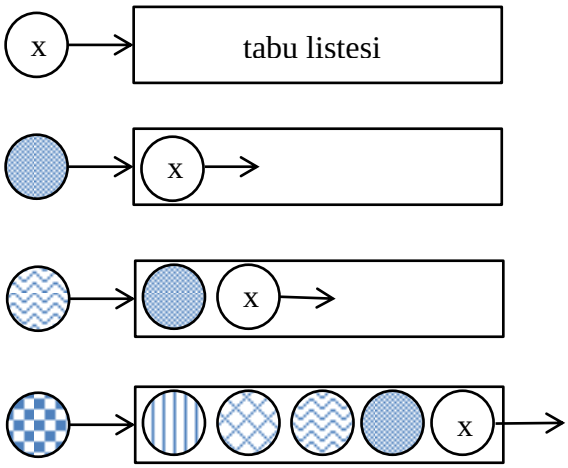
Tabu arama algoritmasındaki bir diğer önemli farklılık ise tabu listesinin varlığıdır. Tabu listesi daha önce üretilmiş çözümlere tekrar dönmeyi veya çözüm uzayının iyi sonuçlar vermeyen bölgelerinde arama yapmayı engellemek amacıyla kullanılır. *H* hareketi eğer tabu listesinde yer alıyorsa bu hareket aspirasyon kriteri adı verilen şartın sağlanması dışında gerçekleştirilmez. Aspirasyon kriteri, bir *h* hareketinin veya çözümün hangi şartlar dâhilinde tabu olma durumunun göz ardı edileceğini belirler. Birçok farklı şekilde tanımlanabileceği gibi, genellikle o ana kadar elde edilen en iyi sonuca ulaştırması aspirasyon kriteri olarak tanılanır. Tabu listesinin çok uzun veya çok kısa olmaması gerekir. Çok uzun bir liste tanımlanmışsa algoritmanın belirli bir süre çalışmasıyla birçok hareket tabu listesine girecek ve komşu çözümler arasında yeterli bir arama yapılamayacaktır. Kısa olduğu durumlarda ise döngülerin oluşması ve işlem süresinin uzaması söz konusu olabilecektir. Tabu arama algoritmasında var olan stratejiler şöyledir [84].

- Tabu listesine eklenecek çözümleri belirleyen yasaklama stratejisi,
- Tabu listesinden çözümlerden hangisinin ne zaman çıkacağını belirleyen serbest bırakma stratejisi
- Denenecek hareketlerin seçimi için yukarıdaki stratejiler arasında ilişkiyi kontrol edecek kısa dönem stratejisi.

Tabu listesine alınan hareketler problemin özelliklerine göre tanımlanacak bir serbest bırakma stratejisine göre tabu olmaktan çıkartılır. Genellikle tabu listesinde çıkartma stratejisi algoritmanın iterasyon sayısı olmaktadır. Şekil 3.10'da çözümlerin tabu listesine eklenmesi ve çıkartılmasına bir örnek verilmiştir. x çözümü beş iterasyon boyunca tabu listesinde kalmış, altıncı iterasyon ile tabu olmaktan çıkartılmıştır.

Birçok arama algoritmasında olduğu gibi Tabu Arama algoritması için de bir durma kriteri belirlenmelidir. Sıklıkla kullanılan durdurma kriterleri aşağıda verilmiştir [89].

- Belirli bir iterasyondan sonra veya belirli bir CPU zamanında sonra,
- Belirli bir iterasyon boyunca amaç fonksiyonun iyileşmesi halinde,
- Amaç fonksiyonu önceden belirlemiş eşik değerine ulaşınca.



Şekil 3.10. Tabu listesinin güncellenmesi

ARP hakkında literatürde yer alan tabu arama hakkındaki en önemli çalışmalar, Osman tarafından önerilen λ-takası yöntemi [15], Taburoute algoritması [90], Taillard'ın algoritması [91], Rochat ve Taillard'ın Adaptif Hafıza Prosedürü [92], Xu and Kelly algoritması [93], Barbarosoğlu ve Özgür algoritması [94] ve Bütünleşik Tabu Arama algoritmasıdır [39,95].

4. ARAÇ ROTALAMA PROBLEMİ İÇİN YENİ BİR ÇÖZÜM ÖNERİSİ

Bu bölümde literatürde yer alan algoritmalardan esinlenilerek geliştirilen; sırasıyla talep noktalarının gruplandırılması, rotalanması ve iyileştirmesi adımlarının uygulandığı bir sezgisel algoritma önerilmiştir. Gruplandırma işlemi için Bulanık c-ortalama yönteminden yararlanılmış ve noktalar literatürdeki uygulamalardan farklı olarak bir küme yerine birden fazla kümeye ait olmaları sağlanmıştır. Noktaların aitlik dereceleri, talep miktarları ve araç kapasiteleri dikkate alınarak en uygun gruplara bölünmüş ve tur oluşturma sezgiselleri yardımıyla başlangıç rotası elde edilmiştir. Başlangıç rotası bir arama algoritması ile geliştirilmeye çalışılmıştır.

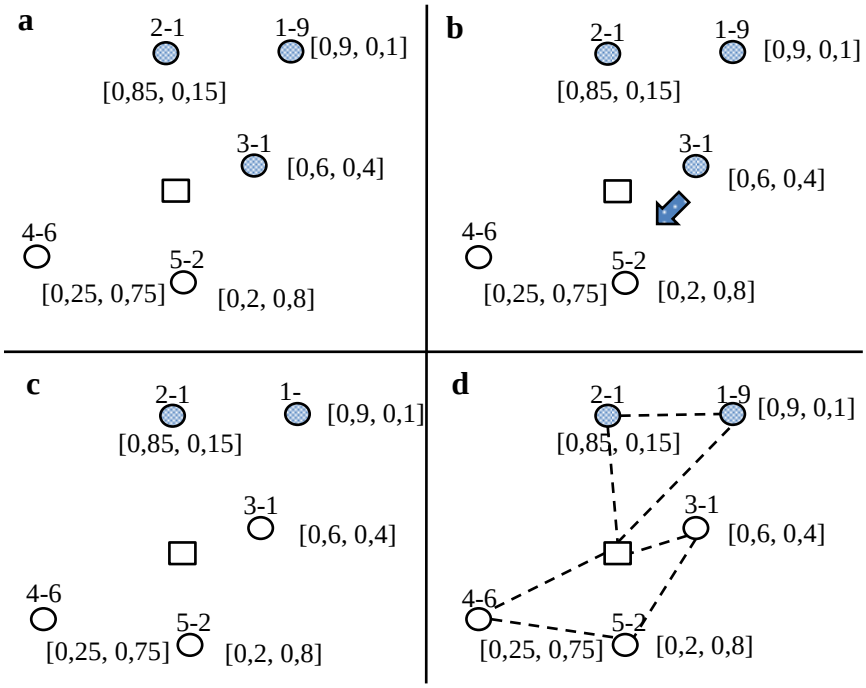
Algoritmanın çıkış noktası büyük boyutlu gezgin satıcı problemlerinin çözülebildiği halde, kesin çözümü elde edilen araç rotalama problemi boyutlarının çok küçük olmasıdır. Bu durumun nedeni elbette ARP'nin araç, kapasite, zaman, vb. kısıtlar içermesidir. Eğer araç rotalama probleminin kısıtları gevşetilerek, gezgin satıcı problemine indirgenebilirse problemin daha hızlı çözülebileceği düşünülmüştür.

4.1. Önerilen Yöntemin Yapısı

Önerilen yöntem önce grupla sonra rotala mantığına dayanmaktadır. Önce grupla sonra rotala algoritmalarında, grup oluşturma işlemi süpürme ve petal algoritması gibi klasik sezgiseller yardımıyla gerçekleştiği gibi, noktaların birbirine göre konumlarını değerlendiren k-ortalama gibi yöntemlerde kullanılmıştır. Fakat literatürdeki kümeleme yöntemleri, talep noktalarının sadece bir küme merkezi ile olan yakınlık ilişkisine göre gruplandırma yapmaktadırlar. Önerilen yöntemde ise kümeleme işlemi Zadeh tarafından önerilen Bulanık Mantık prensiplerine dayanan Bulanık c-ortalama algoritması (BCO) ile yapılmış ve her talep noktasının [0,1] arasında değişen bir aitlik derecesi ile farklı kümelere ait olabilmeleri sağlanmıştır. BCO ile talep noktalarının aitlikleri tayin edilmiş ve böylece noktaların kümelere olan aitlik değerlerinin ne derece yüksek veya zayıf olduğu tespit edilmiştir. İlerleyen adımlarda, aitlik dereceleri kümeler arasında noktaların takaslanması ve komşu çözümlerin elde edilmesinde kullanılacaktır.

Talep noktalarının kümelenmesinin ardından, bir başlangıç rotası oluşturulur ve rota üzerinde çalışmada önerilen arama yöntemi ile daha iyi sonuçlar aranır. Klasik küme teorisinde bir küme içerisindeki elemanların hepsi kümeye eşit derecede aittirler. Bulanık kümelerde ise bir eleman birden fazla kümenin üyesi olabilir. Önerilen yöntemde üyelik (aitlik) değerleri kullanılmış ve komşu çözümler aranırken takaslanacak noktalar bu değerlere göre belirlenmiştir. Bir kümeye aitliği yüksek olan noktaların rotadan kopartılması daha zor bir hale getirilmiştir.

Bu durumu bir örnek ile açıklayalım. Beş adet nokta içeren problemde sırasıyla talepler 9, 1, 1, 6 ve 2 birim olsun. İki adet 10 birim kapasiteli aracımız olduğunu ve birbirine olan uzaklıkları bilinen noktaların bulanık c-ortalama algoritması ile kümelenmiş olduğunu kabul edelim. Şekil 4.1(a)'da köşeli parantez ile belirtilen değerler noktaların aitlik değerlerini, virgülle ayrılmış olan rakamlardan ilki noktanın sırasını, ikincisi ise talep miktarını göstermektedir. Bulanık kümeleme işleminden sonra (1,2,3) noktalar birinci kümeye, (4,5) ise ikinci kümeye dâhil edilmiştir. Ancak bu aidiyet aitlik derecesinin büyük olması durumundan kaynaklıdır. Aslında noktalar her iki kümeye aittirler (3. nokta ilk kümeye 0,6 ikinci kümeye 0,4 dereceyle aittir). İlk kümede yer alan noktaların talepleri toplandığında 9+1+1=11 birim ile araç kapasitesinin aşıldığı görülmektedir. İkinci kümede noktaların taleplerinin toplamı ise 6+2=8 birimdir. Daha önceki bölümlerde bahsi geçen tur iyileştirme sezgisellerinden noktaların takaslanması veya atanması yöntemi kullanılırsa olurlu bir çözüm elde edilebilir. Ancak, bu algoritmalar genellikle bir kümeden rassal olarak bir elemanı seçip diğer kümeye atayarak komşu çözümler üretmektedirler. Yani birinci kümeden 2. talep noktasını kopartıp ikinci kümeye gönderebilirler. Yeni durum (1,3) ve (2,4,5) olacak ve araç kısıtları da sağlanacaktır. Önerilen takas sisteminde ise bir kümeden ilk ayrılacak eleman o kümeye aitliği en az olan elemandır. Şekil 4.1 (b)'de 3.noktanın 0,6 aitlik değeri ile ilk kümeye en az aitliği olan nokta olduğu görülmektedir. Öte yandan 0,4 ile ikinci kümeye olan aitlik değeri azımsanmayacak kadar büyüktür. Böylece birbirine yakın noktaların komşu çözümlerin aranması esnasında kopartılmasının önüne geçilmektedir. Şekil 4.1(c)'de aitlik değerleri dikkate alınarak 3. noktanın ilk kümeden kopartılıp ikinci kümeye dâhil edildiği görülmektedir. Şekil 4.1(d)'de ise takas işlemi sonrasında oluşan rotalar gösterilmiştir.



Şekil 4.1. Bulanık aitlik değerleri ile komşu çözümlerin elde edilmesi

Aitlik değerlerinin kullanıldığı komşu çözüm arama yönteminde, farklı takas kriterleri belirlenebilir. Bir kümeden aitliği en düşük olan noktanın koparılabileceği gibi, bir aralık tanımlanarak, aitlik derecesi tanımlanan aralıkta olan noktalardan rassal olarak seçim yapılması da sağlanabilir. Örneğin $f < 0,5$ aralığı verilirse, kümeye aitliği 0,5 değerinden küçük olan noktalar arasından bir tanesi seçilip farklı bir kümeye aktarılacaktır. Böylece çözüm uzayının kısıtlı bir alanı yerine, daha geniş bir alanı tarama imkânı doğacaktır.

Komşuluk arama esnasında üyeliği en az olan noktaların takaslanması yerine tanımlanan aralığa bağlı olarak rassal seçim yapılacak olursa, aynı seçimlerin tekrar yapılarak algoritmanın bir döngüye takılma ihtimali söz konusu olabilir. Bu durumu engellemek için tabu arama algoritmasındaki gibi bir yasak listesi tanımlanmıştır. Eğer i kümesine ait j noktası takas için seçilmiş ve takas deterministik kısıtlar nedeniyle gerçekleşememişse bu f_{ij} seçimi bir listeye kaydedilir. Aynı seçimin tekrar yapılması halinde, nokta yasak listesinde olmasından dolayı işleme alınmaz ve işlem zamanından tasarruf edilir. f_{ij} değeri yasak listesinden ancak ve ancak bir komşu çözüm üretildiğinde çıkar. Çünkü artık rotalar değişmiştir ve f_{ij} noktasının takaslanmasını engelleyen unsur ortadan kalkmış olabilir. Yasak listesinden çıkarma, yine tabu arama algoritmasında yer alan aspirasyon (yıkma) benzeri bir işlemdir.

4.1.1. Bulanık C-Ortalama Algoritması

BCO, kümeleme işlemlerinde yaygın kullanılan yöntemlerden birisidir. BCO'nun en büyük farkı K-ortalama algoritması gibi keskin bir kümeleme yapmamasıdır. Bulanık kümeler teorisinin, kümeleme algoritmalarında kullanılmasıyla bir nesne aynı anda birden fazla kümeye farklı aitlik dereceleri ile dâhil olabilmektedir [96]. Nesneler kümelerin her birine [0,1] arasında değişen bir üyelik derecesi ile yakın olduğu kümeye aitliği daha büyük olmak kaydıyla ait olacaktır.

BCO, 1981 yılında ise Bezdek tarafından geliştirilmiştir. Temel olarak BCO, Denklem (4.1)'de verilen amaç fonksiyonun minimize edilmesine dayanır [97].

$$J(X,U,V) = \sum_{i=1}^{c}\sum_{k=1}^{N}(u_{ik})^{m}(d_{ik})^{2} \tag{4.1}$$

Burada u_{ik}, *k.* sıradaki verinin *i.* kümeye aitlik derecesini gösterir. *c*, küme sayısını, d_{ik} ise *k.* sıradaki verinin *i.* küme merkezine olan dik uzaklığını ifade eder. *m* ise ağırlık değeridir. d_{ik}, Denklem (4.2) yardımıyla hesaplanır;

$$d_{ik} = \sqrt{(x_k - v_i)^2} \tag{4.2}$$

x_k, *k.* sıradaki veriyi, v_i ise i. kümenin merkez noktasını ifade etmektedir. u_{ik} ve v_i Denklem (4.3) ve Denklem (4.4) ile hesaplanır. Ayrıca, bir verinin *c* adet kümeye olan aitlikleri toplamı 1 değerine eşit olmalıdır.

$$u_{ik}^{\,r+1} = \frac{1}{\sum_{j=1}^{c}\left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{2}{m-1}}}, \qquad 1 \le m \le \infty, \qquad \forall i \in c \tag{4.3}$$

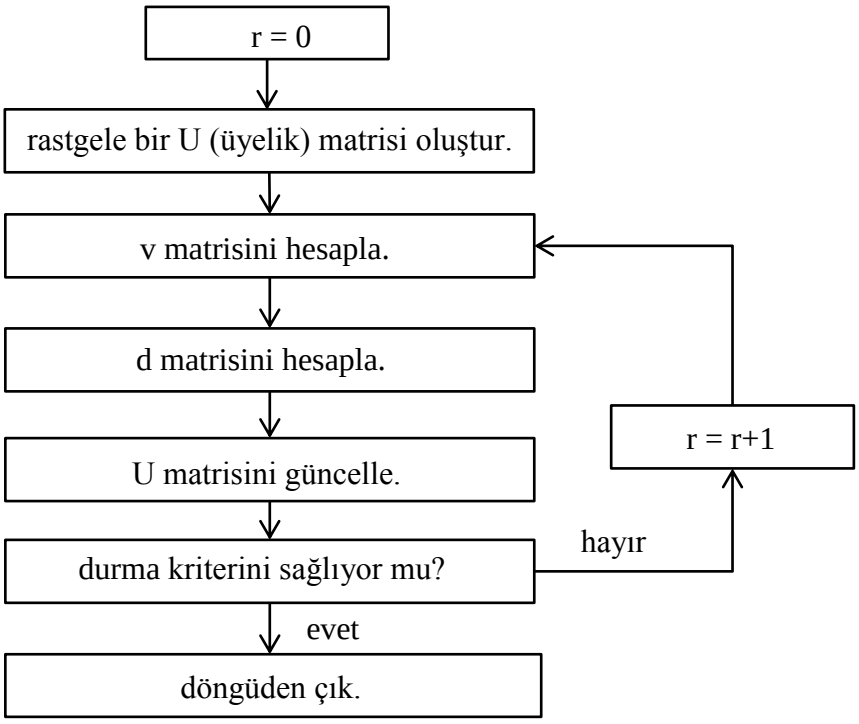
$$v_i = \frac{\sum_{k=1}^{N}(u_{ik})^{m}(x_k)}{\sum_{k=1}^{N}(u_{ik})^{m}}, \qquad \forall i \in c \tag{4.4}$$

Kümeleme algoritması ancak ε gibi bir değerin altına inildiğinde (ulaşıldığında) durmaktadır. Aksi halde sürekli yeni *v,u,d* değerlerini hesaplayarak sonucu

iyileştirmeye çalışacaktır. Durma ölçütü sağlandığında bu döngü sona erecektir. Durma ölçütü 0,001 gibi bir sayıdır. Denklem (4.5)'de durma ölçütünün formüle edilmiş hali görülmektedir. Burada *r* değeri algoritmanın iterasyon sayısıdır.

$$\left\| U^{r+1} - U^{r} \right\| \leq \varepsilon \tag{4.5}$$

Bulanık c-ortalama kümeleme algoritmasına durma ölçütü ε, bulanıklaştırma parametresi *m* ve küme sayısı *c* belirlenerek başlanabilir. *m* kümenin bulanıklık derecesini belirlemektedir ve büyük *m* değeri fonksiyonun bulanıklığını artırmaktadır. Bu değer genellikle 2 olarak alınmaktadır [98]. Önceki sayfalarda verilen formüller dikkate alınarak oluşan algoritmaya ait akış şeması Şekil 4.1'de sunulmuştur [99].



Şekil 4.2. Bulanık c-ortalama algoritması akış şeması

4.2. Önerilen Yöntemin Adımları

Önerilen yöntem, birbirinden bağımsız grupla ve rotala adımlarından oluşması nedeniyle iki algoritmanın birleşimi ile meydana gelmektedir. Gruplama algoritması bulanık c-ortalama yöntemini içeren ve kapasite kısıtlarını dikkate alınarak başlangıç rotasının tayin edildiği adımları içerirken, rotalama algoritması tanımlanmış parametreler doğrultusunda rotaların iyileştirilmesi adımlarını içermektedir.

4.2.1. Gruplama algoritması

Algoritma adımları aşağıdaki gibidir:

Adım 1. Bulanık c-ortalama da kullanılacak küme sayısı *c*, katsayı *m*, değerlerini belirle. Küme sayısı tüm noktaların talebini karşılayacak şekilde araç sayısına bağlı olarak belirlenir. Küme sayısı araç sayısından eşit veya fazla olmalıdır *(c ≥ V).*

Adım 2. Bulanık c-ortalama algoritması ile noktaları kümele. Elde edilen aitlik değerlerini *aitlik[i][j]*, matrisine ekle (*i* kümeler *j* talep noktaları).

Adım 3. Her kümeye atanan noktaların taleplerini topla ve araç kapasitesinin aşılıp aşılmadığını Denklem (4.6)'da verilen denklem ile kontrol et. Kümelerin talepleri araç kapasitelerini aşmıyorsa Adım 7'ye atla.

$$\sum_{i \in C} d_i x_{ic} \leq q_{kc}, \qquad \forall c \in L, \quad \forall k \in V \tag{4.6}$$

d_i, *i* noktasının talep miktarını; x_{ic}, *c* kümesine dâhil edilen *i* noktasını; q_{kc}, *c* kümesine servis verecek *k* aracının kapasitesini temsil eder. L bulanık c-ortalama ile elde edilen grupları, V ise taşıma yapacak araçlar kümesidir.

Adım 4. İçerdiği noktaların talepleri, araç kapasitesini aşan kümeleri belirle. Talepler toplamı ile araç kapasite farkı en fazla olan kümeyi seç ve küme elemanlarından aitliği en az olanı veya belli bir aitlik değerinin altında (*f*) olan noktaları bul ve takas edilecek nokta(lar) olarak belirle.

Adım 5. Belirlenen noktayı aitliği ikinci en büyük olduğu kümeye ekle ve önceki küme listesinden çıkart. Noktaları eklerken uzak köşelerde olan noktaların aktarılmasını engellemek amacıyla aday noktalarının aktarılacağı küme merkezi ile depo arasındaki tasarruf değerlerini bul. Tasarruf değeri en küçük olan noktayı ata.

Adım 6. Yeni kümelerin elemanlarını güncelle. Eğer durma kriteri sağlanmıyorsa Adım 3'e dön. Durma kriteri tüm kümelerin toplam taleplerinin araç kapasitesinin altında olması olarak belirlenir.

Adım 7. Kümeleme işlemini sonlandır.

4.2.2. Rotalama ve iyileştirme algoritması

Başlangıç rotasının oluşturulması ve çözümlerin geliştirilmesi aşamasında kullanılacak olan algoritmanın adımları aşağıda verilmiştir.

Adım 1. Bir başlangıç rotası üret (Tasarruf, en yakın komşuluk, sıralı ekleme) ve her bir kümeye ait rotayı λ-opt ile iyileştir. Her bir grup c için oluşan rotayı R_c kümesine ekle ve rota uzunluğunu u_c değişkeninde sakla. Rotayı mevcut rota olarak belirle (M_{Rota}). Boş bir geçici rota kümesi tanımla (G_{Rota}). Durma kriterini belirle.

Adım 2. Komşuluk arama için gerekli olan parametreleri belirle ve strateji seç (aitliği en düşük olan mı takaslansın yoksa belirli bir aitlik aralığından takaslanacak nokta rassal mı seçilsin). Başlangıç için takas işlemi yapılacak kümeyi rassal olarak belirle. (aitlik derecesi en düşük noktayı içeren kümede seçilebilir)

Adım 3. Belirlenen stratejiye göre kümeden bir nokta seç. Seçilen nokta kümeden çıkartıldığında kümenin kapasite kısıtlarını sağlayıp sağlamadığını kontrol et. Sağlamıyorsa aynı küme içerisinden yeni bir nokta seç. Bu adımda takas edilecek nokta rassal seçilmeyerek gruplama işlemindeki Adım 5'de olduğu gibi takas yapılacak olan noktaların aktarılacağı kümenin merkezi, ve depo arasındaki tasarruf değerleri kontrol edilip, değeri en küçük olan aktarılabilir.

Adım 4. Seçilen talep noktasını aitliği ikinci en yüksek olduğu kümeye ata. Tabu matrisini güncelle (küme i den j ye yapılan hareketi akılda tut)

Adım 5. Takas edilen noktanın geldiği ilk kümeye aitliği fazla olan talep noktalarını kullanarak tasarruf değerlerini hesapla. En az tasarruf değerine sahip olan noktalar arasına yeni noktayı ekle.

Adım 6. Takas yapılan iki kümenin rota uzunluklarını hesapla. λ-opt yöntemi ile yerel olarak rotaları iyileştirmeye çalış. Ayrıca çoklu rota geliştirme algoritmaları ile rotalar arası takaslar yaparak daha iyi sonuç ara. Toplam rota uzunluğu önceki duruma göre iyileşmiş ise ($u_c^{yeni} \geq u_c^{eski}$) ve araç kapasite kısıtları sağlanıyorsa Adım 8'e atla. Aksi halde kötü çözümü geçici rota (G_{Rota}) olarak ata.

Adım 7. İterasyon sayacını artır (S=S+1). Eğer iterasyon sayacı belirli bir değere erişmiş ise Adım 7a'ye geç. Aksi halde Adım 7b'ye atla.

Adım 7a. Aramayı durdur. (G_{Rota}) kümesini boş küme yap. Tabu matrisini temizle. Adım 2 ye dön. (Belli sayıda arka arkaya kötü çözüm üretilirse, aynı arama uzayında daha fazla arama yapılmasının engellenmesi amacıyla bu adıma ihtiyaç vardır)

Adım 7b. Kapasite kısıtının aşıldığı rotayı takas rotası olarak belirle. Adım 3'e dön.

Adım 8. Çözümü mevcut rota (M_{Rota}) olarak ata; (G_{Rota}) kümesini boş küme yap. Tabu matrisini temizle. S=0 olarak ata. Durma kriteri sağlanıyorsa algoritmadan çık ve (M_{Rota})'yı çözüm olarak göster. Sağlanmıyorsa kritere erişene kadar iyileştirmeye devam etmek için Adım 2'ye dön.

4.3. Önerilen Yöntemin Örnek Bir Problem Üzerinde Uygulanması

Önerilen yöntem Christofides ve Eilon tarafından önerilen test problemlerinden 50 adet talep noktası içeren, tek depodan hareketle 5 araç ile rotalama yapıldığı ve her aracın kapasitesinin 160 ile sınırlı olduğu E-n51-k5 üzerinde uygulanmıştır.

4.3.1. Talep noktalarının kümelere ayrılması

E-n51-k5 problemindeki noktalar öncelikle bulanık c-ortalama algoritması ile gruplandırılmıştır. Kümelemeden sonra elde edilen küme merkezi koordinatları Tablo 4.1'de, aitlik değerleri Tablo 4.2'de verilmiştir. Her kümenin araç kapasite kısıtını aşıp aşmadığı kontrol edilmiş ve Tablo 4.3'de kümeler ve talepler toplamı verilmiştir. 2. kümede yer alan noktaların talebi araç kapasitesinin üzerinde olması nedeniyle talebin dengelenmesi için bu kümeden noktalar başka kümeye aktarılmıştır. 2. kümede aitliği en düşük nokta seçilerek komşu kümeye aktarılmış, takas işlemi ardından talep miktarlarındaki değişim Tablo 4.4'de verilmiştir.

Tablo 4.1. BCO işleminden sonra küme merkezleri

	x	y
1	19,74071	55,47487
2	52,41816	33,95772
3	51,07988	60,58620
4	33,15751	20,94463
5	12,17673	18,99084

Tablo 4.2. BCO işleminden sonra talep noktalarının aitlik dereceleri

	Talep Noktaları									
Küme No	1	2	3	4	5	6	7	8	9	10
1	0,314	0,080	0,011	0,069	0,057	0,766	0,870	0,578	0,001	0,043
2	0,173	0,301	0,013	0,054	0,353	0,049	0,027	0,079	0,995	0,546
3	0,357	0,518	0,967	0,028	0,057	0,052	0,048	0,241	0,001	0,059
4	0,099	0,069	0,005	0,303	0,466	0,068	0,027	0,058	0,002	0,291
5	0,057	0,032	0,003	0,546	0,067	0,065	0,028	0,044	0,001	0,061
Küme No	11	12	13	14	15	16	17	18	19	20
1	0,112	0,104	0,061	0,500	0,016	0,032	0,029	0,190	0,018	0,026
2	0,498	0,152	0,030	0,071	0,048	0,794	0,043	0,077	0,016	0,061
3	0,169	0,058	0,021	0,065	0,013	0,103	0,016	0,051	0,009	0,883
4	0,164	0,553	0,087	0,136	0,874	0,052	0,773	0,239	0,069	0,019
5	0,057	0,134	0,801	0,228	0,049	0,019	0,138	0,443	0,888	0,010
Küme No	21	22	23	24	25	26	27	28	29	30
1	0,050	0,132	0,958	0,729	0,323	0,618	0,539	0,119	0,060	0,028
2	0,629	0,103	0,008	0,048	0,073	0,072	0,124	0,068	0,399	0,793
3	0,212	0,689	0,013	0,057	0,061	0,204	0,144	0,753	0,442	0,054
4	0,077	0,048	0,010	0,068	0,155	0,057	0,117	0,036	0,068	0,096
5	0,033	0,028	0,011	0,099	0,389	0,049	0,075	0,025	0,031	0,029
Küme No	31	32	33	34	35	36	37	38	39	40
1	0,288	0,235	0,054	0,027	0,053	0,076	0,002	0,040	0,061	0,058
2	0,094	0,282	0,242	0,813	0,105	0,116	0,004	0,745	0,483	0,051
3	0,514	0,259	0,058	0,071	0,782	0,729	0,001	0,060	0,091	0,030
4	0,060	0,153	0,524	0,066	0,038	0,049	0,986	0,124	0,277	0,153
5	0,044	0,071	0,122	0,023	0,022	0,031	0,006	0,031	0,088	0,707
Küme No	41	42	43	44	45	46	47	48	49	50
1	0,005	0,042	0,677	0,021	0,042	0,234	0,144	0,905	0,027	0,012
2	0,004	0,056	0,062	0,043	0,135	0,223	0,110	0,021	0,751	0,918
3	0,002	0,025	0,092	0,015	0,038	0,119	0,056	0,035	0,039	0,035
4	0,015	0,326	0,074	0,811	0,659	0,301	0,441	0,021	0,153	0,026
5	0,973	0,550	0,095	0,110	0,127	0,124	0,249	0,017	0,030	0,009

Tablo 4.2'deki verilerle her noktanın hangi kümeye dâhil olacağı belirlenmiştir. Örneğin 1., 2. ve 3. noktalar 3. kümeye, 4. nokta 5. kümeye dahil edilecektir.

Tablo 4.3. Kümelerin talepleri ve talep noktaları

Küme No	Talepler	Küme Elemanları
1	154	6-7-8-14-23-24-26-27-43-48
2	172	9-10-11-16-21-30-32-34-38-39-49-50
3	143	1-2-3-20-22-28-29-31-35-36
4	151	5-12-15-17-33-37-44-45-46-47
5	157	4-13-18-19-25-40-41-42
toplam talep	777	

Tablo 4.3 incelendiğinde 2. kümenin talepler toplamının araç kapasitesi olan 160 değerini aştığı görülmektedir. 2. kümenin elemanlarına ait aitlik değerlerini kontrol edildiğinde en düşük aitlik değerinin 0,282 ile 32. noktaya ait olduğu görülür. 32. noktanın ikinci sırada en yüksek ait olduğu küme ise 0.259 ile 3. kümedir. Yani 32.nokta 2. küme ile 3. kümenin neredeyse tam ortalarında konumlanmıştır. 32. talep noktası 3. kümeye aktarılırsa araç kapasitesinin aşılması durumu ortadan kalkmaktadır. Yeni durum Tablo 4.4'de verilmiştir.

Tablo 4.4. Takas sonrası kümelerin talepleri ve talep noktaları

Küme No	Talepler	Küme Elemanları
1	154	6-7-8-14-23-24-26-27-43-48
2	160	9-10-11-16-21-30-34-38-39-49-50
3	155	1-2-3-20-22-28-29-31-32-35-36
4	151	5-12-15-17-33-37-44-45-46-47
5	157	4-13-18-19-25-40-41-42
toplam talep	777	

Kümelerde yer alan noktalara ait talepler toplamının araç kapasitesinden az olması nedeniyle gruplama işlemi tamamlanmıştır Ele alınan örnek problemde sadece bir adımda uygun çözüm elde edilmiştir. Eğer aitlik derecesi en düşük olan nokta talebin dengelenmesi için yeterli olmasaydı, kümede ondan sonra aitlik derecesi en düşük nokta takaslanacak ve kapasite kısıtı sağlanana kadar bir döngü halinde devam edecektir.

4.3.2. Rotaların oluşturulması ve iyileştirilmesi

Kümele işleminin ardından yöntemin ikinci adımına geçilmiştir. İkinci adımda Tablo 4.4'de verilen kümeler birbirinden bağımsız olarak ele alınmış ve depodan başlayan ve sonlanan turlar elde edilmiştir. Başlangıç rotasının oluşturulması aşamasında literatürde yer alan tur kurucu sezgisellerden herhangi biri kullanılabilir. Ancak bu örnekte tasarruf algoritmasından yararlanılmıştır. Her kümeye ait talep noktalarının rotalanması işlemi gezgin satıcı problemi gibi çözülmektedir. Kapasite kısıtlarına göre kümeleme yapıldığından, noktalar birbirine bağlanırken bu kısıtın kontrol edilmesi gerekmez. Öncelikle Denklem (3.1)'de verilen tasarruf hesaplama formülü yardımıyla her rotanın tasarruf değerleri elde edilmiş ve tasarruf değerlerine göre rotalar oluşturulmuştur.

Küme 1'de yer alan noktalara ait ikili tasarruf değerleri Tablo 4.5'de verilmiştir. Bu değerlere göre tasarruflar büyükten küçüğe sıralanmış ve büyük tasarruftan başlanılarak sırasıyla uygun çiftler rotaya atanarak başlangıç çözümü elde edilmiştir. Şekil 4.3'de oluşan rota görülebilir.

Tablo 4.5. Küme 1 için noktaların tasarruf değerleri ve rotalanması

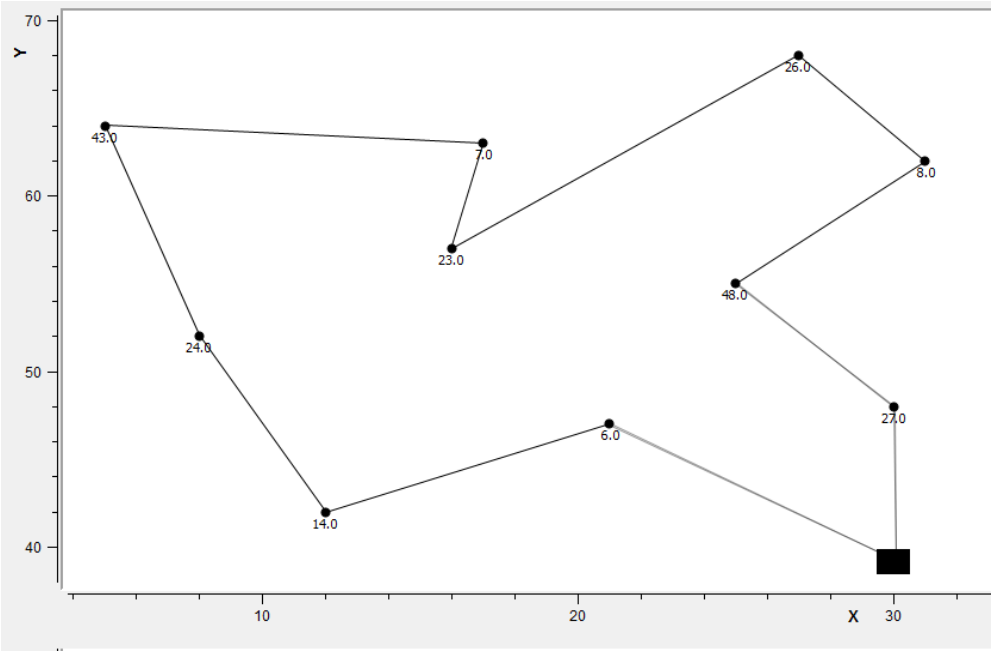
1. Küme									
	7	8	14	23	24	26	27	43	48
6	21,33	15,40	19,22	22,24	22,53	17,72	10,35	22,71	18,27
7		34,41	22,94	43,40	37,27	42,36	14,57	49,03	30,92
8			12,55	28,23	22,00	42,97	15,99	30,60	26,52
14				24,61	32,40	16,25	7,14	29,68	15,54
23					37,65	34,63	13,38	43,64	28,61
24						28,38	10,70	47,35	23,61
26							15,94	40,46	30,82
27								12,97	15,21
43									28,54

Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)
1	**7-43**	9	7-24	17	23-48	25	6-43	33	14-26
2	**24-43**	10	**23-26**	18	43-48	26	6-24	34	8-27
3	23-43	11	7-8	19	24-26	27	6-23	35	26-27
4	**7-23**	12	**14-24**	20	8-23	28	8-24	36	14-48
5	**8-26**	13	7-48	21	**8-48**	29	6-7	37	**27-48**
6	7-26	14	26-48	22	14-23	30	**6-14**	38	
7	26-43	15	8-43	23	24-48	31	6-48	39	
8	23-24	16	14-43	24	7-14	32	6-26	40	

Küme 1'deki talep noktalarının rotalanması için kullanılan yöntemler, geriye kalan tüm kümeler için uygulanmıştır. Diğer kümelere ait tablolar ve resimler çalışmanın ekinde paylaşılmıştır.

Tüm kümeler için başlangıç rotaları oluştuktan sonra toplam tur uzunluğu 542,682 olarak gerçekleşmiştir (Tablo 4.6). E-n51-k5 KKARP örneği, literatürde daha önce yapılan çalışmalarda algoritmaların test edilmesi için kullanılmış ve optimum çözümün 524,61 olduğu bulunmuştur [91]. Hiçbir iyileştirme yapmadan, önerilen BCO ile kümele yapılıp tasarruf algoritması ile optimum çözüme % 3,44 oranında yaklaşılmıştır. Bu değer Denklem (4.7)'de verilen formül ile hesaplanmıştır.

$$yaklaşma\ oranı = \frac{elde\ edilen\ çözüm - optimum\ çözüm}{optimum\ çözüm} \tag{4.7}$$



Şekil 4.3. Küme 1 ‘de tasarruf algoritması ve oluşturulan rota

Elde edilen rotalara λ-opt yerel arama sezgiseli uygulanırsa rotalardan bazılarında iyileşmeler olduğu görülmüştür. Örneğin, küme 1’de (43,7) ve (23,26) bağlantıları kopartılarak (43,23) ve (7,26) olarak değiştirilirse eski tur uzunluğu 111,551 değerinden 108,171’e düşmektedir. Benzer bir iyileşme küme 2’de de gerçekleşmiş rota uzunluğu 112,357 iken 110,424 değerine düşmüştür. λ-opt yerel arama sezgiseli ile elde edilen iyileştirilmiş turlar Tablo 4.7’da verilmiştir. Rotalar üzerinde yapılan iyileştirmelerle toplam tur uzunluğu 542,682’den 537,37’ e düşürülmüştür. Güncel tur uzunluğu ile optimum değere % 2,43 oranında yaklaşılmıştır. Şekil 4.4’de λ-opt yerel arama sezgiselinin uygulanması ile elde edilen 1. kümeye ait güncellenmiş rota verilmiştir.

Tablo 4.6. Rotalar ve tur uzunlukları

Küme No	Tur Uzunluğu	Rota
1	111,551	D-6-14-24-43-7-23-26-8-48-27-D
2	112,357	D-11-16-9-50-21-34-30-39-10-49-38-D
3	115,519	D-1-22-31-28-3-36-35-20-29-2-32-D
4	90,231	D-46-12-5-33-45-15-44-37-17-47-D
5	113,025	D-4-42-19-40-41-13-25-18-D
toplam uzunluk	542,682	

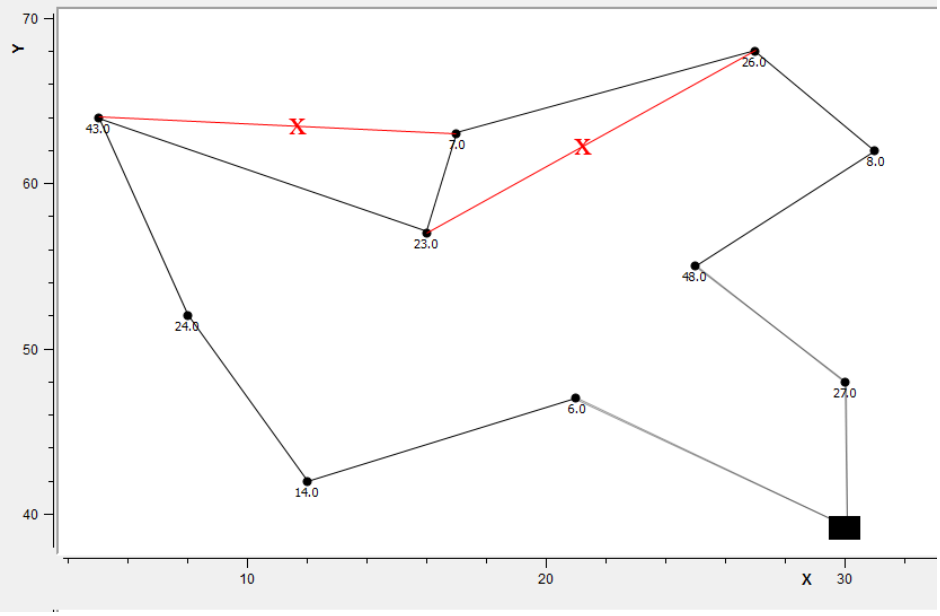
Tablo 4.7. λ-opt ile iyileştirilmiş rotalar ve tur uzunlukları

Küme No	Tur Uzunluğu	Rota
1	108,172	D-6-14-24-43-23-7-26-8-48-27-D
2	110,424	D-11-16-50-21-34-30-39-10-49-9-38-D
3	115,519	D-1-22-31-28-3-36-35-20-29-2-32-D
4	90,231	D-46-12-5-33-45-15-44-37-17-47-D
5	113,025	D-4-42-19-40-41-13-25-18-D
toplam uzunluk	537,371	

Şekil 4.4 incelendiğinde iyileştirme aşamasında uygulanan yöntemin 2-opt yöntemi olduğu anlaşılmaktadır. İki adet nokta çifti arasındaki bağlantı kaldırılmış ve çaprazlanarak tur yeniden oluşturulmuştur. 2-opt algoritması, rassal olarak kaldırılacak bağlantıları seçmesi sebebiyle verilen çözüme hemen ulaşabileceği gibi uzun süren bir aramanın sonunda da ulaşabilir. Aramanın rassal olması nedeniyle bazı sezgisel algoritmalar aynı problem üzerinde her çalıştırıldıklarında farklı sonuç verebilirler. Bu yüzden bir sezgisel algoritma ile problemler çözülürken algoritma bir defadan fazla çalıştırılarak, tüm denemeler boyunca elde ettiği en iyi sonuç değerlendirmeye alınır. Önerilen sezgisel yöntemde de birden fazla sezgisel algoritma kullanılmaktadır. Sezgisel algoritma ile elde edilen çözümler hemen değerlendirmeye alınıp sonraki adıma geçilmemeli, en iyi sonucun henüz bulunmamış olduğu varsayılarak sezgiseller bir defadan fazla çalıştırılmalıdır.

Sıradaki adım, elde edilen tur uzunluğunun, rotalar arasında yapılacak değişikliklerle iyileştirilmeye çalışılmasıdır. Bu noktaya kadar noktalar kümelenmiş, başlangıç turu kurulmuş ve rota üzerinde yerel iyileştirmeler yapılmıştır. Sıradaki adım rotalar arasında takaslar yapılmasıyla daha iyi çözümlerin olup olmadığının araştırılmasıdır.

Komşu çözümlerin aranmasına başlamadan önce bazı algoritmanın adımlarının anlatıldığı bölümde bahsi geçen parametrelerin belirlenmesi gereklidir. Örnek problem için aramaya başlama stratejisini, aitlik derecesi 0,4 değerinden az olan noktaların komşu kümelere atanması ile komşu çözümlerin test edilmesi olarak belirleyelim. Aramaya başlandıktan sonra rotalar arasında yapılacak takaslarda aitlik değerleri en düşük olanlar öncelikli olarak tercih edilsin ve elli defa arka arkaya gittikçe kötüleşen çözüm üretildiğinde arama sonlandırılsın.



Şekil 4.4. Küme 1'de λ-opt algoritması ve geliştirilen rota

Öncelikle her kümede aitliği $f \leq 0{,}4$ olan noktaları belirlenmeli ve bu noktalar seçim için hafızada tutulmalıdır. Tablo 4.8'de talep noktaları verilmiştir. Bu noktalardan bir tanesi seçilerek (rassal veya sırayla seçilebilir) komşuluk arama sezgiseli başlatılmıştır.

Tablo 4.8. Aitlik değerleri $f \leq 0{,}4$ olan talep noktaları

Küme No	Talep Noktası	Aitlik
1	-	-
2	-	-
3	1 - 32	0,357 – 0,259
4	46	0,301
5	25	0,389

Bu örnek için aitlik değeri 0,389 olan nokta seçilmiştir. 25. noktanın kümelere olan aitlik değerlerin sırasıyla; 0,323, 0,073, 0,061, 0,155, 0,389 olduğu görülmüştür. 25. nokta, 5. kümeden sonra 0,323 aitlik derecesi ile 1. kümeye aittir ve 5. kümeden 1. kümeye aktarılır. 1. kümenin talepler toplamı 25. noktasının eklenmesiyle 28 birim yükselmiştir. Şimdi kapasite kısıtını aşmış olan 2. küme incelenir ve aitliği en düşük üyeleri tespit edilir. 2. kümeye aitliği en düşük olan noktalar sırasıyla 8 (23), 27 (15) ve 26 (7)'dır. Parantez içerisinde verilen değerler talep miktarlarıdır. 8. nokta ikinci sırada aitliği yüksek olduğu 3. kümeye atanırsa Toplam talep 23 birim azalmakta ama yeterli olmamaktadır. Aday noktalar arasından 3. kümeye aitliği diğer yüksek

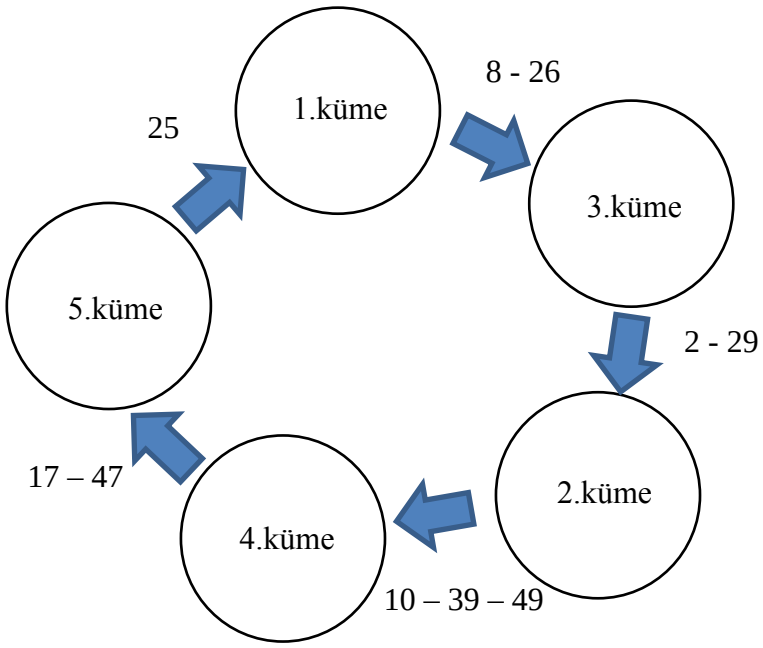
olan nokta 26. noktadır. 26. noktanın da 3. kümeye atanmasıyla 5. kümenin toplam talep miktarı 158'e düşer. 8 ve 26 noktaları 3. kümeye aktarıldıklarından toplam talep 30 birim yükselmiş ve 160 olan araç kapasite sınırını aşmıştır. Önceki adımlardaki gibi aynı yöntem tekrarlanarak en düşük aitlik derecesine sahip olan noktalar belirlenir ve toplam talep 160 birimin altına düşene kadar aitliklerinin yüksek olduğu sonraki kümeye aktarılırlar. 3. kümede en düşük aitlik değerine sahip olan noktalar 2, 29 ve 32 dir. Bunların içerisinde komşu küme olan 2. kümeye en yüksek değerle ait olan noktalar sırasıyla 2, 29 ve 32 olduğundan 2 ve 29 numaralı noktalar 2. kümeye aktarılmış ve toplam talep 160 değerinin altına düşürülmüştür. Bu yöntem aynı mantıkla iteratif bir şekilde algoritmanın başladığı 5. kümeye kadar devam ettirilmiştir. Kümelerin tamamında talepler toplamı en fazla 160 birim olduğunda bulunan çözüm komşu çözüm olarak kabul edilmiş ve rotalar güncellenmiştir. Şekil 4.5'de gerçekleşen nokta takası görsel olarak ifade edilmiş ve Tablo 4.9'da her kümeden çıkan ve dâhil olan noktalar ile toplam talepteki değişim verilmiştir. Elde edilen yeni kümeler, bir komşu çözümdür. Bu adımdan sonra yeni noktaların rota içerisinde nereye yerleştirilecekleri sıralı ekleme sezgiseli yardımıyla belirlenmiştir.

Tablo 4.9. Kümelerden çıkan ve dâhil olan noktalar

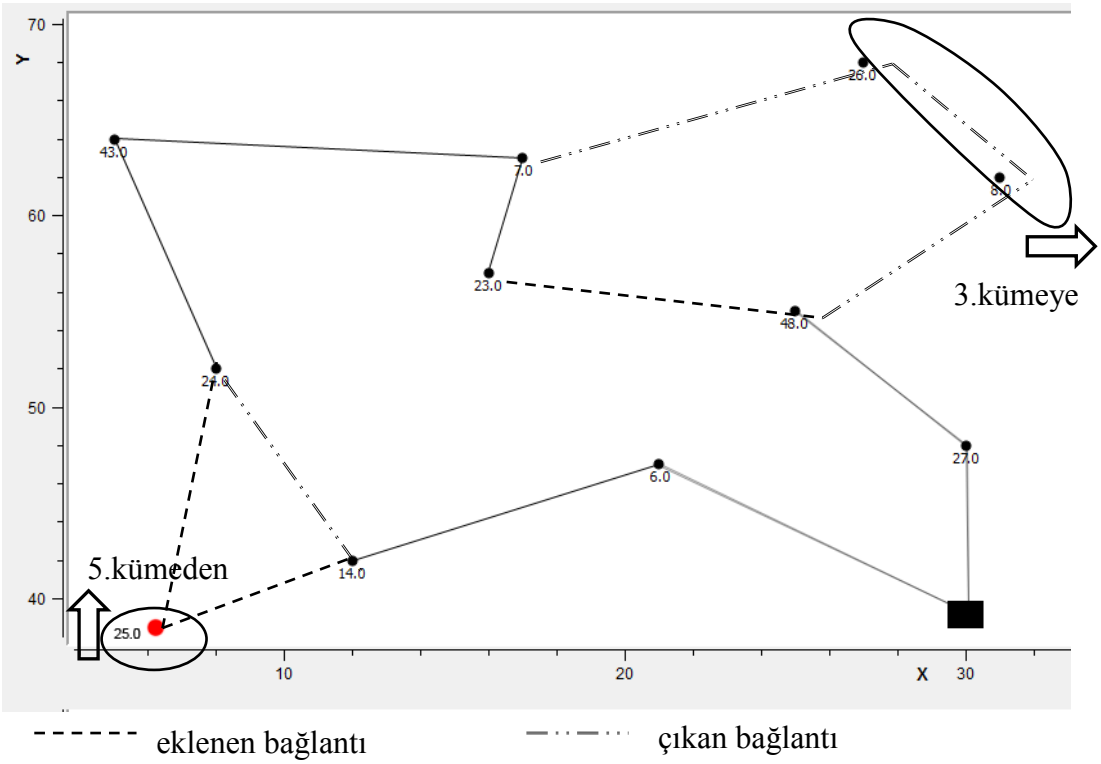
Küme No	Çıkan Noktalar	Dâhil olan Noktalar	İlk Durum	Talepteki Değişim	Son Durum
1	8, 26	25	154	-23-7+28	152
2	10, 39, 49	2, 29	160	-5-14-18+30+6	159
3	2, 29	8, 26	155	-30-6+23+7	149
4	17, 47	10, 39, 49	151	-3-25+5+14+18	160
5	25	17, 47	157	-28+3+25	157

Denklem (3.3)'de verilmiş olan formülün kullanılmasıyla tasarruf değerini minimum olduğu nokta çiftleri tespit edilmiştir. Denklemdeki λ değeri 1 olarak kabul edilmiştir. Şekil 4.6'da 1. kümede yapılan işlemler özetlenmiştir. Komşu çözümün elde edilmesiyle noktalar yer değiştirmiştir. Doğal olarak çıkan noktaların bağlantılarının koparılması kümeye dâhil olan noktaların bağlantılarının ise eklenmesi gereklidir. 25. noktanın hesaplanan tasarruf değeri (14,24) nokta çifti için 6,40 + 14,04 - 1 x 10,77 = 9,67 olarak bulunur. Şekilde görüldüğü gibi 25. noktaya en yakın olan nokta çiftidirler. Diğer tüm ikili çiftlerle yapılacak tasarruf değerleri hesaplansa bile minimum değer (25, 14, 24) arasındaki tasarruf olacaktır. Bu nedenle

25. nokta bu iki noktanın arasına eklenmiş ve rota (14-25-24) olarak güncellenmiştir. Ayrıca kümeden ayrılan 8 ve 26. noktaların bağlantısı koparılmış 48. nokta ise 23. noktaya bağlanmıştır. Böylece yeni bir rota elde edilmiş olur. Tüm kümelerde aynı işlemlerin yapılmasıyla komşu çözümün toplam rota uzunluğu hesaplanmış olacaktır.



Şekil 4.5. Komşu çözümün aranması ve rotalar arası nokta takası



Şekil 4.6. Küme 1'de rotanın güncellenmesi

1. küme haricindeki kümelere ait işlemleri özetleyen şekiller çalışmanın sonunda ekler bölümünde verilmiştir. Komşu çözüm ile oluşan rotalar ve rota uzunlukları aşağıdaki gibi gerçekleşmiştir ve 524,61 değeri elde edilerek literatürdeki çalışmalarda optimum olduğu ispatlanmış olan çözüme ulaşılmıştır [91].

Tablo 4.10. Komşu çözüme ait rotalar ve tur uzunlukları

Küme No	Tur Uzunluğu	Rota
1	98,451	D-27-48-23-7-43-24-25-14-6-D
2	99,333	D-38-9-30-34-50-16-21-29-2-11-D
3	118,519	D-32-1-22-20-35-36-3-28-31-26-8-D
4	99,251	D-12-37-44-15-45-33-39-10-49-5-46-D
5	109,056	D-18-13-41-40-19-42-17-4-47-D
toplam uzunluk	524,610	

4.4. Yöntemin Farklı Problemlere Uygulanması ve Bulgular

Önerilen yöntem Christofides and Eilon tarafından geliştirilen deneysel problemlere uygulanarak test edilmiştir. Test problemleri, literatürde E-n22-k4, E-n23-k3, E-n30-k4, E-n33-k4 ve E-n51-k5 olarak yer almakta ve 21-50 nokta arası şehir ile 3 ile 5 arasında araç içermektedirler. Tablo 4.11'de önerilen yöntemin test problemlerine uygulanması ile elde edilen sonuçlar verilmiştir. Tabloda verilen en iyi sonuç değerleri daha önce yapılmış çalışmalarda elde edilmişlerdir.

Tablo 4.11. Test problemlerinden elde edilen sonuçlar [20, 100]

Problem	Nokta Sayısı	Araç Sayısı	En iyi Sonuç	Elde Edilen Sonuç	Sapma
E-n22-k4	21	4	375,27	375,27	0
E-n23-k3	22	3	568,56	568,56	0
E-n30-k4	29	4	505,01	505,01	0
E-n33-k4	32	4	837,67	880,29	0,0509
E-n51-k5	50	5	524,61	524,61	0

E-n22-k4 probleminde takaslanacak noktaları belirleyecek olan f parametresi ($f \leq 0,6$) olarak belirlendiği zaman en iyi sonuca ulaşılmıştır. E-n22-k4 probleminde her bir aracın kapasitesi 6000 birimdir. Bulanık c-ortalama işlemi sonrasında elde edilen kümeler ve talep değerleri aşağıda verilmiştir.

Tablo 4.12. E-n22-k4'da BCO sonrası kümeler

Küme No	Talepler	Küme Elemanları
1	5200	1-2-5-7-9
2	5900	12-13-14-15-16
3	6900	17-18-19-20-21
4	4500	3-4-6-8-10-11
toplam talep	22500	

Tablo 4.12'de görüldüğü üzere 3. kümenin talepleri toplamı araç kapasitesini aşmaktadır. Kümelerin taleplerinin dengelenmesi yani araç kapasitesinden fazla olmaması gereklidir. Elde edilen yeni kümeler Tablo 4.13'de verilmiştir.

Tablo 4.13. E-n22-k4'da küme taleplerinin dengelenmesi

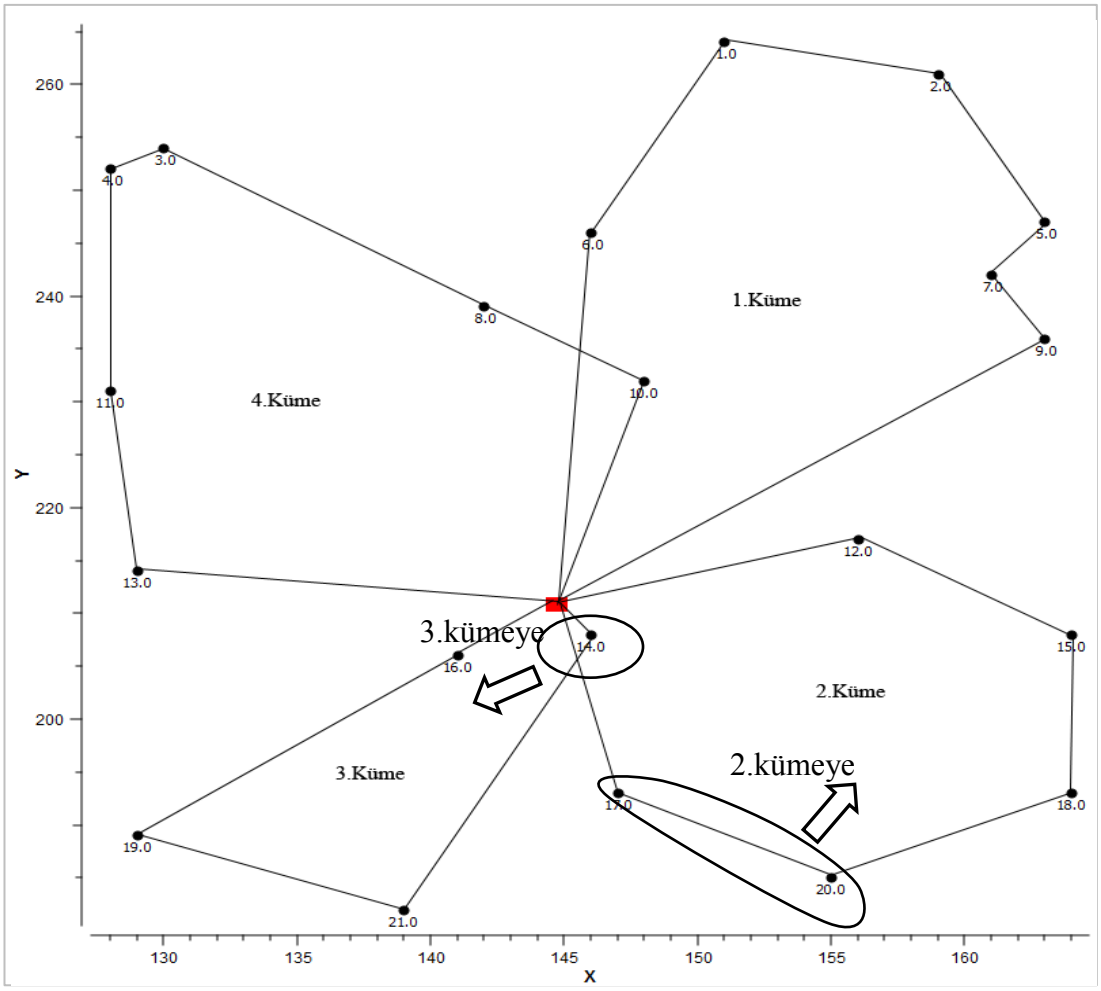
Küme No	Talepler	Küme Elemanları
1	5600	1-2-5-6-7-9
2	5500	12-14-15-16-18
3	6000	17-19-20-21
4	5400	3-4-8-10-11-13
toplam talep	22500	

Kümeler arasında noktaların takaslanmasının ardından rotalama ve iyileştirme algoritması ile komşu çözümler aranmış ve 375,27 değerine ulaşılmıştır. E-n22-k5 probleminin çözümü Şekil 4.7'de özetlenmiştir. 14,17 ve 20. noktalarının rota geliştirme aşamasında farklı kümelere aktarılmasıyla daha iyi bir çözüm üretilmiştir.

E-n23-k3 problemi, 22 talep noktası ve üç araç içeren başka bir test problemidir. Bu problemde BCO kümeleme işlemi ve küme taleplerinin dengelenmesi adımları sonrasında oluşan kümeler Tablo 4.14 ve Tablo 4.15'de verilmiştir. Problemde araç kapasiteleri 4500 olarak verilmiştir. 1. kümenin talepleri toplamının araç kapasitesini aşması nedeniyle kümelerin talepleri 4500 değerinin altında olacak şekilde nokta takasları gerçekleşmiştir.

Tablo 4.14. E-n23-k3'da BCO sonrası kümeler

Küme No	Talepler	Küme Elemanları
1	7450	4-5-7-8-9-10-11-12-13-21
2	1295	18-19-20-22
3	1444	1-2-3-6-14-15-16-17
toplam talep	10189	

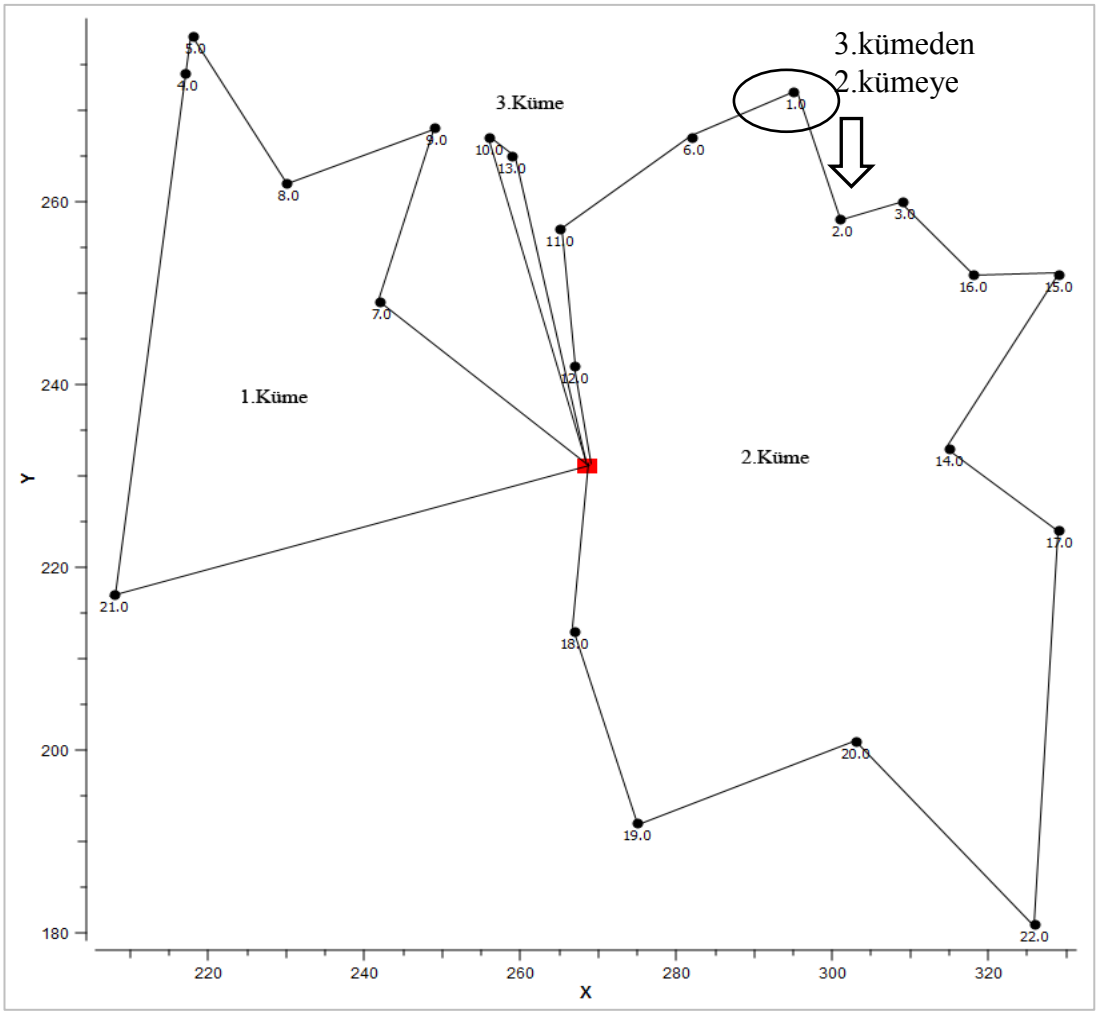


Şekil 4.7. E-n22-k4 test probleminin çözümü

Tablo 4.15. E-n23-k3'da küme taleplerinin dengelenmesi

Küme No	Talepler	Küme Elemanları
1	2575	4-5-7-8-9-21
2	3139	2-3-6-11-12-14-15-16-17-18-19-20-22
3	4475	1-10-13
toplam talep	10189	

E-n23-k3 probleminde *f* değeri 1 olarak ($f \leq 1$) olarak belirlenmek zorunda kalınmıştır. Aksi halde kümelerin taleplerini 4500 değerinin altına indirmek mümkün olmamaktadır. Bu durumun nedeni 10. noktanın talebinin 4100 gibi yüksek bir değer olmasıdır. Talep kısıtının sağlanabilmesi için aitlik değeri 1'e yakın olan noktaların dahi komşu kümelere aktarılması gerekmiştir. Takas işlemi sonrasında iyileştirme adımında sadece 1. noktanın 3. kümeden 2. kümeye aktarılması ile daha iyi bir çözüme ulaşılmıştır. Şekil 4.8'de 568,56 değerini veren rota gösterilmiştir.

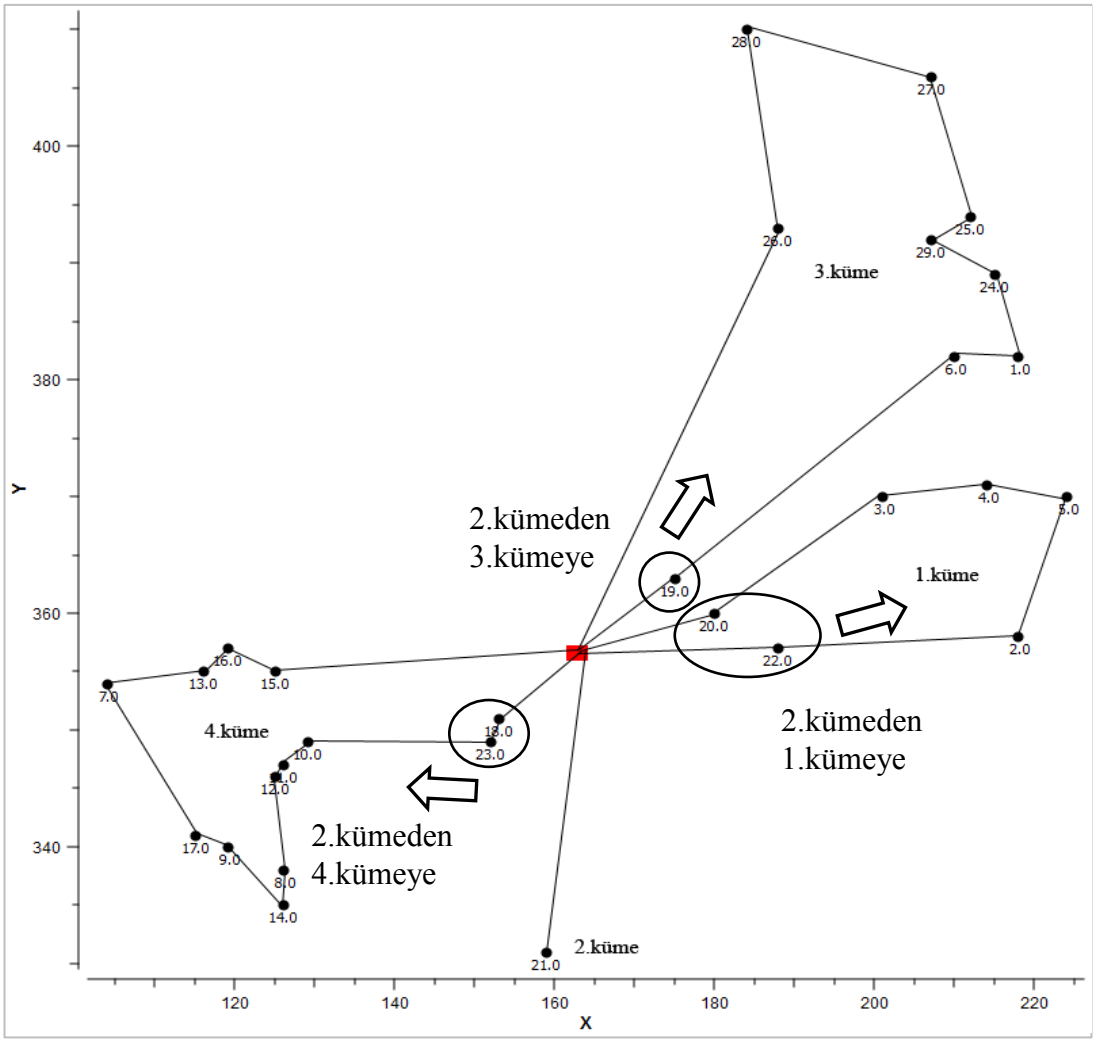


Şekil 4.8. E-n23-k3 test probleminin çözümü

Yöntemin test edildiği bir diğer problem ise E-n30-k4'tür. Bu problemde 29 talep noktası ve her biri 4500 kapasiteye sahip dört adet araç bulunmaktadır. Tablo 4.16'da BCO algoritması ile elde edilen kümeler verilmiştir. Kümelerin talepler toplamı 4500 değerini aşmaması nedeniyle taleplerin dengelenmesi adımına gerek duyulmamış ve doğrudan rotalama ve iyileştirme algoritmasına geçilmiştir. İyileştirme algoritması sonrasında elde edilen rota Şekil 4.9'da gösterilmiştir.

Tablo 4.16. E-n30-k4'de BCO sonrası kümeler

Küme No	Talepler	Küme Elemanları
1	3975	1-2-3-4-5-6
2	2750	18-19-20-21-22-23
3	2850	24-25-26-27-28-29
4	3175	7-8-9-10-11-12-13-14-15-16-17
toplam talep	12750	



Şekil 4.9. E-n30-k4 test probleminin çözümü

E-n33-k4 problemine önerilen yöntem uygulanmış ve literatürde elde edilen sonuçlara ulaşmak mümkün olmamıştır. Problemde 32 nokta ve her biri 8000 kapasiteli, dört araç bulunmaktadır. Tablo 4.17 ve Tablo 4.18'de BCO ve takas işlemleri sonrasında elde edilen kümeler verilmiştir.

Tablo 4.17. E-n33-k4'de BCO sonrası kümeler

Küme No	Talepler	Küme Elemanları
1	7750	1-13-14-15-17-30-31
2	4300	18-19-20-21-22-23-24
3	8200	16-25-26-27-28-29
4	9120	2-3-4-5-6-7-8-9-10-11-12-32
toplam talep	29370	

Tablo 4.18. E-n33-k4'de küme taleplerinin dengelenmesi

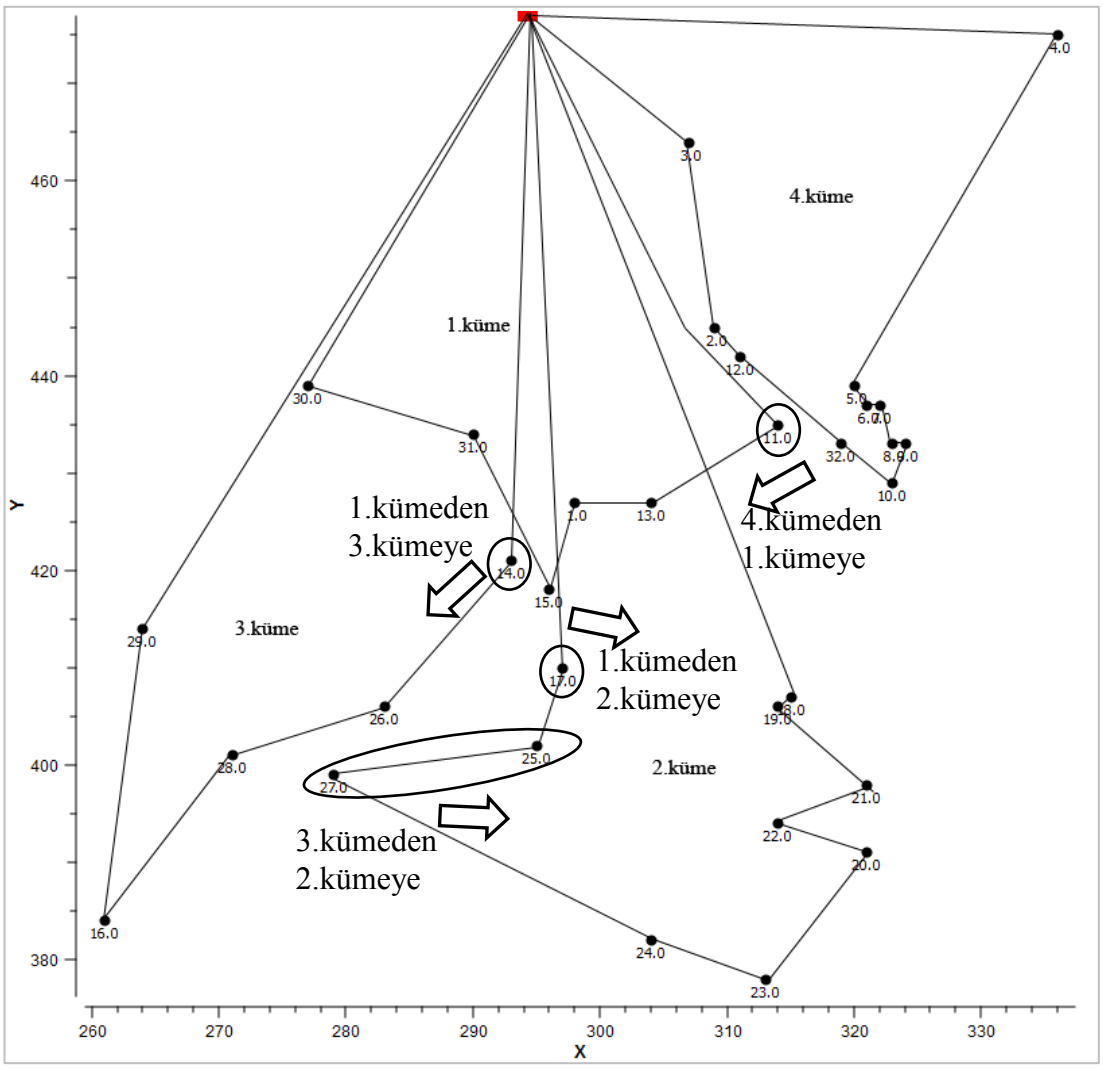
Küme No	Talepler	Küme Elemanları
1	7100	1-11-13-15-30-31
2	6850	17-18-19-20-21-22-23-24-25-27
3	7800	14-16-26-28-29
4	7620	2-3-4-5-6-7-8-9-10-12-32
toplam talep	29370	

E-n33-k4 problemi için optimum çözüm 837,67'dir. Önerilen algoritma ile elde edilen en iyi değer 880,29 olmuştur (Şekil 4.10). Bu değer optimum sonuçtan yaklaşık % 5.1 oranında sapmaktadır. Önerilen yöntem, başlangıç çözümünün noktaların gruplandırılarak elde edilmesinin ardından, özellikle depo merkezlerinin nokta kümelerinin ortasında (merkezinde) olmadığı durumlarda etkili sonuçlar üretememektedir. Yöntem, depo merkezinin kümelerin arasında olduğu örneklerde daha iyi sonuçlar vermiştir.

Bu zayıflığın daha net anlaşılabilmesi için Şekil 4.11'de bir örnek verilmiştir. Örnekte depo merkezi sol üst köşede yer almaktadır. Noktalar üç kümeye dâhil olacak şekilde dağılmıştır. 9-10-11, 1-2-3-4 ve 5-6-7-8 nokta gruplarının aynı kümeyi paylaştıklarını ve ilk kümedeki 10-11 noktalarının araç kapasitesini doldurduğunu varsayalım. Bu durumda 9. noktanın başka bir kümeye aktarılması gerekecektir. Yöntemin, aitlik derecesine göre kümeler arası takas yapılması mantığından dolayı 9. noktanın aktarılacağı küme 1-2-3-4 noktalarının olduğu 2. küme olacaktır. Çünkü 3. kümenin merkezi 9. noktaya çok uzaktır ve aitlik derecesi 0 değerine oldukça yakın olacaktır. Hâlbuki 9. nokta 3. kümeye uzak olmasına rağmen, kümeye servis veren aracın rotası noktanın yakınından geçmektedir. 9. noktanın 3. kümeye aktarılmasıyla iyi bir çözüm elde edebilmek mümkün iken önerilen yöntemle dayandığı aitlik mantığı nedeniyle bu çözüme ulaşılamayabilir. Depo ile küme merkezleri arasında konumlanmış olan küme sayısının artması halinde problemin çözülmesi daha da zorlaşacaktır.

Söz konusu zayıflık, çözümün noktaların sadece birbirlerine olan uzaklıklarına göre değerlendirilerek elde edilmeye çalışılmasından kaynaklanmaktadır. Literatürdeki birçok sezgisel yöntem gibi öncelikli olarak noktaların arasındaki uzaklıklar dikkate alınmaktadır. Hâlbuki noktalardan elde edilen rotanın problemin bütünü içerisindeki

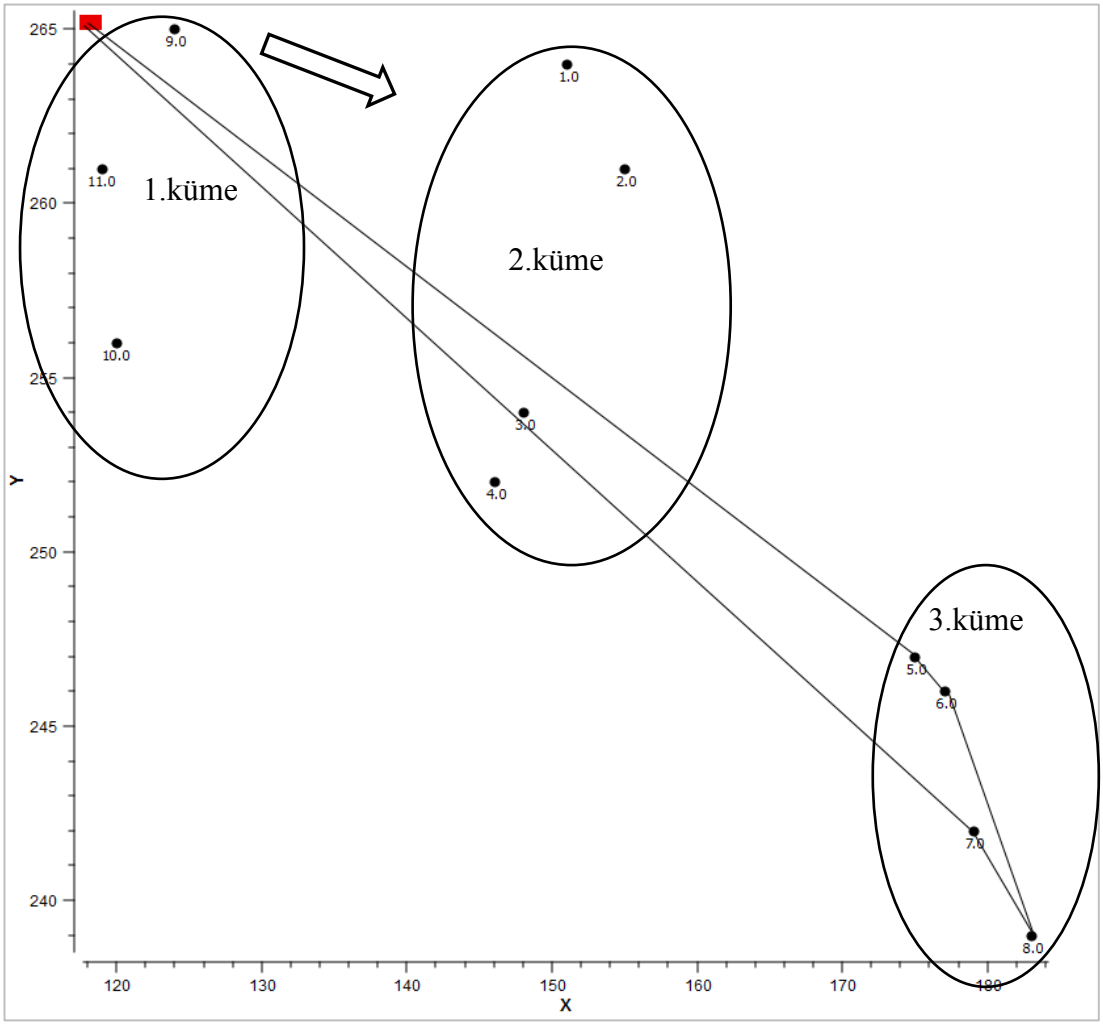
konumu da dikkate alınmalıdır. Tasarruf algoritması gibi sezgiseller öncelikli olarak mesafeyi göz önünde bulundururlar, ancak depo ile noktaların konumu irdeleyen Süpürme algoritması, Taillard'ın paralel iteratif arama metodu [92] benzeri yöntemlerde önerilmiştir.



Şekil 4.10. E-n33-k4 test probleminin çözümü

Süpürme algoritmasında, tepe noktası depo olacak şekilde belirli bir açıyla taranan alanda yer alan noktalar, aralarındaki mesafeden ziyade aynı alan içerisinde yer almalarından ve olası rotaya en yakın noktalar olmalarından dolayı kümelenmektedir. Örnekte izah edilen dezavantajın giderilmesi için hem noktalar arası mesafeleri hem de rotaların problem içerisindeki konumunun değerlendirildiği bir yöntem daha iyi sonuçlar üretmeye açıktır. Örneğin, yöntemle mesafelere göre elde edilen aitlik değerleri gibi, depo ile aynı doğrultuda yer alan fakat farklı

kümelerdeki noktalar içinde ikinci bir aitlik değeri daha tanımlanabilir. İyileştirme adımında da komşu çözümler, hem mesafe aitliklerine hem de rotaya yakın olma durumlarına göre üretilerek daha iyi sonuçlara ulaşmak mümkün olabilir.



Şekil 4.11. Önerilen yöntemin zayıf noktası

Gelecekte yapılacak çalışmada, önerilen yöntemin dezavantajını ortadan kaldırmak için rotaların diğer noktalara olan uzaklıklarının da dikkate alındığı bir kontrol prosedürü yönteme dâhil edilecektir. Böylece depo ile rotalanmış küme arasında yer alan tüm diğer noktaların bulanık aitlik değerleri düşük bile olsa rotaya eklenmesi sağlanmış olacaktır.

Öte yandan, BCO ile kümeleme yapılmasından dolayı, kümeleme algoritmalarının bazı dezavantajları da yönteme yansıyabilir. Özellikle noktalar arası uzaklıkların eşit olduğu verilerin gruplandırılmasında tutarsız çözümler elde edilebilir. Ancak gerçek hayatta yerleşim birimleri eşit uzaklıklarda dağılmamakla birlikte, coğrafi kısıtlar

nedeniyle belirli bölgelerde yığılmalar olmaktadır. Bu nedenle söz konusu zayıflık, önerilen yöntemin gerçek hayat problemlerinde kullanılmasının önünde ciddi bir engel teşkil etmeyecektir. Ayrıca yöntem, sadece talep noktalarının x-y koordinat düzleminde olduğu test problemlerine uygulanmıştır. BCO ile aitlik değerlerinin elde edilmesinde yine noktaların koordinat değerleri kullanılmıştır. Önerilen yöntemin koordinat düzleminde ifade edilen noktalara ait rotaları tespit edebilmektedir. Ancak gerçek hayat problemlerinin çözümünde kullanılabilmesi için gelecekte üzerinde çalışmalar yapılacaktır.

5. SONUÇLAR ve ÖNERİLER

Teknolojinin gelişmesi ve sistemlerin yönetilmesi gittikçe karmaşık bir hale geldikçe optimizasyon hayatımızın vazgeçilmez bir parçası haline gelmektedir. İşletmelerde geride bıraktığımız yüzyılın son çeyreğinde yer verilmeye başlanılan optimizasyon ve optimizasyon teknikleri günümüze kadar ihtiyaçlar doğrultusunda birtakım değişikliklere uğramışlardır. Optimizasyon problemleri ile ilgilenilen erken dönemlerde problem boyutları nispeten küçük olması nedeniyle kesin sonuçlar elde edilmeye çalışılmış ve geliştirilen algoritmalar dönem ihtiyaçlarına paralel olarak kesin çözüm algoritmaları bu yıllarda geliştirilmiştir. Zamanla problemlerin büyümesi ve optimizasyonun işletmelere sağladığı faydaların fark edilmesiyle problemlerin karmaşıklığı ve çeşitliliği artmıştır. Karmaşık problemlerin, kesin algoritmalarla çözülemeyeceği fark edildiğinde yeni bir arayışa girilmiş ve klasik sezgiseller adını verdiğimiz algoritmalar hayatımıza girmiştir. Klasik sezgisel algoritmalar problemlere kısa zamanda kabul edilebilir yaklaşık çözümler üreterek karmaşıklık-çözüm zaman arasında bir denge kurmuşlardır. Sezgisel algoritmaların bu başarısı ile optimum çözümlerin arandığı çözüm uzayının daha sistematik bir şekilde aranmasıyla daha iyi sonuçlara ulaşılabileceği fikri ortaya atılmış ve ileri sezgiseller (heuristics) literatüre kazandırılmıştır. Genellikle bir esin kaynağı olan ileri sezgiseller, karmaşık kombinatoryal problemlere iyi sonuçlar önermektedirler. En çok bilinen ve kullanılanları; Tabu Arama, Karınca Kolonisi Optimizasyonu, Genetik Algoritma, Tavlama Benzetimi'dir. Son yıllarda sezgisel algoritmaların beraber kullanılmasıyla melez yöntemlerde geliştirilmiştir.

Araçların rotalanması problemi, iyi bilinen eski ve çözümü zor bir problemdir. Araçların en uygun yollar üzerinden rotalanarak ürünlerin lojistiğinin yapılması maliyetlerde ciddi bir düşüşe neden olacağından, üzerinde önemle durulmaktadır. Yıllardır üzerinde çalışılmış olmasına rağmen büyük boyutlu problemlere önerilen çözümler tatmin edici olmayabilmektedir. Bunun nedeni problemin çok farklı kısıtlara bağlı olması ve yapısı nedeniyle NP-Zor olarak tanımlanan yüksek karmaşıklığa sahip olmasıdır. Kesin çözüm yöntemlerinin belirli sınırlar dışında

çözüm üretemiyor olmasından dolayı sezgiseller araç rotalama problemlerinde sıklıkla kullanılmaktadırlar. Klasik sezgisel algoritma türlerinden, tasarruf algoritması, tasarruf algoritmasının bir parametreye bağlı geliştirilmiş türleri, süpürme algoritması, petal algoritması, sıralı ekleme sezgiseli en çok bilinenlerdir. Metasezgiseller sınıfında ise daha çok Tabu Arama kullanılmakla birlikte, Genetik algoritma, Karınca Kolonisi ve Tavlama Benzetimi yöntemleriyle problemler çözülmüştür. Bunların dışında son yıllarda ortaya çıkan Memetic algoritma ve Parçacık Sürü Optimizasyonu ile yapılan çalışmalarda mevcuttur.

Sezgisel algoritmaların ARP'ni çözmedeki başarısı nedeniyle bu çalışmada yeni bir çözüm yaklaşımı önerilmiştir. Önerilen yöntem ile araç rotalama problemleri çözümü daha kolay olan gezgin satıcı problemlerine dönüştürerek her araç için rotalama yapılmaktadır. Ardından rotalar arasında bir komşuluk arama algoritması ile komşu çözümler taranarak toplam rota uzunluğu optimize edilmektedir. Problemi gezgin satıcı problemine indirgemek için bulanık c-ortalama algoritması kullanılmıştır. Bu algoritmanın kullanılmasının nedeni, literatürdeki çalışmalarda araçların sadece bir kümeye ait olacak şekilde gruplandırılmış olmalarıdır. Aslında araçlar, diğer kümelere olan uzaklıklarına bağlı olarak her kümenin elemanı olabilmektedirler. BCO algoritması ile birbirine uzak noktaların aynı kümeye olan aitlik değerleri arasında oldukça büyük fark oluşmakta ve noktaların aynı rotada yer almaları zorlaşmaktadır. Ayrıca, iki grubun arasında kalan bir nokta sadece bir kümeye ait olmak yerine eşit aitlik değerleriyle iki kümeye de dâhil olabilmektedir. Çalışmada bulanık kümelerin bu avantajından yararlanılarak çözüm uzayında arama yapılacak bölgenin daraltılması sağlanmıştır. Ardından yine bulanık değerler yardımıyla noktalar, rotalar arasında değiştirilerek komşu çözümler elde edilmiştir. Elde edilen komşu çözümler mevcut çözümlerden iyi olduğunda mevcut çözüm güncellenmiştir.

Önerilen yöntem, Christofides ve Eilon tarafından geliştirilen test problemleri ile test edilmiş ve kümeleme adımında optimum sonuca belli oranda yakın sonuçlar elde edilmiştir. Ardından komşu çözümlerin aranmasıyla sonuçlar iyileştirilmiş ve bazı problemlerin optimum sonuçlarına ulaşılmıştır. Ancak yöntemin bir dezavantajının olduğu görülmüştür. Bu dezavantajın, sadece noktalar arası uzaklıkların dikkate alınması ve bazı noktaların araç rotasına yakın olduğu halde, küme aitlik değerlerinin düşük olması nedeniyle rotaya eklenememelerinden kaynaklandığı tespit edilmiştir.

KAYNAKLAR

[1] Ceran Y., Alagöz A., Lojistik maliyet yönetimi: lojistik maliyetler ve lojistik maliyet muhasebesi, *Yönetim Bilimleri Dergisi*, 2007, **5**, 153-175.

[2] Özkan P., Araç rotalama ve çizelgeleme, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2006, 222134.

[3] Bernhard T., *Information systems in logistics and transportation*, 2th ed., Pergamon Press, Oxford; Tarrytown, N.Y, 1997.

[4] Rushton A., Croucher P., Baker P., *The Handbook of Logistics and Distribution Management*, 4rd ed., Kogan Page, London, 2006.

[5] Hacırüstemoğlu R., Şakrak M., *Maliyet muhasebesinde güncel yaklaşımlar*, 1.baskı, Türkmen Kitabevi, İstanbul, 2002.

[6] Ghiani G., Laporte G., Musmanno R., *Introduction to logistics systems planning and control*, John Wiley & Sons, West Sussex, 2004.

[7] Tek Ö.B., Özgül E., *Modern pazarlama ilkeleri: uygulamalı yönetimsel yaklaşım.* 3.baskı, Birleşik Matbaacılık, İzmir, 2010.

[8] Bowersox D. J., Class D. J., Cooper, M. B., *Supply chain logistics management*, 1th ed., McGraw Hill, Boston, 2002.

[9] Applegate D., Bixby R., Chvatal V., Cook W., On the solution of traveling salesman problems, *Documenta Mathematica*, 1998, 645-656.

[10] Bakır M.A., Altunkaynak, B., *Tamsayılı programlama: teori, modeller ve algoritmalar*, 1.baskı, Nobel Yayın, Ankara, 2003.

[11] Darcan U., Stokastik araç rotalama algoritmalarının karşılaştırmalı incelenmesi, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2007, 201479.

[12] Punnen A. P., The travelling salesman problem: applications, formulations and variations, Editors: Gutin G., Punnen A.P.,*The travelling salesman problem and its variations*, 1th ed., Kluwer Academic, NewYork, 1-24, 2002.

[13] Liu Z. ve Kang L., A hybrid algorithm of n-opt and GA to solve dynamic TSP, *Grid and Cooperative Computing Lecture Notes in Computer Science*. 2004, **3033**, 1030-1033.

[14] Milestones in the Solution of TSP Instances, http://www.tsp.gatech.edu/ history/milestone.html/ (Ziyaret tarihi: 27 Nisan 2013).

[15] Toth P., Vigo D., An overview of vehicle routing problems, Editors: Toth, P., Vigo, D., *The vehicle routing problem*, SIAM, Philadelphia. 1-26, 2002.

[16] Alabaş Ç., Dengiz B., Yerel arama yöntemlerinde yöre yapısı: araç rotalama problemine bir uygulama, *24. Yöneylem Araştırması/Endüstri Mühendisliği Kongresi*, Adana, Türkiye, 16-18 Haziran 2004.

[17] Ballou R.H., *Business logistics management: planning, organizing and controlling the supply chain*, 4th ed. McGraw-Hill, 1999.

[18] Tan K.C., A Framework of Supply Chain Management Literature, *European J. of Purchasing & Supply Management*, 2001, **7**, 39–48.

[19] Laporte G., What you should know about the vehicle routing problem, *Naval Research Logistics*, 2007, **54**, 811–819.

[20] Vural E., Araç rotalama problemleri için populasyon ve komşuluk tabanlı metasezgisel bir algoritmanın tasarımı ve uygulaması, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2006, 180526.

[21] Psaraftis H. N., Dynamic vehicle routing: status and prospects, *Annals of Operations Research*, 1995, **61**, 143-164.

[22] Pillac V., Gendreau M., Guéret C., Medaglia A. L., A review of dynamic vehicle routing problems, *European Journal of Operational Research*, 2003, **225**, 1–11.

[23] Cordeau J. F., Laporte G., Savelsbergh M.W., Vigo D., Vehicle routing, Editors: Barnhart C., Laporte G., *Handbooks in Operations Research and Management Science: Transportation*, North-Holland, Amsterdam, 367-428, 2007.

[24] Bertsimas D. J., A vehicle routing problem with stochastic demand, *Operations Research*, 1992, **40**, 574-585.

[25] Larsen J., Parallelization of the vehicle routing problem with time windows, PhD. Thesis, Technical University of Denmark, Lyngby, 1999. (http://www2.imm.dtu.dk/documents/ftp/phdliste/phd62_99.pdf)

[26] Christofides N., Mingozzi A., Toth P., Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxations, *Mathematical Programming*, 1981, **20**, 255-282.

[27] Fisher M. L., Optimal solution of vehicle routing problems using minimum k-trees, *Operations Research*, 1994, **42**, 626-642.

[28] Laporte G., Mercure H., and Nobert,Y., A branch-and-bound algorithm for a class of asymmetrical vehicle routing problems, *Journal of the Operational Research Society*, 1992, **43**, 469-481.

[29] Toth P., Vigo D., Branch-and-bound algorithms for the capacitated VRP, Editors: Toth, P., Vigo, D., *The vehicle routing problem*, SIAM, Philadelphia. 29-51, 2002.

[30] Augerat P., Belenguer J. M., Benavent E., Corberan A., Naddef D., Seperating capacity constraints in the CVRP using tabu search, *European Journal of Operations Research*, 1998, **106**, 546-557.

[31] Laporte G., The vehicle routing problem: an overview of exact and approximate algorithms, *European Journal of Operation Research*, 1992, **59**, 345–358.

[32] Toth P., Vigo D., Exact solution of the vehicle routing problem, Editors: Crainic T.G., Laporte G., *Fleet Management and Logistic*, Kluwer Academic, Boston, 1-31, 1998.

[33] Laporte G., Osman I.H., Routing problems: a bibliography, *Annals of Operations Research*, 1995, **61**, 227–262.

[34] Clarke G., Wright J. W., Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 1964, **12**, 568-581.

[35] Yellow P., A computational modification to the savings method of vehicle scheduling, *Operational Research Quarterly*, 1970, **21**, 281-283.

[36] Paessens H., The savings algorithm for the vehicle routing problem, *European Journal of Operational Research*, 1988, **34**, 336-344.

[37] Altınel I.K., Öncan T., A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem, *Journal of the Operational Research Society*, 2005, **56**, 954-961.

[38] Gendreau M., Laporte G., Potvin J.Y., Metaheuristics for the capacitated VRP. Editors: Toth, P., Vigo, D., *The vehicle routing problem*, SIAM, Philadelphia. 129-154, 2002.

[39] Cordeau J. F., Laporte G., Mercier A., A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 2001, **52**, 928-936.

[40] Tarantilis C. D., Kiranoudis C. T., Bone route: an adaptive memory-based method for effective fleet management, *Annals of Operations Research*, 2002, **115**, 227-241.

[41] Tarantilis C. D., Solving the vehicle routing problem with adaptive memory programming methodology, *Computers & Operations Research*, 2005, **32**, 2309-2327.

[42] Reimann M., Doerner K., Hartl R. F., D-ants: savings based ant divide and conquer the vehicle routing problem, *Computers & Operations Research*, 2004, **31**, 563-591.

[43] Calvete H.I., Gale C., Oliveros M., Valverde S. B., A goal programming approach to vehicle routing problems with soft time windows, *European Journal of Operation Research*, 2007, **177**, 1720-1733.

[44] Desrosiers J., Dumas Y., Solomon M. M., Soumis F., Time Constrained Routing and Scheduling, Editors: Ball M. O., Magnanti T. L., Monma C. L., Nemhauser G. L., *Handbooks in Operations Research and Management Science: Network Routing*, North-Holland, Amsterdam, 35-139, 1995.

[45] Tan K. C., Lee L. H., Ou K., Artificial intelligence heuristics in solving vehicle routing problems with time window constraints, *Engineering Applications of Artificial Intelligence*, 2001, **14**, 825-837.

[46] Savelsbergh, M. W. P., Local search for routing problems with time windows, *Annals of Operations Research*, 1985, **4**, 285-305.

[47] Gökçe E. İ., A revised ant colony system approach to vehicle routing problems, Msc. Thesis, Sabancı University, Graduate School of Engineering and Natural Sciences, İstanbul, 2004, 153837.

[48] Desrochers M., Desrosiers J., Solomon M., A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research*, 1992, **40**, 342-354.

[49] Kohl, N., Madsen O. B. G., An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean Relaxation, *Operations Research*, 1997, **45**, 395-406.

[50] Cordeau J. F., Desaulniers G., Desrosiers J., Solomon M. M., Soumis F., VRP with Time Windows, Editors: Toth, P., Vigo, D., *The vehicle routing problem*, SIAM, Philadelphia. 157-193, 2002.

[51] Kallehauge B., Larsen J., Madsen O. B. G., Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 2006, **33**, 1464-1487.

[52] Karaçay G., Tersine lojistik: kavram ve işleyiş, *Çukurova Üniversitesi Sosyal Bilimler Enstitüsü Dergisi*, 2005, **14**, 317-332.

[53] Montané F. A. T., Galvão R.D., A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, Computers & Operations Research, 2006, **33**, 595-619.

[54] Min H., The multiple vehicle routing problem with simultaneous delivery and pick-up points, *Transportation Research Part A: General*, 1989, **23**, 377-386.

[55] Dethloff J., Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up, *OR Spektrum*, 2001, **23**, 79-96.

[56] Vural A. V., Çatay B., Eksioglu B., A dual GA approach to capacitated vehicle routing problem with simultaneous pick-up and deliveries, *Annual Industrial Engineering Research Conference*, Atlanta, USA, 14-18 May 2005.

[57] Ropke S., Pisinger D., A unified heuristic for vehicle routing problems with backhauls, *European Journal of Operational Research*, 2006, **171**, 750-775.

[58] Dell'Amico M., Righini G., Salani M., A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection, *Transportation Science*, 2006, **40**, 235-247.

[59] Subramanian A., Uchoa E., Ochi L.S., New lower bounds for the vehicle routing problem with simultaneous pickup and delivery, *Experimental Algorithms: Lecture Notes in Computer Science*, 2010, **6049**, 276-287.

[60] Mirabi M., Ghomi Fatemi S. M. T., Jolai F., Efficient stochastic hybrid heuristics for the multi-depot vehicle routing problem, *Robotics and Computer-Integrated Manufacturing*, 2006, **26**, 564-569.

[61] Archetti C., Savelsbergh M., Speranza M., To split or not to split: that is the question, *Transportation Research Part E: Logistics and Transportation Review*, 2008, **44**, 114-123.

[62] Nowak M., Ergun O., White C.C., An empirical study on the benefit of split loads with the pickup and delivery problem, *European Journal of Operational Research*, 2009, **198**, 734-740.

[63] Gulczynski D., Golden B., Wasil E., The split delivery vehicle routing problem with minimum delivery amounts, *Transportation Research Part E: Logistics and Transportation Review*, 2010, **46**, 612-626.

[64] Archetti C., Feillet D., Gendreau M., Speranza M. G., Complexity of the VRP and SDVRP, *Transportation Research Part C: Emerging Technologies*, 2011, **19**, 741-750.

[65] Jin M., Liu K., Bowden R. O., A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem, *International Journal of Production Economics*, 2007, **105**, 228-242.

[66] Francis P., Smilowitz K., Tzur M., The period vehicle routing problem with service choice, *Transportation Science*, 2006, **40**, 439-454.

[67] Cacchiani V., Hemmelmayr V.C., Tricoire F., A set-covering based heuristic algorithm for the periodic vehicle routing problem, *Discrete Applied Mathematics*, 2012, In Press (Available Online).

[68] Toth P., Vigo D., An exact algorithm for the vehicle routing problem with backhauls, *Transportation Science*, 1997, **31**, 372-385.

[69] Hosny M. I., Investigating heuristic and meta-heuristic algorithms for solving pickup and delivery problems, PhD Thesis, Cardiff University, School of Computer Science and Informatics, Cardiff, 2010.

[70] Başkaya Z., Avcı Öztürk B., Tamsayılı programlamada dal kesme yöntemi ve bir ekmek fabrikasında oluşturulan araç rotalama problemine uygulanması, *Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 2005, **24**, 101-114.

[71] Nagy G., Salhi S., Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries, *European Journal of Operational Research*, 2005, **162**, 126-141.

[72] Laporte G., Gendreau M., Potvin J. Y., Semet F., Classical and modern heuristics for the vehicle routing problem, *International Transactions in Operation Research*, 2000, **7**, 285-300.

[73] Cordeau J. F., Gendreau M., Laporte G., Potvin J.Y., Semet F., A guide to vehicle routing heuristics, *Journal of the Operational Research Society*, 2002, **53**, 512-522.

[74] Bramel J. Simchi-Levi D., A location based heuristic for general routing problems, *Operations Research*, 1995, **43**, 649-660.

[75] Ryan D. M., Hjorring C., Glover F., Extensions of the petal method for vehicle routing, *Journal of the Operational Research Society*, 1993, **44**, 289-296.

[76] Renaud J., Boctor F. F., Laporte G., An improved petal heuristic for the vehicle routing problem, *Journal of the Operational Research Society*, 1996, **47**, 329-336.

[77] Breedam A.V., Comparing descent heuristics and metaheuristics for the vehicle routing problem, *Computers & Operations Research*, 2001, **28**, 289-315.

[78] Osman I.H., Christofides N., Capacitated clustering problem by hybrid simulated annealing and tabu search, *International Transaction in Operational Research*, 1994, **1**, 317-336.

[79] Hertz A., Widmer M., Guidelines for the use of meta-heuristics in combinatorial optimization, *European Journal of Operation Research*, 2003, **151**, 247-252.

[80] Blum C., Roli, A., Metaheuristics in combinatorial optimization: overview and conceptual comparison, *ACM Computing Surveys*, 2003, **35**, 268-308.

[81] Tarantilis C. D., Ioannou G., Prastacos G., Advanced vehicle routing algorithms for complex operations management problems, *Journal of Food Engineering*, 2005, **70**, 455-471.

[82] Tavares R.S., Martins T. C., Tsuzuki M. S. G., Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning, *Expert Systems with Applications*, 2011, **38**, 2951-2965.

[83] Nabiyev V. V., *Yapay zeka insan-bilgisayar etkileşimi*, 3. baskı, Seçkin Yayıncılık, Ankara, 2010.

[84] Karaboğa D., *Yapay zeka optimizasyon algoritmaları,* 2. baskı, Nobel Yayın Dağıtım, Ankara, 2011.

[85] Cordon O., Herrera F., Stutzle T., A review on the ant colony optimization metaheuristic: Basis, models and new trends, *Mathware and Soft Computing*, 2002, **9**, 141-175.

[86] Düzakın E., Demircioğlu M., Araç rotalama problemleri ve çözüm yöntemleri, *Çukurova Üniversitesi İİBF Dergisi*, 2009, **13**, 68-87.

[87] Glover F., Laguna. M., Tabu search. Editor: Reeves C.R., *Modern heuristic techniques for combinatorial problems*, Blackwell Scientific Publications, Oxford, 70-150, 1993.

[88] Dréo J., Pétrowski A., Siarry P., Taillard E., Chatterjee A., *Metaheuristics for hard optimization: methods and case studies*, 1th ed., Springer, Berlin, 2006.

[89] Gendreau M., Potvin J. Y., Tabu search, Editors: Gendreau M., Potvin J. Y., *Handbook of metaheuristics*, 2nd ed., Springer, 41-59, 2010.

[90] Gendreau M., Hertz A., Laporte G., A tabu search heuristic for the vehicle routing problem, *Management Science*, 1994, **40**, 1276-1290.

[91] Taillard E. D., Parallel iterative search methods for vehicle routing problem, *Networks*, 1993, **23**, 661-673.

[92] Rochat Y., Taillard E. D., Probabilistic diversification and ıntensification in local search for vehicle routing, *Journal of Heuristics*, 1995, **1**, 147-167.

[93] Xu J., Kelly J. P., A network flow-based tabu search heuristic for the vehicle routing problem, *Transportation Science*, 1996, **30**, 379-393.

[94] Barbarosoğlu G., Özgür D., A tabu search algorithm for the vehicle routing problem, *Computers & Operations Research*, 1999, **26**, 255-270.

[95] Cordeau J. F., Gendreau M., Laporte G., A tabu search heuristic for periodic and multi-depot vehicle routing problems, *Networks*, 1997, **30**, 105-119.

[96] Kruse R., Borgelt C., Nauck D., Fuzzy data analysis: challenges and perspectives, *IEEE International Conference on Fuzzy Systems*, Seoul, South Korea, 22-25 August 1999.

[97] Chen W. C., Wang M. S., A fuzzy c-means clustering-based fragile watermarking scheme for image authentication, *Expert System with Application*, 2009, **36**, 1300-1307.

[98] Wang X., Wang Y., Wang L. Improving fuzzy c-means clustering based on feature-weight learning, *Pattern Recognition Letters*, 2004, **25**, 1123-1132.

[99] Göktepe A.B., Altun S., Sezer A., Soil clustering by fuzzy c-means algorithm, *Advances in Engineering Software*, 2005, **36**, 691-698.

[100] Known best results, http://neo.lcc.uma.es/radi-aeb/WebVRP/ (Ziyaret tarihi: 15 Mayıs 2013)

EKLER

EK-1a

Tablo 1a. Küme 2 için noktaların tasarruf değerleri ve rotalanması

2. Küme											
	10	11	16	21	30	34	38	39	49	50	
9	39,36	22,32	37,11	41,70	45,47	45,87	31,62	42,06	38,32	43,60	
10		18,43	30,32	36,68	49,97	44,48	28,90	56,61	42,34	37,73	
11			24,06	24,08	21,65	23,21	21,14	19,27	19,36	23,65	
16				44,04	37,66	41,76	28,61	33,39	30,05	42,54	
21					47,41	54,79	29,49	43,18	33,90	50,42	
30						55,94	31,42	57,12	42,45	46,85	
34							31,47	51,96	39,49	51,55	
38								29,69	29,83	30,80	
39									42,89	42,26	
49										35,76	

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

EK-1b

Tablo 1b. Küme 3 için noktaların tasarruf değerleri ve rotalanması

3. Küme											
	2	3	20	22	28	29	31	32	35	36	
1	22,55	27,24	25,46	27,63	27,70	21,64	26,73	17,81	25,99	26,76	
2		38,28	41,43	31,20	32,02	41,09	27,53	19,62	41,33	40,54	
3			57,20	41,16	53,04	44,59	46,58	19,75	61,92	64,41	
20				38,23	45,77	51,52	39,46	19,98	64,79	63,85	
22					40,73	31,57	37,64	19,10	39,34	40,55	
28						34,88	53,47	18,38	49,96	53,80	
29							29,25	19,02	53,00	51,47	
31								16,81	43,53	47,76	
32									20,00	19,96	
35										77,26	

Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)
1	**35-36**	10	29-35	19	2-20	28	2-3	37	**1-22**	46	32-35
2	**20-35**	11	**20-29**	20	2-35	29	20-22	38	2-31	47	20-32
3	**3-36**	12	29-36	21	3-22	30	**22-31**	39	1-3	48	32-36
4	20-36	13	28-35	22	**2-29**	31	28-29	40	1-36	49	3-32
5	3-35	14	31-36	23	22-28	32	2-28	41	1-31	50	**2-32**
6	3-20	15	3-31	24	22-36	33	22-29	42	1-35	51	
7	28-36	16	20-28	25	2-36	34	2-22	43	1-20	52	
8	**28-31**	17	3-29	26	20-31	35	29-31	44	1-2	53	
9	**3-28**	18	31-35	27	22-35	36	1-28	45	1-29	54	

EK-1c

Tablo 1c. Küme 4 için noktaların tasarruf değerleri ve rotalanması

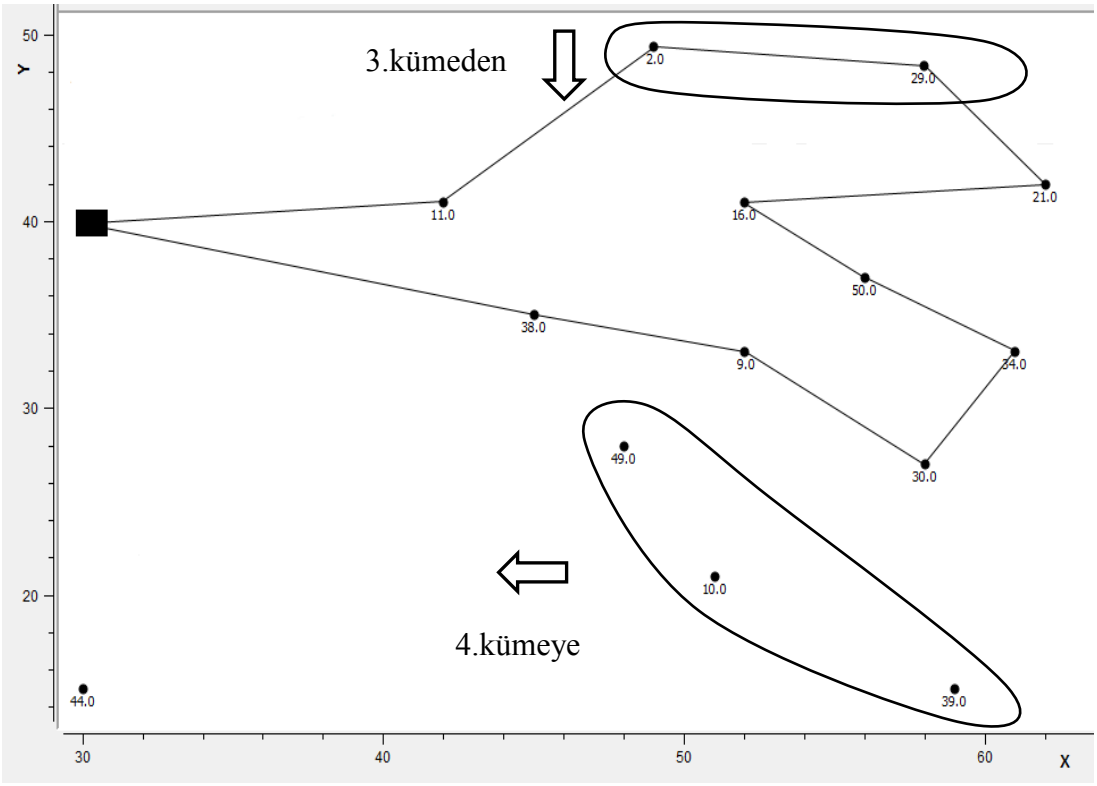
4. Küme									
	12	15	17	33	37	44	45	46	47
5	12,98	24,32	16,64	27,26	20,94	21,11	25,44	4,337	8,443
12		16,04	15,48	15,44	16,12	16,03	15,97	3,227	11,5
15			30,6	47,08	35,64	43,66	49,35	3,629	14,76
17				28,24	30,27	33,72	30,89	2,736	17,48
33					33,67	42,24	58,32	4,034	13,02
37						35,83	35,54	3,347	15,34
44							46,03	3,153	16,71
45								3,724	14,68
46									1,771
Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)
1	**33-45**	10	17-44	19	5-44	28	12-17	37	5-46
2	**15-45**	11	33-37	20	5-37	29	12-33	38	33-46
3	15-33	12	17-45	21	**17-47**	30	37-47	39	45-46
4	44-45	13	15-17	22	44-47	31	15-47	40	15-46
5	**15-44**	14	**17-37**	23	5-17	32	45-47	41	37-46
6	33-44	15	17-33	24	12-37	33	33-47	42	**12-46**
7	**37-44**	16	**5-33**	25	12-15	34	5-12	43	
8	15-37	17	5-45	26	12-44	35	12-47	44	
9	34-45	18	5-15	27	12-45	36	5-47	45	

EK-1d

Tablo 1d. Küme 5 için noktaların tasarruf değerleri ve rotalanması

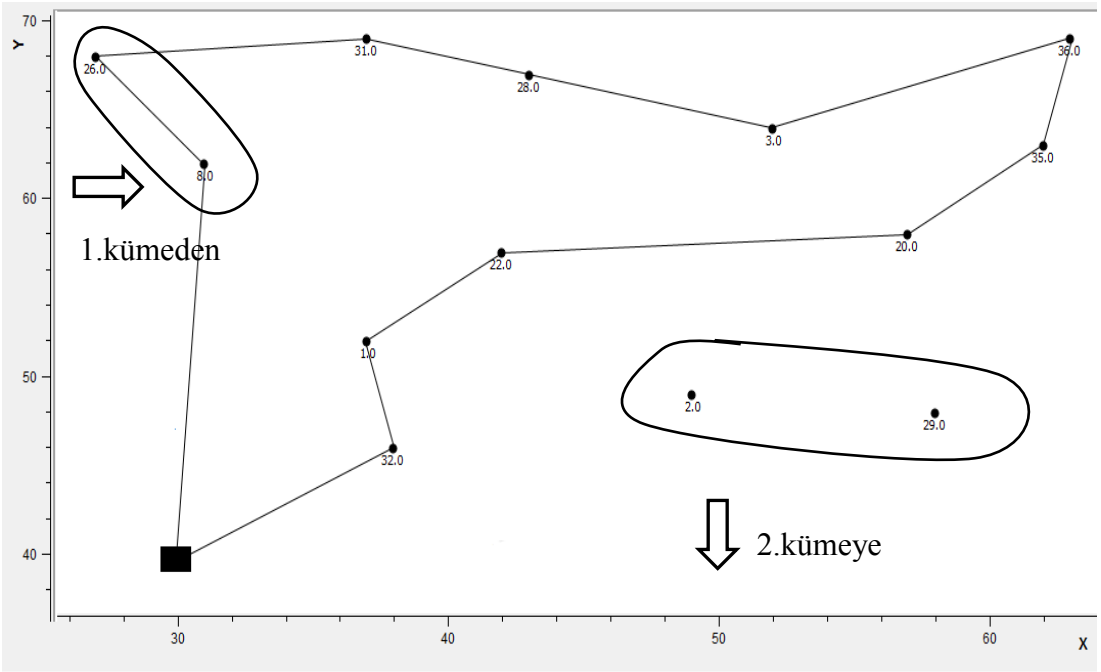
5. Küme							
	13	18	19	25	40	41	42
4	31,33	24,35	34,35	22,6	34,41	34,23	32,49
13		29,5	46,64	39,09	52,36	50,2	38,54
18			26,27	26,67	27,42	27,78	22,74
19				29,28	63,48	57,39	54,68
25					33,23	32,35	23,1
40						60,6	57,03
41							48,76
Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)	Sıra	(i,j)
1	**19-40**	7	**13-41**	13	4-19	19	13-18
2	**40-41**	8	41-42	14	4-41	20	19-25
3	19-41	9	13-19	15	25-40	21	18-41
4	40-42	10	**13-25**	16	**4-42**	22	18-40
5	**19-42**	11	13-42	17	25-41	23	**18-25**
6	13-40	12	4-40	18	4-13	24	

EK-2a

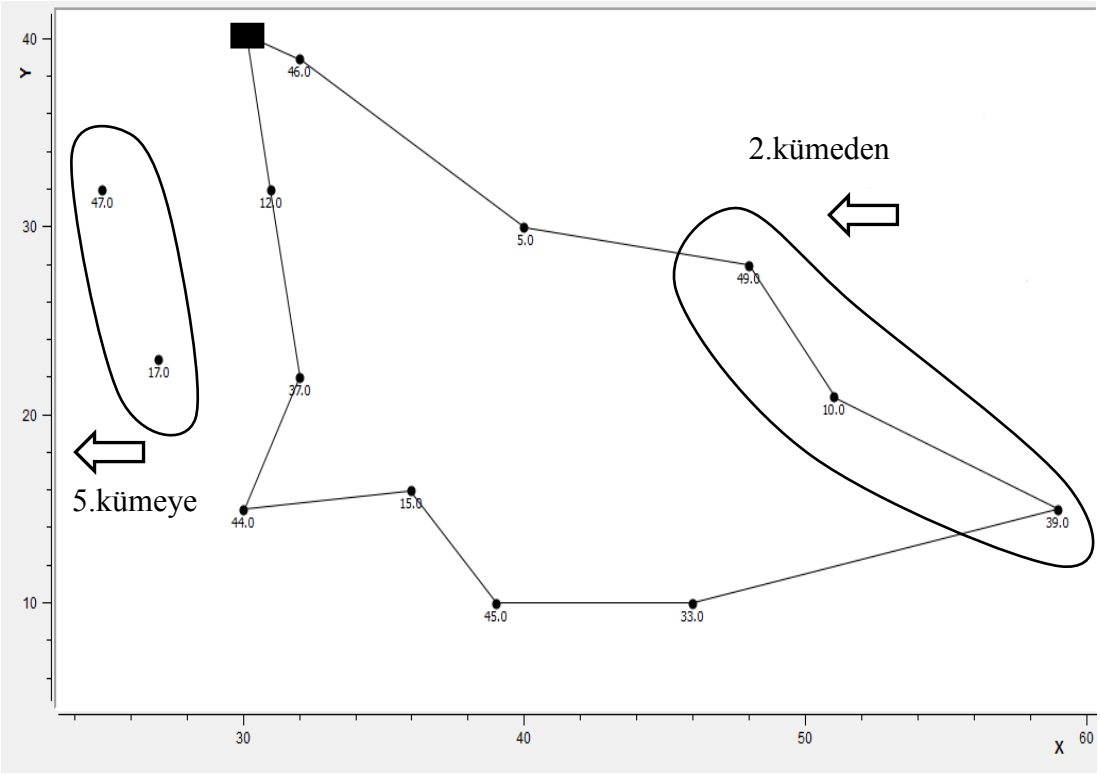


Şekil 2a. Küme 2'de rotanın güncellenmesi

EK-2b

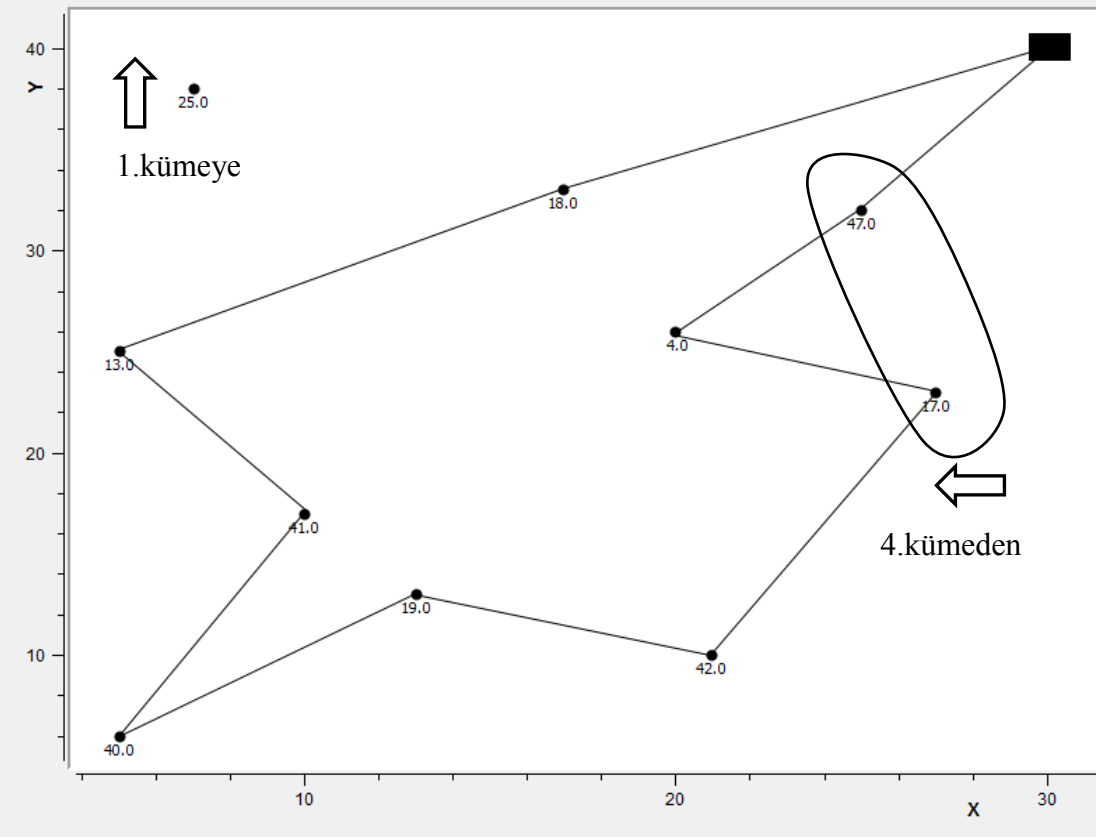


Şekil 2b. Küme 3'de rotanın güncellenmesi

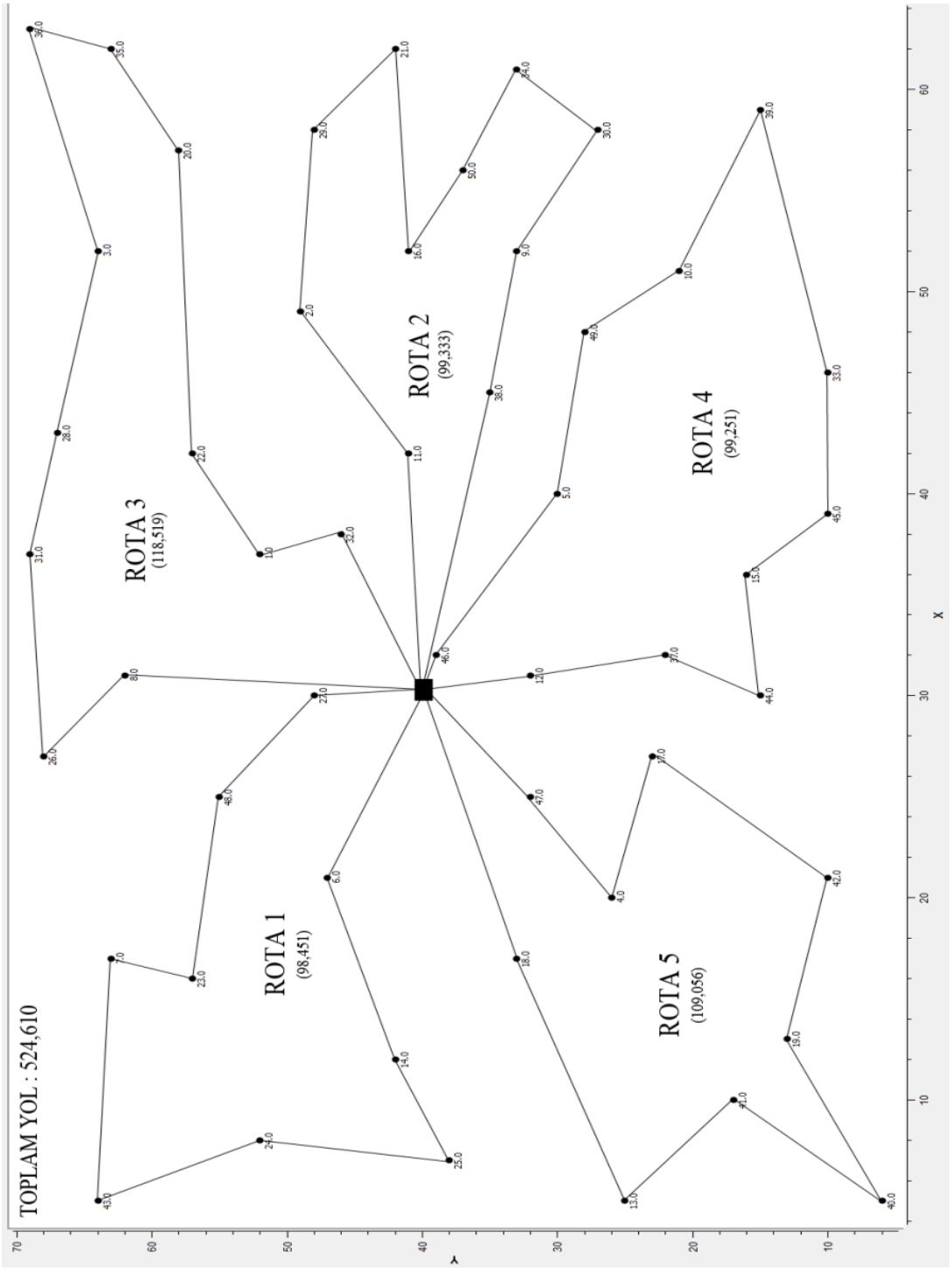


Şekil 2c. Küme 4'de rotanın güncellenmesi

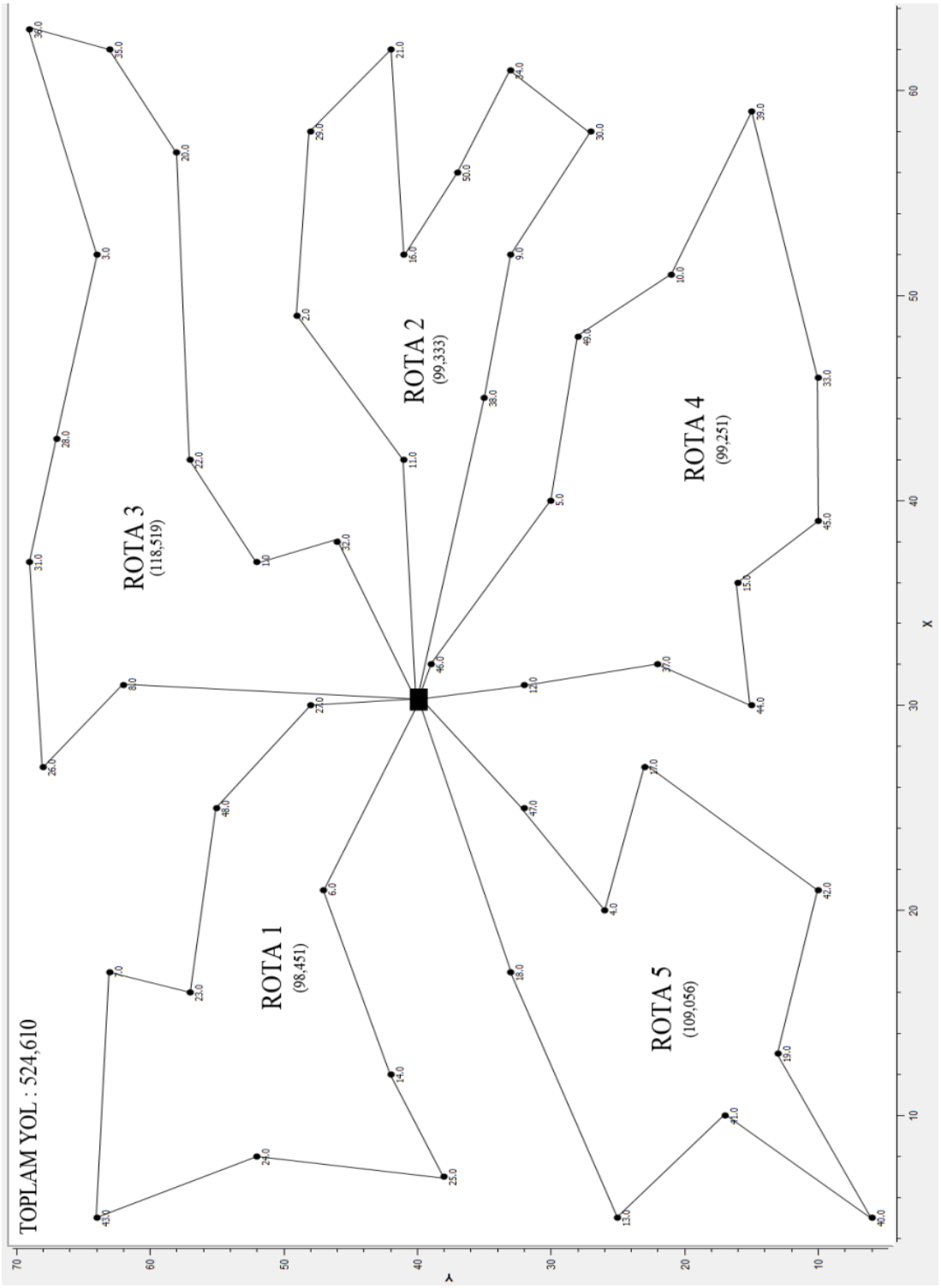
EK-2d



Şekil 2d. Küme 5'de rotanın güncellenmesi



Şekil 3a. E-n51-k5 probleminin kümelenip λ-opt ile iyileştirildikten sonra çözümü



Şekil 3b. E-n51-k5 probleminin optimum çözümü

EK-4a

Tablo 4.a. E-n22-k4 probleminde BCO sonrası talep noktalarının aitlik dereceleri

	Talep Noktaları									
Küme No	1	2	3	4	5	6	7	8	9	10
1	0,509	0,693	0,060	0,056	0,962	0,411	0,961	0,249	0,791	0,407
2	0,069	0,059	0,025	0,026	0,011	0,052	0,012	0,092	0,085	0,239
3	0,037	0,032	0,012	0,013	0,005	0,021	0,005	0,033	0,034	0,067
4	0,386	0,216	0,903	0,905	0,023	0,516	0,022	0,626	0,089	0,287
Küme No	11	12	13	14	15	16	17	18	19	20
1	0,150	0,128	0,094	0,001	0,121	0,017	0,005	0,064	0,049	0,019
2	0,222	0,674	0,567	0,991	0,536	0,862	0,049	0,300	0,281	0,104
3	0,083	0,123	0,188	0,006	0,272	0,103	0,943	0,590	0,613	0,861
4	0,544	0,075	0,151	0,001	0,071	0,018	0,004	0,046	0,057	0,016
Küme No	21									
1	0,022									
2	0,121									
3	0,834									
4	0,023									

EK-4b

Tablo 4.b. E-n23-k3 probleminde BCO sonrası talep noktalarının aitlik dereceleri

	Talep Noktaları									
Küme No	1	2	3	4	5	6	7	8	9	10
1	0,134	0,019	0,009	0,870	0,861	0,298	0,939	0,966	0,956	0,883
2	0,079	0,020	0,011	0,056	0,059	0,112	0,028	0,015	0,016	0,039
3	0,787	0,961	0,979	0,073	0,080	0,591	0,033	0,019	0,028	0,079
Küme No	11	12	13	14	15	16	17	18	19	20
1	0,682	0,499	0,836	0,054	0,048	0,016	0,076	0,189	0,063	0,010
2	0,101	0,214	0,053	0,225	0,097	0,030	0,392	0,640	0,860	0,962
3	0,217	0,288	0,111	0,722	0,855	0,954	0,532	0,171	0,077	0,028
Küme No	21	22								
1	0,600	0,076								
2	0,239	0,739								
3	0,161	0,186								

EK-4c

Tablo 4.c. E-n30-k4 probleminde BCO sonrası talep noktalarının aitlik dereceleri

	Talep Noktaları									
Küme No	1	2	3	4	5	6	7	8	9	10
1	0,735	0,823	0,759	0,998	0,869	0,652	0,025	0,011	0,005	0,008
2	0,023	0,054	0,058	0,000	0,025	0,023	0,090	0,066	0,027	0,054
3	0,233	0,105	0,167	0,002	0,096	0,317	0,026	0,010	0,005	0,008
4	0,009	0,018	0,016	0,000	0,010	0,008	0,859	0,914	0,962	0,931
Küme No	11	12	13	14	15	16	17	18	19	20
1	0,003	0,002	0,009	0,016	0,009	0,011	0,007	0,017	0,147	0,206
2	0,016	0,010	0,042	0,096	0,053	0,053	0,032	0,902	0,661	0,586
3	0,003	0,002	0,009	0,015	0,009	0,011	0,007	0,015	0,120	0,139
4	0,979	0,987	0,940	0,873	0,929	0,925	0,954	0,067	0,072	0,070
Küme No	21	22	23	24	25	26	27	28	29	
1	0,061	0,356	0,020	0,370	0,141	0,160	0,086	0,167	0,063	
2	0,728	0,405	0,875	0,026	0,016	0,069	0,020	0,095	0,007	
3	0,045	0,173	0,017	0,593	0,837	0,744	0,884	0,689	0,926	
4	0,166	0,066	0,088	0,011	0,007	0,026	0,010	0,049	0,003	

EK-4d

Tablo 4.d. E-n33-k4 probleminde BCO sonrası talep noktalarının aitlik dereceleri

	Talep Noktaları									
Küme No	1	2	3	4	5	6	7	8	9	10
1	0,978	0,199	0,280	0,224	0,002	0,007	0,012	0,048	0,054	0,105
2	0,005	0,041	0,090	0,130	0,001	0,003	0,005	0,025	0,029	0,063
3	0,006	0,035	0,089	0,098	0,000	0,001	0,003	0,012	0,013	0,026
4	0,012	0,725	0,541	0,548	0,996	0,989	0,980	0,916	0,904	0,806
Küme No	11	12	13	14	15	16	17	18	19	20
1	0,075	0,132	0,760	0,904	0,812	0,118	0,470	0,145	0,129	0,026
2	0,019	0,029	0,047	0,025	0,059	0,117	0,223	0,682	0,719	0,930
3	0,012	0,022	0,036	0,041	0,072	0,710	0,211	0,064	0,061	0,022
4	0,895	0,818	0,157	0,030	0,057	0,055	0,096	0,109	0,091	0,022
Küme No	21	22	23	24	25	26	27	28	29	30
1	0,034	0,000	0,080	0,093	0,250	0,129	0,026	0,010	0,164	0,554
2	0,913	0,999	0,762	0,703	0,330	0,064	0,021	0,006	0,064	0,083
3	0,023	0,000	0,102	0,148	0,342	0,777	0,944	0,981	0,720	0,202
4	0,031	0,000	0,057	0,056	0,077	0,031	0,008	0,003	0,052	0,160
Küme No	31	32								
1	0,816	0,041								
2	0,035	0,016								
3	0,059	0,008								
4	0,089	0,934								

ÖZGEÇMİŞ

1983 yılında Kayseri'de doğdu. İlk ve orta öğrenimini Niğde'de, lise öğrenimini ise Aksaray'da tamamladı. Gaziantep Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nden Ocak, 2010'da mezun oldu. Kısa bir süre özel sektörde Endüstri Mühendisi olarak görev yaptıktan sonra, aynı yıl içerisinde Bartın Üniversitesi, İktisadi ve İdari Bilimler Fakültesi Yönetim Bilişim Sistemleri Bölümü'nde Araştırma Görevlisi olarak göreve başladı. Bartın Üniversitesi'ndeki görevine halen devam etmektedir.