AĞ VE ARAÇ ROTALARININ OPTİMİZASYONU İÇİN META-SEZGİSEL BİR ÇÖZÜM ÖNERİSİ

2019
DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Dursun EKMEKCİ

AĞ VE ARAÇ ROTALARININ OPTİMİZASYONU İÇİN META-SEZGİSEL BİR ÇÖZÜM ÖNERİSİ

Dursun EKMEKCİ

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Doktora Tezi
Olarak Hazırlanmıştır

KARABÜK
Şubat 2019

Dursun EKMEKCİ tarafından hazırlanan "AĞ VE ARAÇ ROTALARININ OPTİMİZASYONU İÇİN META-SEZGİSEL BİR ÇÖZÜM ÖNERİSİ" başlıklı bu tezin Doktora Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Fuat ŞİMŞİR

Tez Danışmanı, Endüstri Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Doktora tezi olarak kabul edilmiştir. 11/ 02/ 2019

Unvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. Nejat YUMUŞAK (SAÜ)	
Üye : Doç. Dr. İlker TÜRKER (KBÜ)	
Üye : Doç. Dr. Muharrem DÜĞENCİ (KBÜ)	
Üye : Dr. Öğr. Üyesi Fuat ŞİMŞİR (KBÜ)	
Üye : Dr. Öğr. Üyesi Murat İSKEFİYELİ (SAÜ)	

...../...../2019

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Doktora derecesini onamıştır.

Prof. Dr. Filiz ERSÖZ

Fen Bilimleri Enstitüsü Müdür V.

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Dursun EKMEKCİ

ÖZET

Doktora Tezi

AĞ VE ARAÇ ROTALARININ OPTİMİZASYONU İÇİN META-SEZGİSEL BİR ÇÖZÜM ÖNERİSİ

Dursun EKMEKCİ

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Fuat ŞİMŞİR
Şubat 2019, 103 sayfa

Coğrafi bakımdan dağınık konumlardaki müşterilere, dağıtım/toplama hizmeti için bir ya da daha fazla depodan yola çıkacak araçların, en uygun rotalarının belirlenmesi problemi olarak ifade edilen araç rotalama problemi (ARP), uzun zamandır üzerinde değişik teknikler denenerek çözüm aranan ve optimizasyon açısından ilgi çeken problem çeşitlerinden biridir. Esnek bir araç rotalama sistemi; lojistik planların daha önceden hazırlanması, gerektiğinde yeni rotaların hızlı bir biçimde oluşturulabilmesi ve yüklerin; depolarda daha az süre tutulması gibi avantajları da beraberinde getirecektir. Klasik ARP yapısına farklı kıstaslar eklenerek, problemin değişik türleri oluşturulmuş ve daha kompleks hale dönüştürülmüştür. ARP türlerinin genelinde rota maliyeti mesafeyle ilişkilendirilir ve daha kısa mesafeli çözüm, daha başarılı çözüm olarak kabul edilir. En kısa mesafe hedefi, işletmelere maliyet ve zaman açısından önemli avantajlar sağlamaktayken, bu konuyu daha fazla araştırma yapılması için çekici kılmaktadır. Araç rotalama üzerine farklı yön ve alanlarda, çeşitli bakış

açılarıyla oluşturulan problem türleri incelendiğinde, pratikteki uygulamaya en yakın olanının kapasite kısıtlı ve eş zamanlı dağıtım-toplamalı araç rotalama problemi (EDTARP) olduğu söylenebilir. EDTARP modelinde; bir ya da daha fazla depo merkezi, çok sayıda ziyaret edilecek müşteri düğüm ve tamamı depo merkezlerinde bulunan, sınırlı kapasiteli taşıma araçları yer alır. Problem senaryosunda, müşteri düğümlerden, miktarları ve dağıtım noktaları belirlenmiş olarak toplanan yükler, depo merkezlerinde biriktirilmiştir. Bu yükler, ilgili dağıtım noktalarına sevk edilecek ve aynı zamanda, uğranan düğümde, bir sonraki seferde dağıtılmak üzere hazırlanmış, miktarları belli yükler toplanacaktır. EDTARP'de amaç, eş kapasiteli taşıma araçlarının kapasiteleri aşılmadan, ziyaret edilen her bir düğümde dağıtım toplama faaliyetinin eş zamanlı olarak yürütülebilmesi ve bu araçların, toplamda minimum maliyetle depo merkezlerine dönecek biçimde rotalandırılmasıdır. Bu çalışmada, EDTARP için, Yapay Arı Koloni (YAK) algoritması kullanılarak bir çözüm önerisi sunulmuş ve uygulama, literatürde EDTARP için yaygın olarak kullanılan problem setleriyle test edilmiştir. Elde edilen sonuçlar, literatürde aynı test problemleri için belirlenen en düşük maliyetli rota çözümleriyle karşılaştırıldığında, az sayıdaki parametresine rağmen; önerilen yöntemin, literatürdeki en başarılı çözümlerin en çok %1.12'si kadar gerisinde kaldığı gözlemlenmiştir.

Anahtar Sözcükler **:** Dağıtım toplamalı araç rotalama problemi, optimizasyon, yapay arı koloni algoritması.

Bilim Kodu **:** 924.1.172

ABSTRACT

Ph. D. Thesis

A META-HEURISTIC ALGORITHM APPROACH FOR NETWORK AND VEHICLE ROUTING OPTIMIZATION

Dursun EKMEKCİ

Karabük University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Fuat ŞİMŞİR
February 2019, 103 pages

The vehicle routing problem (VRP), which is defined as the problem of determining the most appropriate routes of the vehicles that will depart from one or more depots to the customers in the geographically dispersed locations, is the solution that has been sought for a long time by trying different techniques and the problem of optimization is one of its varieties. A flexible vehicle routing system; preparation of logistics plans before, creation of new routes when necessary, it will bring advantages such as keeping loads in warehouses for less time. In most VRP types, route cost is associated with distance, and a shorter distance solution is considered a more successful solution. While the shortest distance goal provides significant advantages in terms of cost and time to businesses, this makes it attractive for further research. When examining the types of problems having different directions and areas devised from different points of view on vehicle routing, it can be said that the closest approach to practical application is the vehicle routing problem with simultaneous delivery and pickup

(VRPSDP). In VRPSDP model; there are one or more depots, many customer nodes to visit, and limited capacity transport vehicles located at depots. In the problem scenario, the loads collected from the customer nodes with their designated amounts and delivery points are stored in depots. These loads will be shipped to the relevant delivery points and at the same time, certain loads will be collected in the line which is prepared to be distributed next time. The purpose in the VRPSDP model is that to ensure the delivery/pickup activity can be carried out simultaneously on each of the visited nodes without exceeding the capacities of the carrying vehicles, and that these vehicles are routed to return to depots at minimum cost in total. In this study, a solution proposal is presented for the VRPSDP using the Artificial Bee Colony (ABC) algorithm and the application is tested with the benchmark problem data sets commonly used for VRPSDP in the literature. When the results are compared with the least cost route solutions in the literature, it is observed that despite the few parameters, the proposed method can produce low-cost solutions, behind at most 1.12% of the most successful solutions in the literature.

Key Word : Vehicle routing problem with delivery and pickup, optimization, artificial bee colony algorithm.

Science Code : 924.1.172

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üyesi Fuat ŞİMŞİR'e sonsuz teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

N : düğümler kümesi

E : düğümler arası kenarlar kümesi

c_{ij} : $i-j$ düğümleri arası maliyet miktarı

q_i : i düğümündeki talep miktarı

M : araç filosundaki toplam araç sayısı

Q : araç kapasitesi

x_{ij} : düğüm sırası kontrol değişkeni

Ψ : araç filosundaki araç türü seti

L : araçların gidebileceği maksimum mesafe uzunluğu

d_i : i düğümünde dağıtılacak yük miktarı

p_i : i düğümünde toplanacak yük miktarı

U_i : i ve sonraki düğümlere dağıtılacak toplam yük miktarı

V_i : i ve önceki düğümlerde toplanacak yük miktarı

t_i : i düğümündeki müşteri için hizmet süresi

s_i : i numaralı çözüm rotası

λ : tek rotalı iyileştirmede kullanılan yol sayısı

SN : YAK algoritmasında toplam çözüm sayısı

D : YAK algoritmasındaki bir çözümde toplam parametre sayısı

$f(x_i)$: YAK algoritmasında *i.* çözümün toplam maliyeti

$fit(x_i)$: YAK algoritmasında *i.* çözümün uygunluk değeri

r_i : YAK algoritmasında *i.* çözümün seçilme olasılığı

KISALTMALAR

ABC	:	Artificial Bee Colony (Yapay Arı Koloni)
ACS	:	Ant Colony System (Karınca Koloni Sistemi)
ALS	:	Adaptive Local Search (Uyarlamalı Yerel Arama)
AMM	:	Adaptive Memory Methodology (Uyarlamalı Hafıza Metodolojisi)
ARP	:	Araç Rotalama Problemi
AUARP	:	Açık Uçlu Araç Rotalama Problemi
AYARP	:	Asimetrik Yollu Araç Rotalama Problemi
CMT	:	Christofides-Mingozzi-Toth Test Problemleri
ÇDARP	:	Çok Depolu Araç Rotalama Problemi
DARP	:	Dinamik Araç Rotalama Problemi
DKA	:	Değişken Komşuluk Arama
DTARP	:	Dağıtım Toplamalı Araç Rotalama Problemi
EACO	:	Efective Ant Colony System (Efektif Karınca Koloni Sistemi)
EDTARP	:	Eş Zamanlı Dağıtım Toplamalı Araç Rotalama Problemi
EYK	:	En Yakın Komşuluk
GA	:	Genetik Algoritma
GAMS	:	The Genetic Algebraic Modeling System (Genetik Cebirsel Modelleme Sistemi)
GDARP	:	Geri Dönüşlü Araç Rotalama Problemi
GKA	:	Geniş Komşuluk Arama
GSP	:	Gezgin Satıcı Problemi
KDTARP	:	Karışık Dağıtım Toplamalı Araç Rotalama Problemi
KKARP	:	Kapasite Kısıtlı Araç Rotalama Problemi
KKO	:	Karınca Koloni Optimizasyonu
KKS	:	Karınca Koloni Sistemi
KUARP	:	Kapalı Uçlu Araç Rotalama Problemi
LNS	:	Large Neighborhood Search (Geniş Komşuluk Arama)
MILP	:	Mixed-Integer Linear Programming (Karışık Tamsayılı Doğrusal Programlama)
MKARP	:	Mesafe Kısıtlı Araç Rotalama Problemi

MN_GLS : Multiple Neighborhood Guided Local Search (Çoklu Komşuluk Rehberli Yerel Arama)

MTFARP : Müşteri Tipi Farklı Araç Rotalama Problemi

MTSEAS : Modified Tabu Search and Elite Ant System (Modifiye Tabu Arama ve Elit Karınca Sistemi)

PDARP : Parçalı Dağıtımlı Araç Rotalama Problemi

PSO : Particle Swarm Optimization (Parçacık Sürü Optimizasyonu)

PILS : Parallel Iterative Local Search (Paralel İteratif Yerel Arama)

SARP : Statik Araç Rotalama Problemi

SYARP : Simetrik Yollu Araç Rotalama Problemi

TA : Tabu Arama

TB : Tavlama Benzetimi

VRP : Vehicle Routing Problem (Araç Rotalama Problemi)

VRPSDP : Vehicle Routing Problem With Simultaneous Delivery and Pickup (Eş Zamanlı Dağıtım Toplamalı Araç Rotalama Problemi)

YAK : Yapay Arı Koloni

YSA : Yapay Sinir Ağları

ZPARP : Zaman Pencereli Araç Rotalama Problemi

BÖLÜM 1

GİRİŞ

Ulaşım ve iletişim alanındaki teknolojik gelişmeler, ticaret alanını küresel boyuta çıkarmıştır. Bugün itibariyle işletmeler, ticari politikalarını belirlerken, yalnızca bölgelerindeki koşullara göre değil, etkin dağıtım kanalları geliştirerek dünya genelini hedefleyerek adım atmak durumundadırlar. Dolayısıyla günümüz küresel piyasasında yaşanan yoğun rekabet, kısa ömürlü ürünler ve müşterilerin artan beklentileri; üreticileri, dağıtım sistemlerine önem vermeye ve yatırım yapmaya zorlamıştır. Dağıtım sistemleri oluşturulurken, hammadde, yarı ürün ve depolama konularının yanı sıra, dikkat edilmesi gereken diğer bir önemli husus, nihai ürünün kullanıcıya nasıl ulaştırılacağı konusudur [1]. Bu bağlamda işletmeler, pazarda varlıklarını sürdürebilmek ve ön sıralara geçebilmek için lojistik düzenini optimal düzeyde planlamalı ve müşteri memnuniyetini göz önünde tutarak gecikmeleri ortadan kaldırmalıdırlar. Bilhassa kurye sistemiyle çalışan müesseselerde, düşük maliyetli ve esnek güzergâh planı oluşturabilmek, işletme açısından yararlı olacaktır. Çünkü daha düşük maliyetli rotalama; ihtiyaç duyulan taşıma aracı sayısını azaltmak, araç kapasitesini daha yüksek verimle kullanmak, müşteriye daha kısa sürede erişmek gibi avantajları da beraberinde getirecektir. Bu yaklaşım, iletişim ve ulaşım teknolojilerindeki değişimle birlikte, ağ ve araç rotası optimizasyonlarının da sürekli gelişimine neden olmuştur. Ulaştırma faaliyetleri, bazı sektörlerde üretilen malın maliyetinin büyük bir yüzdesini oluşturmaktadır. Bu alanda iyileştirme amaçlı geliştirilen ve uygulanan bilgisayar destekli yöntemlerle, ulaşım maliyetlerinde %5 ile %20 arasında değişen, gözle görülür seviyede önemli tasarrufların elde edildiği savunulmaktadır [2].

Son dönemde etkisi giderek artan, yeniden üretim kapsamındaki geri kazanım süreçleri de işletmelerin, lojistik faaliyetlerine dahil ettikleri önemli konulardan biridir. “Tersine

lojistik" ifadesiyle tanımlanan bu işleyişte, tekrar değer elde etme amacıyla, hammaddelerin, süreçteki envanterlerin, bitmiş malların ve kullanım sonrası oluşan atıkların, düzgün biçimde elden çıkarılarak, tüketim noktasından üretim noktasına, verimli ve maliyet avantajlı akışı, planlı ve kontrollü biçimde yürütülmektedir [3]. Bu sayede işletmeler, ekonomik açıdan kar elde ederken, ekolojik açıdan çevreye daha az zarar vermektedirler.

Gerek kurye sistemlerinde ve gerekse tersine lojistik süreçlerinde uygulanan bu işleyişle birlikte, işletmenin dağıtım faaliyetlerinden oluşan taşıma maliyetine, toplama maliyeti de eklenmektedir. Bu bağlamda, maliyeti en aza düşürebilme adına, gönderilerin tam dolu araçlarla ve hızlı biçimde yapılabilmesi, boşaltılan aracın da yine tam dolu olarak yüksek verimde kullanılabilmesi için milk-run sistemleri geliştirilmiştir. Ancak bu sistemlerin kabiliyeti, ziyaret noktalarında dağıtım, toplama taleplerinin yakınlığıyla ilişkilidir. Ziyaret noktalarında dağıtım, toplama miktarları arasındaki farklılık bu sistemleri başarısız kılmaktadır.

Lojistik faaliyetlerin, başarılı bir şekilde yürütülebilmesi için göz önünde tutulması gereken konulardan biri de yeni müşteri taleplerinin ortaya çıkması, araçlarda oluşabilecek arıza durumları ya da rota güzergahında oluşabilecek sorunlarda, rota planının hızlı biçimde güncellenerek yeniden yapılandırılabildiği, dinamik bir rotalama sistemine sahip olmaktır. Esnek bir araç rotalama sistemi; lojistik planların daha önceden hazırlanması, gerektiğinde yeni rotaların hızlı bir biçimde oluşturulabilmesi ve yüklerin; depolarda daha az süre tutulması gibi avantajları da beraberinde getirecektir. Bu yaklaşımla, lojistik faaliyetler kapsamındaki en önemli problemin, dağıtım ve toplama faaliyetlerinde kullanılacak taşıma araçları için izleyecekleri güzergahların belirlenmesi olduğu düşünülmektedir [4]. Araç rotalama problemi (ARP) olarak isimlendirilen bu problem, genel bir tanımla, coğrafi bakımdan dağınık konumlardaki müşterilere, dağıtım/toplama hizmeti için bir ya da daha fazla depodan yola çıkacak araçların, en uygun rotalarının belirlenmesi problemi olarak ifade edilmiştir [5]. ARP'deki temel yaklaşım, talep miktarları bilinen müşterilerin, bu taleplerini tek bir depodan tedarik etmesi ve her bir müşterinin yalnızca, bir tek taşıma aracıyla ziyaret edilerek tüm dağıtımın sağlanmasıdır [6]. 1959'dan günümüze kadar, otomotivden gıdaya, tekstilden kargoculuğa, üretim ve hizmet sektörlerinin lojistik,

taşımacılık ve dağıtım departmanlarının büyük bir bölümünde, ARP için farklı çözüm önerileri geliştirilmiştir [7]. Literatürde ARP ile ilgili çalışmalar incelendiğinde, rota üzerindeki düğümlerde dağıtım ve toplama faaliyetlerinin birbirinden bağımsız olarak, farklı zamanlarda yapıldığı, değişik türden problemlere rastlamak mümkündür [8]. Bu türler içerisinde, güncel yaşamdaki lojistik faaliyetlere en yakın olarak eş zamanlı dağıtım toplamalı araç rotalama problemi (EDTARP) göze çarpmaktadır. EDTARP modelinde; müşterilere dağıtılacak tüm ürünler, depo merkezlerinden gönderilir ve müşterilerden toplanacak talepler de yine bu merkezlere taşınır. Ayrıca, ağ üzerindeki her bir düğüm, yalnızca bir araç tarafından ziyaret edilir ve araçlarının seyahatleri süresince ziyaret edecekleri her bir düğümde, dağıtım ve toplama faaliyetleri eş zamanlı olarak yürütülür.

Bu çalışmanın amacı, çok noktadan oluşan çeşitli ağ modellerinde, eş zamanlı dağıtım-toplama faaliyeti yapan, sınırlı kapasitelerdeki araçlar için en düşük maliyetli rotalar belirlenmesidir. Çalışma kapsamında, yöntem olarak, Meta-sezgisel yöntemlerin kullanılabildiği diğer tüm alanlarda kullanılabilen ve özellikle bu tarz polinomsal zamanda bir çözümü olduğu ispatlanamayan NP-Zor sınıfındaki problemler için, birçok uygulamada daha başarılı sonuçlar verdiği gözlemlenen Yapay Arı Koloni (YAK) algoritması tercih edilmiştir. Bal arılarının, gerçek yaşamdaki yiyecek arama davranışlarını taklit eden bu meta-sezgisel algoritma, az sayıdaki kontrol parametresiyle, aynı anda pek çok farklı noktadan başlayarak, oldukça geniş bir çözüm uzayını etkin olarak tarayabilen bir yöntemdir. Popülasyona dayalı bu yöntem kullanılarak en düşük maliyetli rotalar oluşturabilen çözüm modeli hedeflenmektedir. Geliştirilen uygulamanın verimliliği CON ve SCA örnek problemleriyle test edilmiştir. CON ve SCA örnekleri, 2001 yılında Dethloff tarafından, EDTARP için tasarlanmış ve literatürde günümüze kadar EDTARP problemi için çözüm geliştiren hemen her yöntem bu örnekler üzerinde test edilmiştir. Önerilen yöntemin elde ettiği çözümler, literatürdeki çözümlerle karşılaştırıldığında, yöntemin diğer sezgisel yöntemlerle rekabet edebilecek seviyede başarılı sonuçlar elde ettiği gözlemlenmiştir.

Hazırlanan bu çalışma, ağ optimizasyonu ve ARP'nin kapsamlı olarak tanıtıldığı, çözüm tekniği olarak kullanılan YAK algoritmasının detaylıca anlatıldığı ve deneysel sonuçların optimizasyon açısından etraflıca analiz edildiği üç ana kısımdan oluşmakta

ve her kısım kendi içinde bölümlere ayrılmaktadır. Birinci bölüm "Giriş" olarak, çalışmanın kısaca özetlendiği bölümdür. İkinci bölümde ağ optimizasyonu ve ARP detaylı olarak tanıtılmış, klasik ARP yapısına farklı kısıtlar eklenerek oluşturulan problem türleri, ana yapıları ve matematiksel formülleriyle açıklanmıştır. Gerçek dünyada, kurye dağıtımlı lojistik faaliyetlere en çok benzeyen ARP türü olduğundan, özellikle, EDTARP üzerinde durulmuş, EDTARP için oluşturulan literatür problemleri incelenmiştir. Bölümde ayrıca farklı türden ARP çözümlerine yönelik geliştirilen teknikler detaylı olarak analiz edilmiş, literatür taraması sunulmuştur. Üçüncü bölümde, EDTARP çözüm önerisinde tercih edilen YAK algoritması irdelenmiştir. YAK algoritmasına ilham olan bal arılarının, doğal yaşamdaki yiyecek arama davranışları, aralarındaki iş bölümü ve haberleşme yöntemleri açıklanmış, ardından YAK algoritmasının, matematiksel modeli, algoritma tasarımı ve akış diyagramı sunulmuştur.

Dördüncü bölümde YAK algoritmasının, EDTARP çözümüne uygulanışı, algoritma bileşenlerinin problem çözümündeki karşılıkları ve algoritma adımlarındaki işlemler açıklanmıştır. Önerilen yöntemin, EDTARP için çözüm arama karakteristiğini analiz edebilmek ve performansını değerlendirebilmek için, uygulama, literatürde EDTARP için geliştirilen test problemleri üzerinde denenmiştir. Bu nedenle, tercih edilen test problemleri incelenmiş ve denemeler için oluşturulan parametre tasarımı paylaşılmıştır. Bölümde ayrıca, deneysel çalışmalardan elde edilen sonuçlar, tablo ve grafiklerle detaylı olarak sunulmuştur.

Beşince ve son bölümde ise, YAK algoritmasının EDTARP test problemlerinde elde ettiği tüm çözümler genel bir yaklaşımla değerlendirilmiş, bulunan en başarılı sonuçlar, literatürdeki en iyi sonuçlarla karşılaştırılmıştır.

BÖLÜM 2

AĞ VE ARAÇ ROTASI OPTİMİZASYONU

"En iyileme" olarak tanımlanan optimizasyon kavramı; ekonomiden inşaata, tasarım optimizasyonundan, lojistik sektörüne kadar, en iyi çözümü hedefleyen her alanda önemli görülen kavramlardan biridir. Hedeflenen en iyi çözüm genellikle, maksimum kar ya da minimum maliyettir. Matematiksel açıdan optimizasyon, fonksiyonun, amaca uygun olarak maksimize ya da minimize edilebilmesi için kısıtlardaki değişkenlerin en iyi sonucu veren değerlerinin bulunması olarak yorumlanmış, bununla ilişkili olarak optimizasyon problemi ise, genel bir tanımla; istenen kıstasları sağlayacak şekilde, uygun parametre değerlerinin bulunması problemi olarak ifade edilmiştir [9]. Optimizasyon problemi çözümlerinde, problemin, belirli koşullar altında, muhtemel çözümleri içinden en iyisi seçilerek "optimum çözümü" hedeflenir. Bu bağlamda ağ tasarımındaki optimizasyon, ağ üzerindeki her bir düğüme en kısa yoldan ulaşabilmeyi ve geziyi en kısa yoldan tamamlayabilmeyi sağlayan kısıtlara ait parametre değerlerinin belirlenmesi işlemi olarak tanımlanabilir.

Ağ optimizasyonu ile araç rotası optimizasyonu bazı durumlarda birbirinden ayrı olarak kabul edilse de ağ optimizasyonu, aracın yolunu da etkilediğinden birbirleriyle ilişkilendirilebilir. Nitekim literatürdeki bazı kaynaklarda birbirinden farklı kavramlar olarak ele alınsa da kaynakların birçoğunda ağ optimizasyonu ve araç rotası optimizasyonu birlikte incelenmiştir. Ağ optimizasyonu, yalnızca karayolu ağları için değil; elektrik-trafo hatları, su kanalları, internet bağlantı kabloları gibi network ağlarını da içermektedir. Karayolu ağı optimizasyonuna ilişkin çalışmalarda, özellikle hat topolojisi, dağıtım merkezleri, şehir içi/şehirlerarası yollar, yol bağlantıları, kavşak durumu, yoldaki araç yoğunluğu, sinyalizasyon gibi unsurlar üzerinde durulmuştur. Araç rotası optimizasyon problemi çözümünde ise toplam seyahat süresini/mesafesini en aza indirme, taşıma maliyetini en aza indirme, toplam araç sayısını en aza indirme

gibi sorunlara yönelik kesin çözüm algoritmaları, matematiksel yöntemler ve sezgisel yaklaşımlar geliştirilmiştir.

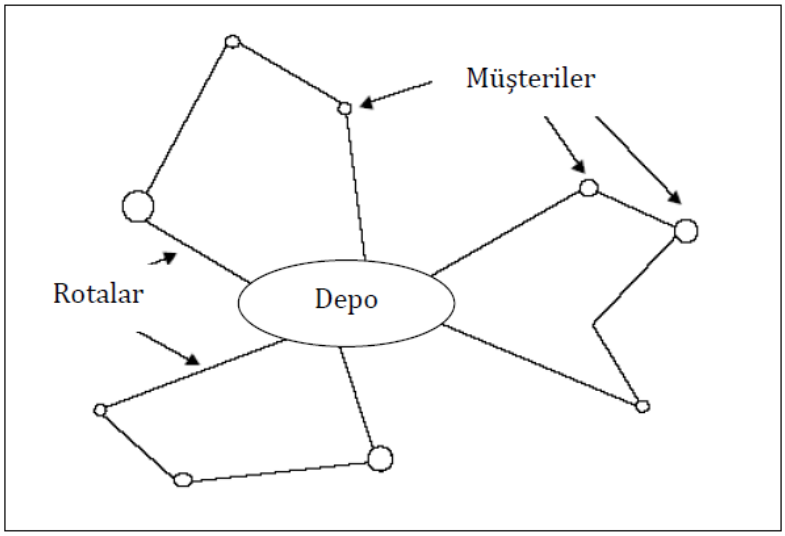
Daha hızlı bir ulaştırma, planlı bir dağıtım/toplama için; araç rotası optimize edebilmeli, diğer bir ifadeyle en kısa yol belirlenmelidir. Bu amaçla tasarlanmış problemler içerisinde en popüler olup üzerinde yıllardır birçok disiplin tarafından değişik yaklaşımlarla çözüm aranan problemlerden biri gezgin satıcı problemi (GSP)dir. GSP'nin kökeni, 1880'lerde Sir William R. Hamilton tarafından bulunan, İkosyan Oyununa dayandırılmaktadır. Bu oyunda amaç, 20 noktadan oluşan bir alanda tüm noktaları mutlaka ve yalnızca bir kez ziyaret edecek en kısa yolu bulmaktır. Bulunan bu yola Hamilton Turu adı verilir [10]. Problemde düğüm sayısı arttıkça problemin çözümü üstlü bir biçimde artan işlem gerektirmektedir. Düğüm sayısının n ve bir düğümden en çok hareket edilebilecek düğüm sayısının m olduğu durumda algoritma karmaşıklığı $O(m^n)$ olarak hesaplanırken; tüm düğümlerin birbiriyle ilişkili ve başlangıç düğümü belirlenmiş problem için, mümkün olan ihtimal sayısı geriye kalan düğümlerin yer değişmesine bağlı olarak (n-1)!'e eşitlemiştir. GSP'de hareketli yalnızca bir tek öğe vardır. Eğer problemde birden çok hareketli öğe var ve bu hareketli öğeler, düğümlere giden taşıma araçlarıysa; bu probleme özel olarak ARP adı verilmektedir. ARP'nin çok düğümlü örneklerinde çözüm için, düğüm noktalarının gruplandırılması da gerekebilir. Dolayısıyla ARP, bu yönüyle de GSP'den ayrılır.

Genel bir tanımla ARP, coğrafi bakımdan dağınık konumlara yayılmış müşteri düğümlere, bir veya daha fazla depodan hizmet veren araçların her biri için, en kısa zamanda ve minimum maliyetli dağıtım/toplama rotalarının tasarlanması olarak ifade edilmiştir [11]. Problemin çözüm uzayı, problemdeki düğüm sayısıyla üstel olarak arttığından, ARP, çözümü zor bir kombinatoryal optimizasyon problemidir [12]. Yıllardır üzerinde çalışılmasına rağmen, ARP için optimum rotayı çizebilen matematiksel bir formül halen bulunamamıştır. ARP'deki operasyonel kıstas sayısı, ağ optimizasyonu için gerekli kıstaslardan çok daha fazla ve kapsamlı olduğundan, ARP için kullanılabilecek çözüm, aynı zamanda ağ optimizasyonu için de kullanılabilecektir. Bu sebeple tez kapsamında ARP konusu ele alınmış, farklı ARP türleri içinden, günlük yaşamdaki lojistik faaliyetleri bire bir temsil eden ve

optimizasyon tekniği açısından ağ optimizasyonuna da çözüm sağlayacak EDTARP türü için çözüm aranmıştır.

2.1. ARAÇ ROTALAMA PROBLEMİ

ARP en genel tanımla, aynı tip ve kapasiteye sahip M adet aracın, ana depodan başlayıp operasyonel kısıtları sağlayarak, alt kümesindeki şube depoları ziyaret ettikten sonra, tekrar ana depoya dönmesiyle sonlanan rotasının oluşturulması problemidir [13]. Klasik ARP modelinde, talepleri önceden belirli şube depolar, sadece bir araçtan hizmet almakta ve her araç yalnızca bir rotayı izleyerek başladığı ana depoya dönmektedir. Dolayısıyla her bir şube, M araç rotalarından birinde mutlaka yer almalı ve her aracın toplam dağıtım miktarı, araç kapasitesini geçmemelidir. Problemdeki ana amaç, maliyet fonksiyonunu en aza indirgerken, bütün kısıtları sağlayıp, kullanılacak olan araç sayısını azaltmak ve toplam mesafeyi veya toplam zamanı minimuma indirmektir [14]. Dolayısıyla bir bakıma müşteri memnuniyeti maksimize edilmektedir. Şekil 2.1'de ARP'nin genel yapısı görüntülenmektedir.



Şekil 2.1. ARP'nin genel yapısı [15].

ARP'nin çözüm uzayı, depo sayısı ve araç sayısı parametreleriyle ilişkili biçimde üstel olarak artar. Bu yönleriyle ARP, NP-Zor kombinatoryal optimizasyon problemi sınıfına dahil edilir. Ayrıca çözüm uzayında çok sayıda yerel minimum nokta

bulunması, bu noktalardan kurtulmayı güçleştirecek, çözüm daha da zorlaşacaktır [16].

ARP'de çözüm uzayının sınırlarını belirleyen parametreler, depo sayısı ve araç sayısıyla sınırlı kalsa da ARP yapısını oluşturan ve çözüm değerlerini etkileyen diğer unsurlar, yol ağı, işletme birimi, müşteriler, sürücüler, kısıtlar ve amaçlar olarak sıralanabilir. ARP'yi oluşturan bu kavramların özellikleri aynı zamanda çözülecek problemin türünü de belirlemektedir.

2.1.1. ARP'nin Bileşenleri

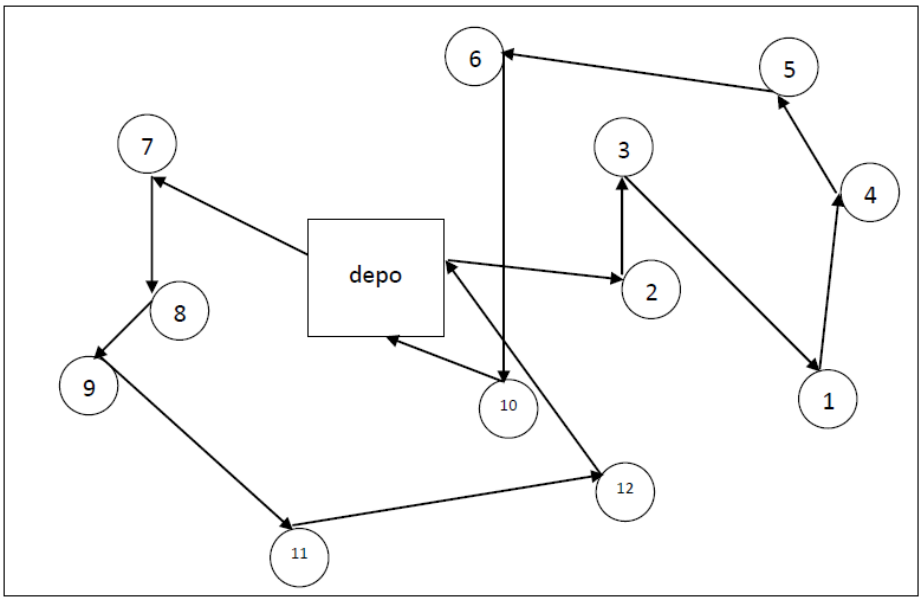
Klasik bir ARP modelinde temel yapı taşları olarak; talep yapısı, taşınacak malzemenin tipi, dağıtım toplama noktaları ve araç filosu kabul edilebilir [17]. Talep, taşınacak yük ya da yolcu olabilir. Talep miktarları ise problem türüne göre, önceden bilinebilir ya da bazı düğümlerde araç, seyahatine devam ederken belirlenebilir. Talep miktarından farklı olarak, talep edilen yer, teslim edilecek mekan, teslim zamanı ve hizmet düzeyini belirleyen müşteri tercihleri de talep kapsamda değerlendirilebilir.

Taşıma araçları ile gıda maddeleri, basın-yayın dağıtımı, çöp toplama gibi malzemeler taşınabilir. Bu tür yükler, probleme ek bir karmaşıklık getirmezler. Ancak tehlikeli madde taşıyan araç rotaları belirlenirken coğrafi koşullar göz önünde bulundurulmalıdır. Diğer taraftan öğrenci servisleri için, güvenlik ve etkinlik gibi amaçlardan kaynaklı daha karmaşık bir durum söz konusudur.

ARP türlerinin birçoğunda, dağıtım noktaları, müşteri düğümler; toplama noktaları ise merkez depolardır. Bazı ARP türlerinde dağıtım ve toplama faaliyetleri aynı düğümde yürütülür. Dağıtım noktaları sabit ve önceden biliniyorsa, hangi düğümleri hangi araçların ziyaret edeceği belirlenmelidir. Dinamik modelde ise dağıtım noktaları potansiyel noktalar arasından seçileceği için ilave bir yerleştirme kararı gerekir. Düğümler arası bağlantı, maliyet ya da uzunlukla tanımlanır. Yollar simetrikse, düğümler arası maliyet, her iki yönde de eşittir. Ancak yolların asimetrik olduğu ağ yapıları da bulunabilmektedir.

Bütün ARP türlerinde, araç kapasitelerinin bilindiği ve filodaki tüm araçların eş kapasitede olduğu kabul edilir. Farklı taşıma kapasitelerine sahip, heterojen bir araç filosunda hangi araç tipinin hangi rotayı dolaşacağına karar vermek ilave bir karmaşıklık getirir. Ayrıca, araçların yakıt tüketimi, hızı ve taşınacak malzemeye uygunluğu gibi kıstaslar, rotalamaya doğrudan etkisi olmayan özellikleridir.

Şekil 2.2'de bir depo merkezi, iki araç ve 12 müşteri düğümden oluşan bir ağ yapısında örnek bir ARP çözümü sunulmaktadır. Yolculuğa depodan başlayan araç rotaları, yine aynı depo merkezinde sonlanmaktadır.



Şekil 2.2. Bir depo merkezi, iki araç ve 12 müşterili ağ için ARP çözüm örneği [15].

2.1.2. ARP Çözümünde Dikkat Edilecek Konular

ARP, NP-Zor kombinatoryal optimizasyon problemi sınıfına dahil edilen, çözümü zor bir optimizasyon problemidir. ARP türlerine göre farklı optimizasyon kıstasları mevcuttur. Yaygın kıstaslardan biri, optimum rota sayısının belirlenmesidir. Bu sayede, taşımacılık için tek bir araç kullanılacaksa, aracın kaç kez seyahat edeceği, birden çok araç kullanılacaksa, kullanılacak araç sayısı belirlenmiş olacaktır. Diğer bir kıstas, oluşturulacak rota uzunluğunun minimize edilmesidir. ARP çözümünde rota maliyetini belirleyen temel parametre rota mesafesidir ve çoğu kere rota maliyeti

yerine rota uzunluğu tabiri kullanılır. Ziyaret edilecek düğümler içinde, öncelik sırası olan düğümler varsa ya da bazı düğümler belli zaman aralıklarında ziyaret edilebiliyorsa, bu kısıtlar da ilave karmaşıklık çıkaracak, çoğu çözümde rota uzunluğunu artıracaktır. Bazı ARP türlerinde ise kıstas, rota uzunluğu yerine rota süresinin minimize edilmesidir. Bu tür problemlerde toplam rota süresini, seyahat, yükleme-boşaltma, bakım-onarım ve dinlenme süreleri belirler. Ayrıca taşımacılığa bağlı olarak, araçların maksimum verimle kullanılması ya da araçların eş yük dağılımında kullanılması kıstaslara dahil edilebilir. Güncel hayattaki lojistik planlamaya, bu kıstaslara ek olarak, aracın gidebileceği yol kapasitesi, yol üzerindeki fiziki şartlar, iklim koşulları, sürücünün çalışma saati aralığı, araç yükleme-boşaltma zamanları gibi operasyonel kısıtlar da dahil edilebilir.

2.1.3. ARP'nin Matematiksel Modeli

Klasik ARP modeli, tüm düğümlerin birbirine bağlı olduğu $G = (N, E)$ grafıyla temsil edilebilir. Bu graf modelinde, $N = (N[0], N[1], N[2], \dots, N[n])$ düğümler kümesini temsil eder. N $[0]$, depo düğüm, diğer düğümler ise müşteri düğümlerdir. E, düğümler arası kenar kümesidir v $E = \{(N[i], N[j]) : N[i], N[j] \in N, i \neq j\}$e olarak ifade edilir. İki düğüm arası maliyet miktarı, her iki yönde de eşittir ve $c_{ij} = c_{ji}$ ifadesiyle belirtilmiştir. i düğümündeki talep miktarı q_i'dir. Araç filosunda bulunan M adet, Q eş kapasiteli taşıma araçlarının tamamı merkez depodadır. x_{ij} değeri, taşıma aracı ile i düğümünden sonra j düğümü ziyaret edilecekse 1, aksi takdirde 0 olacaktır.

Problemde temel amaç, gerekli şartları yerine getirerek, tüm düğümleri ziyaret eden, minimum maliyetli M adet rotayı oluşturmaktır. Bu durumda tek depolu bir ARP modelinde, amaç fonksiyonu eşitlik 2.1 ile gösterilmektedir.

$$min \sum_{k=1}^{M} \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} x_{ijk} \qquad \forall\, i, j \in N \qquad (2.1)$$

Sağlanması gereken koşullar ise aşağıdaki eşitliklerle ifade edilmektedir.

$$\sum_{k=1}^{M}\sum_{j=0}^{N} x_{ijk} = 1 \qquad \forall\, i \in N \tag{2.2}$$

$$\sum_{k=1}^{M}\sum_{i=0}^{N} x_{ijk} = 1 \qquad \forall\, j \in N \tag{2.3}$$

$$\sum_{i=1}^{N} x_{i0k} \leq 1 \qquad \forall\, k \in M \tag{2.4}$$

$$\sum_{i=1}^{N} q_i \sum_{j=0}^{N} x_{ijk} \leq Q \qquad \forall\, k \in M \tag{2.5}$$

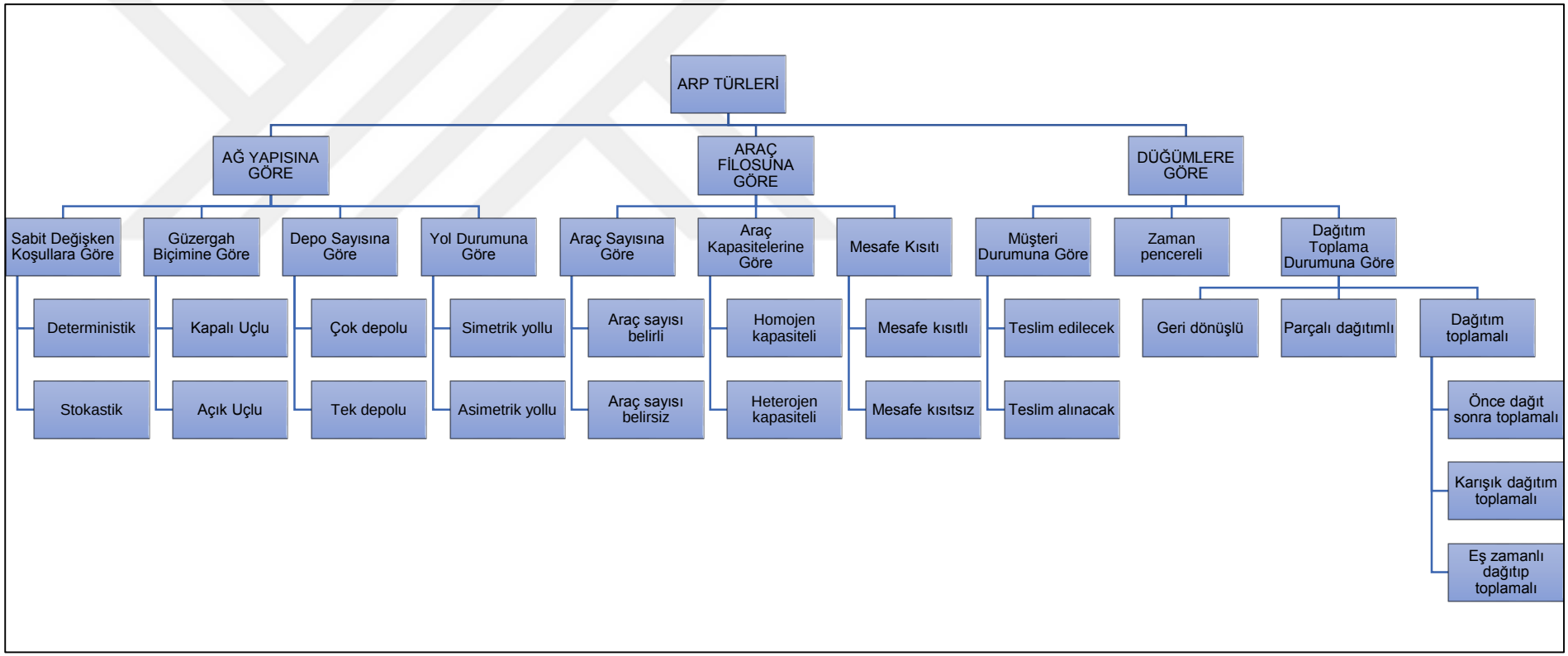
Formüller incelendiğinde 2.2 eşitliğiyle her bir müşteri düğümün yalnızca bir araç tarafından ziyaret edilmesi gerektiği ifade edilmektedir. 2.3 eşitliğinde, müşteri düğüme ulaşan ve müşteri düğümden çıkan yollardan yalnızca bir tanesinin kullanılması gerektiği, 2.4 kıstasında ise her araç için yalnızca bir kere rota oluşturulacağı belirtilmiştir. Eşitlik 2.5'te araçların her bir düğüm arasında taşıyacakları yük miktarının Q araç kapasitesinden daha fazla olamayacağı ifade edilmiştir.

2.2. ARP TÜRLERİ

Araç rotası optimizasyonuna ilişkin problem türlerini; depo durumuna, araç durumuna, dağıtım ve toplama işlemlerinin zamanlama durumuna bağlı olarak değişik sınıflara ayırabilmek mümkündür. ARP örneklerinin bazısında bir tek depo bulunurken bazı örneklerde birden çok depo bulunabilir. Yine örneklerin bazılarında taşımacılık için bir tek araç düşünülmüş, bazılarında ise, kapasite kısıtlı ve eş kapasiteli birden çok araç olduğu varsayılmıştır. Araçlar, problem türüne göre yalnızca dağıtım, yalnızca toplama ya da hem dağıtım hem toplama işlemlerinde kullanılabilir. Düğümlerle ilgili durum ise dağıtım ve toplama faaliyetlerine yöneliktir. Bazı ARP örneklerinde uğranacak düğümlerin her birinde yalnızca dağıtım ya da yalnızca toplama faaliyeti yapılabilirken, bazı örneklerde dağıtım ve toplama eş zamanlı olarak yapılabilmektedir. Düğümlerde karışık olarak dağıtım ve toplamanın yapılabildiği

problem örnekleri geliştirilerek çok depolu haliyle de tanımlanmıştır. Bu tür problemlere ilişkin çözümlerde dağıtım ve toplama farklı araçlarla yapılmaktadır. Ayrıca geliştirilen ARP örneklerinin bir kısmı mesafe bazlı, diğer bir kısmı ise zaman pencereli olarak tasarlanmıştır. Zaman esaslı olarak tasarlanan önce dağıt sonra topla problem örneğinde de dağıtım ve toplama farklı araçlarla yapılır. Ancak düğüm noktasına istenilen zamandan daha erken gelen araç bekletilebilirken, geç gelen araç kabul edilmediğinden, çözüm benimsenmez. Diğer bir örnek türünde ise düğümlerde dağıtım ve toplama farklı sıralarda, araçlar için tanımlanan zaman dilimlerinde yapılır.

Görüldüğü üzere ARP için farklı kıstaslara göre sınıflar oluşturulabilmektedir. Değişen her bir kıstas, problem türünü ve boyutunu da değiştirmektedir. Literatür çalışmaları incelendiğinde, oluşturulan kısıtlamalara göre farklı hiyerarşilerde ARP sınıflandırmalarına rastlamak mümkündür. Bu kısıtlamalar en temelde; ağ yapısına, araç filosuna ve düğümlere dayalı olarak üç ana grupta toplanabilir. Dolayısıyla çalışma kapsamında ARP sınıflandırmaları, bu kısıtlara göre yapılmaktadır. Oluşturulan ARP sınıflandırmalarına ait hiyerarşik düzen Şekil 2.3'te gösterilmektedir.



Şekil 2.3. ARP türleri.

2.2.1. Ağ Yapısına Göre Sınıflandırma

Genel ağ yapısını ilgilendiren kıstaslar dikkate alınarak oluşturulan sınıflandırmalarda, yolculuk sürecinde mevcut koşulların sabit ya da değişkenliğine, güzergah biçimine, depo merkezi sayısına, yol durumuna, mesafe durumuna göre farklı sınıflar oluşturulabilmektedir.

2.2.1.1. Sabit Değişken Koşullara Göre ARP Türleri

Mevcut şartların değişebilme durumuna göre deterministik ya da stokastik olarak da sınıflandırılan ARP türlerinde, seyahat boyunca; yol durumu, iklim şartları, çevre koşulları, kısıtlar, her bir düğümdeki dağıtılacak ve toplanacak yük miktarları biliniyor ve sabit kabul ediliyorsa, problem, statik araç rotalama problemi (SARP) olarak sınıflandırılır. Literatür çözümleri daha çok, bu problem türü üzerine yoğunlaşmıştır. Ziyaret düğümlerinde ani talep değişimleri oluşabiliyor ya da bazı yolların kapanması, trafik sıkışıklığı, zor iklim şartları gibi nedenlerle maliyet miktarı değişebiliyorsa, problem, dinamik araç rotalama problemi (DARP) olarak isimlendirilir. Bu modelde, önceden planlanmayan değişimlerde, güzergahın yeniden çizilebilmesi için hızlı karar verilmesi gerekecek, dolayısıyla problem çözümü güçleşecektir. Gerçekleştirilemeyen dağıtımlar, ertesi güne sarkabilmektedir [18]. DARP, güncel yaşamda, müşterilerine kurye yoluyla dağıtım yapan işletmeler açısından ciddi problemlerden biridir [19].

2.2.1.2. Güzergah Biçimine Göre ARP Türleri

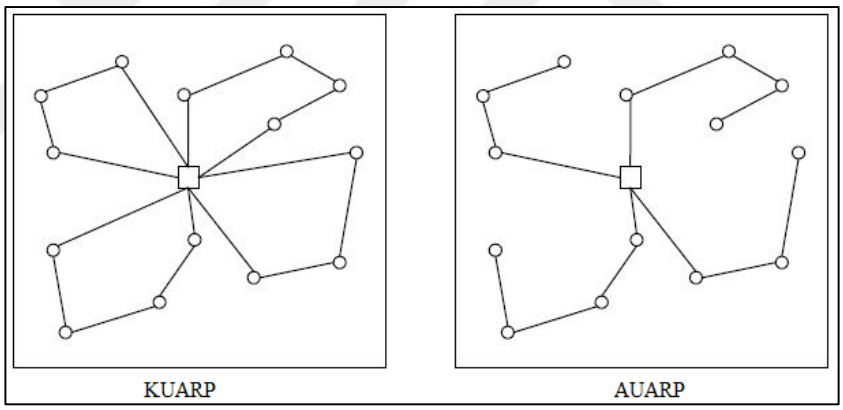
ARP, ağ üzerinde oluşturulabilecek güzergah biçimine göre sınıflandırılırken, araçların, seyahate başladıkları ve seyahatlerini sonlandırdıkları düğüm noktaları referans alınır. Bu bağlamda, araç, seyahatine, merkez depo yerine, bir işletme biriminden başlayıp yine aynı işletmede seyahatini tamamlıyorsa, bu yapı kapalı uçlu araç rotalama problemi (KUARP) olarak isimlendirilir. Tek işletmeye sahip ağ modelinde bu yapının kurulabilmesi için klasik ARP formülasyonuna 2.6 eşitliği eklenmelidir [20].

$$\sum_{j=1}^{N} x_{0jk} = \sum_{i=1}^{N} x_{i0k} \leq 1 \qquad \forall\, k \in M \qquad (2.6)$$

Dağıtım aracı, seyahatine, depo merkezinden başlıyor ancak seyahatini, son dağıtım noktasında tamamlıyorsa bu tür problem türleri açık uçlu araç rotalama problemi (AUARP) olarak isimlendirilir. Bu ARP türünde araç, ziyaret edilecek son düğüme de uğrayarak, rotasını o düğümde tamamlar, depo merkezine geri dönmez.

$$\sum_{j=1}^{N} x_{0jk} + \sum_{i=1}^{N} x_{i0k} = 1 \qquad \forall\, k \in M \qquad (2.7)$$

2.7 eşitliğiyle, her bir taşıma aracının, 0 numaralı depo merkeziyle rotasına başlayabileceği ya da rotasını sonlandırabileceği kısıt formülüze edilmiştir [20].



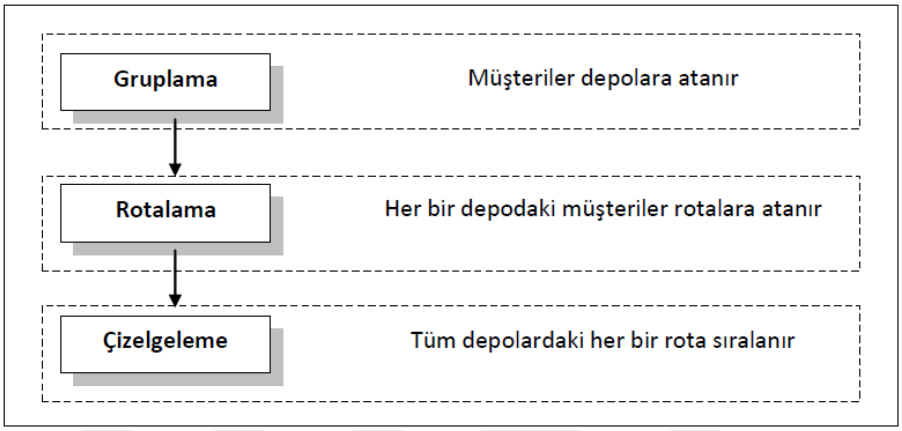
Şekil 2.4. KUARP ve AUARP çözüm örnekleri [1].

Problem türleri incelendiğinde, KUARP'ın GSP'ye, AUARP'ın da Hamilton yol problemine benzediği göze çarpmaktadır. Şekil 2.4'te AUARP ve KUARP için örnek çözümler gösterilmektedir [20].

2.2.1.3. Depo Sayısına Göre ARP Türleri

Ağ modelinde müşteriye hizmet veren depo sayısı birden fazlaysa, bu tür problemler çok depolu araç rotalama problemleri (ÇDARP) olarak sınıflandırılır. Eğer müşteri düğümler, depo merkezi civarında konumlandırılmışlarsa, rotalama için klasik ARP

çözüm teknikleri uygulanabilir. Ancak müşteri düğümler ile merkez depolar iç içe, karışık konumdalarsa; öncelikle müşteri düğümler depolara atanarak gruplandırılır, ardından ilgili depo merkezinden seyahate çıkacak araçlar, tüm müşterileri ziyaret edip başladığı depo merkezine dönecek şekilde rotalandırılır. Son olarak depolara ait rotalar sıralanarak çizelgeleme tamamlanır. ÇDARP'a yönelik çözüm aşamaları Şekil 2.5'te gösterilmektedir [1].



Şekil 2.5. ÇDARP çözüm aşamaları [1].

Ağ üzerinde tek bir depo merkezinin yer aldığı ARP türlerinde, müşteri ihtiyaçlarını karşılamak üzere yalnızca bir araç kullanılıyorsa, araç kapasitesi miktarınca doldurulur ve rotası üzerindeki her bir düğümü ziyaret ettikten sonra depo merkezine geri döner. Talebi henüz karşılanmamış diğer müşteriler için, araç, kapasitesi aşılmayacak şekilde yüklenerek ikinci kez rotalandırılır. Bu işlem tüm talepler karşılanana kadar devam eder. Bu durumda araç, defalarca kez rotalandırılmış olur. Ancak bu durum, ihtiyaçların karşılanmasını geciktirir. Dolayısıyla zaman kısıtının olduğu durumlarda, ağ üzerinde taşımacılık için birden çok araç kullanılır. Rota sayısınca araca ihtiyaç duyulan bu modelde, araçlar, depo merkezinde kapasitelerine göre yüklenerek yola çıkar ve rotaları üzerindeki düğümleri ziyaret edip tekrar depo merkezine geri dönerler [21].

2.2.1.4. Yol Durumuna Göre ARP Türleri

Ağ üzerinde ziyaret edilecek düğüm noktaları arası maliyet iki yönde de eşit ise ($c_{ij} = c_{ji}$) bu tür problemler simetrik yollu araç rotalama problemi (SYARP) olarak isimlendirilir. Eğer iki düğüm için gidiş ile dönüş maliyetleri birbirinden farklıysa ($c_{ij} \neq c_{ji}$) bu tür problemler de asimetrik yollu araç rotalama problemi (AYARP) olarak isimlendirilir. Özellikle rota maliyetinin büyük önem taşıdığı KUARP için, araçların ilk olarak hangi müşteriye gidecekleri belirlenmelidir.

2.2.2. Araç Filosuna Göre Sınıflandırma

Literatürde oluşturulan tüm ARP türlerinde, araçların belirli kapasitelerde oldukları kabul edilir. Bu nedenle öncelikle kapasite kısıtlı araç rotalama problemi (KKARP) türüne değinilmiştir. Araç filosu baz alınarak yapılan ARP sınıflandırmalarında ise araç kapasitelerinin eş olup olmamasına ve araçların gidebilecekleri maksimum mesafeye dikkat edilir.

2.2.2.1. Kapasite Kısıtlı ARP

KKARP, ağ üzerinde müşterilere hizmet veren tüm araçların belirli kapasitelerle (Q) sınırlı olduğu araç rotalama problemleridir. Bu problemler, araçların rotalarına başlayıp rotalarını sonlandırdığı depo merkezlerinden, müşteri düğümler ile müşterilerin talep miktarlarından, maksimum kapasiteye sahip taşıma araçlarından ve ağ üzerindeki her bir düğüm arasındaki taşıma maliyetlerinden oluşur [22]. KKARP türlerinde araç sayısı ya önceden bellidir ya da karar değişkeni olarak bırakılır [23]. Her bir düğüm, yalnızca bir araç tarafından ve yalnızca bir kez ziyaret edilir ve ilgili düğümdeki talep, ziyaretçi araç kapasitesini aşırmamalıdır [24]. İlgili koşul, genel ARP yapısında da ifade edilen 2.5 eşitliğiyle belirtilmektedir.

2.2.2.2. Araç Kapasitelerine Göre ARP Türleri

Araç filosunda bulunan tüm araçlar eş taşıma kapasitelerine sahiplerse bu tür problemler homojen kapasiteli araç rotalama problemleri olarak tanımlanır. Aksi

belirtilmedikçe tüm ARP türlerinde araçların homojen kapasitede oldukları kabul edilmektedir. Araç filosundaki araç türleri ya da kapasiteleri farklıysa bu tür problemler heterojen kapasiteli araç rotalama problemleri olarak isimlendirilir. Ağ yapısında düğümler arası maliyet ve yükleme-boşaltma işlemleri için harcanan zamana bağlı olarak heterojen araçların çeşitliliği değişkenlik gösterir.

Heterojen kapasiteli araç rotalama problemlerinde farklı araç türleri $\psi = \{1..k\}$ setiyle belirtilir. k türlü araçların sayısı n_k'ya kadar ulaşabilmektedir ve bu türdeki araçlar Q_k taşıma kapasitesine sahiptir. k türündeki bir araç ile i müşterisinden j müşterisine olan taşıma maliyeti c_{ijk} iken bu maliyet farklı kapasitelerdeki araçlar için farklılık gösterecektir [25]. Her bir türdeki araç sayıları filoların büyüklüğüne göre değişmektedir.

2.2.2.3. Mesafe Kısıtlı ARP

Taşıma işleminde kullanılan araçların gidebilecekleri maksimum mesafeler kısıtlıysa bu tür problemler mesafe kısıtlı araç rotalama problemleri (MKARP) olarak isimlendirilir. Klasik ARP modeline 2.8 eşitliği ilave edilerek problem MKARP haline dönüştürülebilir.

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} \sum_{j=0}^{N} x_{ijk} \leq L \qquad \forall\, k \in M \qquad (2.8)$$

Formülde verilen L parametresi, aracın gidebileceği maksimum yol mesafesini temsil etmektedir. KKARP'de olduğu gibi MKARP'de de araç filosundaki heterojen yapıya bağlı olarak, her araç türünün gidebileceği yol uzunluğu değişebilmektedir. $(L_k, k = 1..M)$

2.2.3. Düğümlere Göre Sınıflandırma

Düğümlere göre ARP türleri oluşturulurken en temelde, düğümlerdeki müşteri durumu ve dağıtım toplama faaliyetleri dikkate alınmaktadır.

2.2.3.1. Müşteri Durumuna Göre ARP Türleri

Müşteri tipi farklı araç rotalama problemlerinde (MTFARP) ziyaret edilen düğümdeki müşteri, ürün teslim edilecek müşteri ve ürün teslim alınacak müşteri olmak üzere iki gruba ayrılır. İlgili düğümdeki müşteri, ürün teslim ederken aynı zamanda ürün teslim alan müşteri de olabilir. Bu bağlamda hem dağıtım ve hem de toplama işlemi için kullanılan araç, bu düğüme uğradığında önce ürün dağıtımı yapacak, ardından ürün toplayacaktır. Her bir düğüme teslim edilecek talep miktarı için pozitif değerler, düğümden alınacak teslimat için ise negatif değerler kullanılarak MTFARP için çözüm aranır [26].

2.2.3.2. Dağıtım Toplama Durumuna Göre ARP Türleri

Ziyaret edilecek düğümde yürütülecek dağıtım toplama faaliyetine göre de değişik ARP türleri oluşturulmuştur. Eğer düğümlerde hem dağıtım ve hem de toplama işlemleri yapılıyorsa bu tür problemler dağıtım toplamalı araç rotalama problemi (DTARP) olarak isimlendirilir. Bu ARP türünde müşteriler arası ürün alışverişi yoktur. Tüm dağıtım talepleri depodan başlar ve düğümlerdeki toplama talepleri depoda biriktirilir. DTARP çözümünde amaç, toplam rota mesafesini ya da toplam seyahat süresini en aza indirmektir. Klasik ARP türüne ek olarak, dağıtım, toplama ve her bir düğümdeki yükleme durumu, araç kapasitesini aşmayacak şekilde kontrol edilmelidir. d_i, i düğümüne dağıtılacak yük miktarını; p_i, i bu düğümde toplanacak yük miktarını; U_i, i ve sonraki düğümlere dağıtılacak toplam yük miktarını; V_i ise i ve önceki düğümlerde toplanacak yük miktarını temsil ederse, rota boyunca dağıtılacak toplam yükün araç kapasitesini aşmayacağına ilişkin kıstas 2.9 ve yine rota boyunca toplanacak yük miktarının, araç rotasını aşmaması gerektiğine ilişkin kıstas da 2.10 eşitliğiyle ifade edilmektedir.

$$U_j - U_i + Qx_{ij} \leq Q - d_i \qquad \forall\, i,j \in N \qquad (2.9)$$

$$V_i - V_j + Qx_{ij} \leq Q - p_j \qquad \forall\, i,j \in N \qquad (2.10)$$

Dağıtım ve toplama faaliyetlerini kapsayan ARP türleri, düğümlerdeki dağıtım toplama durumuna göre üç farklı şekilde yürütülebilmektedir. Önce dağıtıp sonra toplamalı ARP türlerinde müşteriler dağıtım ve toplama hattı olmak üzere iki gruba ayrılır. Depo merkezlerinden çıkan araçlar önce dağıtım yapılacak düğümlere uğrayıp dağıtım faaliyetlerini tamamlar ve ardından toplama faaliyeti yürütülecek düğümleri ziyaret ederler. Bu ARP modelinde de tek/çok depolu, homojen/heterojen kapasiteli gibi farklı türevler oluşturabilmek mümkündür [27]. Araçlar, dağıtım ve toplama düğümlerini sıralı olarak değil de karışık olarak ziyaret ederlerse bu tür problemler karışık dağıtım toplamalı araç rotalama problemi (KDTARP) olarak adlandırılır [28]. Eğer uğranacak düğümlerin her birinde dağıtım ve toplama faaliyeti aynı ziyaret sürecinde gerçekleştiriliyorsa, bu tür problemler EDTARP olarak adlandırılır. Bu modelde müşteriler eş zamanlı olarak ürün alıp gönderebilirler [29]. Dolayısıyla araçlar, müşteriye bir kez uğrayıp dağıtım ve toplama işlemlerini tamamladıktan sonra ayrılırlar. Gerçek dünyadaki lojistik faaliyetleri temsili açısından çalışma kapsamında EDTARP türü ele alınmıştır. Bu problem türü bir sonraki bölümde ayrıntılı olarak aktarılmıştır.

Ziyaret düğümlerinde müşteriler bazen kendilerine teslim edilen üründen memnun kalmayıp geri iade edebilirler. Bu ARP türleri ise geri dönüşlü araç rotalama problemi (GDARP) olarak isimlendirilir. Bu problem türünde müşterilerden geri dönecek ürünlerin de araca sığma zorunluluğu göz ardı edilmemelidir. GDARP'de iki tür müşteri vardır. Ürün dağıtımı yapılacak müşteriler ve geri dönüş (toplama) yapılacak müşteriler [30].

GDARP'de de amaç diğer türlerde olduğu gibi toplam rota maliyetini minimize etmektir. Problemin uygun çözümleri, dağıtım faaliyetlerinin tamamlandığı ve geri dönüş için toplanacak yüklerin araç kapasitesini aşmadığı rota setlerini içerir. GDARP'nin rota maliyeti klasik ARP modeline benzerlik gösterir. Farkı ise dağıtım, toplama ve taşınan yük miktarlarının her rota için ayrı ayrı kontrol edilme zorunluluğudur.

Müşteri düğümlerde toplanacak yükler bazen parçalanabilir. Toplam rota maliyeti düşürülebilecekse yükün bir bölümü ziyaret eden ilk nakliye aracına, kalanı da diğer

nakliye araçlarına paylaştırılabilir. Parçalı dağıtımlı araç rotalama problemi (PDARP) olarak isimlendirilen bu modelde, müşterilere birden fazla aracın uğramasına izin verilmektedir. Her bir müşteri siparişinin bölünemeyecek kadar küçük parçalara ayrılarak dağıtımına izin vermekle klasik ARP, GDARP modeline dönüştürülmüş olur. Ancak bu modelde uygun çözüm oluşturmak klasik ARP'ye göre çok daha zordur.

Müşteri düğümlerin belirli zaman aralıklarında ziyaret edilebildiği ARP türü ise zaman pencereli araç rotalama problemi (ZPARP) olarak isimlendirilir. ZPARP'nin klasik ARP'den farkı, düğümlerin "zaman penceresi" adı verilen belli zaman aralıklarında ziyaret edilebiliyor olmasıdır. Araç, ilgili düğüme son hizmet zamanından daha erken zamanda ulaşmalı ve daha erken zamanda ulaşmışsa da beklemelidir. Ancak bazı durumlarda araçlara, müşteri düğüme ulaştıklarında hizmete başlamaları yönünde izin verilebilir. Böylece düğümlerde araçlar için bekleme zamanı ortadan kaldırılmış olur. Her araç, merkez depodan 0 zamanında ayrılır ve uğradığı i müşterisine t_i süresinde hizmet eder. ZPARP'de amaç, ilgili zaman aralıklarına uygun bir şekilde hizmet sağlayarak rota maliyetini minimize etmektir.

2.2.4. Eş Zamanlı Dağıtım Toplamalı Araç Rotalama Problemi

EDTARP'deki temel yaklaşım, müşterilere ulaştırılacak tüm malların depodan gönderildiği ve müşterilerden alınacak tüm malların da yine depoya gönderileceği şeklindedir [31]. Rota boyunca "durak noktası" olarak belirlenen her bir düğümde dağıtım ve toplama faaliyetleri eş zamanlı olarak yürütülüp çözüm uzayı düğüm sayısıyla üstel olarak genişlediğinden bu tür problemler NP-zor problemleri sınıfında değerlendirilir [32]. Problemde düğüm sayısı fazlalaştıkça araç kapasitesi yetersiz kalacağından düğüm noktalarının önce gruplandırılıp ardından rotalandırıldığı iki aşamalı çözüm yaklaşımları geliştirilmiştir [33]. EDTARP ayrıca başta dolu şişelerin dağıtılıp boş şişelerin toplandığı meşrubat sektöründe olmak üzere, milk-run sistemi uygulayan birçok sektörde de kullanılabilir [8].

DTARP modelinde, rota boyunca dağıtılacak ve toplanacak yük miktarlarının araç kapasitesini aşmaması gerektiği 2.9 ve 2.10 eşitlikleriyle verilmişti. EDTARP modelinde her bir düğümde dağıtım ve toplama işlemleri eş zamanlı olarak

yürütüldüğünden, düğümlerde güncellenen yük miktarı araç kapasitesini aşmamalıdır. İlgili durum, 2.11 eşitliğinde ifade edilmiştir.

$$U_i + V_i - d_i \leq Q - p_j \qquad \forall i \in N \qquad (2.11)$$

Ayrıca rota üzerindeki her bir düğümde dağıtılacak yük miktarı, eşitlik 2.12 ile ifade edilen değerde, toplanacak yük miktarı da 2.13 ile ifade edilen eşitlik değerinde olmalıdır.

$$d_i \leq U_j \leq Q \qquad \forall i \in N \qquad (2.12)$$

$$p_i \leq V_j \leq Q \qquad \forall i \in N \qquad (2.13)$$

ARP türleri kapsamlı olarak analiz edildiğinde, bilhassa gerçek hayattaki lojistik faaliyetlerine benzerliği açısından EDTARP türü öne çıkmaktadır. Lojistik faaliyetlerin en karmaşık örneklerinden biri olarak kargo firmaları, depo merkezinden yönlendirdikleri taşıma araçlarıyla, önceki gün topladıkları siparişleri, uğradıkları düğümlere teslim etmekte ve bir sonraki gün dağıtılacak siparişleri de teslim almaktadırlar. EDTARP çözümüne yönelik, zaman pencereli ve mesafe bazlı değişik türden test problemleri de oluşturulmuştur. Çalışma kapsamında, çok noktadan oluşan çeşitli ağ modellerinde, eş zamanlı dağıtım-toplama faaliyeti yapan, sınırlı kapasitelerdeki araçlar için en düşük maliyetli rotalar belirlenmeye çalışılmıştır. Geliştirilen uygulama, EDTARP test problemleri üzerinde denenmiş, başarılı sonuçlar elde edilmiştir. Önerilen çözüm yöntemi ve uygulandığı test problemleri Bölüm 4'te detaylı olarak aktarılmaktadır.

2.2.4.1. EDTARP İçin Oluşturulan Test Problemleri

EDTARP için değişik türden, farklı yapılarda test problemleri oluşturulmuştur. Literatür araştırmalarından edinilen bilgiye göre EDTARP için oluşturulan ilk test örnekleri, 1979 yılında Christofides, Mingozzi ve Toth tarafından hazırlanmıştır [34]. Bu kişilerin isimlerinin baş harfleri kullanılarak isimlendirilen CMT örnekleri, KKARP için oluşturulmuştur. 14 örnekten oluşan CMT problemleri, 1999 yılında

Salhi ve Nagy tarafından geliştirilerek, 5 farklı yaklaşımla toplamda 70 örneğe çıkarılmıştır [35].

CMT örneklerinden farklı olarak, EDTARP için, Min [33], Montane ve Galvao [8], Dell'Amico vd. [36], ve Ai ve Kachitvicanukul [31] tarafından geliştirilen problem örnekleri de vardır.

EDTARP için oluşturulan diğer bir test örneği grubu ise Dethloff tarafından 2001 yılında oluşturulmuştur [37]. EDTARP çözümüne yönelik günümüze kadar geliştirilen hemen her yöntem, Dethloff örnekleri üzerinde test edilmiş, başarılı sonuçlar elde edebilen yöntemler, daha fazla düğüm sayısına sahip diğer test problemlerinde de başarılı olabilmişlerdir. Bu bakımdan çalışmada, önerilen çözüm yönteminin performansı, Dethloff test problemleri üzerinde test edilmiştir. Dethloff test problemleri incelendiğinde, bu problemlerin, her biri 10 örnekten oluşan CON3, CON8, SCA3 ve SCA8 isimlerinde dört gruba ayrıldığı gözlemlenmektedir. Dethloff problemleri, düğümler arası mesafeler, dağıtım/toplama talepleri ve araç kapasitelerindeki farklılık nedeniyle, değişik yapılarda çözüm uzaylarına sahiptir. 40 test probleminin her birinde, bir depo merkezi ve ziyaret edilecek 50 düğüm bulunur. Araç filosu, depo merkezinde yer alır ve araçlar seyahatlerini, depo merkezine dönerek tamamlar. Ziyaret edilecek müşteri düğümlerin alana yerleştirilmesinde iki farklı coğrafi senaryo kullanılmıştır. SCA test problemlerinde, düğümlerin tamamı [100x100]'lük bir alanda rassal olarak dağıtılmıştır. CON test problemlerinde ise düğümlerin yarısı [100x100]'lük alanda rassal olarak dağıtılırken, diğer yarısı [(100/3)x(200/3)]'lük alana dağıtılmıştır. Her bir düğümdeki dağıtım miktarları [0-100] arasında rassal olarak belirlenirken, toplanacak yük miktarları $p_i = d_i * (0.5 + \phi_i)$ şeklinde hesaplanmıştır. Formülde ϕ_i, [0-1] arasında rassal olarak türetilmiş bir değerdir. Her bir örnekteki araç kapasiteleri belirlenirken (2.14) eşitliğinden yararlanılmıştır.

$$Q = \sum_{i=1}^{M} d_i/\mu \tag{2.14}$$

CON3 ve SCA3 örnekleri için araç kapasitesini belirlemede kullanılan μ değeri 3 olarak seçilirken, CON8 ve SCA8 örnekleri için bu değer 8 olarak belirlenmiştir.

2.3. ARP ÇÖZÜMÜNDE KULLANILAN TEKNİKLER

Ulaşılan literatür kaynaklarının hepsinde, ARP üzerindeki sistematik ilk çalışma olarak, Dantzig ve Ramser'in 1959 yılında benzin dağıtımı yapan kamyon filosunun rota maliyetini en aza indirmeye yönelik çalışması verilmiştir [38]. Sonraki yıllarda bu metot formülize edilerek acil yardım araçları gibi diğer alanlarda uygulanmıştır. ARP, o yıllardan günümüze, üzerinde farklı yöntemler denenerek çözüm aranan en popüler optimizasyon problemlerinden biri haline gelmiştir.

Günlük yaşamda ARP çözümüyle lojistik dağıtım haricinde farklı birçok alanda da işler daha kısa sürelerde ve daha organize biçimde tamamlanabilmektedir. Bu alanlara örnek olarak, satış personelinin saha dağıtımı, belediyeler için çöp toplama ve sokak temizleme planı, benzin istasyonlarında akaryakıt dağıtım planı, kurye dağıtımı, okul servis araçlarının rotalanması verilebilir. Ayrıca, ARP, bir gruplama ve sıralama problemi olarak düşünülürse, geliştirilen çözümler, farklı birçok alandaki sorunlara da çözüm sağlayacaktır. Bu alanlar kısaca; üretimde tek makine için ürünler arası hazırlık sürelerinin optimizasyonu, okullarda sınav çizelgeleme, matbaalarda baskı makinelerinin aynı anda basacağı sayfa sayısının belirlenmesi, bankacılıkta şubeler arası para transferinin planlanması olarak örneklendirilebilir.

ARP için önerilen çözüm yöntemleri incelendiğinde, çözümlerin en temelde kesin yöntemler ve yaklaşık yöntemler olarak iki grupta toplandığı görülmüştür. Küçük boyutlu problemler için kesin yöntemler kullanılarak optimal sonuç bulunabilirken, problem boyutu genişlediğinde işlem sayısı katlanarak artacağından bu yöntemler tercih edilmemektedir. Yaklaşık yöntemler ise büyük problem türlerinde, daha az işlem yaparak, kısa sürede kaliteli çözümler üretebilen sezgisel yaklaşımlardır.

2.3.1. Kesin Yöntemler

ARP çözümleri için genel olarak sezgisel yöntemler denenmiş olsa da kesin matematiksel yöntemler kullanarak başarılı sonuçlar elde eden çalışmalar da mevcuttur. Bu yöntemler daha çok yol formülasyonu temellidir.

2.3.1.1. Dal ve Sınır Algoritması

Dal ve sınır algoritması yöntemiyle problemin çözüm uzayı alt bölümlere parçalanır ve her bir alt bölüm için kendi içinde optimal çözüm aranır. Bu tekniğin uygulanışında en önemli nokta keskin alt sınırların oluşturulabilmesidir. Keskin alt sınırlar, dal ve sınır ağacının daha fazla budanmasına neden olarak algoritma etkinliğini artırır.

Dal ve sınır algoritması kullanılarak minimizasyon tipli bir problemin çözümündeki temel işleyiş şu şekilde özetlenebilir [39]. Öncelikle tam sayılı programlama yöntemiyle problemin çözüm uzayı, alt gruplara ayrılır. Oluşturulan her bir alt grup için sınır değeri hesaplanarak alt ve üst sınırlardan her bir alt grup içinde çözüm değeri seçilir. Daha sonra bir alt sınır, (ilk alt gruplar arasından en küçüğü) ile alt grup diğer bölümler için seçilir. Daha önce olduğu gibi alt ve üst sınırların her ikisi de hesaplanır. Bölünmeye optimal çözüm bulunana kadar devam edilir. Optimal çözüm herhangi bir alt grup için alt sınırdan daha büyük olamaz. Problem, maksimizasyon tipli ise uygulanacak aşamalar alt üst sınır seçimi dışında minimizasyon tipli problem tipinde olduğu gibidir.

2.3.1.2. Kesme Düzlemi Algoritması

“Tamsayılı Algoritma” ya da “Kesme Düzlemi Yöntemi” adı verilen bu yöntem Gomory tarafından 1959 yılında geliştirilmiştir [40]. Kesme düzlemi algoritması da dal ve sınır algoritması gibi sürekli bir doğrusal programlama probleminin optimal çözümüyle başlar. Farklı olarak bu algoritmada dallanma ve sınırlamadan çok, kesme adı verilen özel kısıtlar oluşturularak çözüm uzayının düzenlenmesi sağlanır.

Kesme düzlemi algoritmasının temel işleyiş sürecinde, öncelikli olarak orijinal sınırlandırmalar tam sayılı hale getirilmelidir [39]. Böylelikle katsayıların tam değer olması için diğer sınırlar da değiştirilmiş olur. Ardından doğrusal programlama probleminin çözüm tablosu oluşturulur. Bu aşamada optimal çözüm değerleri olarak tam sayı değerler elde edilmişse doğrusal programlama problemi için de çözüm sağlanmıştır. Aksi halde, tamsayı tablosunda tamsayı olmayan değişkenlerden biri seçilerek yeni bir kısıtlama eklenerek kesme sağlanmış olur.

2.3.1.3. Dal ve Kesme Algoritması

Dal ve kesme algoritması, dal ve sınır algoritması ile kesme düzlemi algoritmaları birleşiminden oluşan karma bir algoritmadır. Bu algoritma, tam sayılı doğrusal programlamanın çözümü için gerekli değişkenlerin tam sayı değerlerinin bilinmediği ancak sınırlandırıldığı kombinasyonel optimizasyon yöntemidir.

Dal ve kesme algoritmaları, düzenli simpleks algoritmasını da kullanarak tamsayı kısıtı olmayan doğrusal problemler için optimal çözüm elde edebilir. Optimum çözüm bulunduğunda, eğer bu çözüm, tamsayı olarak düşünülen bir değişken için tamsayı olmayan bir değer içeriyorsa, mevcut kesirli çözüm tarafından bozulan tüm uygun tamsayı noktaları için sonraki doğrusal kısıtların bulunmasında kesme düzlemi algoritması kullanılır. Şayet böyle bir eşitsizlik bulunursa, çözüm, doğrusal programa eklenir ve böylece daha az kesirli farklı bir sonuç elde edilmiş olur. Süreç, tam sayılı çözüm bulunana ya da daha fazla kesme düzlemi oluşmayana kadar devam eder [39].

2.3.1.4. Sütun Üretme Algoritması

Sütun üretme algoritması, Dantzig ve Wolfe tarafından 1960 yılında ayrışabilir yapılı doğrusal programların çözümü için önerilmiştir. Algoritma ile doğrusal problemlerin birçoğunda başarılı sonuçlar elde edilmiş, bilhassa çizelgeleme ve rotalama problem çözümlerinde popüler bir metot haline gelmiştir. Genel itibariyle simpleks algoritması tekniğini kullanan bu yöntemde, problem yapısına bağlı olarak, bazı sütun üretme metotlarında sık sık kısmi yavaş yakınsama gözlemlenebilir [41]. Sütun üretme algoritmasında, ana problemin doğrusal probleme dönüşmesiyle elde edilen bilgi

kullanılarak rotalar üretilir ve her bir döngüde amaç fonksiyonuna uygun daha başarılı çözümler aranır. Daha başarılı çözüm elde edilemediğinde algoritma en son çözümle sonlandırılır [42].

2.3.1.5. Dal ve Değer Algoritması

Sütun üretme algoritmasıyla doğrusal programlama problemi için optimal çözüm aranırken, sütun için optimum değerlerin 0 veya 1'e eşit olması gerekir. Dal ve değer algoritmasında sütun değişkenleri için tamsayı değerlerinin sağlanmasıyla, sütun üretme optimizasyon döngüsü, sayısı belirli dal ve sınır çerçevesine dönüşür. Algoritma başlıca üç aşamadan oluşur [43].

Dallanma stratejisi: Bu aşamada dallanma amacıyla bir sonraki düğüm seçilirken "derinlik-önce" (veya "gerileme stratejisi") ve "en iyi-önce" (veya "çıkarım izleme") olarak iki farklı strateji tercih edilebilir.

Dallanma şemaları: Dallanma şemaları oluşturulurken çözüm uzayının gruplara bölünmesine bağlı olarak sütun değişkenleri de dallanır ve var olan kesirli sütun değişkenleri geçersiz sayılır.

Hızlandırma teknikleri: Algoritmanın verimliliğini artırmak için, bazı hızlandırma teknikleri geliştirilebilir. Bu teknikler, başlangıç çözümde sütun üretme optimizasyon döngüsü içinde en alt sınır olarak ve dallanma ağacı boyunca da sütunların eliminasyonu şeklinde uygulanabilir.

2.3.1.6. Dinamik Programlama

Dinamik programlama, karar verme sürecinde bir dizi işlemleri optimize eden matematiksel işlemler bütünü olarak yorumlanabilir. Dinamik programlama ile problem çözümünde, problem parçalara bölünerek her bir parça için çözüm elde edilir ve çözümler kaydedilir. İlgili problem parçaları için yeniden çözüme ihtiyaç duyulduğunda, oluşturulan çözümler genel çözüme eklenerek nihai çözüm elde edilir. Özellikle sistem analizi konusunda yaygın bir kullanım alanına sahip bu yöntem, çok

aşamalı karar verme problemleri gibi silsile halindeki problemler için de kullanılabilir [43].

Dinamik programlama, problemleri parçalara bölmenin sınırlarını oluşturan optimumluk ilkesine sahiptir. Optimizasyon problemine bağlı olarak parçaların yapısı farklılık gösterebildiğinden, dinamik programlama, her bir parçayı optimum kılmak için gerekli hesaplamaların detayını sunmaz. Ayrıntılar, araştırmacı tarafından doğaçlama olarak gerçekleştirilerek tasarlanır [40].

2.3.2. Sezgisel Yöntemler

Optimizasyon alanında sezgisellik ya da sezgisel yöntem; problemin çözüm uzayı, tüm çözümlerin değerlendirilemeyeceği kadar büyük olduğunda, çözüm arayışını kesin biçimde sınırlayan, herhangi kural, strateji, sadeleştirme ve diğer etmenlerin kullanımı olarak yorumlanmıştır [44]. Bir bakıma sezgisellik, karmaşık yapıdaki problem çözümlerinde kullanılan bir anahtardır. Optimizasyon problemi, kesin çözümü bulunamayacak kadar karmaşık yapıda olduğunda sezgisel yöntemlere ihtiyaç duyulur. Ayrıca kullanıcı için, anlaşılırlık açısından sezgisel yöntemlerin çok daha basit olması, sezgisel yöntemleri motive eden sebeplerdendir.

Sezgisel yöntemler, yakınsama özelliğine sahiptir. Kesin çözümü garanti edemezler ancak kesin çözüm yakınındaki çözümü garanti edebilirler. Sezgisel yöntemlerin başlıca aşamaları şunlardır [44]:

1. Muhtemel çözümler içinden herhangi birinin ele alınması
2. Ele alınan çözüme mümkün girişler uygulayarak çözümün değiştirilmesi
3. Çözümün değerlendirilmesi
4. Gereksiz çözümlerin elenmesi
5. İstenilen çözüme ulaşılmışsa döngünün tamamlanması aksi takdirde farklı çözümlerle işlemlerin tekrarı.

Bazı sezgisel algoritmalar, başlangıç çözümlerine bağlı olarak bölgesel optimum çözümler üretirler. Daha çok iteratif gelişme yöntemlerinde karşılaşılan bölgesel

optimum çözümler, global optimum çözümden çok uzak olabilir. Ayrıca ayrık optimizasyon problemlerinin birçoğunda uygun bir başlangıç çözümü seçimi için genel bilgiler bulunmayabilir. Basitlik ve genellik muhafaza edilmek koşuluyla bölgesel arama algoritmalarının bazı dezavantajlarını giderebilmek için farklı teknikler uygulanabilir [9]:

ARP için uygulandığında başarılı çözümler üreten sezgisel algoritmaların şu dört özelliğe sahip olması gerektiği vurgulanmaktadır [16]:

1. Kesinlik: Algoritmanın ürettiği sonucun problemin optimal çözümüne oranıdır. ARP türlerinin hemen hepsinde en iyi çözüm belirlenemediğinden, optimal çözüm olarak literatürde bulunan en iyi çözüm referans alınabilir.
2. Hız: Kullanılan sezgisel yöntemin problem için çözüm bulma zamanıdır.
3. Basitlik: Kolay anlaşılır ve hızlı kodlanabilir olmasıdır. Clark ve Wright algoritmasını popüler kılan sebep de geliştirdikleri algoritmanın kolay anlaşılabilir olması ve kısa sürede kodlanabilmesidir.
4. Esneklik: Başarılı bir sezgisel yöntem, farklı kısıtlamalara kolayca entegre olarak gerçek sistemlere de uygulanabilir olmalıdır.

Sezgisel yöntemler genel olarak klasik sezgisel yöntemler ve Meta-sezgisel yöntemler olarak iki grupta toplanabilir.

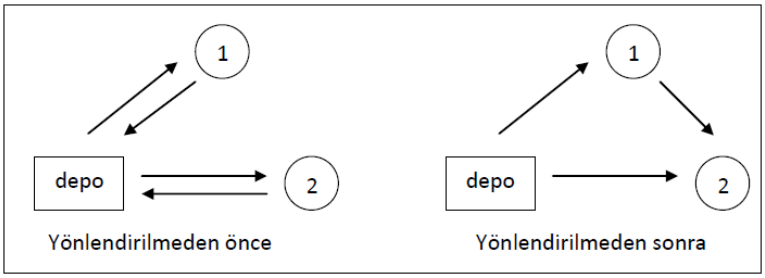
2.3.2.1. Klasik Sezgisel Yöntemler

Klasik sezgisel yöntemler; yapısal, iyileştirmeli ve çok aşamalı yöntemler olarak üç gruba ayrılır [17].

Yapısal sezgisel yöntemler, KKARP için çözüm arayan ilk sezgisel metotlardandır. Günümüzde bile ARP türlerinin birçoğunda çözüm türeten araştırmacılar için, temel yapı niteliğindedir. Bu algoritma işleyişinde, amaç fonksiyonu göz önünde bulundurularak adım adım optimal çözüm bulunmaya çalışılır, ancak genellikle bulunan çözümler üzerinde iyileştirmeler yapılmaz. İlk olarak boş çözümler oluşturulur, ardından tüm araçlar rotalanana kadar her döngüde, bir veya daha fazla

düğüm rotaya eklenerek çözüm tamamlanır. Düğümlerin eklenmesine bağlı olarak bu yöntemlerin, sıralı ve paralel yöntemler olmak üzere iki çeşidi bulunur. Sıralı yöntemlerde bir zaman diliminde yalnızca bir rota için genişleme yapılırken, paralel yöntemlerde tüm rotalar eş zamanlı olarak göz önünde bulundurulur. Yapısal sezgisel yöntemler; başlangıç kriteri, seçim kriteri ve ekleme kriterine göre gruplara ayrılabilir. Ancak KKARP için çözüm üreten iki farklı kriter bulunur. Bunlardan birinde tasarruf kriteri kullanılarak mevcut rotalar birleştirilmeye çalışılır, diğerinde ise bir ekleme maliyetiyle düğümler, adım adım rotalara eklenir.

Tasarruf kriterini kullanan ve ARP çözümünde en bilindik yöntem Clarke ve Write tarafından 1964 yılında geliştirilen tasarruf algoritmasıdır. Algoritma, araç sayısının değişebildiği problem çözümlerine uygulanır. Dolayısıyla bu yöntem kullanılarak araç rotaları oluşturulabildiği gibi, seyahat edecek araç sayısı da belirlenmiş olur. Tasarruf algoritması ile kısa zamanlarda çözümler alınabilmesine karşın, başarılı çözümler elde edilememektedir. Ayrıca algoritma, mevcut rotalar üzerinde değişiklik yapılması zor olduğundan, esnek bir yapıya sahip değildir. Dolayısıyla probleme sonradan ilave edilecek yeni kısıtlamalar çözümü daha da kötüleştirmektedir [16]. Tasarruf algoritmasının temel konsepti Şekil 2.6'da gösterilmektedir.



Şekil 2.6. Tasarruf algoritması konsepti [17].

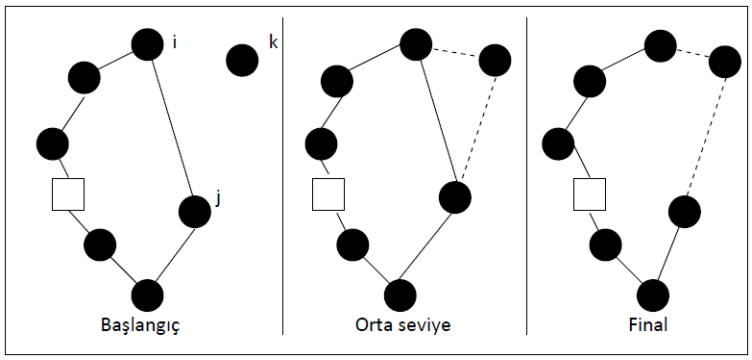
Tasarruf algoritmasında $(0, \ldots i, 0)$ ve $(0, \ldots j, 0)$ şeklinde iki rota, $(0, \ldots i, j, \ldots, 0)$ olarak birleştirildiğinde mesafe tasarrufu, 2.15 eşitliğindeki gibi olur.

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{2.15}$$

Tasarruf algoritmasında araç kapasitesi dolana kadar düğümler rotaya eklenirse rotalar artarda sıralı şekillenmiş olur, rotalar eş zamanlı doldurulursa paralel şekillenmiş olur [17].

Tasarruf kriterini kullanan diğer bir yapısal sezgisel algoritma 1989 yılında Desrochers ve Verhoog tarafından geliştirilen eşleme tabanlı tasarruf algoritmasıdır. Algoritma özellikle çok noktalı KKARP çözümlerinde kullanılır. Bu yönteme göre tasarruf değeri s_{ab} bir kerede hesaplanamayıp, her döngüde a ve b rotalarının birleşiminden elde edilir. s_c, c rotasına ait çözüm kümesi ve $h(s_c)$ bu noktalara ait GSP'nin optimum çözümü ise, tasarruf miktarı, $s_{ab} = h(s_a) + h(s_b) - h(s_a\ U\ s_b)$ ile bulunur. Algoritmaya göre rotalar, tasarruf değerlerine göre, büyükten küçüğe eşlenerek uygun biçimde birleştirilmektedir [45].

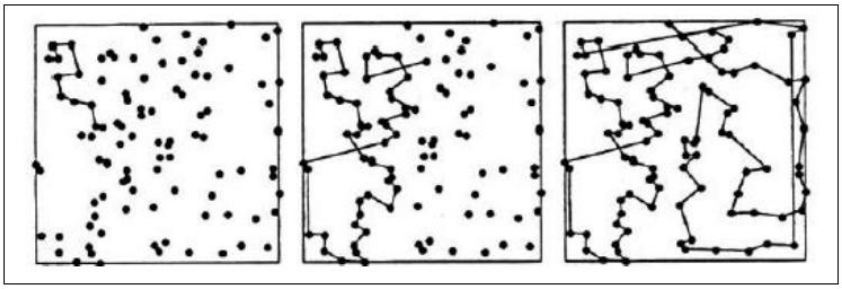
Ekleme sezgisel algoritması, düğümlerdeki yükleme miktarları bilinen ancak araç sayısının belirsiz olduğu KKARP çözümüne yönelik geliştirilmiş bir yöntemdir. Algoritma işleyişinde öncelikle boş rotalar oluşturulur ve her bir döngüde mevcut rotalara, en az maliyet artışına sebep olacak düğüm eklenir. Şekil 2.7'de ekleme sezgiselline ait bir uygulama örneği sunulmaktadır.



Şekil 2.7. Ekleme sezgiselinin uygulama örneği [1].

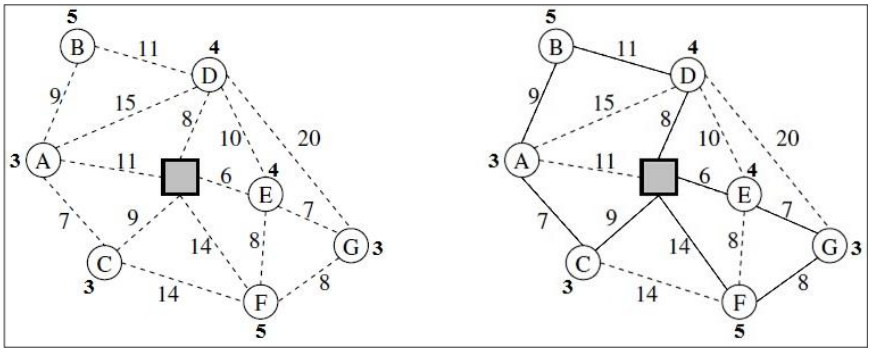
Ekleme sezgisel algoritmasında düğüm ekleme işlemi için iki farklı yaklaşım geliştirilmiştir. 1976 yılında Mole ve Jameson'un önerdikleri yöntemde, bir döngüde yalnızca bir rota için işlem yapılmaktadır. Christofides, Mingozzi ve Toth ise 1979 yılında geliştirdikleri metotla, Mole ve Jameson yöntemine sıralı ve paralel rota oluşturan yordamlar uygulamışlardır [2].

En yakın komşuluk (EYK) sezgisel algoritması GSP çözümüne yönelik geliştirilen en basit yapısal sezgisel yöntemdir. Yöntem ayrıca Hamilton yol problemine de uygulanabilir. Başarı performansı düşük bu algoritmada, başlangıçta rastgele bir düğüm seçilir ve her döngüde, seçilen düğüme, daha önce herhangi bir rotaya atanmamış en yakın komşu düğümün eklenmesiyle rota tamamlanır [46]. Seçilen düğüme en yakın mesafede iki komşu düğüm varsa her bir düğüm için süreç oluşturularak çözüm dalları oluşturulur. Şekil 2.8'de en yakın komşu sezgiseliyle çözüm bulunan bir örnek uygulama sunulmaktadır.



Şekil 2.8. EYK sezgiseliyle bulunan örnek bir çözüm [47].

Şekil 2.9'da ise 15 birim eş taşıma kapasiteli iki araçla, düğümlerdeki talep miktarları belirli bir ağ ortamında KKARP için en yakın komşu sezgiseliyle oluşturulan örnek bir çözüm sunulmaktadır.



Şekil 2.9. EYK sezgiseliyle KKARP çözüm örneği [47].

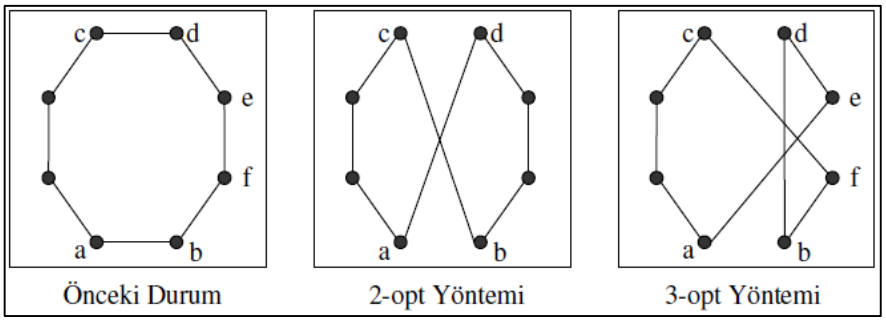
En kısa yol algoritması ise yalnızca bir depo merkezi ve yalnızca bir hedef düğüm bulunan KKARP türlerine uygulanabilir. Problem türüne göre, ağ üzerinde bulunan düğümler, araç kapasitesi dikkate alınarak, birbirlerine; zaman, maliyet, mesafe gibi

farklı kriterlerle birleştirilirler. En kısa rotanın hesaplanmaya çalışıldığı bu algoritmada, merkez düğümle diğer düğümler arasındaki bağlantı için Dijkstra yaklaşımı geliştirilmiş, ancak ağ içindeki herhangi iki düğüm arasındaki en kısa yolu belirleyen Floyd yaklaşımı daha çok benimsenmiştir [40].

İyileştirmeli sezgisel yöntemler, oluşturulan mevcut çözümleri kullanarak daha iyi çözüm arayışındadırlar. Bu iyileştirme, muhtemel çözüm rotaları arasında yol ya da düğüm değiş tokuşuyla sağlanır. Eğer iyileştirme işlemiyle daha başarılı bir çözüm elde edilmişse, bir sonraki döngüde türetmede kullanılacak çözüm, bulunan bu yeni çözüm olacaktır. Eğer iyileştirme işlemleriyle daha iyi sonuçlar türetilemiyorsa, yerel optimum tanımlanır. Algoritma yardımıyla bir tek rota için iyileştirme yapılabileceği gibi, aynı anda birden fazla rota için iyileştirme de sağlanabilir [48].

ARP çözümlerinde tek rotanın iyileştirilmesi amacıyla tasarlanan ve literatürde en sık kullanılan yöntemlerden biri olan λ-opt yöntemi, Min tarafından 1965 yılında geliştirilmiştir. Bu yöntemde, oluşturulan rota üzerinde seçilen λ adet doğru, rotadan çıkarılarak mümkün olan tüm permütasyonlarda, rotanın farklı düğümlerine eklenerek daha iyi çözümler oluşturulmaya çalışılır. Mevcut çözümden daha başarılı çözüm bulunduğunda, bu yeni rota çözüm olarak kabul edilir. λ-opt yöntemiyle ARP'de yaklaşık olarak $O(n^{\lambda})$ sürede çözüm bulması beklenir. Ancak yöntem üzerinde değişiklikler yapılarak çözüm bulma süresi kısaltılabilir [49].

Tek rota iyileştirmeli sezgisel algoritmalarda oluşturulan bir çözüm rotası üzerinden iki yol silinerek mevcut rota iki parçaya bölünebilir. Bu iki parça ters çevrilip birleştirilerek 2-opt ve yine rota üzerinde üç yol silinip, farklı yollarla birleştirilmesiyle de 3-opt elde edilebilir [50]. Şekil 2.10'da mevcut bir rotaya 2-opt ve 3-opt algoritmasının uyarlaması gösterilmektedir.



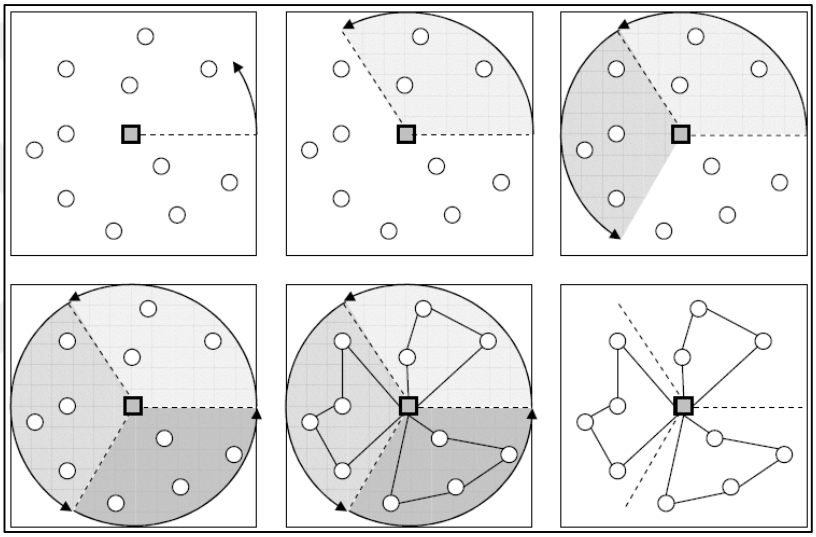
Şekil 2.10. 2-opt ve 3-opt algoritmalarının bir rotaya uyarlanması [1].

Çok rota iyileştirmeli sezgisel yöntemlerde, ARP çözüm rotaları arasında düğüm alışverişi yapılarak düğüm sıralamalarının değiştirilmesiyle daha başarılı çözümler elde edilmeye çalışılır. Yöntem, çeşitli çözümler içinden alınan rotalar üzerinde eş zamanlı olarak uygulanır. EYK ilişkisine göre oluşturulan rotalardaki düğümlerin mutasyonu olarak ifade edilebilen Van Breedem yöntemi, dairesel permütasyonla sıralanan çözüm rotalarından seçilen düğümleri farklı rotalara atayarak yeni çözümler türeten Thompson ve Psaraftis yöntemi ve düğüm ve yolların farklı rotalar arasında yer değiştirdiği Kindervater ve Savelsbergh yöntemi en çok tercih edilen çok rota iyileştirmeli sezgisel algoritmalardır [51].

Çok aşamalı sezgisel yöntemler, KKARP çözümlerini genel olarak iki aşamada gerçekleştiren yöntemlerdir. Bu aşamalardan biri olarak kümeleme aşamasında, talep noktaları, araç kapasiteleri de göz önünde bulundurularak alt setlere ayrılır. Rotalama aşamasında ise araçların seyahatleri boyunca ziyaret edecekleri düğümlerin sırası belirlenir. İlk aşamada kümeleme ardından rotalama yapılabildiği gibi, tersi de mümkündür. Genellikle ikinci aşamada yapılan işlemler, ilk aşamadaki işlemlere geri besleme yaparak sürdürülmektedir [1].

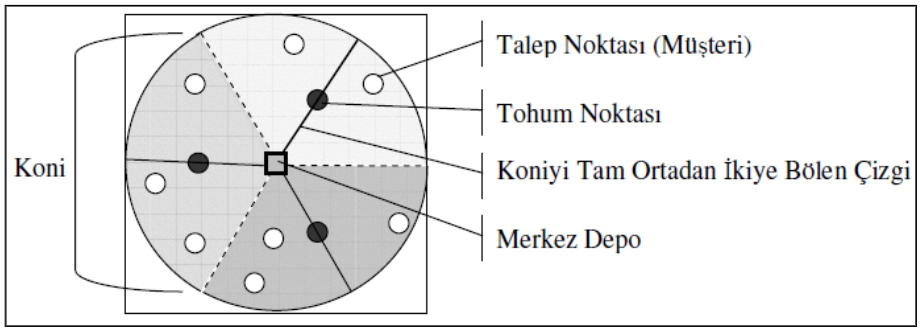
İlk olarak rotalama yapılan yöntemlerde ARP, kapasite kısıtı olmayan, klasik GSP gibi düşünülür ve her düğümü kapsayan tek bir rota oluşturulur. Oluşturulan rota, belirlenen araç adedince alt rotalara bölünerek KKARP için çözüm aranır. Literatür incelendiğinde, iki aşamada tamamlanan bu yöntem kullanılarak çözüm bulan çalışmaların yok denecek kadar az sayıda olduğu görülmektedir. Beasley tarafından 1983 yılında sunulan bir modelde, rotalama aşamasının, klasik en kısa yol problemi olduğu savunulmuştur [2].

İlk aşamada kümeleme yapılan yöntemlerde, düğümlerin uygun kümelere dağıtılabilmesi için farklı teknikler geliştirilmiştir. 1974 yılında Gillet ve Miller tarafından geliştirilen süpürme algoritmasıyla, her bir küme için seçilecek düğümler, depo merkezli bir doğrunun döndürülmesiyle belirlenir. Döndürme sürecinde, doğrunun üzerinden geçtiği düğümler ilgili kümeye atanır ve mesafe ya da kapasite kısıtı aşıldığında küme tamamlanmış olur ve yeni bir küme için döndürme devam eder. Süpürme yöntemi tasarruf algoritmasına kıyasla daha basit uygulamaya sahiptir. Ancak çoğu ARP türü için daha yavaştır [16]. Şekil 2.11'de süpürme algoritması işleyiş süreci gösterilmektedir.



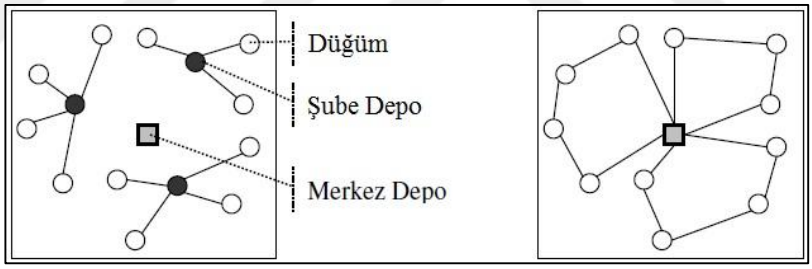
Şekil 2.11. Süpürme algoritması adımları [1].

Fisher ve Jaikumar'ın 1981 yılında KKARP çözümü için geliştirdikleri atama tabanlı kümeleme algoritmasında ise kümeler geometrik olarak değil, genelleştirilmiş atama problemi yardımıyla oluşturulur [16]. Algoritma işleyişinde depo merkezi, koni şeklindeki yapının tepesine oturtulmuş, düğümler ise koni üzerine yerleştirilmiştir. Oluşturulan her bir küme için rotalama, GSP çözüm yöntemleriyle belirlenmektedir [49]. Şekil 2.12'de Fisher ve Jaikumar yaklaşımı şematize edilmektedir.



Şekil 2.12. Fisher ve Jaikumar yaklaşımının genel yapısı [1].

Bramel ve Smichi-Levi konum tabanlı algoritmasında, özetle, düğüm sayısı ve düğümlerdeki talep miktarları belli bir ağ ortamında, inşa edilecek şube depoların konumunu belirlemeye çalışan, kapasite kısıtlı çoklu tesis lokasyonu belirleme problemi için çözüm aranmaya çalışılır. Bu çözümle, düğümlerin depo merkezlerine göre kümelenmesi de sağlanmaktadır [26]. Şekil 2.13'te Bramel ve Smichi-Levi algoritmasının kümeleme işlem adımları sunulmaktadır.



Şekil 2.13. Bramel ve Smichi-Levi algoritması örneği [1].

Christofides, Mingozzi ve Toth tarafından 1979 yılında geliştirilen kısaltılmış dal-sınır yönteminde basit bir arama ağacı yapısı kullanılmaktadır. Ağacın her seviyesinde yalnızca bir adet dal bulunmakta ve her döngüde rotalanmamış bir düğüm seçilerek, seçilen bu düğümün içinde bulunacağı muhtemel rotalar üzerinden sonuç bulunmaya çalışılmaktadır.

Taç yaprağı algoritması ise süpürme algoritmasının geliştirilmiş halidir. Bu yöntemde oluşturulan her bir rota, taç yaprağı ismiyle isimlendirilmekte ve birbirleriyle ortak düğümleri kullanabilen bir rota demeti oluşturulmaktadır. KKARP çözümü için bu demet içinden rastgele seçimler yapılır ve rotalar geliştirilir [49].

Taç yaprağı algoritması, farklı kısıtlamaların kolayca eklenebildiği esnek olma avantajına rağmen diğer klasik sezgisel yöntemlere göre çok karmaşık ve zor kodlanabilen bir algoritmadır [16].

Klasik sezgisel algoritmaların ARP çözümündeki performansları değerlendirildiğinde, bu algoritmaların genel itibariyle basit yapılarına rağmen kısa sürelerde hızlı sonuçlar verdikleri, ancak çözüm kalitesi anlamında başarılı olmadıkları savunulmaktadır. Ayrıca bu yöntemlere güncel yaşamda karşılaşılan kısıtlamaların ilave edilmesinin çok güç olduğu vurgulanmaktadır. Çizelge 2.1'de klasik sezgisel algoritmaların farklı kıstaslara göre performans değerlendirmeleri sunulmaktadır [16].

Çizelge 2.1. ARP çözümlerinde klasik sezgisel yöntemlerin değerlendirilmesi.

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

2.3.2.2. Meta-sezgisel Yöntemler

"Meta-sezgisel" kavramı literatürde ilk kez Glover tarafından 1986 yılındaki çalışmasında kullanmıştır. Dilimizde "modern sezgisel" veya "yapay zeka yaklaşımı" olarak da tercüme edilen bu sözcük, Yununca "daha üst derecede, daha modern olanı" anlamlarına gelen "meta" öneki ile, İngilizce'de "buluşsal, sezgisel" anlamlarını karşılayan "heuristic" kelimesinin birleşiminden oluşur [44]. Bu yöntemler, kombinatoryal problem çözümleri için geliştirilen ve problemin çözüm uzayını etkili biçimde tarayabilmek amacıyla, değişik türdeki sezgisellerle entegre edilerek tasarlanan, döngüsel problem çözme prosedürleridir. Bu algoritmaların işlem sürecinde, her döngüde muhtemel çözümlerden yalnızca biri ya da belirli çözüm grubu ele alınarak, yeni çözümler türetilir. Çoğu Meta-sezgisel yöntem, belirsiz arama

uzayını bilinçli şekilde tarar [52]. Bu sebepten ötürü, Meta-sezgisel yöntemler, modern sezgisel yöntemler ya da yapay zeka yöntemleri olarak da isimlendirilmektedir [9]. Bu yöntemlerin en temel özelliği, yalnızca belli bir problem modelinde değil, tüm optimizasyon problem türlerinde uygulanabilecek seviyede esnek olmalarıdır. Literatür kaynakları incelendiğinde, birbirinden çok farklı alanlarda ve farklı problem türlerinde, Meta-sezgisel yöntemler kullanılarak başarılı sonuçlar elde edilen çalışmalar görülebilir. Dolayısıyla problem çözümlerine genel olarak uygulanabilen bu algoritmalar üzerindeki çalışmalar, gün geçtikçe artmakta, 50 yıla yakın süre içerisinde; biyoloji, fizik, zooloji, bilgisayar, karar verme gibi birçok alanda kullanılarak çözüm önerileri oluşturulmaktadır.

Meta-sezgisel yöntemler genel olarak, gelişmiş komşuluk arama (GKA) kurallarını, hafıza yapılarını ve çözüm kombinasyonlarını birleştirirler. Sezgisel yöntemlere oranla daha başarılı olmalarına karşın, çözüm zamanı daha uzun sürebilmektedir [49]. Problem çözümü için Meta-sezgisel yöntem belirlendikten sonraki ilk adım, yöntemin temel bileşenlerinin probleme doğru biçimde uyarlanmasıdır [53]. Diğer bir ifadeyle Meta-sezgisel yöntemlerin parametre değerleri, problem yapısına göre değişebilmektedir. Zira bu yöntemlerin performansını belirleyen en temel konu, seçilen parametre değerleridir.

Meta-sezgisel yöntemler, klasik yerel arama yöntemlerinden, karmaşık öğrenme yöntemlerine kadar, geniş bir alanı kapsar ve çözüm uzayındaki yerel optimum noktalarından kurtulmak için değişik mekanizmaları kullanırlar. Probleme özgü değillerdir, ancak üst seviye stratejilerle kontrol edilerek probleme özgü bilgi kullanımına izin verirler. Ayrıca bazı Meta-sezgisel yöntemler, daha başarılı çözüm bulabilmek için arama sırasında elde edilen bilgiyi de kullanırlar [46]. Meta-sezgisel algoritmalar için oluşturulma ve literatürdeki uygulanış biçimlerine göre farklı sınıflandırmalar yapılabilir. Bu sınıflandırmalardan birinde bu algoritmalar şu şekilde sınıflandırılmıştır [54]:

1. Doğadan esinlenerek geliştirilenler / Doğadan esinlenmeden geliştirilenler
2. Popülasyon tabanlı algoritmalar / Tek nokta algoritmalar
3. Dinamik amaç fonksiyonlu olanlar / Statik amaç fonksiyonlu olanlar

4. Tek komşu yapılılar / Çok komşu yapılılar
5. Hafıza kullananlar / Hafıza kullanmayanlar

Farklı bir araştırmada ise bu algoritmalar şu şekilde sınıflandırılmışlardır [16]:

1. Yerel arama: Tavlama Benzetimi (TB), Tabu Arama (TA)
2. Popülasyon tabanlı arama: Genetik Algoritma (GA)
3. Öğrenme mekanizması: Yapay Sinir Ağları (YSA) ve Karınca Koloni Optimizasyonu (KKO)

Literatürde ARP çözümleri için birçok farklı sezgisel yöntem kullanılmıştır. Kullanılan bu yöntemler ve elde edilen sonuçlar bir sonraki bölümde aktarılmıştır.

2.4. ARP ÇÖZÜMÜ İÇİN GELİŞTİRİLMİŞ SEZGİSEL YAKLAŞIMLAR

Bu bölümde, literatürde ARP çözümlerine yönelik geliştirilen sezgisel çözüm yaklaşımları incelenmiştir. İlk olarak, klasik ARP ve EDTARP haricindeki ARP türleri için geliştirilen yöntemler incelenmiş, ardından da EDTARP çözümü için geliştirilen sezgisel yaklaşımlar ele alınmıştır.

2.4.1. Klasik ARP ve ARP Türleri İçin Önerilen Sezgisel Çözümler

Balıkesir Ordudonatım Okulu, personel servis araçlarının rota optimizasyonuna yönelik bir çalışmada, çözüm için tasarruf ve rassal tasarruf algoritmaları ile VRP328 yazılımı kullanılarak çözümler elde edilmiş ve bu çözümler tek rota iyileştirme algoritmalarıyla geliştirilmeye çalışılmıştır. Sonuçlar mevcut durumla karşılaştırıldığında daha iyi sonuçlar alındığı gözlemlenmiştir [17]. Zaman pencereli ARP çözümleri için geliştirilmiş yöntemlerde, problem parametrelerindeki belirsizlik ve kısıtlamaların ihmal edildiği düşünülebilir. Bu nedenle üretilen çözümler çoğu kez uygulama aşamasında geçerliliğini yitirir. Bulanık kümeler ve olabilirlik teorilerinden faydalanarak, bulanık karar ortamında da kullanılabilecek zeki bir bulanık programlama modeli önerilmiştir. Önerilen modelde, belirsizliklerin ve esnekliklerin çözümünde bulanık kümeler kullanılmıştır. Ayrıca önerilen modele çözüm için KKO

temelli bir algoritma tasarlanmıştır. Algoritma sürecinde bulanık veriler, kesin verilerin genelleştirilmiş şekli olarak ele alındığından, yöntem, kesin ya da bulanık veri durumlarında da çözüm üretebilmektedir [32].

Araçların birden fazla rotada kullanılabilmesi fikri ilk olarak 1987 yılında Salhi'nin doktora tezinde ortaya atılmıştır. Tez çalışmasında problem, sezgisel yöntemle çözülmüş ve literatüre tanıtılmıştır. Bu problem daha sonraları farklı tekniklerle de çözülmüştür. Literatürde birkaçı haricinde tüm çalışmalarda zaman kısıtı göz ardı edilmiş ve sezgisel yöntemlerle çözüm aranmıştır. Çok kullanımlı ve zaman pencereli ARP olarak isimlendirilen bu probleme, günlük yaşamda, raf ömrü kısa olan ürünlerin dağıtımı gibi kısa sürede dağıtım gerektiren durumlarda karşılaşılır. Karma tamsayılı doğrusal programlama modeli kullanılarak çözüm önerilen bir çalışmada, model, literatürde oluşturulmuş değişik boyutlardaki test problemleri üzerinde denenmiş ve en iyi çözüm için işlem zamanları ve sonuçları üzerinden karşılaştırılmıştır [55].

2.4.2. EDTARP İçin Önerilen Sezgisel ve Meta-sezgisel Çözümler

Bu çalışmada ARP'nin popüler türlerinden biri olarak ele alınan EDTARP çözümü için ilk çalışma, 1989 yılında Min tarafından ortaya atılmıştır [33]. Min'in çözüm aradığı ağ modeli, bir depo ve 22 kütüphaneden oluşmaktadır. Geliştirdiği yaklaşım; müşteri düğümleri için kümeler oluşturmak, oluşturulan her küme için araçlar atamak ve her araç için ait olduğu kümede rota belirlemek tarzında üç aşamadan oluşmaktadır [33]. Dethloff ise ekleme esasına dayanan bir sezgisel yöntem geliştirmiş, Min'den daha başarılı sonuçlar elde etmiştir [37]. Bianchessi ve Righini ise Dethloff'ın geliştirdiği formulasyonu TA algoritmasında kullanmışlar ve Dethloff'tan daha başarılı sonuçlar elde etmişlerdir [56]. Dağıtım ve toplamalı ARP için geliştirilen bir diğer yöntem ise Nagy ve Salhi tarafından ileri sürülmüştür [29]. Algoritma, EDTARP için kullanışlı olduğu gibi, müşterilerin bazılarında yalnızca dağıtım bazılarında ise yalnızca toplama yapılan karışık durumlar için de genel bir çözüm öneriyordu. Model, Min'in geliştirdiği yönteme nazaran daha başarılı sonuçlar elde edebilmekteydi [5]. Tang ve Galvão yerel arama için tur bölümleme yapan ve süpürme algoritması kullanan iki sezgisel çözüm önermiştir [8]. Bu alanda farklı yıllarda çok sayıda çözüm

önerilse de en meşhur olanı Christofides, Mingozzi ve Toth'un 1979 yılında geliştirdikleri algoritma olarak düşünülebilir.

EDTARP çözümü için önerilen ilk Meta-sezgisel yaklaşım, 2005 yılında Crispim ve Brandao tarafından geliştirilmiştir [57]. Önerilen yöntem, tabu arama ve Değişken Komşuluk Arama yöntemlerine dayalı karma bir algoritmadır. Öncelikle süpürme yöntemi kullanılarak başlangıç çözümleri oluşturulur. Oluşturulan başlangıç çözümlerinin geçerliliği, problem kıstasları için kontrol edilir. Geçersiz çözümler varsa çözüm rotalarındaki sıra değiştirilerek uygun çözümler oluşturulur. Çözümlerin iyileştirme aşamasında ise ekleme ve yer değiştirme yöntemleri kullanılır. Farklı bir çözüm yaklaşımında ise, Ai ve Kachitvichyanukul geliştirdikleri matematiksel modeli PSO algoritmasıyla birlikte denemişlerdir [31]. Zhang vd., zaman pencereli araç rotalama problemi için tabu arama ve yapay arı koloni algoritmaları ile karma bir yaklaşım önerdiler [58]. Lojistik dünyasının güncel ihtiyaçlarını karşılamak üzere önerdikleri yaklaşımı önce Solomon'un test verileriyle karşılaştırıp başarılı sonuç elde ettiklerini gördüler ve bu alanda yepyeni karşılaştırma problemleri oluşturdular. Yousefikhoshbakht vd., EDTARP için geliştirilen çözümleri incelediğinde Meta-sezgisel çözümlerin daha başarılı olduklarını fark etmiş ve modifiyeli tabu arama algoritmasıyla elit karınca sistemini kombine ederek karma bir sistem oluşturmuşlardır [59]. Mancini ise gerçek hayatta karşılaşılabilen; çok depolu, çok periyotlu ve heterojen kapasiteli bir taşıma filosunun, eş zamanlı dağıtım toplamalı durumunu örneklendirmiştir [60]. Böyle bir ağ modelinde karışık tam sayılı programlama formulasyonu sunmuş ve Meta-sezgisel tabanlı, uyarlamalı geniş komşuluk arama yaklaşımı önermiştir. Salhi vd., da benzer bir problemi ele almış; çözüm için, komşuluk ve yerel arama operatörlerine ilave olarak yeni özellikler içeren değişken komşuluk arama (DKA) yöntemini eklemişlerdir [61]. Çözüm için geliştirdikleri algoritma ile literatürde yayınlanan 26 örneğin 23'ünde daha başarılı sonuçlar elde etmişlerdir. Kalaycı ve Kaya ise karınca koloni sistemi (KKS) ve değişken komşuluk aramaya dayalı karma bir Meta-sezgisel yöntem geliştirmişlerdir [62]. Yöntemde değişken komşuluk aramanın yerel aramadaki başarısı, karınca koloni siseminin güçlü bellek yapısıyla takviye edilerek algoritma güçlendirilmeye çalışılmıştır. Avcı ve Topaloğlu, EDTARP için önerilen çözüm yaklaşımlarının aynı zamanda karışık dağıtım toplamalı araç rotalama problemi için de uygulanabilirliğini savunmuşlar,

tavlama benzetimi algoritması ile değişken komşuluk arama algoritmalarını kullanılarak karma bir yaklaşımın geliştirmişlerdir [63]. Rieck vd., EDTARP çözümü için "araç-akış" ve "hammadde-akış" isimlerinde, karışık tam sayılı lineer model formülasyonu geliştirmişlerdir [64]. Ayrıca geliştirdikleri modellerin verimliliğini artırmak için, etki alanını azaltan ön işleme teknikleri ve etkin kesme düzlemleri tasarlamışlardır. Zhu vd., etkin bir yerel arama için üç aşamalı model tasarlamışlardır [65]. İlk olarak başlangıç çözümleri için en yakın komşuluk yöntemini kullanmışlardır. İkinci aşamada, başlangıç çözümleri içinde çoklu operatör kullanarak, amaç fonksiyonu ve ceza puanı parametrelerinin değişen değerleriyle, yerel optimal çözümler aramışlardır. Üçüncü aşamada ise yerel optimal çözümlerden en başarılı çözümü seçmişlerdir. Sayyah vd., klasik karınca koloni optimizasyonu yerine, ekleme, yer değiştirme ve 2-Opt hareketleri içeren etkili KKO algoritmasını geliştirmişlerdir [66]. Önerdikleri metot, literatür örnekleriyle test edildiğinde; başarılı sonuçlara ulaşılabilmiş ve aynı zamanda tabu arama, geniş komşuluk arama, parçacık sürü optimizasyonu ve genetik algoritma gibi diğer Meta-sezgisel algoritmalarla da rekabet edebildiği görülmüştür. EDTARP çözümüne yönelik bir çalışmada, Ankara ilinde faaliyet gösteren bir kargo şirketinin daha düşük maliyetle dağıtımını sağlamak üzere, bu şirketin özelliklerine bağlı olarak oluşan kısıtlamalar da dikkate alınarak, matematiksel model oluşturulmuş ve GAMS (The General Algebraic Modeling System) paket programı kullanılarak çözüm geliştirilmiştir. Elde edilen sonuçlar, mevcut durumdaki rota mesafeleri, araç sayısı, araç doluluk oranları gibi verilerle karşılaştırılmıştır [67]. Yücenur ve Demirel, çok depolu araç rotalama problemi çözümü için genetik algoritma ve karınca koloni optimizasyonu yöntemlerini bir arada kullanarak karma bir Meta-sezgisel yöntem geliştirmişler ve iki aşamalı bir çözüm oluşturmuşlardır [47]. İlk aşamada, Thangiah ve Salhi'nin 2001 yılında ortaya koydukları genetik kümeleme yönteminin geliştirilmiş hali ile düğümleri gruplandırmışlar; ikinci aşamada ise Gambardella ve Dorigo tarafından 1997 yılında önerilen karınca koloni optimizasyonu yaklaşımı ile araçları rotalandırmışlardır. Önerdikleri yaklaşımı, literatürde kullanılan problem setleri ile test etmişler ve elde edilen sonuçları, literatür çözümleriyle karşılaştırılmıştır.

Literatür çalışmaları incelendiğinde, EDTARP çözümü için genel olarak Meta-sezgisel yöntemlerin tercih edildiği görülmektedir. Bunun sebebi, Meta-sezgisel

yöntemlerin arama uzayında yararsız çözümlerle oyalanmadan, geçerli çözüm bölgelerini aktif biçimde tarayabilmesidir. Çalışma kapsamında, az sayıda parametresi ve kodlama kolaylığından dolayı yapay arı koloni algoritması tercih edilmiştir. Bu algoritma, arama bölgesi sınırlarında rassal çözümler oluşturarak çözüm uzayına dağılabilen ve ardından zeki arama stratejisiyle etkin bir yerel arama özelliği olan meta-sezgisel bir yöntemdir.

BÖLÜM 3

YAK ALGORİTMASI

Bal arılarının, yiyecek arama, buldukları besin kaynağından nektar toplama, toplanan nektarı kovana taşıma ve besin kaynağının coğrafi konumunu diğer bal arılarına tarif etme tarzındaki sosyal davranışları, sürü zekasının en popüler örneklerindendir. Bal arılarının kendi aralarında oluşturduğu organizasyon ve kurdukları bu iş bölümü, farklı alanlardaki birçok araştırmacının ilgisini çekmiştir. Örneğin bal arısı kolonileriyle ilgili çalışmaları sonucu, arıların dans ederek, birbirlerine yön tarif ettiklerini keşfeden Avusturyalı biyolog Karl Von Frisch, 1973 Nobel Fizyoloji veya Tıp ödülüne layık görülmüştür. Biyolojiden kimyaya, sosyolojiden sağlığa hemen her alanda üzerinde çalışmalar yürütülen bu canlılar, teknik branşlarda da ilgiyle takip edilmektedir. Örneğin bal arısı kolonilerini referans alarak bilgisayar ağlarında sunucu-istemci yapılarını geliştirmek üzere yapılmış çalışmalar mevcuttur.

Bal arılarının, özellikle yiyecek toplama davranışları, sürü zekası temelli farklı Meta-sezgisel algoritmalarının oluşturulmasına da zemin hazırlamıştır. Arılar algoritması (bees algorithm), arı koloni optimizasyonu (bee colony optimization) ve YAK bu algoritmalara örnek verilebilir. Bu bölümde, öncelikle YAK algoritmasının oluşturulmasına ilham olan, bal arılarının sosyal yaşamdaki yiyecek arama davranışlarına kısaca değinilmiş, ardından YAK algoritması, adımlarıyla birlikte detaylı olarak anlatılmıştır.

3.1. BAL ARILARININ YİYECEK ARAMA DAVRANIŞLARI

Gerçek dünyadaki bir bal arısı kolonisinde, yiyecek arama faaliyeti, bu iş için özelleşmiş arılar tarafından yürütülür. Yiyecek arama alanında görevli arılar, aralarında herhangi bir hiyerarşi olmadan organize olmuş, iş bölümü kurarak kolektif zekayı ortaya çıkarmışlardır. Zira iş bölümü yapabilme ve organize olabilme, sürü

zekasının en temel iki özelliğidir. Tereshko and Loengarov 2005 yılında, bir bal arısı kolonisini, ortamdan bilgi toplayan ve davranışlarını buna göre düzenleyen dinamik bir model olarak yorumlamıştır [68]. Onlara göre, bal üretimi için gereken yiyecek arama modeli; besin kaynakları, görevi belirli işçi arılar ve görevi belirsiz işçi arılar olmak üzere üç ana öğeden oluşmaktadır. Bu yaklaşıma göre arıların, yiyecek kaynağını bulma ve besini tükenen kaynağı terk etme tarzında iki farklı davranışı söz konusudur. Bu öğeler kısaca şu şekilde açıklanabilir.

Besin kaynakları; arıların polen, nektar ya da bal elde etmek için ziyaret ettikleri kaynaklardır. Arılar bu kaynaklara ihtiyaç duydukları besini temin etmek için giderler. Bir besin kaynağının değerini belirleyen kıstaslar, kovana yakınlığı, içerdiği polen zenginliği, polen çıkarabilme kolaylığı ve nektar konsantrasyonu olarak sıralanabilir.

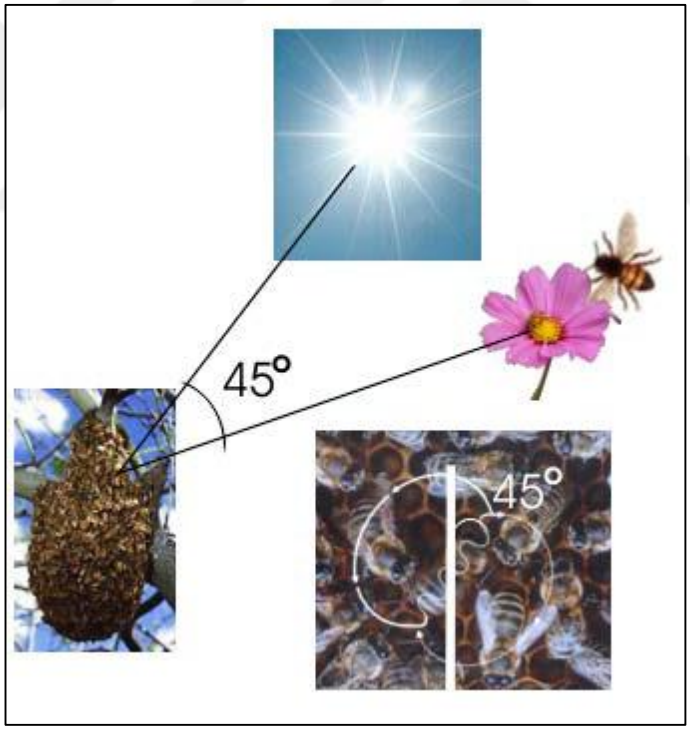
Görevi belirli işçi arılar, konumu daha önceden belirlenmiş besin kaynağına giderek, bu kaynaktan topladıkları besini, yuvaya taşımakla görevli arılardır. Bu arılar ayrıca, yuvaya taşıdıkları besin kaynağıyla ilgili bilgiyi, kovandaki gözcü arılarla paylaşırlar.

Görevi belirsiz işçi arılar ise iki gruptur. Kaşif arı olarak isimlendirilen ilk gruptaki arılar, doğal ortamda rastgele kaynak arayışındaki arılardır. Tüm koloninin %5-10'luk bölümünü oluşturan bu arılar, çevrelerini dolaşarak, farklı besin kaynağı bulmaya çalışırlar. Görevi belirsiz ikinci grup işçi arılar ise, görevi belirli işçi arıları takip ederek, keşfedilen besin kaynağı hakkında bilgi almaya çalışan arılardır. Gözcü arı olarak adlandırılan bu gruptaki arılar, kovana getirilen besinleri tadar ve besini yuvaya taşıyan görevli arıları dikkatle takip eder. Yönelecekleri besin kaynağını ise, kendi tercihlerine göre belirlerler.

Keşfedilen yeni bir besin kaynağına ait bilgi paylaşımı, kovandaki "dans alanı" olarak isimlendirilen bölümde "kuyruk dansı" (waggle dance) ile sağlanır. İlgili kaynağı keşfeden arı, kovana getirdiği besinin bir miktarını, kovandaki gözcü arılara tattırır ve ardından bu kaynağın yerini tarif etmek üzere dans (titreşim, sallanma) etmeye başlar. Bulunan besin kaynağına daha fazla arı gönderebilmek için bu dans, farklı alanlarda defalarca tekrarlanır. Dans şeklini belirleyen farklı unsurlar vardır. Bunlardan biri

nektar kalitesidir. Yiyecek kaynağındaki besinin kıvamı, çokluğu, nektar elde etme zorluğu, hava ve ortam koşulları gibi etmenler dansın şeklini belirler.

Dans şeklini etkileyen diğer bir unsur, besin kaynağının kovana göre konumudur. Bu bağlamda besin kaynağının konumu, kovanın hangi yönünde ve ne kadar uzağında olduğuyla alakalıdır. Bal arıları yön bilgisi için, güneş ışınlarından faydalanır. Besin kaynağının pozisyonunu öğrenen arı, polarize gözleri sayesinde, güneş ışınlarının kovana dik açıyla geldiği doğrultuyu başlangıç doğrusu olarak kabul eder ve bu doğrultudan, dans açısı kadar açıyla yönelerek hedefine doğru uçmaya başlar. Bal arısının polarize göz yapısı ayrıca, kapalı havalarda da güneşin konumunu tayin etmede yardımcı olur. Seçtiği besin kaynağının konumuna göre yönelen bir gözcü arının uçuş biçimi Şekil 3.1'de sunulmaktadır.



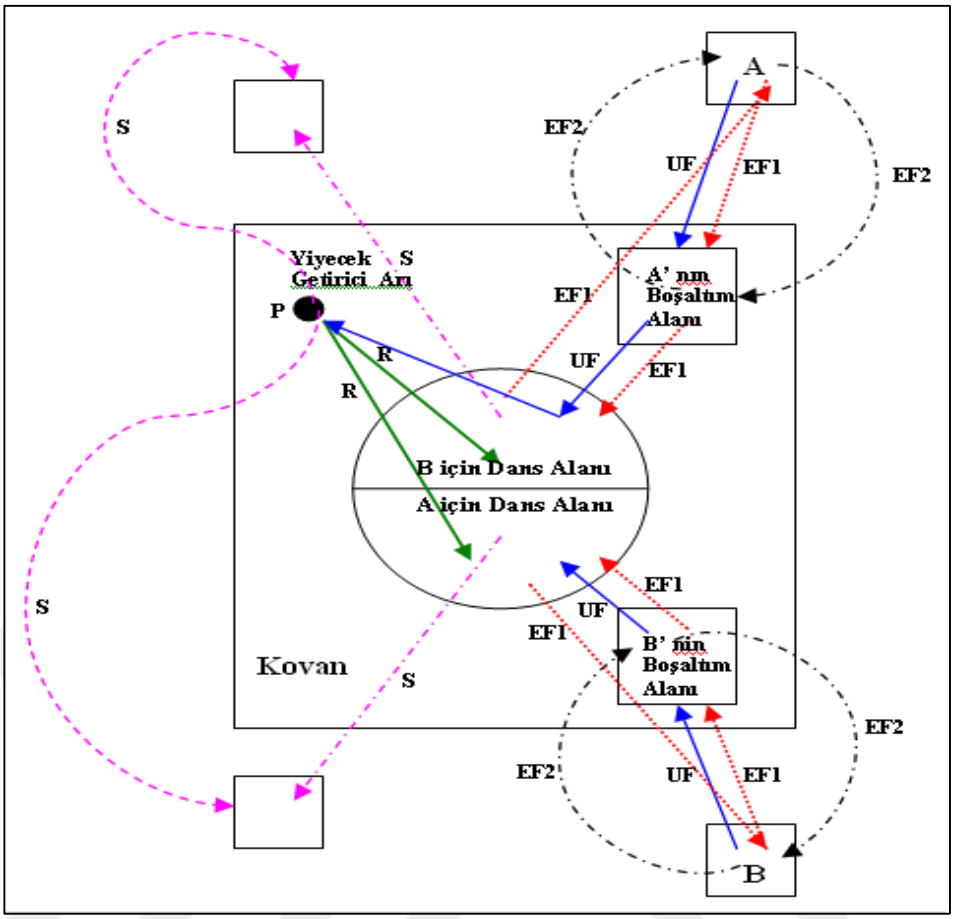
Şekil 3.1. Gözcü arının besin kaynağının konumuna göre yönelmesi [9].

Yeni keşfedilen bir besin kaynağından kovana yiyecek getiren arılar, kovandaki diğer arılara bu kaynağın uzaklığını tarif ederken farklı figürler oluştururlar. Besin kaynağının kovana uzaklığı 100 metre ile 10 kilometre arasındaysa bu desen, Şekil 3.1'de gösterildiği gibi "8" rakamına benzemektedir. Dansın her 15 saniyedeki tekrar

sıklığı, besin kaynağının kovana uzaklığını ifade etmektedir. Titreşim sıklığının az olması, kaynağın kovana daha uzak olduğu anlamındadır. Ayrıca dansçı arıyı takip eden gözcü arı, bu besin kaynağının, kovan-güneş doğrultusunun 45° doğusunda olduğunu anlamaktadır. Dansı izleyen gözcü arılar, besin kaynağının yerini öğrendiklerinde bir titreşim oluştururlar ve bu sayede dans sonlanır.

Besin kaynağı, kovanın 50-100 metre kadar uzağındaysa dans figürü, dairesel şekilde olur. Diğer bir iletişim şekli de arıların petek üzerinde düzensiz tarzda ve yavaş tempoda bacaklarını titreterek ileri, geri, sağa ve sola hareket etmesidir. Bu davranışıyla arı, zengin bir besin kaynağı bulduğunu, ancak kovana işlenebileceğinden daha fazla miktarda nektar geldiğini, dolayısıyla nektarı işleme görevine geçmek istediğini ifade etmektedir. Bu dans, yalnızca dans alanında değil kovanın farklı bölümlerinde de gerçekleştirilmektedir. Bu dans sayesinde, kovan kapasitesi ile yiyecek getirme aktivitesi arasındaki denge korunmaktadır.

Bal arılarının yiyecek arama çevrimi Şekil 3.2'de özetlenmiştir. Buna göre Şekil 3.2'de A ve B, kovan etrafında farklı iki konumda bulunan besin kaynağını ifade etmektedir. Kovandan, besin kaynağı bulmak üzere ayrılan kaşif arılar (Şekil 3.2'de S ile gösterilmektedir), bu besin kaynaklarını keşfederek yeni bir besin toplama faaliyetini başlatabilir.



Şekil 3.2. Bal arılarının yiyecek arama davranışı [9].

Kaşif arı, keşfettiği besin kaynağının konumunu hafızada tutar ve kaynaktan topladığı besinle kovana döner. Bulduğu besin kaynağındaki besin miktarına göre, toplamayı kendisi devam ettirebilir (Şekil 3.2'de EF2 ile gösterilmektedir). Kaynakta fazla miktarda besin varsa, kaynağın konumunu kovandaki diğer gözcü arılarla paylaşabilir (Şekil 3.2'de EF1 ile gösterilmektedir). Kaşif arıları takip ederek besin kaynaklarının konumunu öğrenen gözcü arı, yöneleceği besin kaynağına kendisi karar vererek yiyecek toplama faaliyetine katılacaktır (Şekil 3.2'de R ile gösterilmektedir). Gidilen kaynaktaki besin tükendiğinde, bu kaynak terk edilerek farklı kaynaklar keşfedilmek üzere arılar bağımsız hareket etmektedir (Şekil 3.2'de UF ile gösterilmektedir).

Bu analizlerden sonra, arıların kendi aralarında oluşturdukları organize sürü davranışı şu ilkelerle açıklanmıştır:

1. Pozitif geri besleme: Besin kaynağındaki nektar miktarı arttıkça, bu kaynağa yönelen gözcü arı sayısı da artacaktır.
2. Negatif geri besleme: Nektarı tükenen kaynak terk edilir.
3. Salınımlar: Kaşif arılar besin kaynağı bulabilmek için rastgele dağılırlar.
4. Çoklu etkileşimler: Arılar, besin kaynaklarının konumu ve durumu hakkında dans alanında bilgi paylaşımı yaparlar.

3.2. ALGORİTMA MODELİ

Bal arısı kolonilerinin, özellikle yiyecek arama faaliyeti için, aralarında oluşturduğu iş bölümü, Karaboğa tarafından gözlemlenmiş ve Yapay Arı Koloni (YAK) algoritması olarak modellenmiştir [69]. Algoritma kapsamında her bir besin kaynağı yalnızca bir görevli arı tarafından ziyaret edilir. Dolayısıyla besin toplamakla görevli işçi arı sayısı, besin kaynağı sayısına eşittir. İşçi arı sayısı, aynı zamanda kovanda kendilerini bekleyen gözcü arı sayısına da eşittir. Bu bağlamda koloni boyutu (işçi arı sayısı + gözcü arı sayısı), besin kaynağı sayısının iki katı değerdedir. Bir işçi arı, besin toplamakla görevli olduğu kaynaktaki nektar tükendiğinde, ilgili kaynağı terk edecek ve yeni besin kaynağı bulabilmek üzere kaşif arı olarak görev yapacaktır. Besin kaynakları, optimizasyon probleminin her bir muhtemel çözümüne karşılık gelmektedir. Kaynağın konumu, kovana uzaklığı, nektar konsantrasyonu, çokluğu ve kalitesi, çözümün uygunluk değerine karşılık gelmektedir. Bu bağlamda bal arıları, doğal ortamdaki en uygun besin kaynağını bulmaya çalışırken; YAK algoritması kapsamında, optimizasyon probleminin çözüm uzayındaki optimal çözümü aramaktadırlar.

Bal arılarının yiyecek arama faaliyeti, şu aşamalara ayrıştırılabilir:

1. Kaşif arılar, besin kaynağı bulabilmek için ortama rastgele dağılarak yiyecek arama sürecini başlatırlar.
2. Besin kaynakları belirlendikten sonra kaşif arılar, çalışmalarına, bu kaynaklardan kovana besin taşımakla görevli işçi arılar olarak devam ederler. Kaynaktaki besin tükendiğinde ise yine kaşif arı görevini üstlenerek, ortamda farklı besin kaynağı bulmaya çalışırlar. Bu arılar ayrıca, besin kaynağıyla ilgili

bilgiyi, kovandaki gözcü arılarla paylaşırlar.

3. İşçi arılardan aldığı bilgileri değerlendiren gözcü arılar, yönelecekleri besin kaynağını kendileri seçerler.

YAK modelinde yiyecek arama davranışı, şu adımlardan oluşur:

1. Besin kaynaklarının belirlenmesi.
2. Repeat
3. İşçi arıların besin kaynaklarına yönlendirilmesi.
4. Her bir besin kaynağının kalitesinin hesaplanması ve gözcü arılarca seçilme ihtimalinin belirlenmesi
5. Gözcü arıların, seçilme olasılıklarını referans alarak besin kaynaklarına yönelmesi
6. Besin kaynağını terk etme ve kaşif arıya dönüşme durumu
7. Until (gerekli kısıtlar sağlanana kadar)

3.2.1. Besin Kaynaklarının Belirlenmesi

Yiyecek toplamak üzere kovandan ayrılan bal arıları, kovandan ortalama 15 kilometre kadar uzaklaşabilir ve bu seyahatte 50-100 kadar besin kaynağını dolaşabilirler. Bu bağlamda arı kolonisi için yiyecek arama uzayı, kovan merkezli ve 15 kilometre yarıçaplı dairesel bir alan olarak yorumlanabilir. Kolonideki her bir kaşif arı, rastgele dağılıp, arama bölgesinde bir besin kaynağı bularak yiyecek arama sürecini başlatır. YAK algoritmasında bu işlem, 3.1 eşitliğinde gösterildiği gibi, her bir parametrenin alt ve üst sınırları arasında rastgele değer üretilmesi şeklinde modellenmiştir.

$$x_{i,j} = x_{min\ j} + rand(0,1) * \left(x_{max\ j} - x_{min\ j}\right) \tag{3.1}$$

3.1 eşitliğinde $i = 1..SN$ ve $j = 1..D$ olmak üzere, SN çözüm sayısı ve D parametre sayısıdır. $x_{min\ j}$ ve $x_{max\ j}$ ise j parametresinin alt ve üst sınır değerleridir.

3.2.2. İşçi Arıların Besin Kaynaklarına Yönlendirilmesi

Besin kaynaklarının her birinden besin toplayan ayrı işçi arılar vardır. Bu işçi arılar, nektar çıkarmak üzere görevli oldukları kaynaklara gittiklerinde, bu kaynakların komşuluğundaki diğer kaynakları da kontrol ederler. Bu davranışın matematiksel ifadesi 3.2 eşitliğindeki gibidir.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3.2}$$

3.2 eşitliği ile, çözüm kümesindeki mevcut x_i ve x_k çözümleri kullanılarak, yeni bir v_i çözümü oluşturulmaktadır. x_k çözüm kümesinde rastgele seçilen bir çözümdür. x_i çözümünün rastgele seçilen j. parametresi ile x_k çözümünün j. parametresi arasındaki fark, [-1,1] aralığında rastgele seçilen ϕ_{ij} katsayısıyla çarpılır ve elde edilen sonuç v_i çözümünün j. parametresini oluşturur. Ancak v_{ij}, x_{ij} sınır değerlerinin dışına taşmışsa, değer, x_{ij} sınır değerlerine ötelenir.

Elde edilen v_i çözümünün optimizasyon problemindeki $f(v_i)$ maliyet değeri hesaplanır ve [0,1] aralığındaki $fit(v_i)$ uygunluk değeri belirlenir. Bu işlem 3.3 eşitliğinde gösterilmektedir.

$$fit(v_i) = \begin{cases} 1/(1+f(v_i)) & if\ (f(v_i)) \geq 0 \\ 1 + abs(f(v_i)) & if\ (f(v_i)) < 0 \end{cases} \tag{3.3}$$

Eğer bulunan yeni besin kaynağı, mevcut besin kaynağından daha kaliteli değilse, işçi arı bir sonraki seferinde yine yönelmekte olduğu mevcut besin kaynağına gidecektir. Ancak keşfedilen bu yeni kaynak daha kaliteliyse, sonraki uçuşlarında bu kaynağa gidecektir. YAK ifadesiyle, elde edilen yeni çözüm, mevcut çözümden daha başarılı değilse, mevcut çözümün başarısızlık (failure) sayacı 1 artırılır ve bir sonraki döngüde ilgili çözüm üretme işlemi bu mevcut çözüm ile tekrarlanır. Ancak elde edilen yeni çözüm mevcut çözümden daha başarılıysa, başarısızlık sayacı 0'a eşitlenir ve bir sonraki çevrimde, bu yeni çözüm kullanılarak farklı çözümler türetilir.

3.2.3. Her Bir Besin Kaynağının Kalitesinin Hesaplanması ve Gözcü Arılarca Seçilme İhtimalinin Belirlenmesi

İşçi arılar, görevli oldukları besin kaynaklarından nektar toplayıp kovana döndüklerinde, bu besinin bir kısmını, kovandaki diğer gözcü arılara da tattırır ve besin kaynağının konumunu dans alanında dans ederek tarif ederler. YAK algoritması için bu işlem, algoritmanın altında çoklu etkileşim sergilendiğinin kanıtıdır. Algoritmadaki olasılıkla seçme işlemi, 3.4 eşitliğinde gösterildiği gibi, elde edilen çözümlerin uygunluk değerlerine uygulanarak sağlanır. Sıralamaya dayalı, stokastik, turnuva, örnekleme gibi farklı seçme yöntemleri kullanılabilse de bu aşamada genellikle rulet tekerleği yöntemi tercih edilmektedir. Böylece başarılı çözümlerin seçilme olasılıkları artırılmaktadır.

$$r_i = \frac{fit(v_i)}{\sum_{i=1}^{SN} fit(v_i)} \tag{3.4}$$

3.4 eşitliğinde v_i türetilen çözümü, r_i çözümün seçilme olasılığını, *fit* çözümün uygunluk değerini, *SN* çözüm sayını göstermektedir.

3.2.4. Gözcü Arıların Seçilme Olasılıklarını Referans Alarak Besin Kaynaklarına Yönelmesi

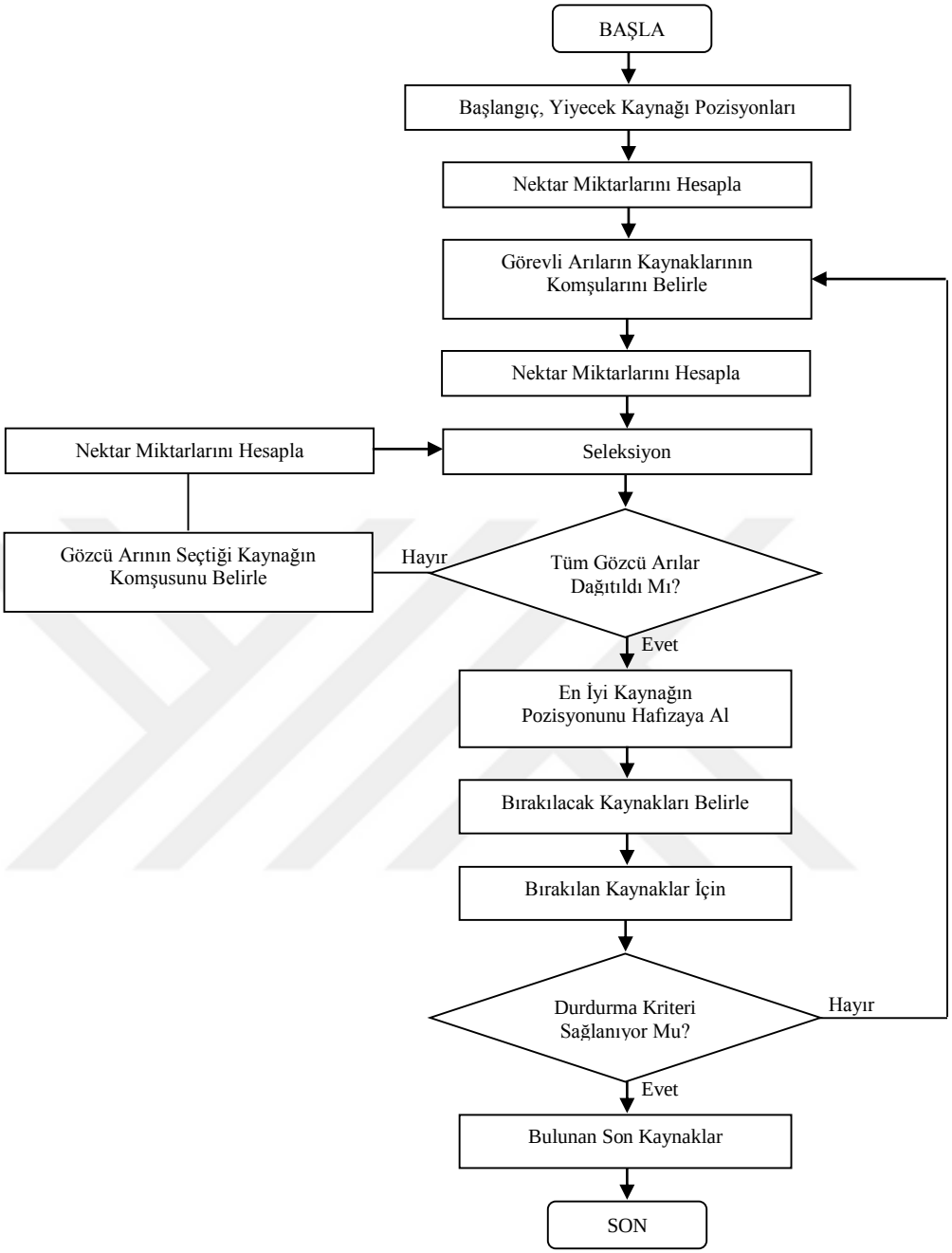
İşçi arılardan aldıkları bir miktar besini tadıp bu besin kaynağının konum bilgisini öğrenen gözcü arılar, yöneleceği besin kaynağını kendileri seçerler. YAK algoritmasında 3.4 eşitliğiyle her bir çözümün seçilme olasılığı belirlenir ve bu değer rulet tekerleğine yerleştirilir. Rastgele seçme yöntemi uygulanarak seçilen çözüm için 3.2 eşitliği uygulanarak yeni bir çözüm üretilir. Üretilen bu yeni çözümün uygunluk değeri belirlenip 3.3 eşitliğiyle uygunluk değeri hesaplanır. Eğer üretilen yeni çözüm daha başarılıysa, mevcut çözüm yerine bu çözüm kullanılır ve ilgili çözümün başarısızlık sayacı sıfırlanır. Üretilen çözüm yeni çözümden daha başarılı değilse, eski çözümün başarısızlık değeri 1 artırılır. Bu işlemler tüm gözcü arılar için tekrarlanır.

3.2.5. Besin Kaynağını Terk Etme ve Kaşif Arıya Dönüşme Durumu

İşçi ve gözcü arılar ziyaretlerini tamamlayıp kovana döndüklerinde, her bir besin kaynağının durumu değerlendirilir. Belirlenen limit değer sayısınca, ziyaret edilen besin kaynağının komşuluğunda daha kaliteli bir besin kaynağı bulunamamışsa, bu kaynak terk edilir. İlgili besin kaynağı için görevlendirilen işçi arı, bu kaynağı terk ederek kaşif arı olarak davranır ve farklı bölgelere dağılarak yeni bir besin kaynağı aramaya başlar. YAK metodundaki modelde, bir çevrim sonunda eldeki tüm çözümlerin başarısızlık değerleri kontrol edilir. Başarısızlık değerleri, belirlenen limit değerine ulaşan çözümler için 3.1 eşitliği uygulanarak yeni bir çözüm oluşturulur.

3.2.6. YAK Akış Diyagramı

Tüm adımlar değerlendirildiğinde YAK algoritması için genel akış diyagramı Şekil 3.3'deki gibi oluşturulmuştur.



Şekil 3.3. YAK algoritması akış diyagramı [9].

3.2.7. YAK Adımları

YAK algoritmasının her bir adımında yapılan işlemler şu şekilde sözde kodla ifade edilebilir.

1. Eşitlik 3.1 ile başlangıç çözümlerinin oluşturulması ve her bir çözüm için

başarısızlık sayaçlarının başlatılması ($failure_i = 0$)

2. Oluşturulan her bir çözümün fonksiyon değerlerinin ($f(x_i)$) ve bu değere karşılık gelen uygunluk değerlerinin ($fit(x_i)$) hesaplanması
3. *Repeat*
4. *for i=1 To SN do*
5. Eşitlik 3.2 yardımıyla x_i çözümünü kullanarak yeni bir v_i çözümü oluşturulması.
6. Bu v_i çözümünün fonksiyon değerinin ($f(v_i)$) belirlenmesi ve 3.3. eşitliği ile uygunluk değerinin ($fit(v_i)$) hesaplanması.
7. *if* $fit(v_i) > fit(x_i)$ *Then*
8. Geçerli çözüm = v_i
9. $failure_i = 0$
10. *Else*
11. $failure_i = failure_i + 1$
12. *End If*
13. *End For*
14. Eşitlik 3.4 ile gözcü arıların yöneleceği besin kaynağını seçerken dikkat edecekleri seçilme ihtimallerinin (r_i) hesaplanması.
15. t=0, i=0
16. *Repeat*
17. *if random*<r_i *Then*
18. Eşitlik 3.2 yardımıyla x_i çözümünü kullanarak yeni bir v_i çözümü oluşturulması.
19. Bu v_i çözümünün fonksiyon değerinin ($f(v_i)$) belirlenmesi ve 3.3. eşitliği ile uygunluk değerinin ($fit(v_i)$) hesaplanması.
20. *if* $fit(v_i) > fit(x_i)$ *Then*
21. Geçerli çözüm = v_i
22. $failure_i = 0$
23. *Else*
24. $failure_i = failure_i + 1$
25. *End If*
26. t=t+1
27. *End If*

28. *Until(t=SN)*
29. *if* $failure_i$ *> limit Then*
30. Eşitlik 3.1'i kullanarak, x_i yerine yeni bir çözüm oluşturulması.
31. *End If*
32. En başarılı çözümün hafızada tutulması.
33. *Until (Durdurma kısıtlamaları sağlanana kadar)*

3.2.8. YAK Algoritmasının Genel Özellikleri

YAK algoritması;

1. Bal arılarının yiyecek arama davranışına çok yakın biçimde modellenmiştir.
2. Popülasyona dayalı bir algoritmadır.
3. Sürü zekasına dayalı bir algoritmadır.
4. Nümerik optimizasyon problemlerinde olduğu gibi, ayrık problemlerde de başarılıdır.
5. Az sayıda kontrol parametresiyle çözüm arar.
6. Kodlaması basit, yapısı esnektir.
7. Kaşif arılarla küresel, işçi ve gözcü arılarla yerel arama yapılır.

BÖLÜM 4

EŞ ZAMANLI DAĞITIM TOPLAMALI ARAÇ ROTALAMA PROBLEMİ İÇİN GELİŞTİRİLEN ÇÖZÜM ÖNERİSİ

Geliştirilen uygulamanın; lojistik, nakliye ve kargo faaliyetlerinde, gerçek yaşamda karşılaşılabilen problemlere de çözüm teşkil edebilmesi için, çalışma sürecinde kurye dağıtımı yapan işletmelerin faaliyetleri incelenmiş, nakliye faaliyetlerinde karşılaşılan problemler ve çözüm yaklaşımları konusunda gözlemler yapılmıştır. Bu bilgiler kapsamında ARP türleri değerlendirildiğinde, gerçek yaşamdaki örneklere en yakın tür olarak EDTARP türü öne çıkmaktadır. EDTARP, Bölüm 2'de kapsamlı olarak bahsedildiği üzere, dağıtım ve toplama faaliyetlerinin eş zamanlı olarak yürütüldüğü, çok noktadan oluşan ağ yapısında, taşıma araçlarının en az maliyetli rotalarının belirlenmesi problemidir. Taşıma araçları için en az maliyetli rotaların oluşturulması, en erken sürede teslimat ve araç kapasitesinin en yüksek verimde kullanılması gibi avantajları da beraberinde getirmektedir. Çalışmanın bu bölümünde, öncelikle EDTARP çözümünde dikkat edilmesi gereken noktalar açıklanmış, ardından YAK algoritmasının, EDTARP çözüme uyarlanışı anlatılmıştır. Geliştirilen algoritma, modüler yapıda, esnek biçimde tasarlanmış ve .net platformunda C# dili kullanılarak kodlanmıştır.

4.1. EDTARP ÇÖZÜMÜNDE ÖNEMLİ HUSUSLAR

Bölüm 2'de genel yapısı, matematiksel modeli ve kısıtlamaları verilen EDTARP çözümlerinde dikkat edilecek noktalar şu şekilde sıralanabilir:

1. Problemdeki toplam düğüm sayısı, merkez depo ve müşteri düğümlerin toplam sayısı kadardır. (Örneğin 1 depo merkezi ve bu merkezden beslenen 50 müşterili bir ağ modelinde, toplam 51 adet düğüm bulunmaktadır.)
2. İlk düğüm (0 numaralı düğüm) depo merkezini, sonraki düğümler müşterileri

temsil eder.

3. Her bir çözüm 0 numaralı düğüm ile başlayıp 0 numaralı düğüm ile sonlandırılmalıdır.
4. Her bir müşteri düğüm, "yalnızca ve mutlaka" bir kez ziyaret edilebilir. Ancak depo merkezine birden fazla kere dönülebilir.
5. Herhangi bir çözüm dizisinde 0 numaralı düğümün kullanılmış olması, depo merkezine dönüşü ifade eder. Dolayısıyla ilgili araç için rotalama tamamlanmış, bir sonraki araç için rotalama başlatılmıştır. Bu bağlamda, çözümdeki 0 düğümleri sayısının 1 eksiği kadar araç için rotalama yapılmış olur. (Araç sayısında kısıtlama yoktur.)
6. Çözüm oluşturulduktan sonra, çözümdeki her bir rota kontrol edilir. Araç, seyahate başlarken ya da seyahat boyunca uğranılacak düğümlerde, dağıtım/toplama sonrası taşınacak yük miktarı, araç kapasitesini aşmamalıdır.

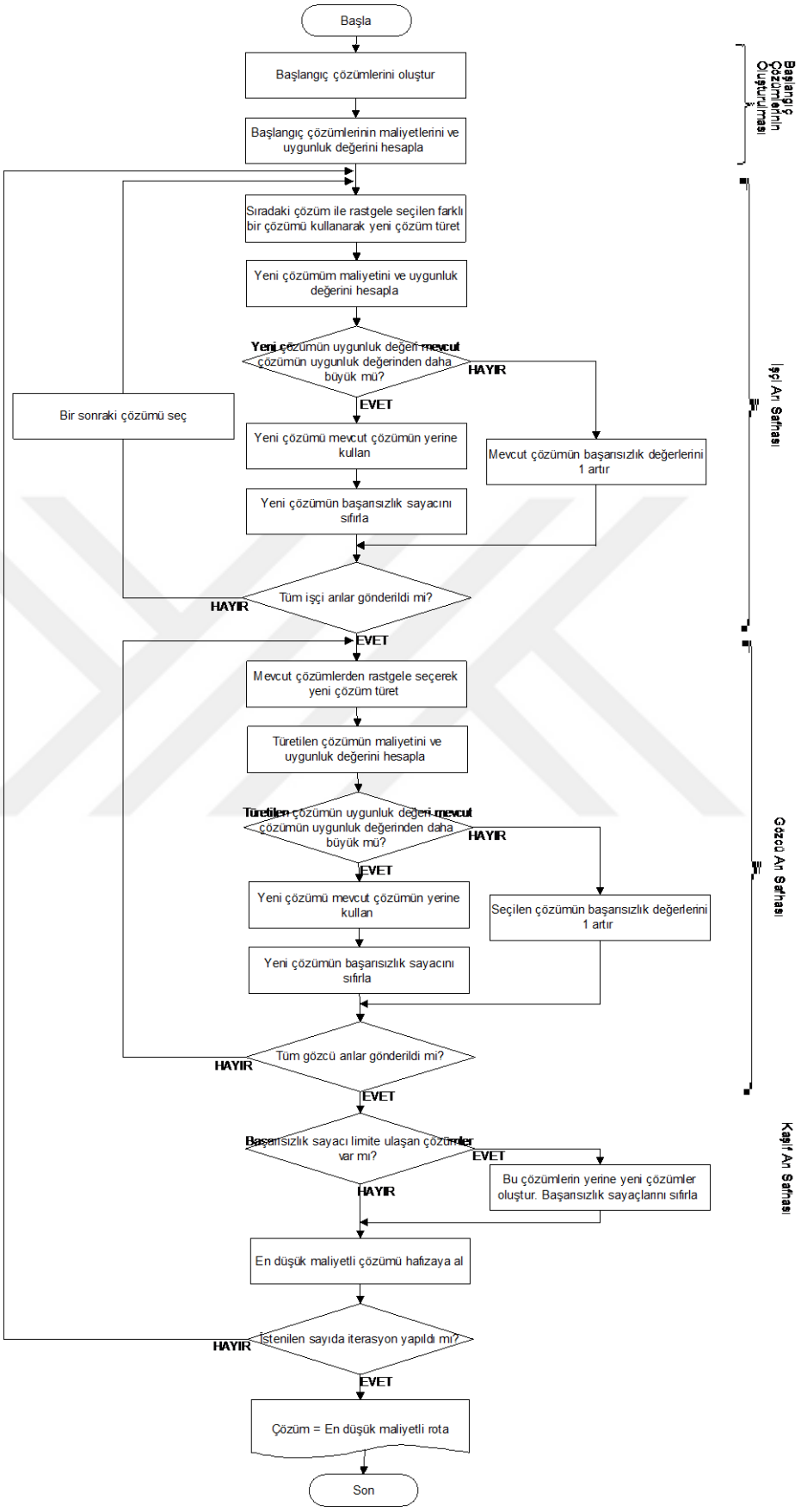
EDTARP'deki amaç, yukarıdaki maddeler çerçevesinde en kısa rotaların oluşturulmasıdır.

4.2. EDTARP ÇÖZÜMÜ İÇİN OLUŞTURULAN YAK MODELİ

Bölüm 3'te etraflıca anlatılan YAK algoritması, nümerik optimizasyon problemlerinin çözümlerinde kullanılabilir nitelikte aktarılmıştır. Ancak EDTARP, kombinatoryal optimizasyon problemleri kapsamında NP-zor sınıfındaki ayrık yapılı bir problem modelidir. Bu bölümde, YAK algoritmasının EDTARP çözümüne uygulanışı, adımlarıyla birlikte ele alınmıştır.

4.2.1. EDTARP Çözümü İçin YAK Akış Diyagramı

YAK algoritmasının, EDTARP çözümü için oluşturulan akış diyagramı Şekil 4.1'de gösterilmektedir.



Şekil 4.1. EDTARP çözümü için oluşturulan YAK akış diyagramı.

4.2.2. Başlangıç Çözümlerinin Oluşturulması

Algoritma, belirlenen besin kaynağı sayısınca rastgele çözümlerin oluşturulmasıyla başlar. Başlangıç çözümleri oluşturulurken, öncelikle her bir çözüm dizisinin ilk elemanı, "0" olarak belirlenmektedir. Dizinin sonraki elemanları, 0 çok kez, diğer rakamlar yalnızca bir kez kullanılabilecek şekilde seçilir. Burada dikkat edilecek nokta 0 düğümlerinin artarda kullanılmamasıdır. Bu bağlamda, Şekil 4.2'de oluşturulan örnek bir başlangıç çözümü sunulmaktadır.

0	28	41	37	16	…	19	25	***0***	49	32	11	***0***	3	29	…	***0***

Şekil 4.2. YAK algoritması ile EDTARP için oluşturulmuş bir çözüm örneği

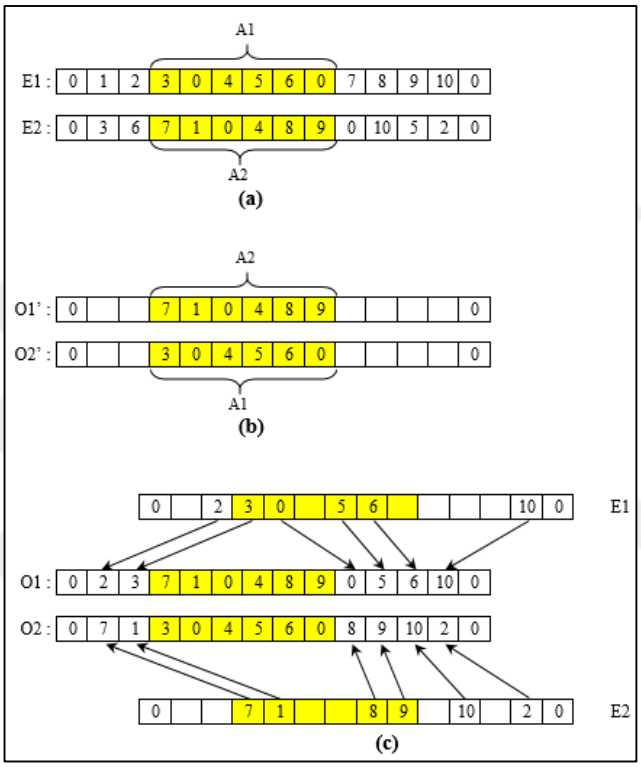
4.2.3. Çözüm Maliyetlerinin ve Uygunluk Değerlerinin Hesaplanması

Çözüm maliyeti için, nakliye masrafı, benzin masrafı, işçi ücreti gibi farklı kıstaslar da kullanılabilir. Ancak ARP'lerde rota maliyeti olarak kastedilen, genellikle rotanın toplam mesafesidir. Bu bağlamda, oluşturulan her bir çözümün maliyeti, rotalandırılan tüm araçların, depo merkezinden seyahate başlayıp tekrar depo merkezine dönüşüne kadar kat ettikleri mesafelerin toplamıdır. Her bir çözümün uygunluk değeri ise, toplam mesafeyle ters orantılıdır. Daha uzun mesafeli bir çözümün uygunluk değeri, daha düşük seviyededir.

4.2.4. İşçi Arı Safhası

Görevli oldukları besin kaynağının komşuluğunda yeni besin kaynakları arayan işçi arılar, YAK metodunda yeni çözümlerin üretimine vesile olurlar. İşçi arı safhasında yeni çözüm üretilirken kullanılan yöntemde, başlangıçta iki çözüm rotası belirlenir ve bu çözüm rotaları üzerinde, aralarında eşit sayıda düğüm kalacak şekilde, yine rastgele iki kesim noktası seçilir. Yeni çözümler bulunurken, kesim noktaları arasında kalan düğümler karşılıklı yer değiştirir ve arta kalan düğümler de mevcut çözümlerdeki sırasına göre dizilirler. Şekil 4.3a'da gösterilen E1 ve E2 çözüm rotaları üzerinde, A1

ve A2 kesim noktaları arasındaki düğümler, O1’ ve O2’ isimlerinde yeni kaynak üretimi için Şekil 4.3b’deki gibi karşılıklı yer değiştirir. Her bir çözümde, araç sayısınca çevrim olacağından, “0” numaralı düğüm sayısı da Q+1 adedince olacaktır. Dolayısıyla arta kalan düğümlerin dizilişinde “0” numaralı düğüm eklenirken, Şekil 4.3c’de gösterildiği gibi, bu düğümlerin art arda gelmemesine ve araç sayısını aşmamasına dikkat edilmelidir.



Şekil 4.3. İşçi arı safhasında çözüm oluşturma algoritmasına ait bir örnek.

İşçi arı safhasında, yeni çözüm türetirken gerekli olan iki çözümden ilki sıradaki çözüm, diğeri ise mevcut çözüm dizisi içinden rastgele seçilecek çözümdür. Bu safhada, sırasıyla her bir çözüm ve bundan farklı diğer bir çözüm kullanılarak yeni çözümler türetilir. Eğer türetilen yeni çözümün uygunluk değeri, mevcut çözümün uygunluk değerinden daha büyükse, çözüm matrisinde mevcut çözüm yerine, yeni çözüm güncellenir ve başarısızlık değeri sıfırlanır. Aksi takdirde yalnızca mevcut çözümün başarısızlık değeri 1 artırılacaktır.

4.2.5. Uygunluk Değerlerinin Rulet Tekerleğine Yerleştirilmesi

İşçi arılar kovana döndükten sonra, buldukları her bir kaynağın kalitesi belirlenir ve besin kaynağına ait tüm bilgiler, gözcü arılarla paylaşılır. Gözcü arılar, işçi arılardan farklı olarak, gidecekleri besin kaynağını kendileri seçerler. YAK algoritması yaklaşımında, işçi arı safhasından sonra, çözüm matrisinde yer alan güncel çözümlerin maliyetleri ve buna ilişkin oluşan uygunluk değerleri, rulet tekerleğine yerleştirilmektedir. Gözcü arı safhasında türetilecek çözümler için, gereken mevcut çözümler, rulet tekerleğine göre seçilecektir. Bu sayede başarılı çözümlerin seçilme olasılığı artırılmaktadır.

4.2.6. Gözcü Arı Safhası

Uygulamada gözcü arı safhasındaki yeni çözüm türetme işlemi, işçi arı safhasındakine benzer biçimde, OX tekniğiyle yapılır. Ancak işçi arı safhasındakinden farklı olarak, çözüm türetmede gerekli iki çözümün her ikisi de rulet tekerleğinden seçilir. Seçilen ilk çözüm "temel çözüm", ikinci çözüm ise "yardımcı çözüm" olarak kabul edilir. Bu iki çözüm kullanılarak yeni bir çözüm türetilir ve yine türetilen yeni çözümün maliyeti ve uygunluk değeri hesaplanır. Türetilen çözümün uygunluk değeri, temel çözümün uygunluk değerinden daha yüksek değerdeyse, çözüm matrisinde temel çözüm yerine türetilen çözüm aktarılır ve başarısızlık değeri sıfırlanır. Aksi takdirde temel çözümün başarısızlık değeri 1 artırılacaktır.

4.2.7. Besin Kaynağının Terk Edilmesi ve Kaşif Arı Safhası

Tüm bal arıları kovana döndüğünde besin kaynaklarının durumu gözden geçirilir ve bazı kaynaklar terk edilir. YAK algoritmasında, her bir çevrim tamamlandığında çözümlerin başarısızlık sayaçları kontrol edilir. Başarısızlık sayacı limit değere ulaşan çözümler terk edilerek, bu çözümlerin yerine yeniden çözümler oluşturulur. Oluşturulan bu yeni çözümlerin de maliyetleri ve uygunluk değerleri hesaplanır.

4.2.8. Algoritmanın Durdurulması ve Bulunan Çözüm

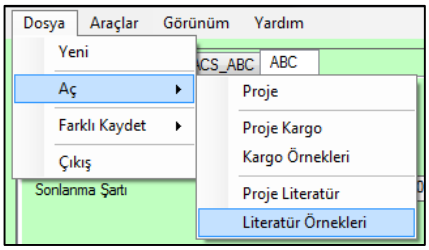
YAK algoritmasının kontrol parametreleri, "besin kaynağı sayısı" ve "limit"tir. Algoritma standardında işçi arı ve gözcü arı sayıları besin kaynağı sayısına eşittir. Algoritmanın sonlanma parametresi olarak belirlenen "çevrim sayısı" parametresi de algoritmanın bir parametresi olarak yorumlanmıştır.

İteratif algoritmalarda, sonlanma için farklı kıstaslarla da kullanılabilir. Ancak geliştirilen yöntemde algoritma, belirlenen iterasyon sayısınca çevrim tamamlandıktan sonra sonlandırılmaktadır. Her bir çevrimde, uygunluk değeri en yüksek çözüm, en başarılı çözüm olarak hafızada tutulur. Algoritma, istenilen çevrim sayısı kadar çalıştığında, elde edilen en başarılı çözüm algoritmanın bulduğu çözüm değer olarak kaydedilir.

4.3. GELİŞTİRİLEN MODELİN ÇALIŞMA ARAYÜZÜ

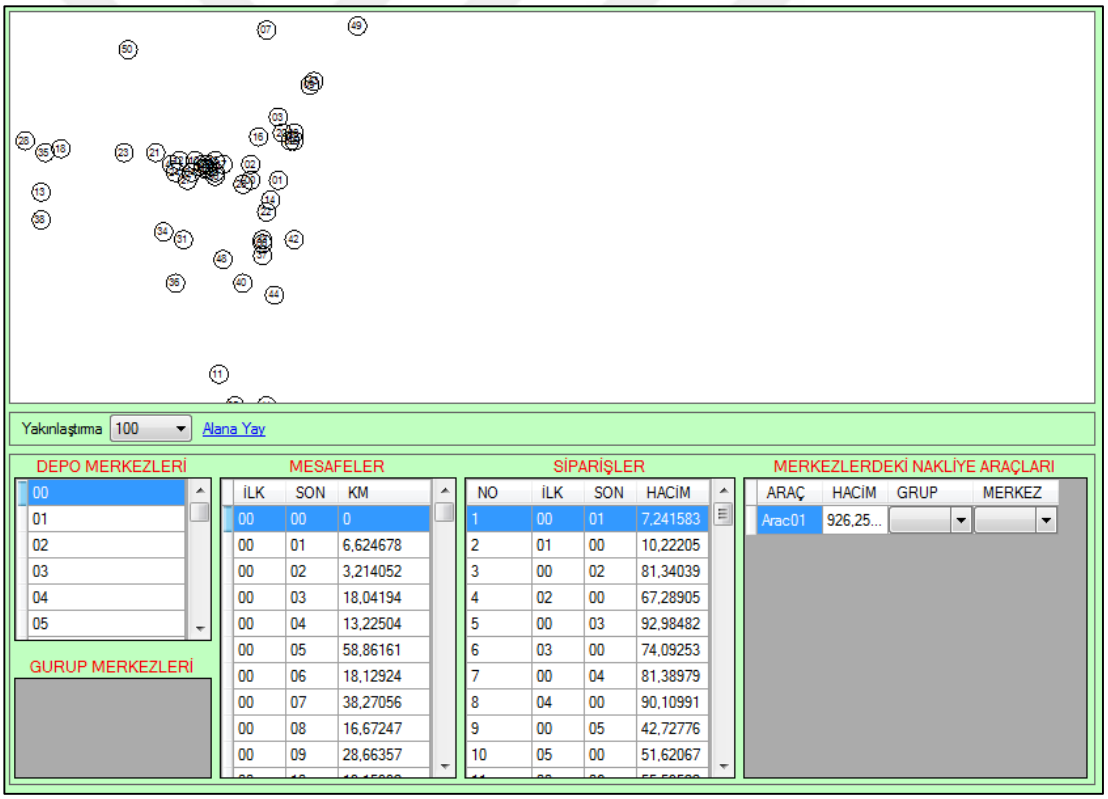
EDTARP çözümü için YAK tekniği kullanarak hazırlanan yazılım, sonradan gerekecek güncellemelere olanak sağlayabilecek esneklikte tasarlanmış ve modüler yapıda kodlanmıştır. Uygulama ayrıca, kullanıcı dostu bir ara yüze sahiptir. Kullanıcının birçok bilgiyi aynı ekranda görebilmesi için, ana ekran zenginleştirilmiş, menüler, klasik programlardaki tanıdık içerikleriyle oluşturulmuştur. Kullanıcı bu menüler yardımıyla test problemlerine ait farklı detayları da görebilecek, algoritmanın elde ettiği sonuçları kaydedebilecektir.

Şekil 4.4'te gösterildiği gibi, programın ana ekranındaki "Dosya\Aç\Literatür Örnekleri" menü yoluyla açılan ekranda, bilgisayarda kayıtlı Dethloff örnekleri açılabilir.



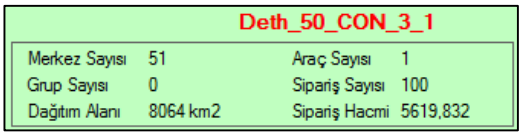
Şekil 4.4. Uygulamada Dethloff test problemleri açılırken kullanılan menü.

40 adet Dethloff test örneklerinden herhangi biri seçildiğinde, problemdeki tüm düğümlerin yerleşimi (depo merkezi ve müşteri düğümler), mesafeler, siparişler ve problemdeki araç kapasitesi bilgileri, ekranın ilgili kısımlarında gösterilmektedir. Şekil 4.5'te CON3-1 test örneğindeki detaylar gösterilmektedir.



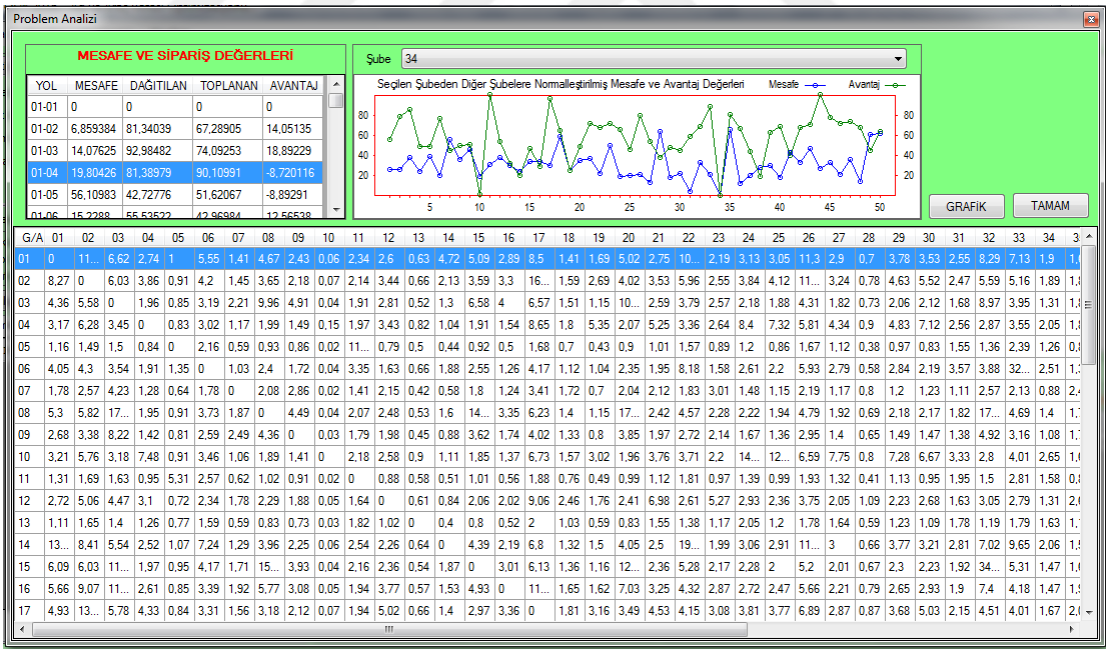
Şekil 4.5. CON3-1 test örneğinin uygulamadaki görüntüsü.

Uygulamada, seçilen test örneğindeki depo merkezi sayısı, düğüm sayısı, düğümlerde gruplandırma yapılmışsa gurup sayısı, tüm düğümlerin yayıldığı toplam alan, kullanılan araç türü sayısı ve dağıtım/toplama talep hacmi bilgileri de ekranda gösterilmektedir. Şekil 4.6'da CON3-1 problemine ait ilgili detaylar sunulmaktadır.



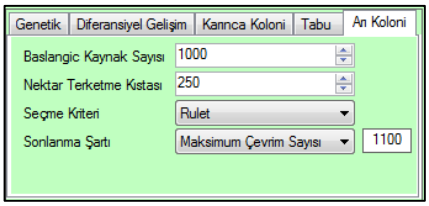
Şekil 4.6. CON3-1 test örneğinde sipariş, araç türü ve talep bilgileri.

"Araçlar\Problem Analizi" menü yoluyla açılan ekranda, seçilen test problemindeki dağıtım/toplama talepleri ve düğümler arası mesafelere ilişkin detaylar gösterilmektedir. Şekil 4.7'de örnek olarak, CON3-1 test problemindeki "34" numaralı düğüm ele alınmış, bu düğümden diğer düğümlere olan mesafeler (grafikte mavi renk) ve diğer düğümlerde dağıtım/toplama farkı (grafikte yeşil renk), [0, 100] arasında normalize edilmiştir. Buna göre ilgili grafiğe bakarak, CON3-1 test örneğinde 34 numaralı düğüme en yakın düğümün 31 numaralı düğüm olduğu ve araç kapasitesi açısından en avantajlı (dağıtım – toplama talepleri arasındaki en yüksek fark) düğümün 44 numaralı düğüm olduğu görülebilmektedir.



Şekil 4.7. CON3-1 test örneğinde mesafe ve taşıma bilgilerine ait analiz.

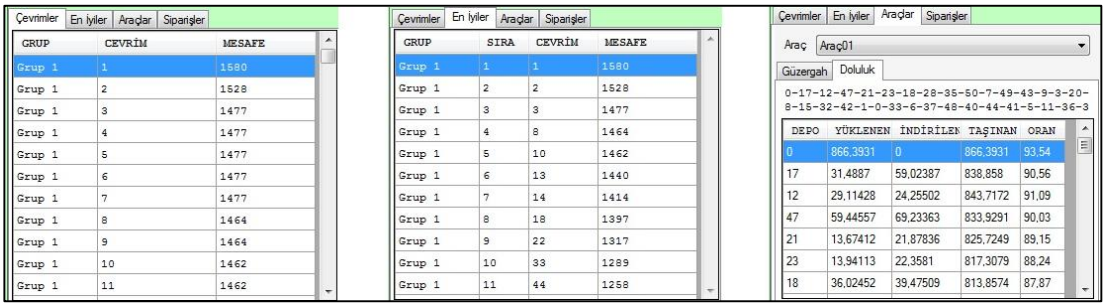
Seçilen test problemi için YAK algoritmasıyla sonuç aranırken ekranda algoritmaların gösterildiği sekmeden "Arı Koloni" sekmesi seçilir. Bu sekmede Şekil 4.8'de de görüldüğü gibi YAK algoritmasına ait parametreler bulunmaktadır.



Şekil 4.8. Uygulamadaki Arı Koloni sekmesi.

Şekil 4.8'de YAK algoritmasında başlangıç çözümleri için kaç adet çözüm istendiğini ifade eden "Başlangıç Kaynak Sayısı" alanı, limit parametresini ifade eden "Nektar Terketme Kıstası" alanı, gözcü arıların seçeceği çözümlerde kullanılan yöntem olarak "Seçme Kriteri" alanı ve algoritmanın sonlandırılması için gereken kriter için "Sonlandırılma Şartı" alanları yer almaktadır.

Parametre değerleri belirlendikten sonra "Uygula" düğmesi ile program istenilen sayıda çevrim yapacak ve algoritma sonlandırılacaktır. Algoritmanın sonlandırılmasıyla birlikte, ekranın sol alt bölümünde, Şekil 4.9'da da görüldüğü üzere, elde edilen sonuçların listelendiği sekmeler aktif hale gelmektedir.

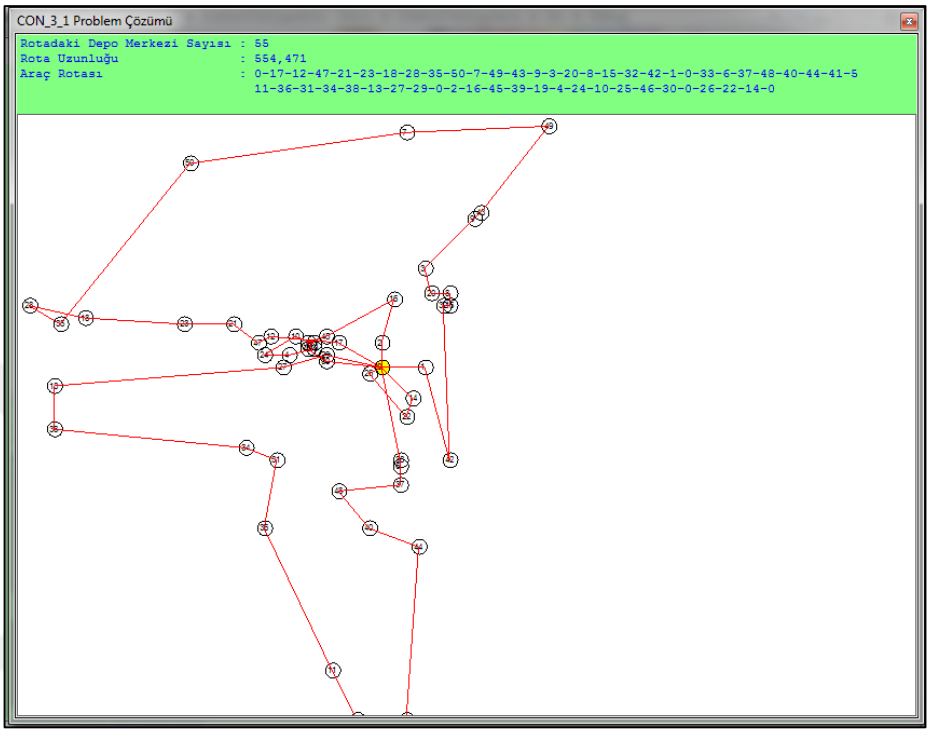


(a) (b) (c)

Şekil 4.9. Uygulamanın, CON3-1 test problemi için elde ettiği sonuçlar.

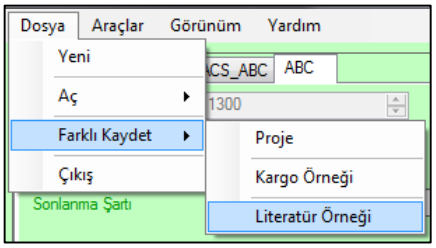
Şekil 4.9a'da her bir çevrimde elde edilen en kısa mesafeler listelenmektedir. Şekil 4.9b'de daha başarılı çözümlerin elde edildiği çevrimler ve bu çözümlerde elde edilen sonuçlar listelenmektedir. Şekil 4.9c'deki "Araçlar" sekmesinde ise rotalandırılan araçlara ait bilgiler görülmektedir. Oluşturulan çözümde rotalandırılan tüm araçlar, açılır listede yer almaktadır. Bu listeden seçilecek araca ilişkin bilgiler iki sekme halinde düzenlenmiştir. "Doluluk" sekmesinde, aracın rotası boyunca ziyaret edeceği düğümler sıralanmış ve aracın, ziyaret ettiği her bir düğümde dağıtıp/toplayacağı yük

miktarları, güncellenecek yükü miktarı ve bu miktarın araç kapasitesine oranı listelenmektedir. "Güzergah" sekmesinde "Araç rotasını görmek için tıklayınız" yazısına basılarak, Şekil 4.10'da görüldüğü gibi, YAK algoritmasının seçilen test problemi için elde ettiği rotaya ait grafik sunulmaktadır.



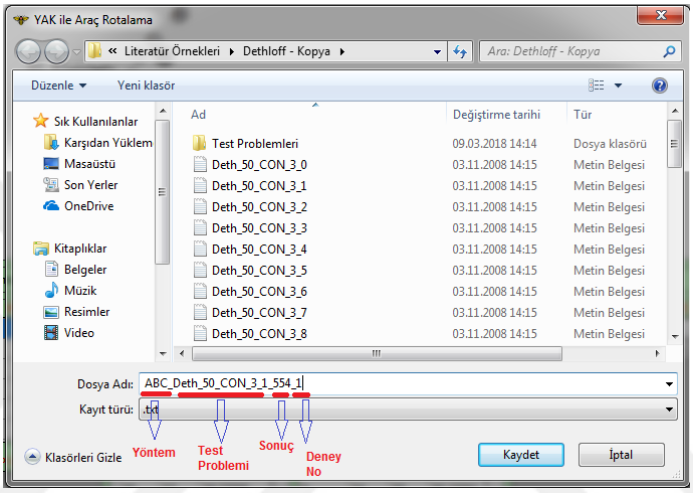
Şekil 4.10. CON3-1 test problemi için elde edilen çözüm rotası grafiği.

Uygulama, test problemleri çözümünde elde edilen tüm verilerin kaydedilmesine imkan sağlamaktadır. Şekil 4.11'de gösterildiği gibi "Dosya\Farklı Kaydet\Literatür Örneği" menü yolu kullanılarak, seçilen test problemi için elde edilen bilgiler, seçilen dizinde text dosyası halinde kaydedilebilecektir.



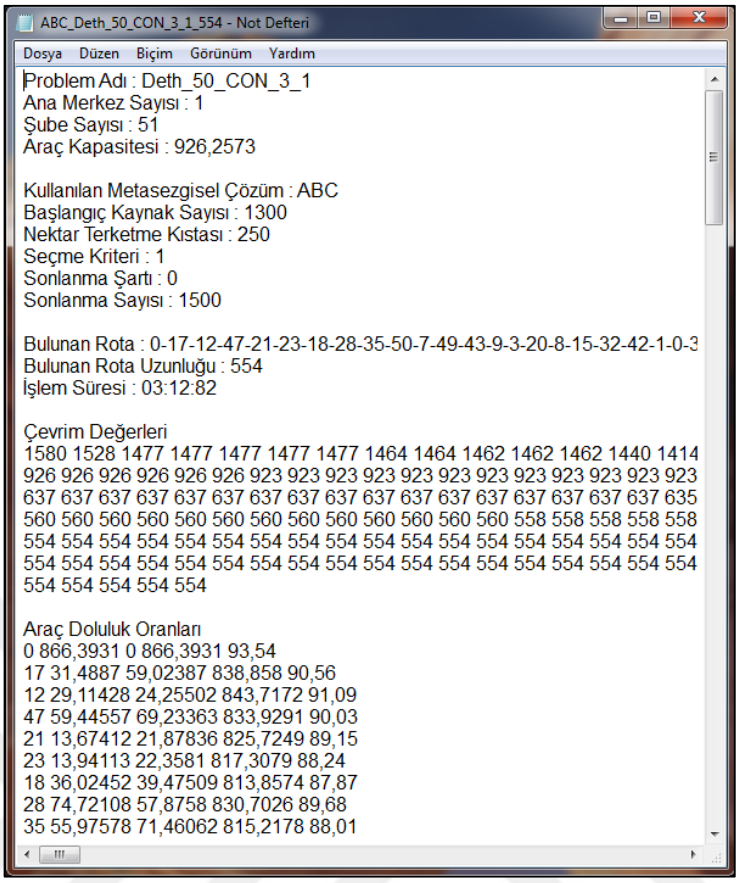
Şekil 4.11. Literatür örnekleri için elde edilen çözümlerin kaydedilmesi.

Uygulama ile aynı test problemi için, YAK algoritmasının aynı parametre değerleriyle, birden fazla sayıda deneme yapmak mümkündür. Test problemi seçilip programa yüklendikten sonra, kaç kez deneme yapılacağı, ilgili alanda seçilir ve "Toplu Uygula" düğmesiyle istenen sayıda deneme yapılması sağlanır. Her bir deneme tamamlandıktan sonra, elde edilen sonuçlar, otomatik olarak kaydedilmektedir. Şekil 4.12'de gösterildiği gibi kaydedilen her bir dosya adı, kullanılan algoritma adı, seçilen test problemi, elde edilen sonuç (tamsayı değer) ve deney numarasından oluşmaktadır.



Şekil 4.12. Aynı test problemi için elde edilen sonuçların toplu halde kaydedilmesi.

Şekil 4.13'te de gösterilen, sonuçların kaydedildiği bir text (.txt) dosyasında, test problemi adı, depo merkezi sayısı, toplam düğüm sayısı, araç kapasitesi, kullanılan çözüm algoritması, algoritma parametreleri için seçilen değerler, bulunan rota, rota mesafesi, çözüm oluşturmada harcanan süre, her bir çevrimde elde edilen sonuçlar, rota boyunca her bir düğümde güncellenecek yük miktarının araç kapasitesine oranı (araç doluluk oranları), düğümler arası mesafeler ve her bir düğümde dağıtılıp/toplanacak talep miktarları yer almaktadır.



Şekil 4.13. Kaydedilen sonuç örneği.

4.4. DENEYSEL ÇALIŞMALAR

EDTARP, yıllardır üzerinde düşünülen, çözümü için kesin ve sezgisel metotlar geliştirilen NP-zor sınıfındaki problem örneklerinden biridir. EDTARP, günlük yaşamdaki kargo ve lojistik faaliyetlerini neredeyse birebir temsil ettiğinden işletmelerin ilgisini çektiği gibi; problem parametreleriyle, kombinatoryal optimizasyon alanındaki araştırmacıların da odaklandığı konular arasında yer almaktadır. Geliştirilen yöntemlerin doğruluk kontrolü, performans analizleri ve karşılaştırılabilmeleri için farklı yön ve yapılarda matematiksel test örnekleri oluşturulmuştur.

Optimizasyon problemleri için önerilen çözüm tekniklerinin performansları, farklı türden benchmark problemleriyle test edilebilmektedir. EDTARP için oluşturulan matematiksel test örnekleri incelendiğinde, kompleks yapısı, geniş ve dalgalanmalı çözüm uzayı itibariyle, araç rotalama alanında mesafe bazlı çözüm bulmaya çalışan

araştırmacıların tercih ettiği Dethloff örnekleri dikkat çekmektedir. Oluşturulduğu zamandan günümüze dek, EDTARP için hazırlanan hemen her çözüm önerisi, bu problemler üzerinde test edilmiştir. Bu test örnekleri üzerinde uygulandığında başarılı sonuçlar elde edebilen yöntemler, daha büyük hacimli problem çözümlerinde de başarılı olabilmişlerdir. Bu bakımdan, YAK algoritması kullanılarak, EDTARP'ye yönelik geliştirilen Meta-sezgisel çözüm önerisi de Dethloff test problemleri üzerinde denenmiştir. Dethloff örneklerinin oluşturulma yöntemi, Bölüm 2'de detaylı olarak açıklanmıştı. Bu bölümde, öncelikle EDTARP parametreleri ele alınmış, Dethloff test problemleri analiz edilmiş ve YAK algoritmasının bu problem çözümleri için hazırlanan deney setleri açıklanmıştır. Elde edilen çözümler, çözüm değerlerinin literatürde bugüne dek elde edilen en başarılı çözümlerle karşılaştırılması ve YAK algoritmasının EDTARP çözümlerindeki performansı, bir sonraki bölümde tablo ve grafik verileriyle tartışılmıştır.

4.4.1. EDTARP Parametreleri

Genel bir yaklaşımla EDTARP çözümlerinde sonucu etkileyen temel öğeler: düğümler arası mesafeler, taşınacak yük miktarları ve araç kapasiteleri olarak düşünülebilir. Düğümler arası mesafelerin sonuca olan etkisini incelemek üzere Şekil 4.14'te, A-B-C-D düğümlerinden oluşan iki farklı ağ (network) örneği gösterilmektedir. Şekil 4.14a'da, network'lerdeki düğümler arası mesafe matrisleri verilmiştir. Şekil 4.14b'de her iki network için tüm düğümleri ziyaret eden tüm çözümler belirlenmiş ve Şekil 4.14c'de ise network'lerin çözüm uzayları grafiksel olarak karşılaştırılmıştır. Her iki network'teki tüm düğümlerin, diğer komşu düğümlere toplam uzaklıkları eşit (18 birim), ancak düğümler arası mesafeler farklı değerlerdedir. Network 1'de düğümler arası mesafelerin standart sapma değeri 3.11 olurken, Network 2'de düğümler arası mesafelerin standart sapma değeri 5.92'dir. Her iki network'ün çözüm uzayındaki 24 adet sonuç değerleri karşılaştırıldığında, Network 1'e ait çözüm maliyetlerinin aritmetik ortalaması 17.917 ve standart sapma değeri 1.381 olarak hesaplanmış, Network 2'ye ait çözüm maliyetlerinin aritmetik ortalaması ise 17.875 ve standart sapma değeri 8.999 olarak hesaplanmıştır.

Model 1				
	A	**B**	**C**	**D**
A	0	5	6	7
B	5	0	7	6
C	6	7	0	5
D	7	6	5	0

Model 2				
	A	**B**	**C**	**D**
A	0	1	13	4
B	1	0	4	13
C	13	4	0	1
D	4	13	1	0

(a) Mesafeler

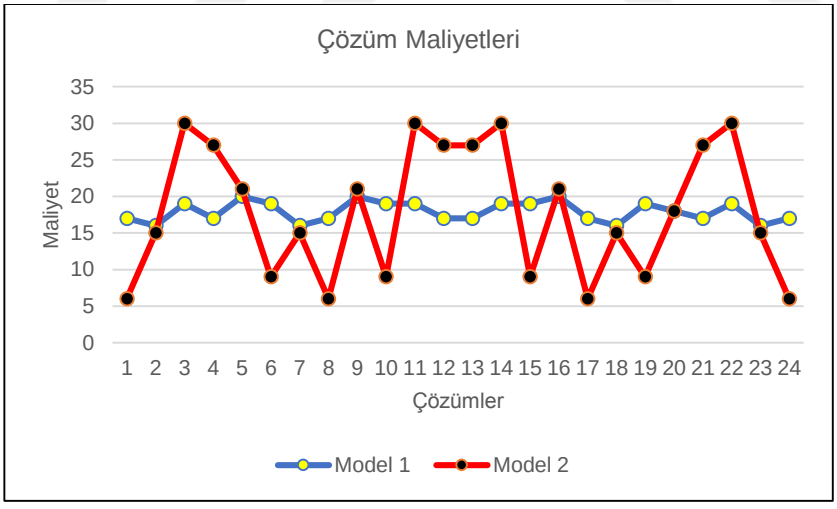
Çözümler				Maliyet	
				M1	**M2**
A	B	C	D	17	6
A	B	D	C	16	15
A	C	B	D	19	30
A	C	D	B	17	27
A	D	B	C	20	21
A	D	C	B	19	9
B	A	C	D	16	15
B	A	D	C	17	6
B	C	A	D	20	21
B	C	D	A	19	9
B	D	A	C	19	30
B	D	C	A	17	27

Çözümler (Devamı)				Maliyet	
				M1	**M2**
C	A	B	D	17	27
C	A	D	B	19	30
C	B	A	D	19	9
C	B	D	A	20	21
C	D	A	B	17	6
C	D	B	A	16	15
D	A	B	C	19	9
D	A	C	B	18	18
D	B	A	C	17	27
D	B	C	A	19	30
D	C	A	B	16	15
D	C	B	A	17	6

(b) Çözümler

	Model 1		**Model 2**	
	Ortalama	Standart Sapma	Ortalama	Standart Sapma
Mesafeler Arası	6,00	3,11	6,00	5,92
Çözümler Arası	17,92	1,38	17,88	9,00

(c) Çözümlerin analizi



(d) Çözümlerin karşılaştırılması

Şekil 4.14. İki farklı ARP örneği ve çözüm maliyetlerinin karşılaştırılması.

Şekil 4.14'teki örnekten net olarak görülebileceği üzere, hemen hemen aynı genişlikte alana yayılmış ve eşit sayıda düğüm içeren iki network'te, tüm düğümleri ziyaret eden çözüm rotaları karşılaştırıldığında, düğümler arası mesafe değerleri farklılık

gösterdikçe, oluşabilecek çözümler arası farklılık da artmaktadır. Şekil 4.14c'de sunulan, Network 2'nin çözüm uzayındaki dalgalanma, Network 1'in çözüm uzayına nazaran daha yüksek seviyededir.

4.4.2. Dethloff Test Problemlerinin Genel Analizi

Dethloff test problemlerinin hazırlanma süreci dikkate alındığında, test problemlerinin her biri için, düğümler arası mesafeler ve dağıtım toplama talepleri parametrelerine ait değerlerde önemli dalgalanmalar oluşacağı beklenmektedir. YAK algoritmasının farklı problem örneklerindeki performansını detaylı ve objektif olarak analiz edebilmek için; öncelikle, Dethloff problemlerindeki mesafe değerlerine (4.1) denklemiyle normalizasyon uygulanmış, (4.2) denklemiyle her bir düğümde taşınacak yük miktarı değişimi belirlenmiş ve belirlenen yük miktarı değişimi (4.3) denklemiyle araç kapasitesine oranlanmıştır. Ardından, elde edilen dizilerin, ortalama ve standart sapma değerleri hesaplanmıştır.

$$c'_{ij} = \frac{(c_{ij} - c_{min})}{(c_{\max} - c_{\min})} \tag{4.1}$$

$$load_i = d_i - p_i \tag{4.2}$$

$$rate_i = load_i \,/\, Q \tag{4.3}$$

Bu işlemler, Dethloff test problemlerinin her birine ayrı ayrı uygulandıktan sonra, oluşan son durum Tablo-1'de gösterilmektedir. Çizelge 4.1 verileri incelendiğinde; CON türü örneklerde, düğümler arası mesafelerin standart sapma değerleri, SCA türündeki düğümler arası mesafelerin standart sapma değerlerine oranla daha düşük seviyelerde olduğu gözlemlenmektedir. Dolayısıyla CON örneklerindeki çözüm değerleri, SCA örneklerindeki çözüm değerlerine nispeten, birbirine daha yakın seviyelerde olacaktır. Düğümlerdeki dağıtım/toplama taleplerinin araç kapasitelerine oranları incelendiğinde ise CON3 ve SCA3 türü problemlerdeki oranların, CON8 ve SCA8 türü problemlerdeki oranlara nispeten daha düşük seviyelerde olduğu gözlemlenmektedir.

Çizelge 4.1. Dethloff problemlerinde normalizasyon sonrası oluşan son durum.

Dethloff Problemleri				
Problem	Mesafeler		Araç Kapasitesine Göre Yükleme Oranı	
	Ortalama	Standart Sapma	Ortalama (e-2)	Standart Sapma (e-1)
SCA3-0	0,477	0,240	-0,072	0,217
SCA3-1	0,516	0,259	-0,086	0,214
SCA3-2	0,507	0,260	-0,384	0,185
SCA3-3	0,505	0,258	-0,130	0,209
SCA3-4	0,533	0,258	-0,167	0,195
SCA3-5	0,511	0,256	-0,104	0,185
SCA3-6	0,473	0,242	0,216	0,208
SCA3-7	0,523	0,257	-0,460	0,211
SCA3-8	0,543	0,267	0,374	0,202
SCA3-9	0,525	0,260	-0,284	0,214
SCA8-0	0,477	0,240	-0,191	0,579
SCA8-1	0,516	0,259	-0,230	0,571
SCA8-2	0,507	0,260	-1,023	0,495
SCA8-3	0,505	0,258	-0,346	0,558
SCA8-4	0,533	0,258	-0,444	0,521
SCA8-5	0,511	0,256	-0,277	0,494
SCA8-6	0,473	0,242	0,576	0,553
SCA8-7	0,523	0,257	-1,227	0,562
SCA8-8	0,543	0,267	0,997	0,540
SCA8-9	0,525	0,260	-0,756	0,570
CON3-0	0,374	0,228	-0,153	0,211
CON3-1	0,360	0,221	-0,135	0,171
CON3-2	0,325	0,205	0,160	0,205
CON3-3	0,385	0,236	-0,512	0,174
CON3-4	0,356	0,206	-0,073	0,168
CON3-5	0,381	0,229	0,238	0,207
CON3-6	0,312	0,189	0,306	0,197
CON3-7	0,412	0,230	0,386	0,205
CON3-8	0,360	0,227	0,025	0,174
CON3-9	0,367	0,227	0,092	0,227
CON8-0	0,374	0,228	0,408	0,562
CON8-1	0,360	0,221	-0,359	0,455
CON8-2	0,325	0,205	0,427	0,546
CON8-3	0,385	0,236	-1,364	0,465
CON8-4	0,356	0,206	-0,194	0,448
CON8-5	0,381	0,229	0,634	0,553
CON8-6	0,312	0,189	0,815	0,524
CON8-7	0,412	0,230	1,029	0,547
CON8-8	0,360	0,227	0,066	0,465
CON8-9	0,367	0,227	0,244	0,606

Dethloff problemlerinde çözüm uzayları, düğümler arası mesafeler ve yük oranı parametreleriyle şekillenecek, bu parametreler için standart sapma değerleri düşük olan problemlerde, çözümler, birbirine daha yakın seviyelerde oluşacaktır. Test problemleri incelendiğinde, CON3-1 problemi, düğümler arası mesafeler ve dağıtım/toplama talepleri itibariyle standart sapma verilerinin düşük olduğu

örneklerden biridir. Dolayısıyla CON3-1 problemi, çözüm uzayındaki birbirine yakın çözüm değerleri arasında en düşük maliyetli çözüm aranan, en zor Dethloff örneklerinden biri olarak düşünülebilir. Çalışmada, YAK algoritmasının, EDTARP çözümü için değişen parametre değerlerindeki arama yaklaşımını analiz etmek üzere CON3-1 problemi seçilmiştir. Algoritma daha sonra, farklı çözüm uzaylarına sahip diğer tüm Dethloff problemleri üzerinde denenerek performansı incelenmiştir.

4.4.3. YAK Algoritmasının CON3-1 Test Problemi İçin Deney Tasarımı

YAK algoritması, CON3-1 test problemi için farklı parametre değerleriyle denendiğinde; algoritmanın, nektar=1000, limit=150 ve iterasyon=1200 parametre değerleriyle daha başarılı sonuçlar elde ettiği gözlemlenmiştir. Dolayısıyla deney tasarımı, daha başarılı sonuçlar elde eden parametre verilerine yakın değerlerin, farklı kombinasyonlarıyla oluşturulmuştur. Bu bağlamda parametreler;

1. Nektar = 500..1500 (Δ=250)
2. Limit = 50..250 (Δ=50)
3. İterasyon = 1000..1500 (Δ=100)

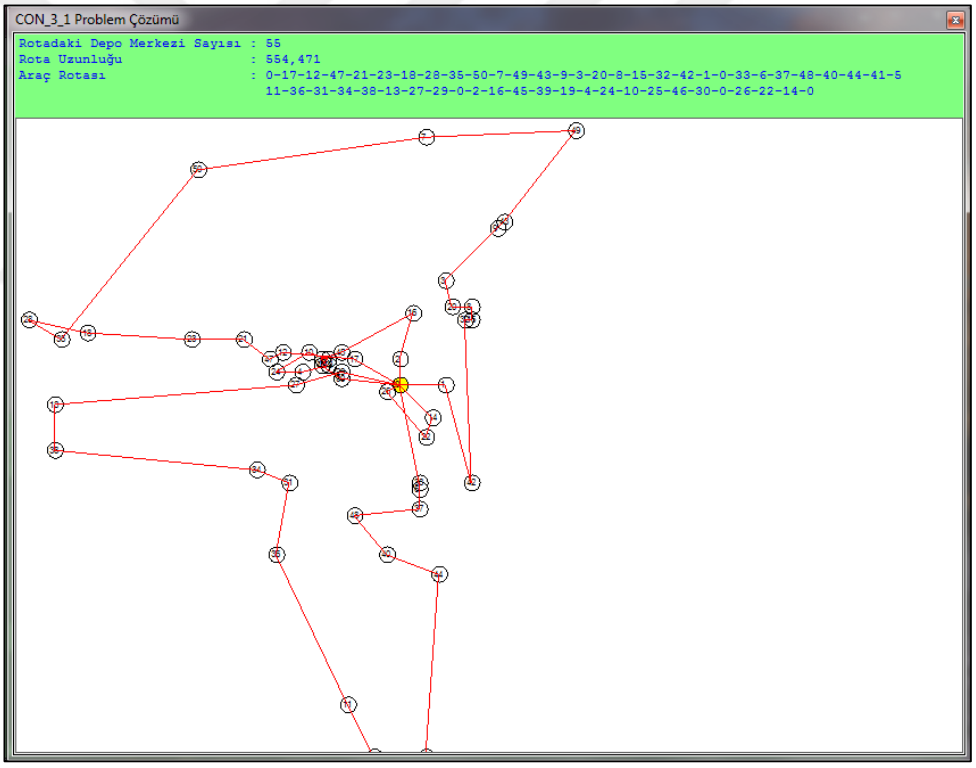
olarak seçilmiş, parametrelerden birine ait değer sabit tutulmak koşuluyla, diğer parametre değerleri değiştirilerek, 150 farklı durum oluşturulmuştur. Algoritma, durumların her biri için 40'ar kez denenmiş, her bir denemede elde edilen en başarılı çözüm rotaları, rota maliyetleri (tam sayı olarak), araç doluluk oranları ve çözüm için harcanan CPU süreleri kaydedilmiştir.

4.5. DENEYSEL SONUÇLAR

YAK algoritmasının EDTARP için çözüm arama yaklaşımı ve algoritma kontrol parametrelerinin çözüm arama faaliyetlerindeki etkisi; algoritmanın, farklı parametre değerleriyle CON3-1 probleminden elde ettiği sonuçlar üzerinden değerlendirilmiştir. Ardından algoritma, diğer Dethloff problem örnekleri üzerinde denenerek elde edilen sonuçlar, literatürde bugüne dek bulunan en başarılı çözümlerle karşılaştırılmış, YAK algoritmasının EDTARP çözümündeki performansı analiz edilmiştir.

4.5.1. CON3-1 Test Problemi Çözümünde Elde Edilen En Başarılı Sonuç

CON3-1 problemi, Dethloff problemleri içinde, problemin parametre değerleri arasındaki standart sapma verilerinin en düşük seviyede olduğu problem örneklerinden biridir. Bu doğrultuda, problemin çözüm uzayında oluşacak sonuç değerlerinin de birbirine yakın seviyelerde olması beklenir. Dolayısıyla CON3-1 problemi için, çözümünün diğer problemlere nazaran daha zor bir Dethloff modeli olduğu düşünülebilir. YAK algoritmasının, kontrol parametreleri için belirlenen değerler, kombinatif halde düzenlenerek, deney seti oluşturulmuş ve CON3-1 çözümü için uygulanmıştır. YAK algoritmasının CON3-1 problemi için elde ettiği en başarılı çözüm Şekil 4.15'te gösterilmektedir.

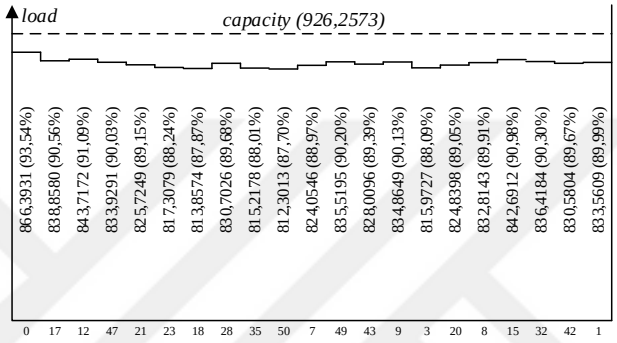


Şekil 4.15. YAK ile CON3-1 problemi için elde edilen en başarılı çözüm.

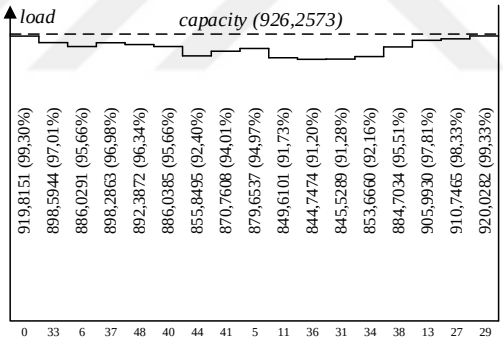
Şekil 4.15'ten de görülebileceği üzere, çözüm için (depo merkezine dönüşler de dahil) toplam 55 düğüm ziyaret edilmekte ve toplamda 554,471 birim mesafe kat edilmektedir. CON3-1 problemi için YAK algoritmasının elde ettiği bu çözüm sonucu, literatürde bugüne kadar elde edilen en başarılı sonuca eşittir. Ziyaret edilecek düğümler şu şekilde sıralanmıştır:

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

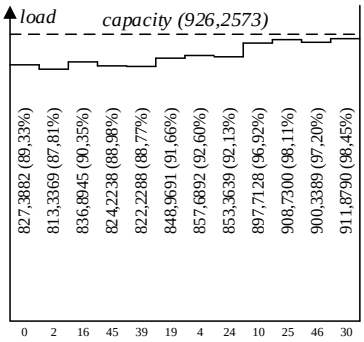
Belirlenen çözümde, taşımacılık faaliyeti için dört adet nakliye aracı rotalandırılmıştır. Her biri 926,2573 birim, eş kapasiteli bu dört aracın, rotaları üzerindeki her bir düğümde dağıtım/toplama sonrası güncellenen yük miktarları ve taşınan yük miktarlarının araç kapasitesine oranları Şekil 4.16'da sunulmaktadır.



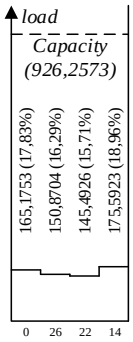
(a) Araç 1



(b) Araç 2



(c) Araç 3



(d) Araç 4

Şekil 4.16. CON3-1 çözümünde taşıma araçlarının doluluk durumu.

4.5.2. CON3-1 Test Problem Çözümü İçin YAK Algoritması Parametre Analizi

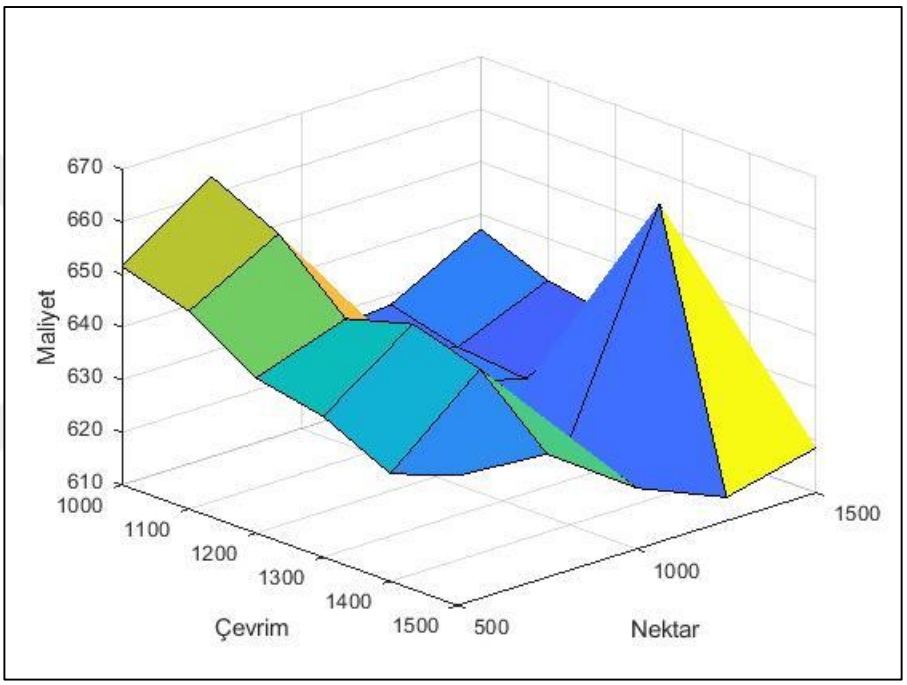
YAK algoritmasının, çözümü zor EDTARP test problemlerinden biri olan CON3-1 örneğindeki parametre analizi için, problem çözümünden elde edilen sonuçlar, algoritma parametreleriyle birlikte ele alınmıştır. Bu bağlamda Çizelge 4.2'de, YAK algoritmasının farklı parametre değerleriyle, CON3-1 problem çözümü için denemelerden elde edilen en düşük ve en yüksek maliyetli sonuçlar, sonuçların aritmetik ortalamaları ve standart sapma değerleri, birlikte sunulmaktadır.

Çizelge 4.2 CON3-1 problem çözümünde ABC parametrelerinin farklı değerleriyle elde edilen sonuç analizi

CON 3.1																														
			Limit = 50 Çevrim						Limit = 100 Çevrim						Limit = 150 Çevrim						Limit = 200 Çevrim						Limit = 250 Çevrim					
			1000	1100	1200	1300	1400	1500	1000	1100	1200	1300	1400	1500	1000	1100	1200	1300	1400	1500	1000	1100	1200	1300	1400	1500	1000	1100	1200	1300	1400	1500
Nektar	500	Ortalama	727,50	709,05	711,75	697,80	678,48	698,45	645,43	652,05	640,03	634,38	633,35	625,88	638,48	625,65	626,23	624,15	624,30	612,48	628,08	625,23	610,55	615,10	611,83	619,43	618,30	626,08	608,83	612,38	604,35	617,00
		Maksimum	831,00	832,00	820,00	825,00	823,00	844,00	736,00	775,00	740,00	758,00	712,00	757,00	712,00	717,00	718,00	686,00	711,00	735,00	692,00	701,00	667,00	674,00	664,00	663,00	670,00	961,00	653,00	672,00	655,00	684,00
		Minimum	609,00	596,00	619,00	600,00	612,00	600,00	587,00	594,00	590,00	587,00	587,00	569,00	585,00	585,00	579,00	580,00	586,00	571,00	577,00	583,00	573,00	574,00	556,00	573,00	569,00	582,00	585,00	576,00	570,00	565,00
		Standart Sapma	73,24	73,94	73,24	73,25	61,37	77,70	40,23	45,13	33,06	33,98	29,72	32,46	31,63	32,18	37,35	22,62	28,38	27,84	24,44	24,05	21,06	23,34	24,01	21,42	24,07	58,76	16,56	21,11	16,86	28,07
	750	Ortalama	761,38	763,43	735,53	754,80	738,35	707,50	680,93	661,13	646,50	656,13	644,20	629,53	642,30	634,05	625,18	619,50	626,63	619,98	624,73	617,03	612,95	606,30	614,35	607,98	606,93	609,18	606,90	608,08	601,35	602,13
		Maksimum	838,00	823,00	831,00	826,00	829,00	815,00	751,00	754,00	736,00	755,00	749,00	730,00	715,00	697,00	722,00	681,00	711,00	698,00	721,00	688,00	669,00	682,00	661,00	657,00	653,00	664,00	645,00	662,00	674,00	681,00
		Minimum	632,00	622,00	612,00	597,00	600,00	612,00	579,00	586,00	583,00	597,00	580,00	582,00	590,00	578,00	583,00	588,00	580,00	574,00	580,00	568,00	578,00	574,00	566,00	571,00	570,00	573,00	581,00	567,00	568,00	568,00
		Standart Sapma	64,11	58,70	68,86	68,00	69,25	71,57	45,25	45,24	40,02	45,22	43,36	28,74	36,42	29,13	34,82	23,27	31,92	26,98	30,56	29,56	21,04	23,55	19,91	20,81	18,11	20,39	15,38	20,07	23,86	19,49
	1000	Ortalama	690,95	684,78	679,45	680,83	687,70	676,68	617,43	632,78	624,30	633,83	625,00	630,25	611,98	623,23	611,68	613,78	612,75	604,98	606,13	608,53	614,50	605,95	605,50	598,33	605,95	591,59	607,80	596,68	600,43	597,75
		Maksimum	804,00	788,00	784,00	801,00	805,00	778,00	697,00	732,00	712,00	709,00	748,00	725,00	700,00	703,00	657,00	663,00	697,00	651,00	656,00	674,00	685,00	661,00	640,00	649,00	684,00	655,00	642,00	661,00	652,00	667,00
		Minimum	581,00	579,00	579,00	588,00	589,00	588,00	574,00	570,00	570,00	577,00	573,00	588,00	573,00	581,00	554,00	574,00	571,00	565,00	559,00	573,00	575,00	570,00	575,00	562,00	558,00	561,00	572,00	571,00	574,00	568,00
		Standart Sapma	77,63	67,99	61,60	66,35	63,96	60,09	32,58	39,75	33,88	37,99	36,09	32,90	27,25	28,96	21,91	22,68	28,46	22,36	24,30	21,42	25,83	24,03	19,33	21,68	28,03	23,34	18,06	21,39	19,70	23,10
	1250	Ortalama	712,05	711,83	695,45	691,85	684,45	668,18	610,18	610,03	610,83	615,88	618,25	609,60	608,18	604,88	608,10	607,05	612,63	603,30	608,05	599,95	602,53	598,60	808,40	595,40	602,43	596,98	599,73	600,13	605,08	595,93
		Maksimum	777,00	767,00	768,00	752,00	764,00	762,00	668,00	661,00	669,00	712,00	705,00	666,00	653,00	652,00	657,00	655,00	660,00	652,00	678,00	660,00	659,00	647,00	1033,0	655,00	669,00	654,00	643,00	654,00	666,00	636,00
		Minimum	617,00	611,00	614,00	610,00	600,00	590,00	573,00	559,00	570,00	574,00	577,00	567,00	570,00	567,00	568,00	570,00	574,00	559,00	560,00	567,00	566,00	567,00	585,00	564,00	558,00	559,00	562,00	568,00	566,00	561,00
		Standart Sapma	38,59	45,00	45,31	40,21	48,68	50,50	23,97	26,60	24,52	27,65	27,11	23,38	21,51	24,27	20,21	18,48	23,17	24,13	24,85	20,70	19,82	19,56	196,07	23,86	24,83	19,53	19,19	17,26	20,55	20,17
	1500	Ortalama	714,70	705,88	707,93	706,78	693,45	692,70	639,45	627,90	629,05	625,58	627,30	624,93	620,10	617,48	606,70	604,50	602,63	598,35	612,65	603,50	602,80	599,28	595,93	588,73	599,13	605,35	601,15	597,90	589,93	587,83
		Maksimum	769,00	776,00	769,00	751,00	768,00	739,00	680,00	678,00	670,00	672,00	677,00	690,00	676,00	658,00	665,00	653,00	647,00	645,00	671,00	645,00	639,00	632,00	637,00	634,00	640,00	654,00	644,00	639,00	613,00	630,00
		Minimum	623,00	602,00	593,00	595,00	609,00	606,00	592,00	575,00	570,00	573,00	598,00	588,00	569,00	580,00	571,00	574,00	572,00	574,00	561,00	567,00	575,00	573,00	568,00	567,00	558,00	571,00	567,00	573,00	569,00	565,00
		Standart Sapma	41,20	38,75	37,88	36,20	37,51	35,07	24,21	26,65	25,41	24,14	20,39	25,41	24,20	17,93	21,74	19,32	19,61	15,25	22,88	20,33	17,78	16,03	15,24	14,16	17,52	21,23	18,77	17,24	10,51	12,70

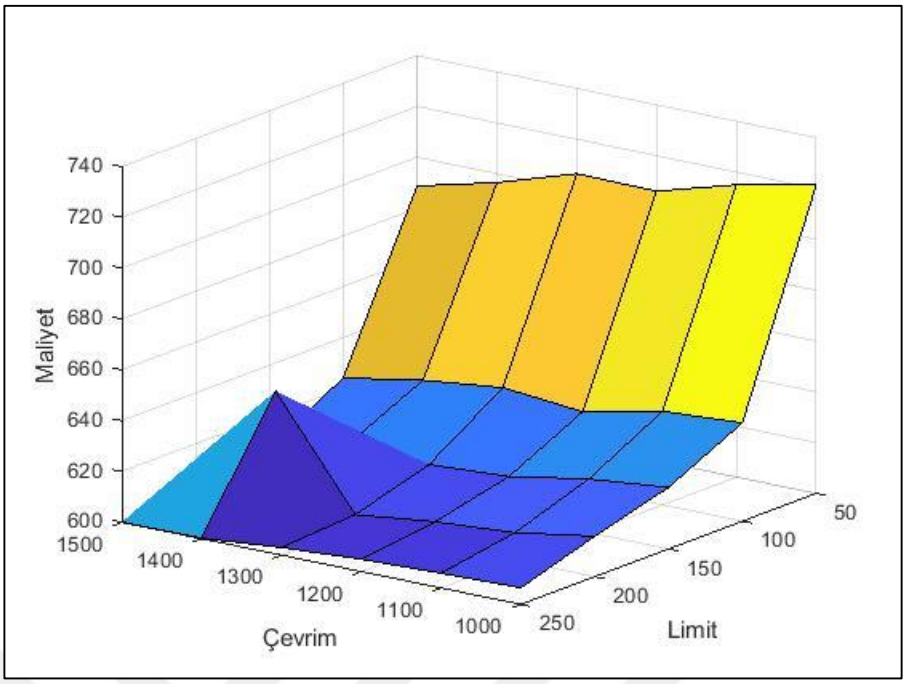
Çizelge 4.2'deki standart sapma verileri karşılaştırıldığında, nektar sayısı, limit ve iterasyon parametrelerine ait değerler artırılarak, algoritma ile daha kararlı çözümler elde edilebildiği görülmektedir.

Çizelge 4.2'deki veriler kullanılarak, YAK algoritması parametrelerinin çözüme etkisini değerlendirebilmek üzere grafikler oluşturulmuştur. Şekil 4.17'deki grafikte, CON3-1 çözümünde, YAK algoritması Çevrim ve Nektar parametrelerinin değişen değerleriyle elde edilen ortalama sonuçlar sunulmaktadır.



Şekil 4.17. CON3-1 çözümünde YAK algoritması Çevrim-Nektar parametre analizi.

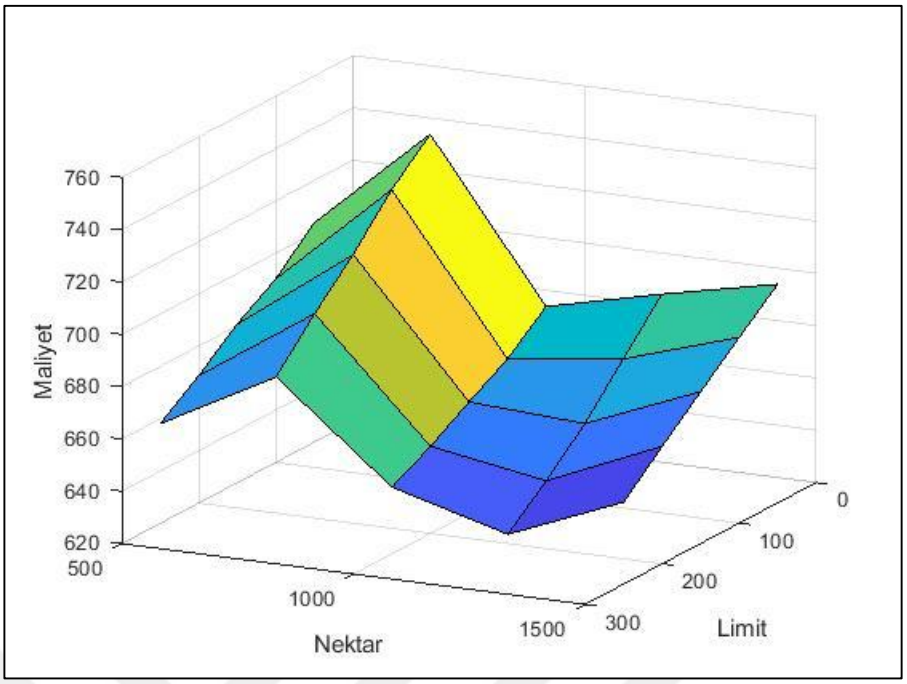
Şekil 4.17'ye bakarak, Çevrim ve Nektar parametrelerinin, çözüm performansı üzerinde anlamlı bir etkisinin olmadığı görülebilir. Algoritmanın ilgili parametreleri için değerler sabit aralıklarla artırılmış, ancak artan/azalan, düzensiz sonuçlar elde edilmiştir. Şekil 4.18'deki Çevrim-Limit parametre analizinde, ilgili parametre değerleri artırıldığında daha başarılı çözümlere ulaşıldığı ancak belirleyici unsurun, limit parametresi olduğu fark edilmektedir.



Şekil 4.18. CON3-1 çözümünde YAK algoritması Çevrim-Limit parametre analizi.

Şekil 4.18'deki Nektar-Limit grafiğinde ise, en başarılı sonuçların nektar parametresinin 1250 değerlerinde elde edildiği ve limit parametresinin etkisinin nektar sayısıyla orantılı olduğu gözlemlenmektedir.

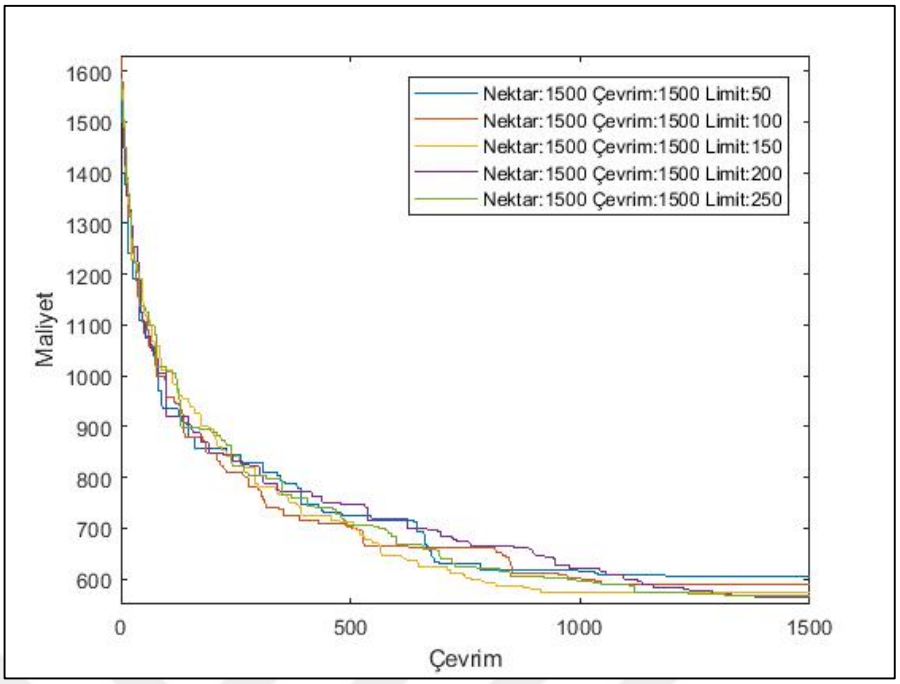
Algoritmanın, yerel optimum çözüme takılmadan, farklı arama bölgelerine dallanabilmesini sağlayan parametre, limit parametresidir. Limit parametresi için düşük değerler seçilerek, ortamda bağımsız dolaşan kâşif arı sayısı artırılabilir. Ancak daha etkin bir yerel arama isteniyorsa, limit parametresi için yüksek değerler tercih edilmelidir. Bölüm 2'de detaylı olarak incelenen CON3-1 problemi, çözüm maliyetleri birbirine yakın değerlerden oluşması beklenen bir optimizasyon problemidir. Şekil 4.17, Şekil 4.18 ve Şekil 4.19'daki sonuçlar dikkate alınarak, "daha geniş koloni" ile "farklı bölgelere sıçrayarak" nispeten daha başarılı çözümlere ulaşılabilse de CON3-1 problem çözümü için etkili bir yerel aramanın daha avantajlı olduğu görülmektedir.



Şekil 4.19. CON3-1 çözümünde YAK algoritması Nektar-Limit parametre analizi.

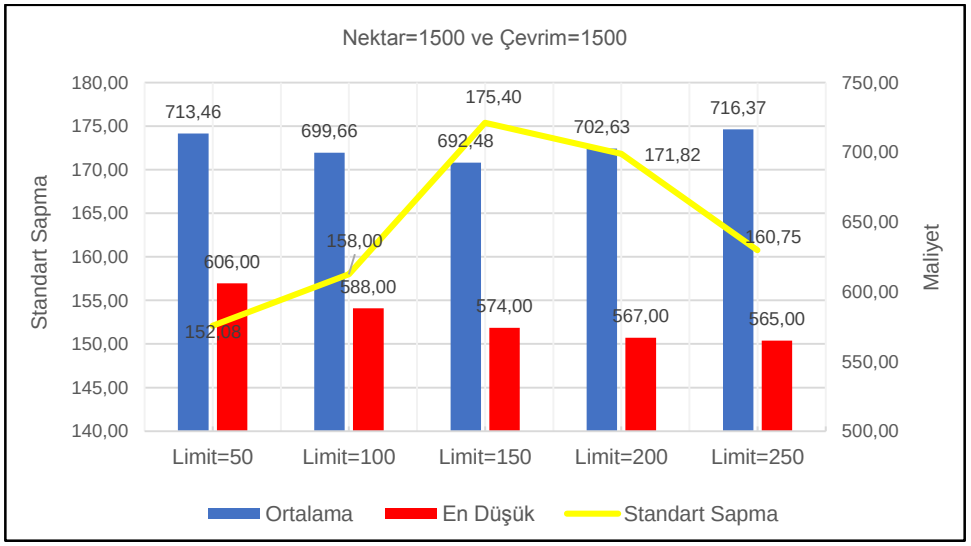
Bahsedildiği üzere YAK algoritmasında kâşif arı oluşumunu etkileyen parametreler, limit, koloni boyutu ve çevrim sayısı parametrelerine ait değerlerdir. Çizelge 4.2'deki veriler incelendiğinde, en çok sayıda kâşif arı oluşturacak parametre değerleri; nektar=1500 ve çevrim=1500 değerleri olarak yorumlanabilir. YAK algoritmasının, nektar ve çevrim parametreleri 1500 değerinde iken, limit parametresinin değişen değerlerinde (50..250 Δ=50) elde edilen en başarılı çözümler, Şekil 4.20'da gösterilmektedir.

Şekil 4.20'deki grafiğe bakarak algoritmanın belirlenen parametre değerleriyle başarılı bir arama yaptığı, yerel en iyi sonuca takılmadan, daha başarılı çözümler arama eğiliminde hareket ettiği söylenebilir.



Şekil 4.20. CON3-1 çözümünde YAK algoritması limit parametre analizi.

Belirlenen parametre değerleriyle yapılan denemelerden alınan en başarılı çözümlere ait ortalama ve standart sapma verileri Şekil 4.21'de gösterilmektedir. Şekil 4.21'deki sonuç grafiğine bakarak, CON3-1 problem çözümündeki en başarılı çözümlerde de yine farklı bölgelere sıçramaktan ziyade, yerel aramada yoğunlaşmak gerektiği görülebilmektedir.



Şekil 4.21. CON3-1 çözümünde Limit parametre analizi için elde edilen veriler.

4.5.3. YAK Algoritması performans Analizi

Algoritma çıktılarının detaylı analizleri verilen CON3-1 probleminde olduğu gibi, diğer tüm CON ve SCA örnekleri için de YAK algoritmasıyla çözüm aranmıştır. Elde edilen sonuçlar, yine bu test problemleri için çözüm elde eden literatür çalışmalarının en başarılı sonuç değerleriyle karşılaştırılmıştır. Çizelge 4.3'te, CON ve SCA örnekleri için sonuç elde eden farklı yaklaşımlar sunulmaktadır.

Çizelge 4.3 CON ve SCA problemleri için geliştirilen literatür çalışmaları.

Yöntem
Geniş Komşuluk Arama (Large Neighborhood Search – LNS) [70]
Karınca Koloni Sistemi (Ant Colony System – ACS) [30]
Paralel İteratif Lokal Arama (Parallel Iterative Local Search – PILS) [25]
Uyarlamalı Hafıza Metodoloji (Adaptive Memory Methodology – AMM) [71]
Karma Parçacık Sürü Optimizasyonu (hybrid Particle Swarm Optimization – h_PSO) [72]
Uyarlamalı TA ve Elit Karınca Sistemi (Modified Tabu Search and Elite Ant System – MTSEAS) [59]
DKA ile Güçlendirilmiş KKS (Ant Colony System Empowered Variable Neighborhood Search – ACSEVNS) [62]
Uyarlamalı Yerel Arama (Adaptive Local Search – ALS) [63]
Karışık Tamsayılı Doğrusal Programlama (Mixed-Integer Linear Programming – MILP) [64]
Çoklu Komşuluk Rehberli Yerel Arama (Multiple Neighborhood Guided Local Search - MN_GLS) [65]
Etkin Karınca Koloni Optimizasyonu (Effective Ant Colony Optimization – EACO) [66]

Çizelge 4.4'te ise CON ve SCA için farklı yaklaşımlar denenerek geliştirilmiş çözümlerin elde ettiği en başarılı sonuçlar sunulmaktadır.

Çizelge 4.4. Literatürdeki çalışmalarda elde edilmiş en başarılı sonuçlar.

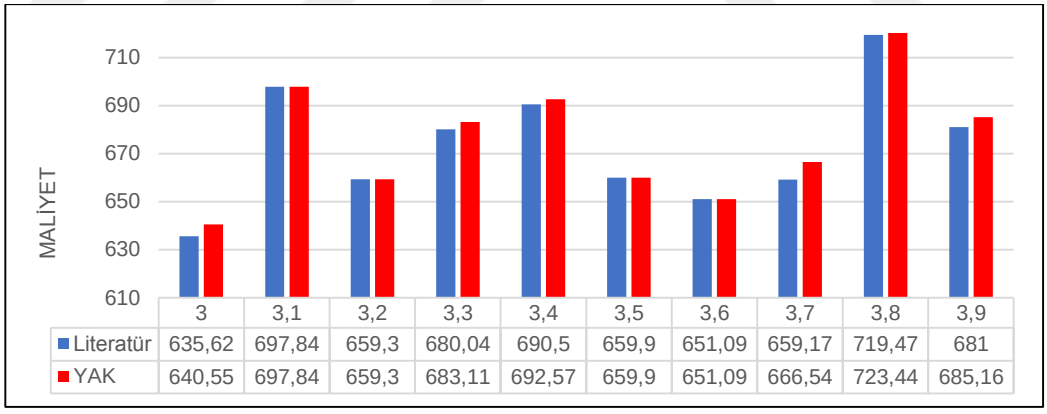
Test Problemi	LNS	ACS	PILS	AMM	h_PSO	MTSEAS	ACSEVNS	ALS	MILP	MN_GLS	EACO
SCA3-0	636,10	635,62	635,62	635,62	635,62	635,62	635,62	635,62	677,35	635,66	635,93
SCA3-1	697,84	697,84	697,84	697,84	697,84	697,84	697,84	697,84	758,90	697,84	697,84
SCA3-2	659,30	659,34	659,34	659,34	659,34	659,34	659,34	659,34	735,18	659,34	659,34
SCA3-3	680,60	680,04	680,04	680,04	680,04	680,04	680,04	680,04	735,79	680,04	680,04
SCA3-4	690,50	690,50	690,50	690,50	690,50	690,50	690,50	690,50	741,75	690,50	690,50
SCA3-5	659,90	659,90	659,90	659,90	659,90	659,91	659,91	659,90	702,45	659,96	659,90
SCA3-6	651,10	651,09	651,09	651,09	651,09	651,09	651,09	651,09	707,72	651,09	651,09
SCA3-7	666,10	659,17	659,17	659,17	659,17	659,17	659,17	659,17	708,24	659,17	659,17
SCA3-8	719,50	719,47	719,48	719,47	719,47	719,48	719,48	719,47	771,94	719,47	719,47
SCA3-9	681,00	681,00	681,00	681,00	681,00	681,00	681,00	681,00	726,77	681,00	681,00
SCA8-0	975,10	961,50	961,50	961,50	961,50	961,50	961,50	961,50	1026,79	961,50	964,81
SCA8-1	1052,40	1049,65	1049,65	1049,65	1049,65	1052,04	1049,65	1049,65	1127,41	1049,65	1049,65
SCA8-2	1044,50	1042,69	1039,64	1039,64	1039,64	1039,64	1039,64	1039,64	1126,12	1039,64	1042,64
SCA8-3	999,10	983,34	983,34	983,34	983,34	983,34	983,34	983,34	1062,99	983,34	983,34
SCA8-4	1065,50	1065,49	1065,49	1065,49	1065,49	1065,49	1065,49	1065,49	1114,12	1065,49	1065,49
SCA8-5	1027,10	1027,08	1027,08	1027,08	1027,08	1027,08	1027,08	1027,08	1085,96	1027,08	1027,08
SCA8-6	977,00	971,82	971,82	971,82	971,82	971,82	971,82	971,82	1038,59	971,82	975,19
SCA8-7	1061,00	1052,17	1051,28	1051,28	1051,28	1061,00	1051,28	1051,28	1114,17	1051,28	1051,28
SCA8-8	1071,20	1071,18	1071,18	1071,18	1071,18	1071,18	1071,18	1071,18	1165,08	1071,18	1071,18
SCA8-9	1060,50	1060,50	1060,50	1060,50	1060,50	1060,50	1060,50	1060,50	1145,71	1060,50	1062,34
CON3-0	616,50	616,52	616,52	616,52	616,52	616,52	616,52	616,52	667,46	616,52	616,52
CON3-1	554,50	554,47	554,47	554,47	554,47	554,47	554,47	554,47	590,82	554,47	554,47
CON3-2	521,40	518,00	518,00	518,00	518,00	518,01	518,00	518,00	558,89	518,00	519,89
CON3-3	591,20	591,19	591,19	591,19	591,19	591,19	591,19	591,19	634,93	591,19	591,19
CON3-4	588,80	588,79	588,79	588,79	588,79	588,79	588,79	588,79	627,95	588,79	588,79
CON3-5	563,70	563,70	563,70	563,70	563,70	563,70	563,70	563,70	603,56	563,70	563,70
CON3-6	500,80	499,05	499,05	499,05	499,05	500,80	499,05	499,05	539,58	499,05	500,21
CON3-7	576,50	576,48	576,48	576,48	576,48	576,48	576,48	576,48	627,05	576,48	576,48
CON3-8	523,10	523,05	523,05	523,05	523,05	523,05	523,05	523,05	561,65	523,05	523,05
CON3-9	586,40	578,25	578,25	578,25	578,25	578,25	578,25	578,25	619,95	578,25	578,25
CON8-0	857,20	857,17	857,17	857,17	857,17	857,17	857,17	857,17	918,21	857,23	859,93
CON8-1	740,90	740,85	740,85	740,85	740,85	740,85	740,85	740,85	772,44	740,85	740,85
CON8-2	716,00	712,89	712,89	712,89	712,89	712,89	712,89	712,89	738,99	712,89	712,89
CON8-3	811,10	811,07	811,07	811,07	811,07	811,07	811,07	811,07	857,35	811,07	811,07
CON8-4	772,30	772,25	772,25	772,25	772,25	772,25	772,25	772,25	816,81	772,25	772,25
CON8-5	755,70	754,88	754,88	754,88	754,88	755,70	754,88	754,88	798,07	754,88	754,88
CON8-6	693,10	678,92	678,92	678,92	678,92	678,92	678,92	678,92	718,36	678,92	678,92
CON8-7	814,80	811,96	811,96	811,96	811,96	814,80	811,96	811,96	863,39	811,96	812,55
CON8-8	774,00	767,53	767,53	767,53	767,53	767,53	767,53	767,53	808,17	767,53	767,79
CON8-9	809,30	809,00	809,00	809,00	809,00	809,00	809,00	809,00	843,84	809,00	809,00

Çizelge 4.5, Çizelge 4.6 Çizelge 4.7 ve Çizelge 4.8'de, YAK algoritmasının CON ve SCA test problemleri çözümlerinde elde ettiği sonuçlar ve Çizelge 4.4'te listelenen literatür çalışmalarının bu problem çözümlerinde elde ettiği en başarılı sonuçlar sunulmaktadır. Çizelgelerde YAK algoritması çözümlerine ilişkin; her bir problem

örneğinde elde ettiği en başarılı sonuçlar, sonuçların ortalama değerleri, standart sapma verileri, en iyi sonucu bulurken harcanan işlemci zamanı ve her bir problem çözümü için harcanan ortalama işlem zamanları da verilmektedir. Şekil 4.22, Şekil 4.23, Şekil 4.24 ve Şekil 4.25'te ise YAK ile elde edilen en başarılı sonuçlar, literatürde elde edilen en iyi sonuçlarla karşılaştırılmaktadır.

Çizelge 4.5. SCA3 Problem çözümlerinde YAK ile elde edilen çözümler ve literatürde elde edilmiş en iyi sonuçlar.

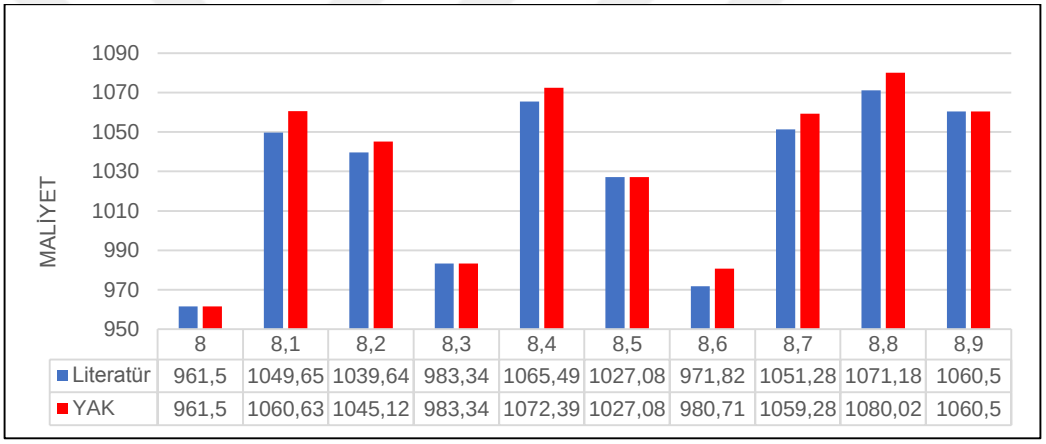
Test Problemi	Dethloff	Literatür En İyi	YAK ALGORİTMASI SONUÇLARI						
			En İyi Çözüm	Ortalama	Standart Sapma	% Oran (Dethloff'a Göre)	% Oran (Literatüre Göre)	İşlem Süresi (En İyi Çözüm İçin)	Ortalama İşlem Süresi
SCA3-0	689,00	635,62	640,55	664,05	17,88	7,03	-0,78	07:47,59	09:33,97
SCA3-1	765,60	697,84	697,84	715,60	12,16	8,85	0,00	08:16,62	07:45,83
SCA3-2	742,80	659,30	659,30	685,43	18,88	11,24	0,00	08:45,92	10:41,11
SCA3-3	737,20	680,04	683,11	701,93	17,40	7,34	-0,45	07:34,52	09:45,85
SCA3-4	747,10	690,50	692,57	718,13	13,09	7,30	-0,30	09:44,89	08:32,53
SCA3-5	784,40	659,90	659,90	686,73	12,18	15,87	0,00	08:36,93	09:57,41
SCA3-6	720,40	651,09	651,09	668,07	15,17	9,62	0,00	08:44,29	07:37,69
SCA3-7	707,90	659,17	666,54	687,83	15,17	5,84	-1,12	03:29,28	03:17,13
SCA3-8	807,20	719,47	723,44	742,13	19,10	10,38	-0,55	08:01,57	07:16,45
SCA3-9	764,10	681,00	685,16	704,50	11,51	10,33	-0,61	09:24,36	12:35,67



Şekil 4.22. SCA3 problemleri çözümlerinde YAK sonuçlarının literatürdeki en iyi sonuçlarla karşılaştırılması.

Çizelge 4.6. SCA8 Problem çözümlerinde YAK ile elde edilen çözümler ve literatürde elde edilmiş en iyi sonuçlar.

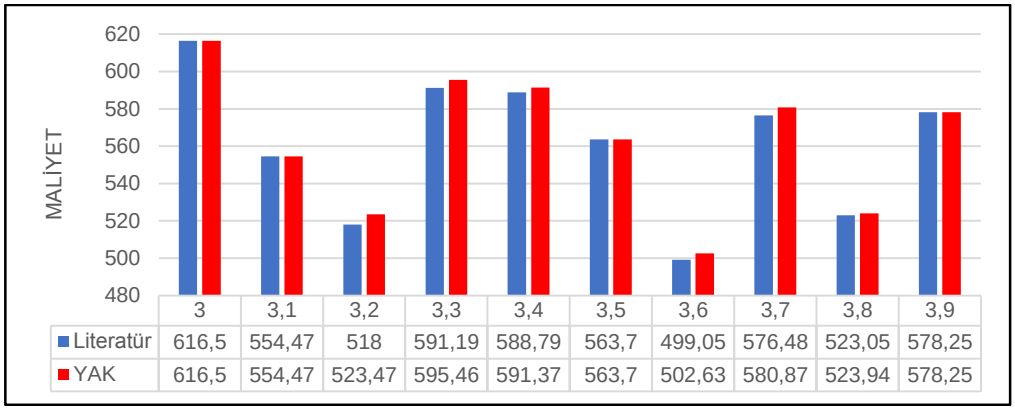
Test Problemi	Dethloff	Literatür En İyi	YAK ALGORİTMASI SONUÇLARI						
			En İyi Çözüm	Ortalama	Standart Sapma	% Oran (Dethloff'a Göre)	% Oran (Literatüre Göre)	İşlem Süresi (En İyi Çözüm İçin)	Ortalama İşlem Süresi
SCA8-0	1132,90	961,50	961,50	995,65	15,19	15,13	0,00	08:01,33	07:09,46
SCA8-1	1150,90	1049,65	1060,63	1001,80	18,60	7,84	-1,05	07:41,27	08:11,98
SCA8-2	1100,80	1039,64	1045,12	1092,20	16,37	5,06	-0,53	05:15,43	09:03,07
SCA8-3	1115,60	983,34	983,34	1011,93	8,47	11,86	0,00	03:38,78	03:43,82
SCA8-4	1235,40	1065,49	1072,39	1121,13	13,73	13,19	-0,65	08:35,33	08:03,47
SCA8-5	1231,60	1027,08	1027,08	1059,73	15,07	16,61	0,00	08:08,24	08:21,11
SCA8-6	1062,50	971,82	980,71	1039,40	15,43	7,70	-0,91	08:04,65	10:31,59
SCA8-7	1217,40	1051,28	1059,28	1096,24	14,11	12,99	-0,76	19:13,45	19:41,83
SCA8-8	1231,60	1071,18	1080,02	1131,53	12,53	12,31	-0,83	14:37,42	13:11,68
SCA8-9	1185,60	1060,50	1060,50	1112,97	19,59	10,55	0,00	15:11,89	14:01,46



Şekil 4.23. SCA8 problemleri çözümlerinde YAK sonuçlarının literatürdeki en iyi sonuçlarla karşılaştırılması.

Çizelge 4.7. CON3 Problem çözümlerinde YAK ile elde edilen çözümler ve literatürde elde edilmiş en iyi sonuçlar.

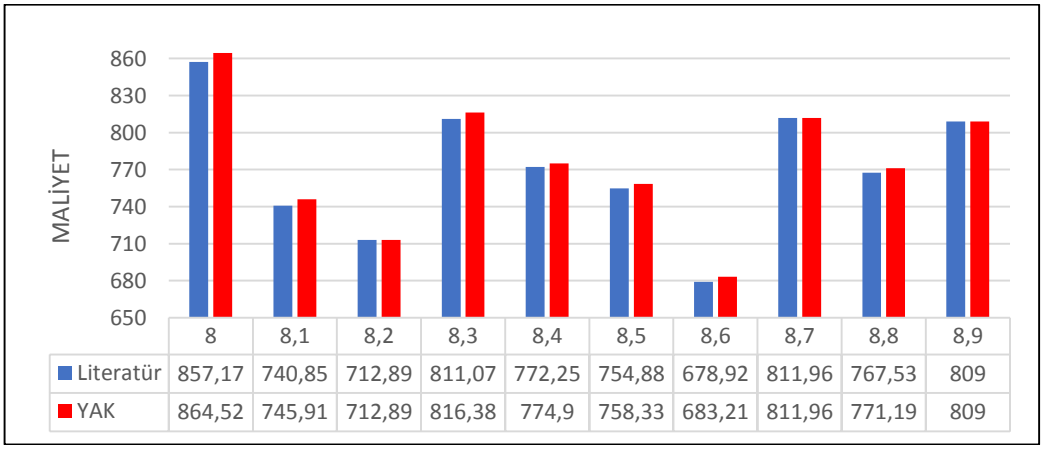
Test Problemi	Dethloff	Literatür En İyi	YAK ALGORİTMASI SONUÇLARI						
			En İyi Çözüm	Ortalama	Standart Sapma	% Oran (Dethloff'a Göre)	% Oran (Literatüre Göre)	İşlem Süresi (En İyi Çözüm İçin)	Ortalama İşlem Süresi
CON3-0	672,40	616,50	616,50	637,93	19,19	8,31	0,00	08:52,90	09:17,22
CON3-1	570,60	554,47	554,47	572,47	8,26	2,83	0,00	03:34,42	03:21,65
CON3-2	534,80	518,00	523,47	538,97	7,35	2,12	-1,06	08:10,18	08:58,10
CON3-3	656,90	591,19	595,46	611,33	14,75	9,35	-0,72	10:03,95	10:20,21
CON3-4	640,20	588,79	591,37	610,73	15,75	7,63	-0,44	09:52,37	10:12,49
CON3-5	604,70	563,70	563,70	598,77	16,42	6,78	0,00	02:14,40	04:33,77
CON3-6	521,30	499,05	502,63	517,43	14,34	3,58	-0,72	08:40,99	09:39,63
CON3-7	602,80	576,48	580,87	611,87	16,70	3,64	-0,76	19:34,81	19:14,58
CON3-8	556,20	523,05	523,94	540,03	17,76	5,80	-0,17	17:35,51	18:42,03
CON3-9	612,80	578,25	578,25	603,90	11,94	5,64	0,00	19:32,49	15:05,47



Şekil 4.24. CON3 problemleri çözümlerinde YAK sonuçlarının literatürdeki en iyi sonuçlarla karşılaştırılması.

Çizelge 4.8. CON8 Problem çözümlerinde YAK ile elde edilen çözümler ve literatürde elde edilmiş en iyi sonuçlar.

Test Problemi	Dethloff	Literatür En İyi	YAK ALGORİTMASI SONUÇLARI						
			En İyi Çözüm	Ortalama	Standart Sapma	% Oran (Dethloff'a Göre)	% Oran (Literatüre Göre)	İşlem Süresi (En İyi Çözüm İçin)	Ortalama İşlem Süresi
CON8-0	967,30	857,17	864,52	879,77	10,12	10,63	-0,86	10:41,46	11:38,27
CON8-1	828,70	740,85	745,91	757,03	6,76	9,99	-0,68	07:40,20	09:16,71
CON8-2	770,20	712,89	712,89	739,77	8,89	7,44	0,00	04:11,91	04:35,79
CON8-3	906,70	811,07	816,38	840,58	16,29	9,96	-0,65	18:29,62	17:13,93
CON8-4	876,80	772,25	774,90	791,80	7,66	11,62	-0,34	05:45,82	06:28,74
CON8-5	866,90	754,88	758,33	769,83	5,68	12,52	-0,46	06:05,37	06:11,39
CON8-6	749,10	678,92	683,21	707,87	5,41	8,80	-0,63	02:52,88	02:37,54
CON8-7	929,80	811,96	811,96	828,77	8,78	12,67	0,00	02:48,61	02:45,37
CON8-8	833,10	767,53	771,19	786,90	5,26	7,43	-0,48	04:54,96	04:46,69
CON8-9	877,30	809,00	809,00	824,70	4,88	7,79	0,00	08:20,58	07:11,44



Şekil 4.24. CON3 problemleri çözümlerinde YAK sonuçlarının literatürdeki en iyi sonuçlarla karşılaştırılması.

Sonuçların karşılaştırıldığı çizelge ve grafiklere bakarak, önerilen yöntemin, Dethloff'a oranla çok daha düşük maliyetli rotalar üretebildiği ve literatürde yakın zamana kadar elde edilmiş en başarılı sonuçların en çok %1,12'si kadar göz ardı edilebilecek seviyede gerisinde kaldığı görülebilmektedir. Süreler ve standart sapma verileri göz önüne alındığında, YAK algoritmasının makul sürelerde ve düşük standart sapma değerlerinde kararlı çözümler üretebildiği görülmektedir.

BÖLÜM 5

TARTIŞMA VE ÖNERİLER

Bu çalışmada, üzerinde farklı yaklaşımlarla yıllardır çözüm aranan, NP-zor problemler sınıfındaki ayrık yapılı kombinatoryal optimizasyon problemlerinden biri olan EDTARP için, YAK algoritması kullanılarak yeni bir çözüm yöntemi önerilmiştir. Geliştirilen algoritma, .net platformunda C# programlama dili ile kodlanmış ve bu alanda hazırlanan test problemleri üzerinde çalıştırılmıştır. Denemelerden elde edilen her bir çözüme ait tüm sonuçlar kaydedilmiş, ilgili bilgiler tablo ve grafikler halinde sunularak, önerilen yöntemin çözüm arama stratejisi ve performansı tartışılmıştır. Bu sayede, EDTARP tarzındaki problemlere çözüm arayacak sonraki çalışmalara, yapay arı koloni kontrol parametreleri için en uygun parametre değerlerinin seçimi konusunda ışık tutulmaktadır. Zira kombinatoryal optimizasyon problemleri çözümünde en karmaşık ve en çok zaman alan durum, seçilen yöntem için en uygun parametre değerlerinin belirlenmesi sürecidir.

Önerilen yöntemin, test problemlerinde elde ettiği en başarılı çözüm sonuçları, literatürde bugüne dek bulunan en başarılı sonuçlarla karşılaştırılmış, sonuç olarak; YAK algoritmasının, kısa sürelerde elde ettiği sonuçlarla, EDTARP gibi değişik dalgalanma seviyelerindeki kombinatoryal optimizasyon problemlerine uygulanabileceği ve bu alanda diğer Meta-sezgisel yöntemlerle rekabet edebilecek derecede başarılı olduğu gözlemlenmiştir. Çalışma kapsamında, klasik haliyle uygulanan YAK algoritması, az sayıdaki kontrol parametresiyle, literatürde bugüne dek kaydedilen en başarılı sonuçların en çok %1,12'si kadar gerisinde kalmıştır. Algoritma ayrıca, birbirine yakın çözümlerden oluşan arama bölgeleri için, yerel aramadaki yeteneğiyle, kararlı biçimde, başarılı sonuçlar üretebilmiş, dalgalanma seviyesi daha yüksek problem çözümlerinde ise daha kısa sürelerde başarılı sonuçlara ulaşabilmiştir.

Uygulamanın kaydettiği sonuç değerleri analiz edildiğinde, algoritmanın, yerel en iyi çözümlere takılmadan, arama uzayının farklı bölgelerine de yönelebildiği gözlemlenmiştir. Sonraki süreçte, önerilen yöntemin algoritma performansına yönelik; parametrelerin başarılarını artırmak ve algoritmanın çalışma sürelerini kısaltabilmek tarzında iyileştirmeler yapılabilir. Ayrıca uygulamaya, YAK algoritmasıyla birlikte kullanılabilecek diğer yöntemler entegre edilerek, karma çözüm sistemleri de geliştirilebilir.

Lojistik alanında gerçek yaşamda uygulanabilirliği artırabilmek adına, opsiyonel sayı ve kapasitelerde araçlar içeren esnek yapılı ARP modeli için çizelgeleme yapabilen, aynı zamanda seyahat sürecinde karşılaşılabilecek beklenmedik sorunlara dinamik çözümler üretebilen Meta-sezgisel yöntemler oluşturulabilir. Ayrıca, lojistik firmalarından elde edilebilecek verilerle "önce gruplandır, sonra rotala" yaklaşımıyla, uygulamanın gerçek hayatta kullanılabilirliği değerlendirilebilir.

KAYNAKLAR

1. Vural, E., "Araç Rotalama Problemleri İçin Popülasyon ve Komşuluk Tabanlı Metasezgisel Bir Algoritmanın Tasarımı ve Uygulaması", ***Yıldız Teknik Üniversitesi***, (2006).

2. Toth P., D. V., "The Vehicle Routing Problem", ***Society For Industrial And Applied Mathematics Philadelphia***, (2001).

3. Erbao, C., Mingyong, L., and Kai, N., "A Differential Evolution & Genetic Algorithm for Vehicle Routing Problem with Simultaneous Delivery and Pick-up and Time Windows", ***IFAC Proceedings Volumes***, 41 (2): 10576–10581 (2008).

4. Chen, J. F. and Wu, T. H., "Vehicle routing problem with simultaneous deliveries and pickups", ***Journal Of The Operational Research Society***, 57 (5): 579–587 (2006).

5. Çetin, S. and Gencer, C., "Heterojen Araç Filolu Zaman Pencereli Eş Zamanlı Dağıtım-Toplamalı Araç Rotalama Problemleri : Matematiksel Model", ***International Journal Of Research And Development***, 3 (1): 19–27 (2011).

6. Baker, B. M. and Ayechew, M. A., "A genetic algorithm for the vehicle routing problem", ***Computers & Operations Research***, 30 (5): 787–800 (2003).

7. Hezer, S. and Kara, Y., "Eşzamanli dağitimli ve toplamali araç rotalama problemlerinin çözümü için bakteriyel besin arama optimizasyonu tabanli bir algoritma", ***Journal Of The Faculty Of Engineering And Architecture Of Gazi University***, 28 (2): 373–382 (2013).

8. Montané, F. A. T. and Galvão, R. D., "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service", ***Computers And Operations Research***, 33 (3): 595–619 (2006).

9. Akay, B., "Nümerik Optimizasyon Problemlerinde Yapay Ari Kolonisi (Artificial Bee Colony) Algoritmasinin Performans Analizi", ***Erciyes Üniversitesi***, (2009).

10. Ateş, E., "Gezgin Satıcı Probleminin Çözümü Ve 3 Boyutlu Benzetimi", (2012).

11. Laporte, G., Nobert, Y., and Taillefer, S., "Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems", ***Transportation Science***, 22 (3): 161–172 (1988).

12. Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T., "A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service", ***Expert Systems With Applications***, 36 (2 PART 1): 1070–1081 (2009).

13. Düzakin, E. and Demircioğlu, M., "Araç Rotalama Problemleri ve Çözüm Yöntemleri", ***Çukurova Üniversitesi İİBF Dergisi***, 68–87 (2009).

14. Yu, B., Yang, Z. Z., and Yao, B., "An improved ant colony optimization for vehicle routing problem", ***European Journal Of Operational Research***, 196 (1): 171–176 (2009).

15. Yücenur, G. N., "Optimizasyon Problemlerinin Çözümünde Melez Metasezgisel Bir Algoritmanin Tasarimi", ***Yıldız Teknik Üniversitesi***, (2011).

16. Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F., "A guide to vehicle routing problem", ***Journal Of The Operational Research Society***, 53 (5): 512–522 (2002).

17. Eryavuz, M. and Gencer, C., "Araç Rotalama Problemine Ait Bir uygulama", ***Süleyman Demirel Üniversitesi***, 6 (1): 139–155 (2001).

18. Montemanni, R., Gambardella, L. M., Rizzoli, A. E., and Donati, A. V., "A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System", ***Istituto Dalle Molle Di Studi Sull'Intelligenza Artificiale (IDSIA)***, 02 (23): 73–90 (2002).

19. Potvin, J. Y., Xu, Y., and Benyahia, I., "Vehicle routing and scheduling with dynamic travel times", ***Computers And Operations Research***, 33 (4): 1129–1137 (2006).

20. Brandão, J., "A tabu search algorithm for the open vehicle routing problem", ***European Journal Of Operational Research***, 157 (3): 552–564 (2004).

21. Tan, K. C., Lee, L. H., Zhu, Q. L., and Ou, K., "Heuristic methods for vehicle routing problem with time windows", ***Artificial Intelligence In Engineering***, 15 (3): 281–295 (2001).

22. Mazzeo, S. and Loiseau, I., "An Ant Colony Algorithm for the Capacitated Vehicle Routing", ***Electronic Notes In Discrete Mathematics***, 18: 181–186 (2004).

23. Lacomme, P., Prins, C., and Sevaux, M., "A genetic algorithm for a bi-objective capacitated arc routing problem", ***Computers And Operations Research***, 33 (12): 3473–3493 (2006).

24. Alba, E. and Dorronsoro, B., "Computing nine new best-so-far solutions for Capacitated VRP with a cellular Genetic Algorithm", ***Information Processing Letters***, 98 (6): 225–230 (2006).

25. Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S., and Farias, R., "A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery", ***Computers And Operations Research***, 37 (11): 1899–1911 (2010).

26. Toth, P. and Vigo, D., "1. An Overview of Vehicle Routing Problems", ***The Vehicle Routing Problem***, (January): 1–26 (2002).

27. Gencer, C. and Yaşa, Ö., "Ulaştırma Komutanlığı Ring Seferlerinin Eş Zamanlı Dağıtım Toplama Karar Destek Sistemi", ***Gazi Üniv. Müh. Mim. Fak. Der.***, 22 (3): 437–449 (2007).

28. Ganesh, K. and Narendran, T. T., "CLOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up", ***European Journal Of Operational Research***, 178 (3): 699–717 (2007).

29. Nagy, G. and Salhi, S., "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries", ***European Journal Of Operational Research***, 162 (1): 126–141 (2005).

30. Gajpal, Y. and Abad, P., "An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup", ***Computers And Operations Research***, 36 (12): 3215–3223 (2009).

31. Ai, T. J. and Kachitvichyanukul, V., "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery", ***Computers And Operations Research***, 36 (5): 1693–1702 (2009).

32. Kiliç, S. and Kahraman, C., "Bulanık karar ortamında karınca kolonisi optimizasyonu yöntemiyle araç rotalama", ***Itüdergisi/D***, 8 (4): 160–172 (2009).

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

34. Christofides, N., Mingozzi, A., and Toth, P., "Contributions to the quadratic assignment problem", ***European Journal Of Operational Research***, 4 (4): 243–247 (1980).

35. Nagy, G. and Salhi, S., "A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling", ***Journal Of The Operational Research Society***, 50 (10): 1034–1042 (1999).

36. DellAmico, M., Righini, G., Salani, M., "A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection", ***Transportation Science***, 40(2) (44): 235–247 (2006).

37. Dethloff, J., "Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up", ***OR Spektrum***, 23 (1): 79–96 (2001).

38. Dantzig, G. B. and Ramser, J. H., "The Truck Dispatching Problem", ***Management Science***, 6: 80–91 (1959).

39. Çevik, D. D. O., "Tam sayili do rusal programlama le planlamasi ve b r uygulama gücü", 157–171.

40. Yan, L., Taha, H. A., and Landers, T. L., "A Recursive Approach for Enumerating Minimal Cutsets in a Network", ***IEEE Transactions On Reliability***, 43 (3): 383–388 (1994).

41. Rousseau, L. M., Gendreau, M., and Feillet, D., "Interior point stabilization for column generation", ***Operations Research Letters***, 35 (5): 660–668 (2007).

42. Archetti, C., Bianchessi, N., and Speranza, M. G., "A column generation approach for the split delivery vehicle routing problem", ***Networks***, 58 (4): 241–254 (2011).

43. Internet: Çetin, E., "Dinamik Programlama İle Sınır Tenörü Optimizasyonu", **http://www.maden.org.tr** (2019).

44. Nabiyev, V. V., "Yapay Zeka – Problemler Yöntemler Algoritmalar", ***Seçkin Yayıncılık***, Ankara, (2003).

45. Van Breedam, A., "Comparing descent heuristics and metaheuristics for the vehicle routing problem", ***Computers And Operations Research***, 28 (4): 289–315 (2001).

46. Dağ, S., "Akış Tipi Çizelgeleme Problemlerinin Sezgisel Yöntemlerle Optimizasyonu", ***İstanbul Üniversitesi***, (2012).

47. G.Nilay Yücenur, N. Ç. D., "Çok Depolu Araç RotalamProblemlerinin ÇözümİçinGenetikAlgoritma Ve KarincKolonisiOptimizasyonundan Oluşan MelezAlgoritma Tasarimi Özet", ***Journal Of Engineering And Natural Sciences***, (1997): 340–350 (2011).

48. Cordeau, J. F., Laporte, G., Savelsbergh, M. W. P., and Vigo, D., "Chapter 6 Vehicle Routing", ***Handbooks In Operations Research And Management Science***, 14 (C): 367–428 (2007).

49. Laporte, G., Gendreau, M., Potvin, J. Y., and Semet, F., "Classical and modern heuristics for the vehicle routing problem", ***International Transactions In Operational Research***, 7 (4–5): 285–300 (2000).

50. Özkan, B., Çevre, U., and Uğur, A., "Melez Bir Eniyileme Yöntemi İle Rota Planlama", ***Akademik Bilişim '08***, 46 (2008).

51. Savelsbergh, M. and Sol, M., "Drive : Dynamic Routing of Independent Vehicles", ***Operations Research***, 46 (June 2015): 474–490 (1998).

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

53. Hertz, A. and Widmer, M., "Guidelines for the use of meta-heuristics in combinatorial optimization", ***European Journal Of Operational Research***, 151 (2): 247–252 (2003).

54. Tarantilis, C. D., Ioannou, G., and Prastacos, G., "Advanced vehicle routing algorithms for complex operations management problems", ***Journal Of Food Engineering***, 70 (3): 455–471 (2005).

55. Koç, Ç. and Karaoğlan, İ., "Çok Kullanimli Ve Zaman Pencereli Araç Rotalama Problemi İçin Bir Matematiksel Model", ***Gazi Üniv. Müh. Mim. Fak. Der.***, 27 (3): 569–576 (2012).

56. Bianchessi, N. and Righini, G., "Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery", ***Computers And Operations Research***, 34 (2): 578–594 (2007).

57. Crispim, J. and Brandão, J., "Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls", ***Journal Of The Operational Research Society***, 56 (11): 1296–1302 (2005).

58. Zhang, D., Cai, S., Ye, F., Si, Y. W., and Nguyen, T. T., "A hybrid algorithm for a vehicle routing problem with realistic constraints", ***Information Sciences***, 394–395: 167–182 (2017).

59. Yousefikhoshbakht, M., Didehvar, F., and Rahmati, F., "A combination of modified tabu search and elite ant system to solve the vehicle routing problem with simultaneous pickup and delivery", ***Journal Of Industrial And Production Engineering***, 31 (2): 65–75 (2014).

60. Mancini, S., "A real-life Multi Depot Multi Period Vehicle Routing Problem with a Heterogeneous Fleet: Formulation and Adaptive Large Neighborhood Search based Matheuristic", ***Transportation Research Part C: Emerging Technologies***, 70: 100–112 (2016).

61. Salhi, S., Imran, A., and Wassan, N. A., "The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation", ***Computers And Operations Research***, 52: 315–325 (2014).

62. Kalayci, C. B. and Kaya, C., "An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery", ***Expert Systems With Applications***, 66: 163–175 (2016).

63. Avci, M. and Topaloglu, S., "An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries", ***Computers And Industrial Engineering***, 83: 15–29 (2015).

64. Rieck, J. and Zimmermann, J., "Exact Solutions to the Symmetric and Asymmetric Vehicle Routing Problem with Simultaneous Delivery and Pick-Up", ***Business Research***, 6 (1): 77–92 (2013).

65. Zhu, H., Feng, J., and Li, H., "MN _ GLS for VRP with Simultaneous Delivery and Pickup", 28 (6): 1–12 (2017).

66. Sayyah, M., Larki, H., and Yousefikhoshbakht, M., "Solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery by an Effective Ant Colony Optimization", 3 (1): 15–38.

67. Atmaca, E., "Bir Kargo Şirketinde Araç Rotalama Problemi", ***TÜBAV Bilim Dergisi***, 5 (2): 12–27 (2012).

68. Tereshko, V. and Loengarov, A., "Collective decision-making in honey bee foraging dynamics", ***Computing And Information Systems Journal***, 9 (3): 1–7 (2005).

69. Karaboga, D., "An Idea Based on Honey Bee Swarm for Numerical Optimization", (2005).

70. Ropke, S. and Pisinger, D., "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls", ***European Journal Of Operational Research***, 171 (3): 750–775 (2006).

71. Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T., "An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries", ***European Journal Of Operational Research***, 202 (2): 401–411 (2010).

72. Goksal, F. P., Karaoglan, I., and Altiparmak, F., "A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery", ***Computers And Industrial Engineering***, 65 (1): 39–53 (2013).

EK AÇIKLAMALAR A.

GELİŞTİRİLEN UYGULAMADA KULLANILAN BAZI KODLAMALAR

```
private void BaslangicCozumleriniOlustur(int BesinKaynagiSayisi, int DugumSayisi)
{
    int i, ZiyaretEdilenDugumSayisi = 0, GidilecekDugum = 0,DugumSayac=0;
    int[] ZiyaretEdilenDugumler;
    Random DugumSec = new Random();

    for (i = 0; i < BesinKaynagiSayisi; i++)
    {
        BesinKaynaklari[i, 0] = 0;
        DugumSayac = 1;
        ZiyaretEdilenDugumSayisi = 1;
        ZiyaretEdilenDugumler = new int[DugumSayisi];
        do
        {
            do
            {
                GidilecekDugum = DugumSec.Next(DugumSayisi + 1);
            } while ((GidilecekDugum != 0) && (Array.IndexOf(ZiyaretEdilenDugumler, GidilecekDugum) != -1));
            BesinKaynaklari[i, DugumSayac] = GidilecekDugum;
            DugumSayac++;
            if (GidilecekDugum > 0)
            {
                ZiyaretEdilenDugumler[ZiyaretEdilenDugumSayisi] = GidilecekDugum;
                ZiyaretEdilenDugumSayisi++;
            }
        } while (ZiyaretEdilenDugumSayisi < DugumSayisi);
    }
}
```

Şekil Ek A.1. Başlangıç çözümlerini oluşturmak için kullanılan C# kod bloğu.

```
double CozumunUygunlukDegeriniHesapla(int[] Cozum)
{
    int i;
    double ToplamMesafe = 0, UygunlukDegeri = 0;

    for (i = 0; i < Cozum.Length-1; i++)
        ToplamMesafe += Mesafeler[Cozum[i], Cozum[i + 1]];
    UygunlukDegeri = 1 / ToplamMesafe;
    return UygunlukDegeri;
}
```

Şekil Ek A.2. Çözümün uygunluk değerini hesaplamak için kullanılan C# kod bloğu.

```
int[] SiraliCaprazlama(int[]Cozum1, int[]Cozum2)
{
    int i, j,sayac=0, KesmeNoktasi1 = 0, KesmeNoktasi2 = 0;
    Random CozumNoktasiSec = new Random();
    int[] YeniCozum = new int[Cozum1.Length];
    int[] KesimNoktalariArasindakiler;
    do
    {
        KesmeNoktasi1 = CozumNoktasiSec.Next(Cozum1.Length);
        KesmeNoktasi2 = CozumNoktasiSec.Next(Cozum1.Length);
    } while (KesmeNoktasi1 >= KesmeNoktasi2);

    KesimNoktalariArasindakiler = new int[KesmeNoktasi2 - KesmeNoktasi1];
    sayac = 0;
    for (i = KesmeNoktasi1; i < KesmeNoktasi2; i++)
    {
        YeniCozum[i] = Cozum2[i];
        KesimNoktalariArasindakiler[sayac] = Cozum2[i];
        sayac++;
    }

    sayac = 0;
    for (i = 0; i < Cozum1.Length; i++)
    {
        if (Array.IndexOf(KesimNoktalariArasindakiler, Cozum1[i]) == -1)
        {
            YeniCozum[sayac] = Cozum1[i];
            sayac++;
        }
        if (sayac == KesmeNoktasi1)
            sayac = KesmeNoktasi2;
    }
    return YeniCozum;
}
```

Şekil Ek A.3. Yeni çözüm türetmede kullanılan C# kod bloğu.

```
private void IsciAriSafhasi()
{
    int i, j, RastgeleCozumNo = 0;
    Random CozumSec = new Random();
    int[] MevcutCozum = new int[500];
    int[] RastgeleCozum = new int[500];
    int[] TuretilenCozum = new int[500];
    double TuretilenCozumunUygunlukDegeri = 0;

    for (i = 0; i < BesinKaynagiSayisi; i++)
    {
        do
        {
            RastgeleCozumNo = CozumSec.Next(BesinKaynagiSayisi);
        } while (RastgeleCozumNo == i);

        for (j = 0; j < 500; j++)
        {
            MevcutCozum[j] = BesinKaynaklari[i, j];
            RastgeleCozum[j] = BesinKaynaklari[RastgeleCozumNo, j];
        }
        TuretilenCozum = SiraliCaprazlama(MevcutCozum, RastgeleCozum);
        TuretilenCozumunUygunlukDegeri = CozumunUygunlukDegeriniHesapla(TuretilenCozum);
        if (TuretilenCozumunUygunlukDegeri > UygunlukDegerleri[i])
        {
            for (j = 0; j < 500; j++)
                BesinKaynaklari[i, j] = TuretilenCozum[j];
            UygunlukDegerleri[i] = TuretilenCozumunUygunlukDegeri;
            BasarisizlikSayaci[i] = 0;
        }
        else
            BasarisizlikSayaci[i]++;
    }
}
```

Şekil Ek A.4. İşçi arı safhasındaki işlemler için kullanılan C# kod bloğu.

```
private void UygunlukDegerlerininRuletTekerlegineYerlesimi()
{
    int i, j;
    double UygunlukDegerleriToplami = 0;

    for (i = 0; i < BesinKaynagiSayisi; i++)
        UygunlukDegerleriToplami += UygunlukDegerleri[i];

    for (i = 0; i < BesinKaynagiSayisi - 1; i++)
        RuletTekeri[i] = UygunlukDegerleri[i] * 1000000 / UygunlukDegerleriToplami;
    RuletTekeri[BesinKaynagiSayisi - 1] = 1000000;
}
```

Şekil Ek A.5. Uygunluk değerlerinin rulet tekerleğine yerleşimi için gereken C# kod bloğu.

```
private void GozcuAriSafhasi()
{
    int i, j, TemelCozumNo = 0, YardimciCozumNo = 0;
    Random CozumSec = new Random();
    int[] TemelCozum = new int[500];
    int[] YardimciCozum = new int[500];
    int[] TuretilenCozum = new int[500];
    double TuretilenCozumunUygunlukDegeri = 0;

    for (i = 0; i < BesinKaynagiSayisi; i++)
    {
        do
        {
            TemelCozumNo = CozumSec.Next(1000000);
            for (j = 0; j < BesinKaynagiSayisi; j++)
                if (TemelCozumNo < RuletTekeri[i])
                {
                    TemelCozumNo = j;
                    break;
                }
            YardimciCozumNo = CozumSec.Next(1000000);
            for (j = 0; j < BesinKaynagiSayisi; j++)
                if (YardimciCozumNo < RuletTekeri[i])
                {
                    TemelCozumNo = j;
                    break;
                }
        } while (TemelCozumNo == YardimciCozumNo);

        for (j = 0; j < 500; j++)
        {
            TemelCozum[j] = BesinKaynaklari[TemelCozumNo, j];
            YardimciCozum[j] = BesinKaynaklari[YardimciCozumNo, j];
        }

        TuretilenCozum = SiraliCaprazlama(TemelCozum, YardimciCozum);
        TuretilenCozumunUygunlukDegeri = CozumunUygunlukDegeriniHesapla(TuretilenCozum);
        if (TuretilenCozumunUygunlukDegeri > UygunlukDegerleri[TemelCozumNo])
        {
            for (j = 0; j < 500; j++)
                BesinKaynaklari[TemelCozumNo, j] = TuretilenCozum[j];
            UygunlukDegerleri[TemelCozumNo] = TuretilenCozumunUygunlukDegeri;
            BasarisizlikSayaci[TemelCozumNo] = 0;
        }
        else
            BasarisizlikSayaci[TemelCozumNo]++;
    }
}
```

Şekil Ek A.6. Gözcü arı safhasındaki işlemlerde kullanılan prosedür.

ÖZGEÇMİŞ

Dursun EKMEKCİ Karabük'te doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı ve Karabük Mustafa Yazıcı Lisesi Yabancı Dil Ağırlıklı Bölümü'nden mezun oldu. 2000 yılında Sakarya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde başladığı üniversite öğreniminden 2004 yılında mezun oldu. Mezuniyeti sonrası vatani görevini yaptığı Balıkesir Astsubay MYO'da astsubay olarak terhis oldu. 2006 yılında Maden Tetkik ve Arama Genel Müdürlüğü'nde mühendis olarak göreve başladı. Aynı yıl Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar ve Bilişim Mühendisliği Anabilim Dalı'nda başlamış olduğu yüksek lisans programını 2008 yılında tamamladı. 2009 yılında Karabük Üniversitesi Türkiye Odalar ve Borsalar Birliği Meslek Yüksek Okulu (KBU TOBB MYO)'nda Öğretim Görevlisi olarak göreve başlamış, uzunca bir dönem, aynı üniversitenin Uzaktan Eğitim Uygulama ve Araştırma Merkezi'nde görevlendirilmiştir. Görevine halen KBU TOBB MYO'da devam etmektedir.

ADRES BİLGİLERİ

Adres : Karabük Üniversitesi
TOBB MYO
Bilgisayar Programcılığı Bölümü
Balıklarkayası Mevkii / KARABÜK

Tel : (505) 846 0434

E-posta : dekmekci@karabuk.edu.tr