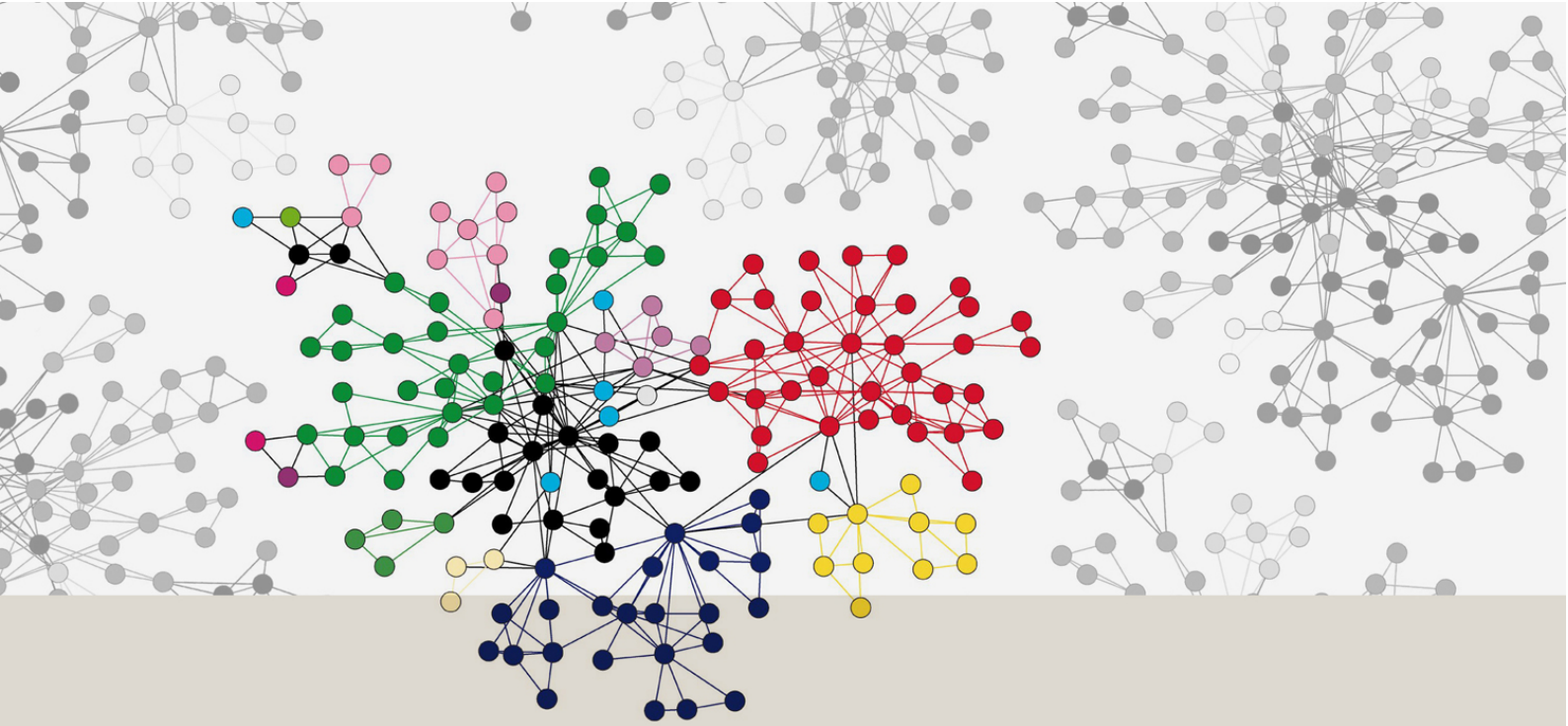


ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMİNİN GELİŞTİRİLMİŞ YAPAY ARI KOLONİSİ ve ATEŞ BÖCEĞİ ALGORİTMALARI ile ÇÖZÜMÜ

DR. NAZİFE ŞAHİN MACİT
DOÇ. DR. YUSUF ŞAHİN

ÇİZGİ KİTABEVİ | e-Kitap

ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMİNİN GELİŞTİRİLMİŞ YAPAY ARI KOLONİSİ ve ATEŞ BÖCEĞİ ALGORİTMALARI İLE ÇÖZÜMÜ

Dr. NAZİFE ŞAHİN MACİT
Doç. Dr. YUSUF ŞAHİN

ÇİZGİ KİTABEVİ

Çizgi Kitabevi Yayınları (e-kitap)

Ekim 2022

ISBN: 978-605-196-878-0
Yayıncı Sertifika No:52493

KÜTÜPHANE BİLGİ KARTI
- Cataloging in Publication Data (CIP) -
ŞAHİN MACİT, Nazife
ŞAHİN, Yusuf
ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMİNİN GELİŞTİRİLMİŞ
YAPAY ARI KOLONİSİ VE ATEŞ BÖCEĞİ ALGORİTMALARI İLE ÇÖZÜMÜ

Yayına Hazırlık: Çizgi Kitabevi Yayınları
Tel: 0332 353 62 65- 66

ÇİZGİ KİTABEVİ

Sahibiata Mah.	Alemdar Mah.
M. Muzaffer Cad. No:41/1	Çatalçeşme Sk. No:42/2
Meram/**Konya**	Cağaloğlu/**İstanbul**
(0332) 353 62 65 - 66	(0212) 514 82 93

www.cizgikitabevi.com
/ cizgikitabevi

İÇİNDEKİLER

ÜÇÜNCÜ BÖLÜM

DÖRDÜNCÜ BÖLÜM

TEŞEKKÜR

Çalışmanın her aşamasında bilgi ve desteğini esirgemeyen, önerileri ile bana yol gösteren çok değerli hocam ve danışmanım Doç. Dr. Yusuf ŞAHİN'e en derin teşekkürlerimi sunarım. Yorum ve önerileri ile çalışmaya katkı sağlayan Sayın Doç. Dr. Muhammet Burak KILIÇ, Doç. Dr. Erdal AYDEMİR, Doç. Dr. Kenan KARAGÜL, Doç. Dr. Kenan Oğuzhan ORUÇ ve Doç. Dr. Emre ÇOMAK hocalarıma çok teşekkür eder ve saygılarımı sunarım.

Dr. Öğr. Üyesi Mustafa Bilgehan İMAMOĞLU, Öğr. Gör. Hasan ŞEN, Öğr. Gör. Gökhan ÇAYBAŞI ve Dr. Öğr. Üyesi Oğuz KÖSE hocalarıma desteklerinden dolayı ayrıca teşekkür ederim. Yorucu ve uzun süren çalışmalarım boyunca desteğini esirgemeyen ve özellikle hayatım boyunca üzerimde çok emeği olan, haklarını asla ödeyemeyeceğim babam İbrahim ŞAHİN, annem Ümmü ŞAHİN ve canım kardeşim Derya ŞAHİN'e maddi ve manevî desteklerinden dolayı sonsuz teşekkürlerimi sunarım.

Son olarak, onlara ayırmam gereken zamandan çalarak çalışmaya yoğunlaştığım son zamanlarda, göstermiş oldukları hoşgörü ve eşsiz destekleri için biricik oğlum Yiğit MACİT ve canım eşim Hasan Hüseyin MACİT'e çok teşekkür ederim. İyi ki varsınız iyi ki benim ailemsiniz.

Dr. Nazife ŞAHİN MACİT

KISALTMALAR DİZİNİ

$\boldsymbol{F_{best}}$: En iyi uygunluk değeri
$\boldsymbol{f_i}$: i. yiyecek kaynağının uygunluk değeri
$\boldsymbol{l_i}$: i. ateş böceğinin ışık yoğunluğu
ABA: Ateş Böceği Algoritması
ARP: Araç Rotalama Problemi
BTARP: Bölünebilir Talebli Araç Rotalama Problemi
BYA: Bakteriyel Yiyecek Arama
ÇDARP: Çok Depolu Araç Rotalama Problemi
ÇDTDARP: Çok Depolu Topla Dağıt Araç Rotalama Problemi
DARP: Dinamik Araç Rotalama Problemi
EA: Evrimsel Algoritma
EYK: En Yakın Komşu
EZTDARP: Eş Zamanlı Topla Dağıt Araç Rotalama Problemi
GA: Genetik Algoritma
GKA: Guguk Kuşu Arama
HAA: Harmoni Arama Algoritması
HFARP: Heterojen Filolu Araç Rotalama Problemi
KK: Karınca Kolonisi
KKARP: Kapasite Kısıtlı Araç Rotalama Problemi
KSA: Kurbağa Sıçrama Algoritması
LP: Lineer Programlama
MA: Memetik Algoritma
MKARP: Mesafe Kısıtlı Araç Rotalama Problemi
ÖDSTARP: Önce Dağıt Sonra Topla Araç Rotalama Problemi
PARP: Periyodik Araç Rotalama Problemi
PSO: Parçacık Sürü Optimizasyonu
SARP: Statik Araç Rotalama Problemi
TA: Tabu Arama
TB: Tavlama Benzetimi
WOA: Balina Optimizasyon Algoritması
YA: Yarasa Algoritması
YAA: Yerel Arama Algoritması
YAKA: Yapay Arı Kolonisi Algoritması
YK: Yiyecek Kaynağı
YUA: Yusufçuk Algoritması
ZPARP: Zaman Pencereli Araç Rotalama Problemi
$\boldsymbol{NV}$: Araç Sayısı
$\boldsymbol{md}$: mesafe değeri
$\boldsymbol{sd}$: süre değeri

GİRİŞ

Küreselleşme bugünün iş ortamında birçok zorlukları beraberinde getirmiştir. Endüstri Devrimi'nden bu yana, pazar hızlı bir şekilde genişlemiş ve rekabetçi pazarda ayakta kalabilmek için şirketlerin daha büyük rekabet avantajı kazanması gerekmiştir. Ayrıca, internet dönemi, çevrimiçi rekabet piyasasına yol açan tüketicilere çevrimiçi alışveriş getirmiştir. Sonuç olarak, rekabet avantajı elde etmek için şirketler, müşterilerini tutmalarını ve yenilerini çekmelerini sağlayacak daha iyi bir müşteri memnuniyet oranı almaya yönelmişlerdir (Petelina, 2016: 2). Günümüz iş ortamının giderek daha rekabetçi hale gelmesi birçok sektördeki birçok şirket için büyük bir baskı yaratmıştır. Böyle bir ortamda, şirketlerin yeni ürünler tasarlamanın ve üretmenin yollarını sürekli araştırması ve bu ürünleri verimli ve etkili bir şekilde dağıtması gerekmiştir. Uzun yıllar boyunca şirketler, imalat işlemlerinde ve diğer işlemlerde ortaya çıkan maliyetleri azaltma çabalarına odaklanmışlardır. Dağıtımı inceleyen ve maliyet azaltmada onu son sınır olarak kabul eden çok sayıda şirket vardır.

Lojistik bir sistemde, dağıtım maliyeti genellikle en yüksek tek giderdir ve bu genellikle depolama, stok ve sipariş işleme gibi maliyetlerden daha yüksektir. Dağıtım, hızlı ücret ve navlun enflasyonu, nakliye maliyetlerinde önemli bir artış ve düzenleme, envanter taşıma maliyetinin yüksek olması ve petrol piyasası belirsizlikleri nedeniyle yönetimin dikkatini çekmiştir. Tedarik, üretim, dağıtım, depolama, envanter ve bilgi sistemleri önemli lojistik fonksiyonlardır. Bunlar arasında dağıtım tüm lojistik sistemde önemli bir fonksiyondur ve tedarik zincirindeki üreticiler ve müşteriler arasındaki anahtar bağlantıdır. Ayrıca, dağıtım bir şirkette kârlılığın en büyük itici gücüdür, çünkü hem lojistik maliyeti hem de müşteri deneyimi üzerinde doğrudan bir etkiye sahiptir. Dolayısıyla, şirketler genel lojistik ve tedarik zinciri maliyetlerini azaltma hedefine ulaşmak, dağıtım maliyetlerini düşürmek için çeşitli yaklaşımlar benimsemiştir. Ürün özellikleri, kalite ve fiyat müşteriler için önemli faktörler olmasına rağmen, lojistik ve tedarik zinciri performansı bir şirketin başarısının anahtarıdır.

Bir dağıtım ağının iyi bir tasarımı ile düşük işletme maliyetinden yüksek müşteri hizmeti seviyesine kadar değişen bir dizi lojistik ve tedarik zinciri hedefine ulaşılmaktadır. Günümüz rekabetçi iş dünyasında, maliyet, kalite, verimlilik ve müşteri hizmetleri seviyesinin boyutları artık bir şirket için değişmez. Aynı anda dikkate alınmaları gerekir. Bu hedeflere ulaşmak için, tüm dağıtım ağını en uygun şekilde yeniden tasarlamak kritik öneme sahiptir ve çoğu zaman gereklidir (Yang, 2013: 1). Lojistik sisteminin temel işlevi olan dağıtım, doğrudan son müşteriler ile bağlantılıdır. Dağıtım fonksiyonunun performansı ve hizmet seviyesi, şirketlerin lojistik maliyetini ve müşterinin tüm lojistik hizmet için memnuniyet seviyesini doğrudan etkilemektedir. Dağıtımın ana parçası, dağıtım araçlarının sınıflandırılması ve dağıtımı gerçekleştirmesi ve malların takviye edilmesi sürecidir. Bunlar arasında, araçlara yönelik dağıtım rotalarının makul bir şekilde optimizasyonu, tüm lojistik taşımacılığın hızı, maliyeti ve verimliliği açısından son derece önem arz etmektedir. Ulaşım maliyetinin, lojistik merkezin maliyetinin çoğunu içerdiğinden ve şehir içi ulaşımın son yıllarda kalabalıklaşmasından dolayı, dağıtım araçlarının güzergâhını optimize etmenin önemi desteklenmektedir. Lojistik dağıtım merkezleri, nakliye maliyetini etkin bir şekilde azaltmayı amaçladığından dağıtım araçlarının rotası ve program planlaması çok önemli operasyon kararlarıdır ve bu tür kararlar araç rotalama problemi olarak adlandırılmaktadır (Ding ve Zou, 2016: 956).

Mal ve hizmetlerin verimli bir şekilde dağıtılması günlük faaliyetlerin merkezinde yer almaktadır. Bu dağıtım sorunları genellikle araçların rotalanmasını içerir ve bir okul otobüsü, bir yerleşim mahallesinin sokaklarında dolaşan çöp kamyonu, ayda bir kez müşterisinin sayaçlarını okuyan bir elektrik şirketi ya da bir mahalledeki tüm evleri ziyaret eden postacı gibi faaliyetler dağıtım problemi olarak ele alınabilir. Bu faaliyetlere katılan araçların toplam maliyeti çok büyük olabilir ve bu araçları mümkün olduğunca verimli bir şekilde yönlendirmek önemlidir. Ancak, bu tür sorunlara etkin çözümler geliştirmek, son derece zor olduğu kanıtlanan bir birleşimsel optimizasyon sorununu çözmemizi gerektirir. Bu tür sorunlar, ilk kez Dantzig ve Ramser tarafından 1959'da tanıtılmış ve son elli yılda yüzlerce çalışma yayınlanmıştır.

Klasik ARP'de, homojen bir araç filosu ve her müşterinin bilinen bir talebi olduğu bir dizi müşteri yeri verilmiştir. Görev, tüm müşterilerin talebini karşılayacak minimum maliyette bir rota seti oluşturmak ve ayrıca bireysel rotaların araç kapasitesine ve rota uzunluğu kısıtlamalarına uymasını sağlamaktır. ARP'nin ifade edilmesi basit ve anlaşılması kolay olmasına rağmen pratikte çözülmesi çok zor olan bir problemdir (Groer, 2008:1). Problemde, bir dizi kısıtlama göz önüne alındığında, bir dizi müşteriye hizmet vermek üzere bir araç filosuna en uygun rotanın tasarlanması amaçlanmıştır. ARP'ler çok sayıda çeşidi olan problem türüdür. Bunlar, gereken hizmetin kalitesine, taşınan malların, müşterilerin ve araçların özelliklerine göre formüle edilir (Kumar ve Panneerselvam, 2012: 66).

ARP'nin bir uzantısı olan ZPARP, NP-zor kombinatoryal optimizasyon problemi sınıfına ait olduğundan bu problemlerin makul bir süre içerisinde kesin çözümü çok zordur. Bu nedenle bu tarz problemlerin çözümünde genellikle sezgisel yöntemler tercih edilmektedir. Uygun bir çözüme ulaşmanın yeterli olmadığı, ancak çözüm kalitesinin kritik olduğu durumlarda, pratik olarak kabul edilen süre içinde mümkün olan en iyi çözümleri elde etmek için etkili prosedürlerin araştırılması önem kazanmaktadır. Genellikle gerçek hayat verilerinin karmaşıklığı ile birlikte müşteri sayısı, sorunun tam olarak çözülmesine izin vermez. Bu durumlarda sezgisel ve metasezgisel yöntemler kullanılabilir. Bu yöntemler en iyi çözümü garanti etmezler. Yaklaşım algoritmaları için en kötü durum sapması bilinmesine rağmen sezgiseller için bilinen bir şey yoktur. Ancak tipik olarak çok iyi performans gösterecek şekilde ayarlanabilirler. Günümüzde ZPARP'nin çözümünde benzetimli tavlama, tabu arama, genetik algoritma gibi meta-sezgisel yöntemler başarılı bir şekilde kullanılmaktadır (El-Sherbeny, 2010: 123-124).

ZPARP'de büyük veri setlerinin kesin yöntemlerle çözülmesinin çok uzun zaman alması çoğu araştırmacıları sezgisel ve meta-sezgisel yöntemlere yöneltmiştir. Probleme daha kısa sürede ve en uygun çözümün bulunmasında daha etkili olan bu metotlar literatürde oldukça önemli bir yere sahip olmuştur ve olmaya da devam etmektedir. Bu çalışma kapsamında, problemin kısıtlamalarından olan araç kapasitelerini, seyahat süresi kısıtlamalarını ve müşteriler tarafından belirlenen zaman penceresini kısıtlamalarını ihlal etmeden tüm müşterilere minimum maliyetle (seyahat mesafesi vb.) hizmet etmeleri için araçlara uygun rotaları bulmak amacıyla yapay arı kolonisi algoritması ve ateş böceği algoritması önerilmiştir.

Çalışmanın birinci bölümünde lojistik ve lojistik yönetimi konuları ele alınmaktadır. Lojistiğin tanımı, tarihsel gelişimi, lojistik yönetiminin tanımı ve önemi gibi konulara bu bölüm kapsamında kısaca değinilmiştir. İkinci bölümde ARP ile ilgili kavramlar incelenmiştir. ARP'nin tanımı, tarihsel gelişimi ve çeşitleri, KKARP'nin tanımı ve matematiksel modeli, ZPARP'nin tanımı ve matematiksel modeli, ZPARP'nin çözümünde kullanılan yöntemler ile ilgili literatür araştırması bu kısımda ele alınmıştır. Çalışmanın üçüncü bölümünde çalışma kapsamında kullanılan meta-sezgisel yöntemler olan yapay arı kolonisi, ateş böceği algoritmaları açıklanmış ve belirtilen probleme bu yöntemlerin uyarlanması anlatılmıştır. Dördüncü bölümde yapılan deneysel çalışmalar ile geliştirilen yöntemlerin karşılaştırmaları sunulmuştur. Son bölümde ise yapılan çalışma özetlenmiş ve çalışma ile ilgili sonuç ve önerilere yer verilmiştir.

BİRİNCİ BÖLÜM

LOJİSTİK VE LOJİSTİK YÖNETİMİ

Bugünün global pazarlarında artan rekabet, ömrü kısa ürünlerin pazara sunulması ve müşteri beklentilerinin yüksek olması, üretim işletmelerini yatırım yapmaya ve lojistik sistemlerindeki stratejilerini geliştirmeye yöneltmiştir. Mobil iletişim ve bir gecede teslimat gibi iletişim ve ulaşım teknolojilerindeki değişiklikler, lojistik sistemlerin yönetiminin gelişimine katkı sağlamıştır. Bu sistemlerde, ürünlerin fabrikalarda üretimi gerçekleştikten sonra muhafaza edilmesi için depolara, daha sonra aracı kuruluşlara veya nihai müşterilere sevki gerçekleştirilir. Bu sevk süresince maliyeti minimum ve hizmet kalitesini yüksek düzeyde tutmak için lojistik stratejileri lojistik ağındaki çeşitli seviyelerin etkileşimlerini gözönünde bulundurmalıdır. Dağıtım ağı; ürün ya da hizmetlerin tedarikçilerden müşterilere doğru akışını kapsayan ve bu süreç içerisindeki üretim merkezleri, depolar, dağıtım merkezleri ve perakendeci satış noktalarının yanı sıra, hammadde, süreç içi envanter ve tesisler bütününden oluşmaktadır (Bramel ve Simchi-Levi, 1997: 1).

Kuruluşlara değer yaratmanın ve müşterilere teslim etmenin yeni yollarını bulmaya yönelik baskıların giderek daha da güçlenir hale gelmesi, geleneksel olarak kullanılanlardan daha verimli lojistik sistemleri geliştirme ihtiyacını doğurmuştur. Ayrıca etkin bir lojistik yönetimi sayesinde firmaların maliyet azaltma ve hizmet iyileştirme gibi hedeflerini önemli ölçüde gerçekleştireceğine dair artan bir görüşün olması lojistik yönetimine olan ilgiyi de arttırmıştır (Rego ve Alidaee, 2005: 3). Giderek daha da artan öneme sahip olan lojistik ve lojistik yönetimi ile ilgili kavramlara çalışmanın bu bölümünde yer verilmiştir.

Lojistiğin Tanımı

Lojistik kavramı, iş ortamındaki değişiklikler gibi faktörlere cevap olarak zaman içinde gelişmiştir. Grunnet (BTRE 2001), 1950'lerde envanterin, 1960'lı yıllarda dağıtımın, 1970'lerde üretimin, 1980'lerde satın alma / üretim / satışın ve 1990'lardaki iş sürecinin belirtilen yılların dikkat çeken terimleri olduğunu ileri sürmüştür. Lambert ve arkadaşları 1998 yılında "herhangi bir organizasyondaki lojistiğin temel amacının kurumun müşteriye olan hizmet hedeflerini etkin ve verimli bir şekilde desteklemektir." olduğunu belirtmiştir.

Lambert ve Stock (1993) lojistik yönetimini, "Müşteri gereksinimlerini dikkate alarak, hammaddelerin, mamul malların, süreç içi envanterin ve amaca yönelik bilgilerin üretim noktasından tüketim noktasına uygun maliyetli akışını ve depolanmasını planlama, uygulama ve kontrol etme sürecidir." şeklinde tanımlamıştır. Lojistik yönetiminin ana vurgusu verimliliktir. Broeke vd. (1989), hammaddelerin alımından bitmiş ürünlerin müşteriye teslimine kadar satın alma, taşıma ve depolama faaliyetlerinin organizasyonu, planlaması, uygulaması ve kontrolünün lojistik olduğunu belirtmiştir (Tseng, 2004: 14-15).

Lojistik, genel olarak malların bir yerden diğerine hareketi olarak bilinmektedir. Daskin (1985), lojistiği "malların zaman ve mekânın üstesinden gelmesine izin vermek için gerekli olan fiziksel, yönetsel ve bilgi sistemlerinin tasarımı ve çalışması" olarak tanımlamıştır. Lojistik Yönetimi Konseyi tarafından ilan edilen bir diğer tanım ise:

"Müşterilerin ihtiyaçları doğrultusunda hammaddelerin, mamul malların, süreç içi envanterin ve ilgili bilgilerin çıkış noktasından tüketim noktasına kadar etkin, uygun maliyetli taşınmasını ve depolanmasını planlama, uygulama ve kontrol etme sürecine lojistik denir." şeklindedir (Chiu, 1995: 5; Elzarka, 2010: 50).

Lojistiğin anlamı, önemi ve içeriği ve hatta onu neyin oluşturması gerektiği hakkında çok şey yazılmış, ancak bu konudaki tartışmalar henüz tamamlanmamıştır. Çalışmaya değer bir disiplin olarak, lojistik nispeten geç gelişmiştir ve bu nedenle de nispeten genç bir bilimsel disiplindir. Bu yüzden bugün hala lojistik tanımı konusunda bazı şüpheler söz konusudur. Elbette, aynı kelimelerin aynı sırada kullanılması gerektiği anlamında tek bir tanım beklenilmesi söz konusu değildir. Tekdüze tanım, öncelikle lojistik unsurları ile alt bölümleri arasındaki ilişkiyi açıklığa kavuşturmak yerine, bir disiplin olarak lojistik çerçevesini verecektir. Günümüzde endüstri ve ticaret başta olmak üzere birçok sektörde lojistik sektörünün önemi artmaktadır. Lojistik, tüm malzemenin entegre yönetimini ve tedarik malzemelerinin nihai tüketiciye kadar olan geçişi sırasında tedarikçilerin ilgili bilgi akışını ele alan bir bilim olarak kabul edilir. Birleştirilmiş lojistik tanımı olmasa da yazarların çoğu bu tanımın olabileceği konusunda hemfikirdir. Farklı ülkelerden ve kıtalardan gelen konular üretim ve iş dünyasıyla bütünleştiğinde, özellikle çağdaş küresel ortamda, malzeme ve bilgi akışının önemi ve hacmi artar. Malzeme ve bilgi akışını başarılı bir şekilde yönetmek için, hacmi ve yapısı ile ilgili iyi bir genel bakışa sahip olmak gerekir (Kukovic vd., 2014: 111-112).

Son tanımların ortak noktası olarak lojistik, müşterileri memnun etmek ve rekabet kazandırmak için, üretim, satış süreci ve atık bertarafının başından sonuna kadar malzeme ve malların taşınması ve işlenmesi sürecidir. Tilanus'a göre lojistik, müşteri ihtiyaçlarını ve isteklerini önceden belirlemek; bu ihtiyaçları ve istekleri yerine getirmek için gerekli olan sermayeye, materyallere, insanlara, teknolojilere ve bilgilere sahip olma; müşteri taleplerini yerine getirmek için mal veya hizmet üreten ağı optimize etme ve müşteri taleplerinin zamanında karşılamak için ağdan yararlanma sürecidir. Basitçe söylemek gerekirse lojistik, müşteri odaklı operasyon yönetimidir (Tseng vd., 2005: 1658).

Lojistiğin Tarihsel Gelişimi

Sermaye çalışması "Lojistik Sistemleri", tüm insan faaliyetlerini sentezlemiş ve her birinin ayrı bir lojistik alt sistem olduğunu göstermiştir. Ayrıca, "lojistik" kelimesinin kaynağı, onun doğuşunun, on sekizinci yüzyıldaki buhar motorunun icadı ile zamansal olarak ilişkili olduğu görüşüne dayanmaktadır. Bu nedenle, lojistik kavramının doğuşu, sanayi devrimi ile doğrudan ilişkilidir, ancak üretim ve ticaret olgusu daha eski zamanlara dayanır. Şu anda yaygın olan görüş, lojistik kavramının ilk kez İsviçre General Baron de Jomini (1779 −1869) tarafından kullanıldığıdır. Lojistik kelimesinin her ikisi de Fransız kökenli olmak üzere iki tip versiyonu vardır. İlk "logistique", "Marechal de Logis" askeri rütbesinden türetilmiştir ve askeri destek birliklerinin örgütlenmesi anlamına gelir. Diğeri mekânsal bir askeri organizasyon anlamına gelen "loger" dir. Ondokuzuncu yüzyılın sonunda, lojistik terimi Amerika Birleşik Devletleri'ne gelmiş ve askeri literatür, askeri destek hizmetleri bilimi, yani birliklere ulaşım ve tedarik anlamında "lojistik" terimini benimsemiştir. İkinci Dünya Savaşı'nda, "lojistik" terimi, müttefik birliklerin donatımında ve tedarik edilmesinde, planlama ve yönetim süreciyle ilişkili olarak kullanılmıştır (Tepic vd., 2011: 379).

Dağıtım ve lojistik unsurları, elbette, mal ve ürünlerin imalatı, depolanması ve taşınması için her zaman temel olmuştur. Bununla birlikte, dağıtım ve lojistiğin iş ve ekonomik çevrede hayati işlevler

olarak kabul edilmesi ancak nispeten yakın zamanda olmuştur. Lojistiğin rolü değişti, çünkü artık birçok farklı operasyon ve organizasyonun başarısında önemli bir rol oynuyor. Özünde, lojistik için temel kavramlar ve mantık yeni değildir. Dağıtım ve lojistiğin gelişiminde birkaç farklı aşama olmuştur.

1950'lerden önce lojistik keşfedilmemişti. 1950'li yıllar ve 1960'lı yılların başında dağıtım sistemleri plansız ve formüle edilmemiş durumdaydı. Üreticiler üretmiş, perakendeciler perakende satışı yapmış ve bir şekilde üretilen ürünler mağazalara ulaşmıştır. Dağıtım, genel olarak taşımacılık endüstrisi ve üreticilerin kendi hesap filoları tarafından temsil edilmiştir. Çok az pozitif kontrol vardı ve dağıtımla ilgili çeşitli fonksiyonlar arasında gerçek bir bağlantı yoktu (Tseng vd., 2005: 1660; Rushton vd., 2010: 5-6).

1960'lı yıllar ve 1970'li yılların başında 'karanlık kıta'nın gerçekten de yönetimsel katılım için geçerli bir alan olduğunun kademeli olarak anlaşılmasıyla fiziksel dağıtım kavramı geliştirilmiştir. Bu, nakliye, depolama, malzeme taşıma ve paketleme gibi birbirine bağlanabilecek ve daha etkin bir şekilde yönetilebilecek bir dizi birbiriyle ilişkili fiziksel aktivitenin olduğunun kabul edilmesiyle gerçekleştirilmiştir. Özellikle, bir sistem yaklaşımı ve toplam maliyet perspektifinin kullanılmasını sağlayan çeşitli fonksiyonlar arasında bir ilişkinin olduğu kabul edilmiştir. Bir fiziksel dağıtım yöneticisinin gözetimi altında, hem gelişmiş hizmeti hem de düşük maliyeti sağlamak için bir dizi dağıtım zorluğu planlanabilir ve yönetilebilir olmuştur. Başlangıç faydalar, ürünlerinin tedarik zinciri boyunca akışını yansıtmak için dağıtım operasyonlarını geliştiren üreticiler tarafından fark edilmiştir (Rushton vd., 2010: 6).

1970'li yıllar, dağıtım kavramının geliştirilmesinde önemli bir on yıl olmuştur. Büyük değişikliklerden birisi de, bazı şirketlerin, bir organizasyonun fonksiyonel yönetim yapısına dağıtımı dâhil etme ihtiyacını tespit etmesi olmuştur. On yılda dağıtım zincirinin yapısı ve kontrolünde değişiklik görülmüştür. Üreticilerin ve tedarikçilerin gücünde bir düşüş yaşanmış ve büyük perakendecilerde inanılmaz derecede artış meydana gelmiştir. Daha büyük perakende zincirleri, başlangıçta mağazalarını tedarik etmek için bölgesel veya yerel dağıtım depolarına dayanan kendi dağıtım yapılarını geliştirmiştir. 1970'lerden itibaren, giderek daha fazla lojistik uygulaması ve araştırması ortaya çıkmıştır. 1973 yılında petrol fiyatlarındaki artış nedeniyle lojistik faaliyetlerin işletmeler üzerindeki etkileri artmıştır. Pazarın yavaş büyümesi, ulaşım kontrolünün serbest bırakılması ve üçüncü dünyanın ürün ve malzemeler üzerindeki rekabetleri, o zamanlar lojistik sistemin planlama ve iş üzerindeki önemini artırmıştır (Rushton vd., 2010: 6; Tseng vd., 2005: 1660).

1980'li yıllarda oldukça hızlı maliyet artışları ve gerçek dağıtım maliyetlerinin daha net tanımlanması, dağıtımdaki profesyonellikte önemli bir artışa katkıda bulunmuştur. Bu profesyonellik ile uzun vadeli planlamaya doğru bir adım atılmış ve maliyet tasarruf yöntemlerini belirleme ve takip etme girişimleri olmuştur. Bu önlemler arasında merkezi dağıtım, stok tutmada ciddi düşüşler ve gelişmiş bilgi ve kontrol sağlamak için bilgisayarın kullanımı yer almıştır. Üçüncü taraf dağıtım hizmeti endüstrisinin büyümesi de büyük önem taşımış; bu şirketler bilgi ve donanım teknolojisindeki gelişmelere öncülük etmişlerdir. Şirketler, organizasyonlarının içindeki ve dışındaki fiziksel akışı yönetmek ve koordine etmek için birbirleriyle anlaşmaya başladılar. Entegre lojistik sistemleri kavramı ve ihtiyacı, dağıtım faaliyetlerine katılan ileriye dönük şirketler tarafından tanınmıştır (Rushton vd., 2010: 6-7; Ezzat vd., 2019:2) .

1980'lerin sonlarında ve 1990'ların başını kapsayan dönemde, bilgi teknolojisindeki ilerlemelere bağlı olarak, örgütler bütünleştirilebilecek işlevler açısından bakış açılarını genişletmeye başlamıştır. Kısacası bu, malzeme yönetiminin (gelen taraf) fiziksel dağıtımla (giden taraf) birleştirilmesini kapsamıştır. Bu kavramı tanımlamak için "lojistik" terimi kullanılmıştır. Bu bir kez daha, müşteri hizmetlerini iyileştirmek ve ilgili maliyetleri azaltmak için ek fırsatlara yol açmıştır. Bu dönemde yapılan en önemli vurgu, etkili bir lojistik stratejisinin sağlanmasında bilgi yönlerinin fiziksel yönler kadar önemli olduğu olmuştur.

1990'larda, süreç sadece bir örgütün kendi sınırları içindeki temel işlevleri değil, aynı zamanda bir ürünün nihai bir müşteriye sağlanmasına katkıda bulunan dışındaki işlevleri de kapsayacak şekilde geliştirilmiştir. Bu tedarik zinciri yönetimi olarak bilinmektedir. Tedarik zinciri kavramı, pazara ürün elde etmekle uğraşan birkaç farklı kuruluş olabileceğine güven vermiştir. Bu nedenle, örneğin, üreticiler ve perakendeciler, doğru ürünlerin verimli ve etkili bir şekilde nihai müşteriye akmasını sağlayan bir lojistik boru hattı oluşturulmasına yardımcı olmak için ortaklıkla birlikte hareket etmiştir.

2000 ve sonraki yıllarda, iş örgütleri rakiplerine karşı pozisyonlarını korumak veya geliştirmek için çaba sarf ettikleri, yeni ürünleri piyasaya sürdüğü ve faaliyetlerinin kârlılığını arttıracağı için birçok zorlukla karşı karşıya kalmıştır. Bu, özellikle iş hedeflerinin yeniden tanımlanmasında ve tüm sistemlerin yeniden mühendisliğinde tanınan iyileştirme için pek çok yeni fikir geliştirilmesine yol açmıştır. Lojistik ve tedarik zinciri nihayet genel iş başarısının anahtarı olan bir alan olarak kabul edilmiştir. Gerçekten, birçok kuruluş için lojistikteki değişiklikler, işlerinde büyük iyileştirmeler için katalizör sağlamıştır. Lider kuruluşlar, lojistikteki çeşitli işlevlerin yalnızca başka herhangi bir etkiden bağımsız olarak en aza indirilmesi gereken bir maliyet yükü olduğu geleneksel görüşten ziyade, lojistiğin sunabileceği olumlu bir "katma değer" rolünün olduğunu fark etmiştir. Bu nedenle, lojistiğin rolü ve önemi, iş geliştirme için önemli bir yardımcı olarak kabul edilmeye devam etmiştir (Rushton vd., 2010: 6-8). Lojistiğin tarihsel gelişimi, özet olarak Tablo 1'de ifade edilmiştir.

Tablo 1: Lojistiğin Gelişimi

AŞAMALAR	YÖNETİM MERKEZİ	ÖRGÜTSEL TASARIM
	1960 Yılları	
Depolama ve Ulaştırma	✓ Satış Pazarlama, ✓ Depolama, ✓ Stok Denetimi, ✓ Ulaştırma Etkinliği,	✓ Dağınık lojistik faaliyetler ✓ Lojistik faaliyetler arasında zayıf bağlantı ✓ Düşük lojistik yönetimi otoritesi işletme başarısını etkiler
	1980 Yılları	
Toplam Maliyet Yönetimi	✓ Lojistiğin merkezileştirilmesi ✓ Toplam maliyet yönetimi ✓ Süreç optimizasyonu ✓ Rekabetçi bir avantaj olarak lojistik	✓ Merkezileşmiş lojistik faaliyetler ✓ Büyüyen lojistik yönetimi otoritesi ✓ Bilgisayar uygulamaları
	1990 Yılları	
Entegre Lojistik Yönetimi	✓ Lojistik planlama ✓ Tedarik zinciri stratejileri ✓ İşletme faaliyetleri ile bütünleşme ✓ Süreç kanalları ile bütünleşme	✓ Lojistik faaliyetlerde genişleme ✓ Tedarik zinciri planlama ✓ Toplam kalite yönetimi için destek ✓ Lojistik yönetim faaliyetleri
	2000 Yılları	
Tedarik Zinciri Yönetimi	✓ Stratejik tedarik zinciri görüşü ✓ Extranet teknoloji kullanımı ✓ Kanal güçlerini ortak bir kuvvet aracı kullanmak için tedarik zinciri TQM göstergelerinde işbirliği yapmak	✓ Ticari ortaklık ✓ Sanal örgüt ✓ Talepteki değişimler ✓ Benchmarking ve yeniden yapılanma
	2000 Yılı ve Sonrası	
E-Tedarik Zinciri Yönetimi	✓ SCM kavramına internetin uygulanması ✓ Düşük maliyetli anında veri tabanı paylaşımı ✓ Elektronik Bilgi ✓ SCM senkronizasyonu	✓ Tedarik zinciri ağı ile ticaret ortaklığı yapmak ✓ .com, -e eklentisi vb. piyasa değişiklikleri ✓ Örgütsel çeviklik ve ölçülebilirlik

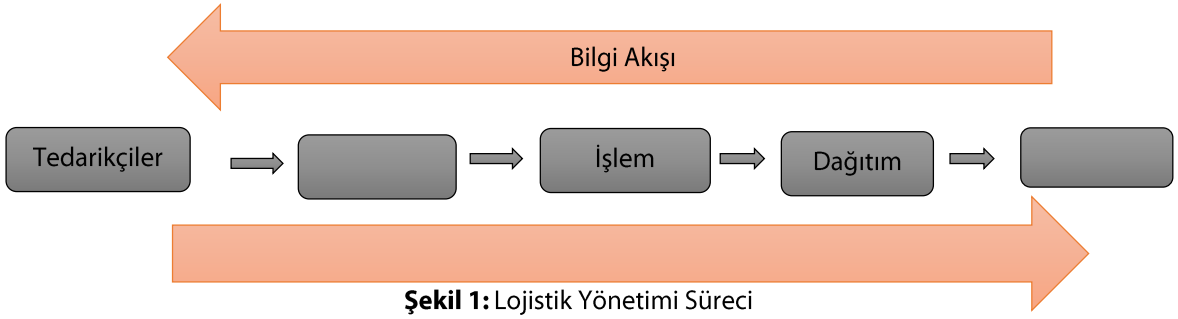
Kaynak: (Gülenç ve Karagöz, 2008: 77)

Lojistik Yönetimi

Ticari personelin kâr amacı gütmeyen bir kuruluşu olan Lojistik Yönetimi Konseyine göre, lojistik yönetimi; müşteri gereksinimlerini göz önünde bulundurarak malların, hizmetlerin ve ilgili bilgilerin menşe noktasından tüketime kadar verimli, etkin bir şekilde akışını ve depolanmasını planlama, uygulama ve kontrol etme sürecidir (Bramel ve Simchi-Levi, 1997: 1).

Tedarik Yönetimi Uzmanları Konseyi'nin (2011) tanımına göre, lojistik yönetimi "Müşterilerin gereksinimlerini karşılamak için üretim ve tüketim noktası arasındaki ilgili bilgilerin, araç-gereç ve malların, akışını ve depolanmasını tersine çeviren, etkin bir şekilde ilerlemeyi planlayan, uygulayan ve kontrol eden Tedarik Zinciri Yönetiminin önemli bir kısmıdır" şeklinde tanımlanmaktadır. Ayrıca, lojistik yönetimi, bütün lojistik faaliyetlerini kontrol eden, en iyi şekilde kullanan ve bu lojistik faaliyetleri pazarlama, satış üretimi, finans ve bilgi teknolojisi gibi diğer fonksiyonlarla bütünleştiren bir fonksiyondur.

Şekil 1'de lojistik yönetiminin süreci ayrıntılı bir şekilde gösterilmiştir. Tüm süreç, tedarikçi veya üretici ile başlar ve daha sonra tedarik, işlem, dağıtım ve müşteri tarafından bitirme gibi şirketin tüm lojistik operasyonlarında devam eder. Ayrıca, müşteriden gelen malzeme akışının yönünü ve bilgi akışını net bir şekilde anlamak çok önemlidir. Şekil 1'e dayanarak, lojistik yönetiminin temel kavramlarını tanımlamak kolaydır.



Şekil 1: Lojistik Yönetimi Süreci

Lojistik yönetiminin her iki tanımını da inceledikten ve Şekil 1'i analiz ettikten sonra, lojistik yönetiminin hammadde yönetiminden nihai ürünün teslimatına kadar organizasyonu kapsadığını söylemek mümkündür (Petelina, 2016: 16-17). Bu toplam sistem bakış açısına göre, pazardan, firma ve operasyonları ve bunun ötesinde tedarikçilere tedarik eden malzeme ve bilgi akışlarının koordinasyonu yoluyla müşterilerin ihtiyaçlarının karşılanması anlamına gelir. Bu şirket çapında entegrasyonun başarılması, geleneksel organizasyonda tipik olarak karşılaşılandan oldukça farklı bir oryantasyon gerektirir. Örneğin, uzun yıllardır pazarlama ve imalat, organizasyon içinde büyük ölçüde ayrı faaliyetler olarak görülmüştür. İmalat öncelikleri ve hedefleri tipik olarak, uzun üretim süreleri, minimize edilmiş kurulumlar ve değişimler ve ürün standardizasyonu yoluyla elde edilen işletme verimliliğine odaklanmıştır. Diğer taraftan, pazarlama çeşitliliği, yüksek servis seviyeleri ve sık ürün değişimleri ile rekabet avantajı elde etmeye çalışmıştır.

Son yıllarda hem pazarlama hem de üretimin yenilenen dikkatin odağı haline gelmesi tesadüf değildir. Pazarlama ve müşteri odaklılık felsefesi olarak pazarlama artık her zamankinden daha geniş bir kabul görmektedir. Artık genel olarak müşteri gereksinimlerini anlama ve karşılama ihtiyacının hayatta kalmak için bir ön şart olduğu kabul edilmektedir. Aynı zamanda, artan maliyet rekabetçiliği arayışında, üretim yönetimi büyük bir devrimin konusu olmuştur. Son on yılda, esnek üretim sistemlerine, malzeme ihtiyaç planlamasına ve tam zamanında üretime dayalı envantere ait yeni yaklaşımlar hızlı bir şekilde tanıtılmıştır. Aynı şekilde, entegre lojistik sürecinin bir parçası olarak, satın almanın rekabet avantajı yaratmada ve sürdürmede kritik rol oynar. Önde gelen kuruluşlar artık stratejik planlarının geliştirilmesine düzenli olarak arz yönlü sorunları dahil etmektedirler. Çoğu kuruluşta yalnızca satın alınan malzemelerin ve tedariklerin maliyeti toplam maliyetlerin önemli bir bölümünü oluşturmakla kalmaz, aynı zamanda alıcıların ve tedarikçilerin lojistik süreçlerinin daha yakın

entegrasyonu yoluyla tedarikçilerin yetenek ve yeterliliklerinden yararlanmak için büyük bir fırsattır. Bu nedenle, bu açıdan lojistik, temel olarak firmanın sistem çapında bir bakış açısını geliştirmeyi amaçlayan bütünleştirici bir kavramdır (Christopher, 2005: 15-16).

Lojistik Yönetiminin Önemi

Lojistik yönetimi, günümüzde ekonomik zorluklarla yüzleşmek için kullanılan araçlardan biridir; kuruluşun iş ve temel faaliyetlerinin bir karışımıdır. Bütünleşik bir şekilde ele alınan tedarik ve dağıtım faaliyetleri, lojistik faaliyetlerini oluştururlar. Bir iş organizasyonu içindeki lojistik faaliyetler, müşterilerin ihtiyaçlarını ve satın alma gücünü göz önünde bulundurarak, zaman ve yer ile ilgili pazar zorluklarını ve ayrıca sağlanan hizmetin maliyeti ve kalitesi aracılığıyla müşterileri memnun etmeye çalışır. Müşteri memnuniyeti önemlidir çünkü pazarlamacılara ve işletme sahiplerine işlerini yönetmek ve geliştirmek için kullanabilecekleri bir ölçü sağlar. Müşteri memnuniyeti aynı zamanda müşterilerin sadakatini ölçerek işletmenin veya bir ürün yaşamının sürekliliğini belirlemenin bir yoludur. Müşterilerin mutlu ve memnun olması satışların devamlılığını yani işin devamlılığını sağlayacaktır.

Tüm araştırmacılar ve iş danışmanları, lojistik yönetiminin, bir müşterinin ihtiyacının karşılanmasında doğrudan veya dolaylı olarak yer alan tüm taraflardan (üreticiler, pazarlamacılar, tedarikçiler, nakliyeciler, depolar, perakendeciler ve hatta müşteriler dâhil) oluştuğu konusunda hemfikirdir. Lojistiğin ana hedefleri, müşteriye olan ürün veya hizmet sunumunu iyileştirerek genel organizasyon performansını ve müşteri memnuniyetini iyileştirmektir. Lojistik analiz, kalite ve süreyi dikkate alarak tedarikçilerden son kullanıcıya kadar olan maliyeti düşürmeyi amaçlar. Başlıca müşteri memnuniyeti göstergelerinden ikisi maliyetler ve bekleme süresidir. Her iki müşteri memnuniyeti göstergesi de ucuz bir ürün (ucuz hammadde kullanımı, en ucuz nakliye yöntemini seçme, düşük işçilik maliyeti ile yüksek üretim, düşük maliyetli depolama ve teslimat) ile sonuçlanan lojistik süreçte ima edilmektedir (Ghoumrassi ve Tigu, 2017: 292-296). Asıl hedefi müşteriyi memnun etmek olan lojistik sisteminin her bir bileşeni, müşterinin doğru ürünü, doğru yerde, minimum maliyetle doğru zamanda alıp almadığını etkileyebilir. Günümüzde birçok şirket lojistik yönetimini müşteriyi memnun etmek ve nihayetinde müşteri hizmetlerini iyileştirmek amacıyla uygulamaktadır. Gereken hizmetin minimum maliyetle sağlanması ise etkili bir lojistik yönetimi sayesinde gerçekleşmektedir (Kaveh ve Samani, 2009: 16-17).

Lojistik sistemlere ait dağıtım ağı biçimi, üretim, envanter kontrolü, çapraz yerleştirme, envanter ve dağıtım entegrasyonu, araç filosu yönetimi, araç rotalama, paketleme, dağıtımın belirli zaman aralıklarında gerçekleştirilmesi, toplama ve dağıtım sistemleri gibi karmaşık bir yapıya sahip olan lojistik problemleri bir şirketin performansı üzerinde önemli bir etkiye sahiptir. Dolayısıyla şirket yönetiminin bu tarz karmaşık lojistik problemlere optimum çözümler bulabilmesi için lojistik yönetimine gereken önemi vermesi, planını etkin bir şekilde önceden yapması ve yönetmesi gerekir. Lojistik yönetimini devam ettirmesiyle müşterilerin ihtiyacını karşılayacak ve müşterilerden olumlu geri dönüşler alacağı için gelirini arttıracaktır. Diğer taraftan en uygun ulaşım ve taktiksel planlama yöntemlerini seçerek işletme maliyetlerini ve toplam ulaşım maliyetlerini de azaltacaktır (Petelina, 2016: 18; Bramel ve Simchi-Levi, 1997: 3-6; Friedrich ve Gumpp, 2014: 47).

İKİNCİ BÖLÜM

ARAÇ ROTALAMA PROBLEMİ

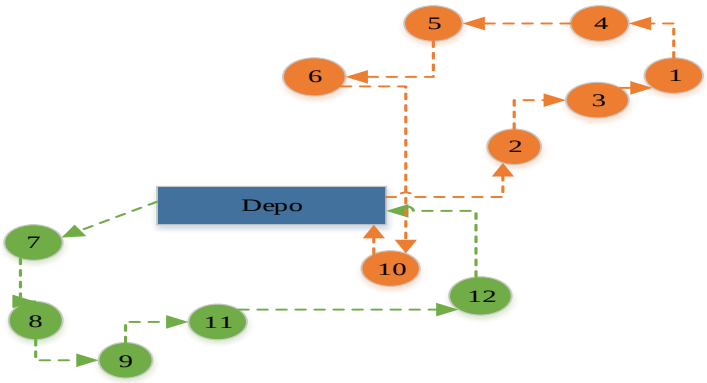
Lojistik, hammadde alımından nihai ürünün teslimine kadar, toplam ürün akışının etkin yönetimi ile ilgilidir (Golden, 1975: 1). Malların ve hizmetlerin verimli dağıtımı, birçok günlük aktivitenin merkezinde yer almaktadır. Bir okul otobüsü tarafından toplanan bir çocuk, bir yerleşim mahallesinin sokaklarında dolaşan çöp kamyonu, ayda bir defa müşterilerinin sayaçlarını okuyan bir elektrik şirketi gibi dağıtım sorunları araçların rotalanması ile ilgili sorunlardır. Bu faaliyetlere katılan araçların toplam maliyeti çok büyük olabilir ve bu araçları mümkün olduğunca verimli bir şekilde yönlendirmek önemlidir. Bununla birlikte, bu tür sorunlara etkin çözümler geliştirmek, son derece zor olduğu kanıtlanan bir kombinatoryal optimizasyon problemini çözmemizi gerektirir (Groer, 2008: 1-2).

Tam bir lojistik sistem, hammaddelerin bir dizi tedarikçiden veya satıcıdan nakledilmesini, üretim veya işleme için fabrika tesisine teslim edilmesini, ürünlerin çeşitli ambarlara veya depolara taşınmasını ve sonunda müşterilere dağıtılmasını içerir. Hem tedarik hem de dağıtım prosedürleri etkin taşımacılık yönetimi gerektirir. İyi bir taşımacılık yönetimi, bir şirkette toplam dağıtım maliyetini önemli miktarda azaltabilir. Potansiyel maliyet tasarrufları; daha uygun rotalar sayesinde daha düşük kamyon maliyeti ve daha kısa mesafeler, azaltılmış kurum içi alan ve ilgili maliyetler, geç teslimat nedeniyle daha az ceza gibi faktörleri oluşturur. En önemli taşımacılık yönetimi önlemlerinden birisi de araçların etkili bir şekilde rotalanmasıdır. Çeşitli kısıtlamalar verilen araçlar için rotaların optimize edilmesi, araç rotalama problemlerinin kaynağıdır. Birçok gerçek nakliye lojistiği ve dağıtım problemi araç rotalama problemi olarak ifade edilebilir. Bu tarz problemlerdeki amaç, bilinen talepleri olan bir dizi müşteriye hizmet etmek için minimum maliyetle bir rota planlamaktır. Her müşteri yalnızca bir rotaya tahsis edilir ve herhangi bir rotanın toplam talebi araç kapasitesini aşmamalıdır. Tipik bir araç rotalama problemi Şekil 2'de ifade edilmiştir. Çözüm iki rotadan oluşmaktadır. Tek depo ve 12 müşteriden oluşan problemde her rota depoda başlar ve müşterileri ziyaret ettikten sonra depoya geri döner (Tan vd., 2001: 281- Alzaqebah vd., 2016: 1).

Rota 1: Depo → 7→8→9→11→12→Depo

Rota 2: Depo→2→3→1→4→5→6→10→Depo

Bazen depo 0 (sıfır) olarak belirtilir.



Şekil 2: Araç Rotalama Problemi (ARP)

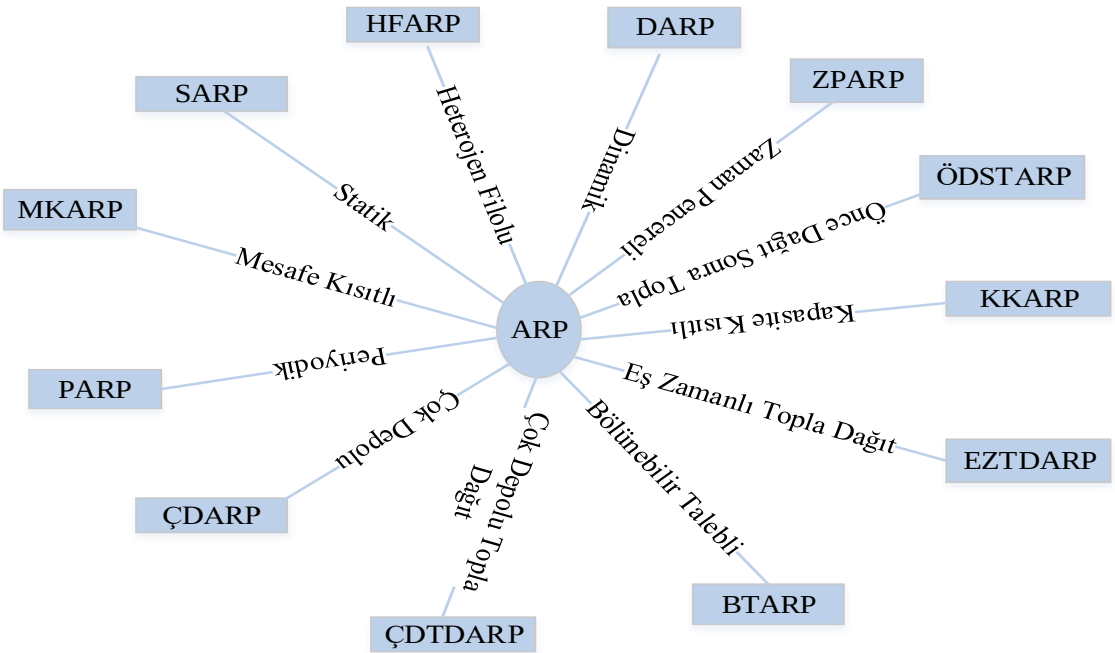
Kaynak: (Tan vd., 2001: 281-282)

Bir dizi yan kısıtlamaya tabi olan ve bir dizi coğrafi olarak dağılmış müşteri ile en düşük maliyetli teslimat rotalarının tasarlanmasından ibaret olan araç rotalama problemi, dağıtım yönetiminde merkezi bir yere sahiptir ve dünya çapında on binlerce taşıyıcı tarafından günlük olarak karşı karşıya kalınmaktadır. Problem, uygulamada karşılaşılan çeşitli kısıtlamalardan dolayı çeşitli formlarda ortaya çıkmaktadır. 50 yıldan fazla bir süredir, araç rotalama problemi, operasyonel araştırma topluluğunun büyük bir kısmının dikkatini çekmiştir. Bu, kısmen sorunun ekonomik önemi nedeniyle değil, aynı zamanda ortaya koyduğu metodolojik zorluklardan da kaynaklanmaktadır. Örneğin, Araç Rotalama Probleminin (ARP) kapasite kısıtı eklenmemiş hali olan gezgin satıcı problemi artık binlerce ve hatta on binlerce tepe için çözülebilmektedir. Buna karşılık, araç rotalama probleminin çözülmesi çok daha zordur. Örneğin, 100 veya 200 müşterili yalnızca kapasite kısıtlamalarının mevcut olduğu basitleştirilmiş araç rotalama problemini, kesin algoritmalar aracılığıyla çözmek hala zordur. Bu nedenle son yıllarda, araştırma girişimcilerinin çoğu güçlü meta-sezgisellerin geliştirilmesine yönelmiştir (Laporte vd., 2013: 1-2).

Araç rotalama problemi ilk olarak Dantzig ve Ramser tarafından 1959 yılında ortaya konmuştur. İki matematikçi, bir petrol dağıtım şirketinin verilerini kullanarak ARP'yi çözen ilk algoritmayı uygulamıştır. Ardından, Clarke ve Wright (1964) geliştirdikleri Tasarruf (Savings) yöntemi ile mevcut ARP çözümlerini geliştirmişlerdir (Namany, 2017: 12). Klasik ARP'de, homojen bir araç filosu, her müşterinin bilinen bir talebi olduğu bir dizi müşteri lokasyonları verilmiştir. Görev, tüm müşterilerin talebini karşılayacak minimum maliyette bir rota seti oluşturmak ve ayrıca bireysel rotaların, araç kapasitesine ve rota uzunluğu kısıtlamalarına uymasını sağlamaktır. ARP'nin ifade edilmesi basit ve anlaşılması kolay olmasına rağmen, deneyim pratikte çözülmesinin çok zor olduğunu göstermiştir. Ayrıca uygulamada, firmanın birden fazla araca sahip olması, belirli müşterilerin belirli bir süre boyunca hizmete ihtiyaç duyabilmesi, planlama ufkunun bir zamanda birkaç gün içermesi gibi ortaya çıkan birçok başka komplikasyon da söz konusudur. ARP'yi çözmedeki güçlük ve pratikte ortaya çıkan ek hususların sayısı nedeniyle, gerçek boyuttaki problemlerin tipik olarak sezgisel yöntemlerle çözülmesi gerekir. ARP için kesin çözüm yöntemleri olarak tamsayılı programlama veya dinamik programlama gibi yöntemler kullanılır ve hatta modern bilgi işlem kaynaklarında bile, bu yöntemler tipik olarak 100'den fazla müşteri lokasyonuna sahip optimallik problemlerini çözemez. ARP, NP-Zor (polinomsal zamanda çözümü olduğu ispatlanamayan) problemler kategorisine giren iyi bilinen bir tamsayılı programlama problemi olduğundan dolayı bu problemi kesin yöntemlerle çözmek problem boyutu arttıkça zorlaşmaktadır. Dolayısıyla gerçek dünyadaki ARP örnekleri binlerce düğüme sahip olabileceğinden ve problemi daha da karmaşıklaştıran birçok ek kısıtlama içerebileceği için, yüksek kaliteli sezgisel yöntemlere olan ihtiyaç açıktır (Groer, 2008: 1-2; Altman, t.y.: 6).

"Araç rotası" ifadesinin göründüğü ilk makale Golden vd. (1972) tarafından yazılmıştır. Araç Rotalama Probleminin (ARP) diğer versiyonları 70'lerin başında ortaya çıkmıştır. Liebman ve Marks (1970), atıkların toplanmasıyla ilişkili rotalama problemini temsil eden matematiksel modeller sunmuşlardır. Levin (1971), filo taşıtları sorununu ortaya koymuştur. Wilson ve diğerleri (1971) engelli kişiler için ulaşım hizmetinde telefon talebi sorununu ve Marks ve Stricker (1971) kamu hizmeti araçlarını rotalamak için bir model sunmuşlardır. Bazı olasılıksal kavramlar, Golden ve Stewart (1975) tarafından ARP ile ilişkilendirilmiştir. Solomon (1987)'de zaman pencerelerinin kısıtlamalarını probleme dâhil etmiştir. Sarıklis ve Powell (2000), aracın, yolculuğun başladığı noktaya dönmediği bir problem önermiştir (Toro vd., 2016: 363). Araç rotalama problemi için kesin algoritmaların geliştirilmesi 1981 yılında Christofides, Mingozzi ve Toth'un Networks'te yayınladığı iki makalenin yayınlanmasıyla başlamıştır. Bu makalelerin ilki, durum uzayı gevşetmeli dinamik programlamaya dayalı bir algoritma sunarken, ikincisi, q-yollarını ve k-en kısa yayılma ağaçlarını kullanan iki matematiksel formülasyon önermiştir. Laporte vd. (1984), araç rotalama problemi için bir tamsayı modelinin doğrusal gevşeme çözümünü temel alan ilk düzlem kesme yaklaşımını önermişlerdir. Bu seminal kavramlar daha yeni algoritmalardan bazılarının içerisine girmeyi başarmıştır. O zamandan beri, matematiksel programlama formülasyonlarına dayanan çeşitli kesin algoritmalar önerilmiştir. Bazı formülasyonlar araç akışı veya mal akışı değişkenlerini içerir ve genellikle dal ve kesme ile çözülür. ARP, bazı geçerli eşitsizliklerin eklendiği bir küme bölümleme problemi olarak da formüle edilebilir. Fukasawa vd. (2006) ve Baldacci vd. (2008) tarafından gerçekleştirilen en başarılı uygulamalardan bazıları bu metodolojiye dayanmaktadır. ARP için modern sezgisellerin gelişimi, 1990'larda meta-sezgisellerin ortaya çıkışıyla başlamıştır. En iyi meta-sezgiseller, çözüm alanını aynı anda geniş ve derin bir şekilde araştıran ve problemin çeşitli türevlerini çözebilen olmuştur (Laporte vd., 2013: 2).

Son yıllarda, ARP, yük dağıtım planlaması ve yönetiminin kombinatoryal optimizasyonu üzerinde en çok çalışılanlardan biri olmuştur; bunun sebebi, gerçek hayattaki uygulamalardaki büyük önemi ve ayrıca ciddi zor olmasından kaynaklanmaktadır (Santana, 2016: 1). Rekabetin artması ve çevre koşullarının değişmesi sebebiyle rotalama problemlerine giderek daha fazla kısıtın eklenmesi araç rotalama problemlerine uygun çözümler bulmayı giderek zorlaştırmaktadır (Erol, 2006: 7). Bu sebepten dolayı araç rotalama problemi, problemin bileşenleri olan kısıtlar, araçlar, müşteriler, yollar ve rotalar gibi çeşitli kriterlere göre Şekil 3'teki gibi sınıflandırılmıştır.



Şekil 3: Araç Rotama Problemi Çeşitleri
Kaynak: (Şahin ve Eroğlu, 2014: 338; Keskintürk vd., 2015: 81)

Kapasite Kısıtlı Araç Rotalama Problemi (KKARP)

Kapasiteli Araç Rotalama Problemi (KARP), nakliye, dağıtım ve lojistik konularındaki pratik uygulamalarla birlikte kombinatoryal optimizasyonundaki temel problemlerden biridir. KARP'nin amacı, tek bir depoya dayalı bir kapasitedeki araç filosu için aşağıdaki sınırlamalar altında bir dizi müşteriye hizmet vermek için minimum toplam maliyet rotalarını bulmaktır:

- ✓ Her rota depoda başlamalı ve depoda sonlanmalıdır,
- ✓ Her müşteri tam olarak bir kez ziyaret edilmelidir,
- ✓ Her bir güzergâhın toplam talebi, aracın kapasitesini aşmamalıdır (Borcinova, 2017: 463).

Araçların ortak bir depodan müşterilere minimum transit maliyetle ulaştırılması için homojen oldukları ve belirli bir kapasiteye sahip oldukları kabul edilmektedir (Santana, 2016: 9-10).

Kapasiteli araç rotalama problemi, aşağıdaki gibi tanımlanabilir:

$V' = \{0,1, \dots, n\}, n+1$ tane köşelerin kümesi ve E, kenarların kümesi olmak üzere yönlendirilmemiş bir $G = (V', E)$ grafiği verilsin. 0 köşesi depoya ve $V = V' \setminus \{0\}$ tepe noktası n müşteriye karşılık gelmektedir. Negatif olmayan bir d_{ij} maliyeti, her bir $\{i, j\} \in E$ kenarı ile ilişkilidir. q_i yükleri, depo 0'dan $(q_0 = 0)$ tedarik edilmektedir. Depo 0' da Q kapasiteli özdeş m araçların kümesi yerleştirilmiştir ve araçlar müşterilere tedarik etmek için kullanılmaktadır. Bir rota, ziyaret edilen köşelerin toplam talebinin araç kapasitesini geçmeyeceği şekilde depo 0'dan geçen en az basit bir grafik G grafik devresi olarak tanımlanır (Yeun vd., 2008: 207).

Kapasite kısıtlı araç rotalama probleminin notasyonlarını, amaç fonksiyonunu ve kısıtlama denklemlerini içeren tipik bir matematiksel formülasyonu aşağıda verilmiştir (Kao vd., 2012: 4-5):

Notasyonlar:

0: depoların indeksi

N: toplam müşteri sayısı

K: toplam araç sayısı

C_{ij}: i müşterisinden j müşterisine seyahat ederken ortaya çıkan maliyet

S_i: i müşterisi için gereken servis zamanı, $S_0 = 0$

Q: bir aracın maksimum yük kapasitesi

T: bir aracın maksimum sürüş mesafesi

d_i: i müşterisinin talebi, $d_0 = 0$

X_{ij}^k: $0-1$ karar değişkeni

$$X_{ij}^k = \begin{cases} 1, & i \text{ müşterisinden } j \text{ müşterisine } k \text{ aracı tarafından seyahat} \\ 0, & \text{aksi halde} \end{cases}$$

p: ceza katsayısı

R: bir araç tarafından hizmet edilen müşterilerin kümesi,

$|R|$: R'nin kardinalitesi

Amaç Fonksiyonu:

$$Min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} C_{ij} X_{ij}^k \qquad (2.1)$$

Kısıtlar

$$\sum_{k=1}^{K}\sum_{i=0}^{N} X_{ij}^{k} = 1, \;\; j = 1,2,\dots,N \tag{2.2}$$

$$\sum_{k=1}^{K}\sum_{j=0}^{N} X_{ij}^{k} = 1, \;\; i = 1,2,\dots,N \tag{2.3}$$

$$\sum_{i=0}^{N} X_{iu}^{k} - \sum_{j=0}^{N} X_{uj}^{k} = 0, \qquad k = 1,2,\dots,K; u = 1,2,\dots,N \tag{2.4}$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^{k} d_i \le Q, = 1,2,\dots,K \tag{2.5}$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^{k}\left(C_{ij} + S_i\right) \le T, \;\; k = 1,2,\dots,K \tag{2.6}$$

$$\sum_{j=1}^{N} X_{ij}^{k} = \sum_{j=1}^{N} X_{ji}^{k} \le 1, i = 0; k = 1,2,\dots,K \tag{2.7}$$

$$\sum_{ij\in R} X_{ij}^{k} \le |R| - 1, R \subseteq \{1,2,\dots,N\}; 2 \le |R| \le N - 1; k = 1,2,\dots,K \tag{2.8}$$

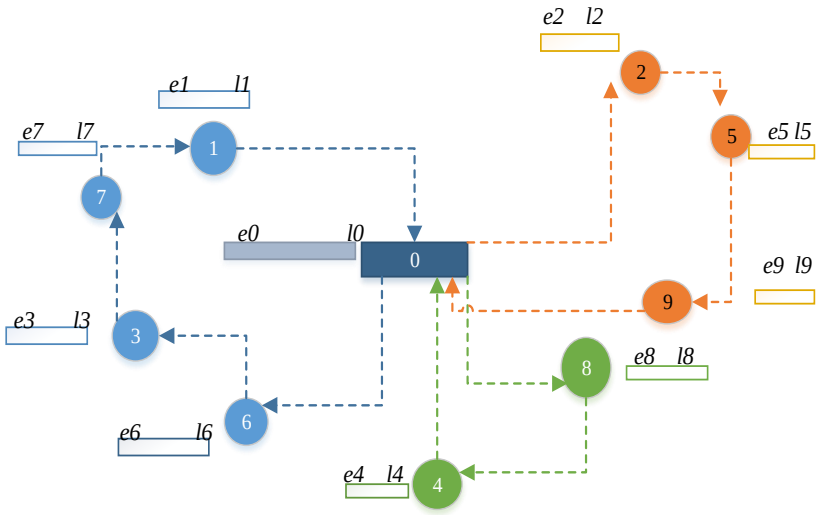
$$X_{i,j}^{k} \in \{0,1\}, i, j = 0,1,\dots,N; k = 1,2,\dots,K \tag{2.9}$$

(2.1) denklemi kapasite kısıtlı araç rotalama probleminin amaç fonksiyonudur. (2.2) ve (2.3) denklemi her müşteriye yalnızca bir araç tarafından hizmet verilebilmesini sağlar. (2.4) denklemi, her araç için her düğümde sürekliliği korur. (2.5) denklemi, bir aracın toplam müşteri talebinin aracın maksimum kapasitesini aşmamasını ifade eder. Benzer şekilde (2.6) denklemi, bir aracın toplam rota mesafesinin rota uzunluğu sınırını aşmamasını sağlar. Denklem (2.7), her aracın en fazla bir defa kullanılabileceğinden ve depoda başlayıp depoda sonlanması gerektiğinden emin olur. Alt tur eleme kısıtları (2.8) denkleminde verilmiştir. (2.9) denklemi tamsayı kısıtıdır.

Zaman Pencereli Araç Rotalama Problemi (ZPARP)

Zaman pencereli araç rotalama problemi, tartışmasız şu anda en geniş çapta çalışılan araç rotalama problemi çeşididir. Klasik zaman pencereli araç rotalama problemi 1987'de Solomon, ardından 1999'da Gehring ve Homberger tarafından tanıtılmıştır. Araç rotalama probleminin önemli bir uzantısı olan zaman pencereli araç rotalama problemi, birçok gerçek dünya problemini modelleyebilen ve bilinen talepleri olan müşterilere hizmet veren bir araç filosu için, bir merkezi depodan çıkan ve depoda sonlanan bir dizi minimum maliyetli rotaları tasarlamaktan oluşan temel bir dağıtım yönetimi problemidir. Müşteriler, araç kapasitelerinin aşılmaması için araçlara tam olarak bir kez tayin edilmelidir. Bir müşterideki servis, müşterinin hizmet başlangıcına izin verdiği en erken zaman ve en son zaman olarak belirlenen zaman penceresi içerisinde başlamalıdır. Banka teslimatları, posta teslimatları, endüstriyel atık toplama, ulusal franchise restoran teslimatları ve okul otobüsü rotalama ve güvenlik devriyesi hizmetleri gibi alanlar problemin uygulama alanları arasındadır (El-Sherbeny, 2010: 124; Santana, 2016: 13). ZPARP, önceden belirlenmiş zaman dilimi içerisinde, müşteri setinin sadece bir kez ziyaret edildiği bir dizi araç yolunun belirlenmesinden oluşur. Zaman penceresi, zaman aralığı olarak tanımlanır ve $v \in V$ müşterisine atanan aracın zaman penceresi içerisinde ulaşması gerekir. Genellikle zaman penceresi sınırlayıcıları, müşteriye hizmet verilmesi gereken $[e_0, l_0]$ zaman penceresi içerisinde erken varış zamanı e_0, geç varış zamanı l_0 olarak tanımlanır. Öncelikli amaç, araç filosunu,

seyahat süresini ve bekleme süresini en aza indirmektir. Tüm filonun kapasiteyi aşmadan nakliye rotasını depoda başlatması ve depoda sonlandırması gerekmektedir. Her rota depo ile ilişkilendirilmiş zaman içerisinde başlamalıdır. Müşterinin zaman penceresine başlamadan önce aracın geldiği durum da olabilir. Bu durumda, araç müşterinin bulunduğu yerde beklemek zorunda kalır ve sonuç olarak rota üzerinde ekstra bekleme süresi sağlar. Ancak yine de geç varış yasaktır, çünkü araç müşteriye üst sınırdan sonra teslimat yapılırsa çözüm mümkün olmaz. Her birinin belirli bir zaman penceresi olan 9 müşterinin 3 farklı rotaya tahsis edildiğini gösteren zaman pencereli araç rotalama problemine ait bir örnek çözüm Şekil 4'te gösterilmektedir. Her bir rota bir araçla geçildiğinden dolayı toplam rota sayısı kadar toplam araç sayısı söz konusudur.



Şekil 4: Zaman Pencereli Araç Rotalama Problemi (ZPARP)

Kaynak: (Pan vd., 2013: 2256)

Şekil 4'te ifade edilen ZPARP'i örneğine ait rotaların oluşturmuş olduğu çözüm aşağıda ifade edilmiştir.

0	6	3	7	1	0	8	4	0	2	5	9	0

Araç 1 (Rota 1): $\boldsymbol{Depo(0)}\rightarrow\mathbf{6}\rightarrow\mathbf{3}\rightarrow\mathbf{7}\rightarrow\mathbf{1}\rightarrow\boldsymbol{Depo(0)}$

Araç 2 (Rota 2): $\boldsymbol{Depo(0)}\rightarrow\mathbf{8}\rightarrow\mathbf{4}\rightarrow\boldsymbol{Depo(0)}$

Araç 3 (Rota 3): $\boldsymbol{Depo(0)}\rightarrow\mathbf{2}\rightarrow\mathbf{5}\rightarrow\mathbf{9}\rightarrow\boldsymbol{Depo(0)}$

Zaman penceresi kısıtlamasına göre, ZPARP'nin iki farklı durumunu ayırt edebiliriz. Sıkı Zaman Pencereli Araç Rotalama Problemi: Zaman penceresi kısıtı, yerine getirilmesi gereken problemdir. Esnek Zaman Pencereli Araç Rotalama Problemi: Zaman penceresi kısıtlamasının ihlaline izin verilir, başka bir deyişle, daha sonraki bir servis, çözümün uygulanabilirliğini etkilemez. Bununla birlikte, çözüm maliyetlerini artıran amaç fonksiyonuna yansıyan bir ceza ile gelir. (Pan vd., 2013: 2256; Santana, 2016: 13).

Problemin Tanımı

ZPARP, farklı coğrafi bölgelerde yer alan çeşitli taleplere ve belirli zaman aralıklarına sahip bir dizi müşteriye hizmet vermek için depodan çıkan müşterilerin taleplerini karşıladıktan sonra başlangıç noktasına geri dönen bir araç filosundan ibarettir. Amaç, araçların kapasitelerini, seyahat süresi kısıtlamalarını ve müşteriler tarafından belirlenen zaman penceresi kısıtlamalarını ihlal etmeden tüm müşterilere minimum maliyetle (seyahat mesafesi vb.) hizmet etmeleri için araçlara uygun rotaların bulunmasıdır.

- Her bir rota deponun zaman aralığına uygun bir şekilde depoda başlayıp, depoda bitmelidir.
- Bütün müşterilerin talepleri tek seferde ve tek bir araçla karşılanmalıdır.
- Aynı rotadaki müşterilerin toplam talep miktarları maksimum aracın kapasitesi kadar olmalıdır.
- Her bir müşterinin talebi müşterilere ait zaman pencereleri içerisinde karşılanmalıdır. Araç müşterinin açılış zamanından önce gelmişse o zamana kadar beklemek durumundadır.

ZPARP'nin kısıtları; bir dizi aynı araç, merkezi bir depo düğümü, bir dizi müşteri düğümü, depo ve müşteri birbirine bağlayan bir ağdan oluşmaktadır. $N+1$ tane müşteri ve K tane araç vardır. Depo düğümü, 0 olarak ifade edilir. Ağdaki her yay, iki düğüm arasındaki bir bağlantıyı temsil eder ve aynı zamanda hareket ettiği yönü gösterir. Her rota depodan başlar, müşteri düğümlerini ziyaret eder ve daha sonra depoya geri döner. Ağdaki rotaların sayısı kullanılan araçların sayısına eşittir. Her araç bir rotaya tahsis edilmiştir. Bir c_{ij} maliyeti ve bir t_{ij} seyahat süresi ağın her bir yayı ile ilişkilidir. Ağdaki her müşteri sadece bir araç tarafından ziyaret edilmektedir. Her araç aynı q_k kapasitesine ve her müşteri değişen bir m_i talebine sahiptir. q_k, k aracı tarafından seyahat edilen rotadaki tüm taleplerin toplamına eşit veya daha büyük olmalıdır, bu, hiçbir aracın aşırı yüklenemeyeceği anlamına gelir. Zaman penceresi kısıtı, en erken varış zamanı ve en son varış zamanı olmak üzere önceden tanımlanmış bir zaman aralığı ile gösterilir. Araç, en geç varış saatinden daha geç olmamak kaydıyla müşterilere ulaşmalıdır. Eğer en erken varış saatinden daha erken ulaşırsa, beklemelidir. Her müşteri aynı zamanda, malların yükleme / boşaltma zamanını dikkate alarak rotaya bir servis süresi uygular. Araçların bireysel rotalarını da esasen deponun zaman penceresi olan toplam rota süresi içerisinde tamamlamaları gerekir. Zaman pencereli araç rotalama probleminde üç tip temel karar değişkeni vardır. x_{ijk} $(i,j \in \{0,1,\dots,N\}; k \in \{1,2,\dots,K\}; i \neq j)$, k aracı i müşterisinden j müşterisine seyahat ettiğinde 1 değerini aksi halde 0 değerini alan temel karar değişkenidir. t_i karar değişkeni bir aracın müşteriye vardığı zamanı ve w_i karar değişkeni ise i düğümündeki bekleme zamanını gösterir. Amaç, toplam seyahat maliyetini en aza indirerek aynı anda tüm kısıtlamaları karşılayan bir ağ tasarlamaktır (Tan vd., 2001: 283; Göçken vd., 2018: 776-777).

ZPARP'nin matematiksel modeli, modelde kullanılan değişkenler ve parametreler aşağıda ifade edilmiştir (Jawarneh ve Abdullah, 2015: 3-5):

Karar Değişkeni

$$x_{ijk} = \begin{cases} 1, & eğer\ k\ aracı\ i.\ müşteriden\ j.\ müşteriye\ giderse \\ 0, & aksi\ halde \end{cases}, i \neq j, i, j \in \{0,1,\dots N\}$$

Parametreler

$t_i - i$ Noktasına varış zamanı

$w_i - i$ Noktasında bekleme zamanı

$K -$ Araçların sayısı

$N -$ Müşterilerin sayısı (0 merkezi depo olarak tanımlanır)

$c_{ij} - i$ Müşterisi ile j müşterisi arasındaki seyahat maliyeti

$t_{ij} - i$ Müşterisi ile j müşterisi arasındaki seyahat süresi

$m_i - i$ Müşterisinin talebi

$q_k - k$ Aracının kapasitesi

$e_i - i$ Müşterisi için en erken varış zamanı

$l_i - i$ Müşterisi için en son varış zamanı

$f_i - i$ Müşterisi için servis zamanı

$r_k - k$ Aracı için maksimum rota süresi

Amaç Fonksiyonu

$$Min \sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk} \quad (2.10)$$

Kısıtlar

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} \leq K \ , \quad i = 0 \quad (2.11)$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{jik} \leq 1, \quad i = 0; k \in \{1,2,\dots,K\} \quad (2.12)$$

$$\sum_{k=1}^{K} \sum_{j=0, j\neq i}^{N} x_{ijk} = 1, \quad i \in \{1,2,\dots,N\} \quad (2.13)$$

$$\sum_{k=1}^{K} \sum_{i=0, i\neq j}^{N} x_{ijk} = 1 \ , \quad j \in \{1,2,\dots,N\} \quad (2.14)$$

$$\sum_{i=1}^{N} m_i \sum_{j=0, j\neq i}^{N} x_{ijk} \leq q_k \ , \quad k \in \{1,2,\dots,K\} \quad (2.15)$$

$$\sum_{i=1}^{N} \sum_{j=0, j\neq i}^{N} x_{ijk}\left(t_{ij} + f_i + w_i\right) \leq r_k, \quad k \in \{1,2,\dots,K\} \quad (2.16)$$

$$t_0 = w_0 = f_0 \quad (2.17)$$

$$\sum_{k=1}^{K} \sum_{i=0, j\neq i}^{N} x_{ijk}\left(t_i + t_{ij} + f_i + w_i\right) \leq t_j, \qquad j \in \{1,2,\dots,N\} \quad (2.18)$$

$$e_i \leq (t_i + w_i) \leq l_i, \qquad i \in \{1,2,\dots,N\} \quad (2.19)$$

$$c_{ij} = \sqrt{(i_x - j_x)^2 + \left(i_y - j_y\right)^2} \quad (2.20)$$

(2.10) denklemi, zaman pencereli araç rotalama probleminin temel amaç fonksiyonunu ifade eder. (2.11) nolu kısıt, maksimum K rotasını veya depodan çıkan araçları gösterir. Her rotanın depoda başlayıp ve depoda bitmesi gerektiğini belirten kısıt ise (2.12)'tür. (2.13) ve (2.14) nolu kısıtlar, her müşterinin yalnızca bir araç tarafından tam olarak bir kez ziyaret edilmesini sağlar. (2.15) ve (2.16) denklemleri sırasıyla kapasite ve azami seyahat süresi kısıtlamalarını tanımlar. (2.17), (2.18) ve (2.19) kısıtları zaman penceresini açıklar. Denklem (2.20) ise seyahat maliyetini hesaplar. Burada i_x i müşterisinin x koordinatı ve i_y ise i müşterisinin y koordinatıdır.

ZPARP için Çözüm Yöntemleri

ZPARP'yi çözmek için kesin, sezgisel ve meta-sezgisel yöntemlerden faydalanılır. Bu problemler, polinomsal zamanda bir çözümü olduğunu ispatlayamadığımız karar problemlerinin karmaşıklık sınıfına girdiğinden kesin yaklaşımlar sadece küçük boyutlu problemlerin çözümünde etkilidir. Dolayısıyla büyük boyutlu problemlerin çözümü için sezgisel ve meta-sezgisel yöntemler kullanılır (Ai ve Kachitvichyanukul, 2009: 521). ZPARP'nin çözümü için araştırmacılar tarafından geliştirilmiş pek çok yöntem literatürde yer almaktadır. Bu yöntemler, optimal çözüme ulaşıp ulaşmaması durumuna göre kesin ve sezgisel olmak üzere iki gruba ayrılır. ZPARP'nin çözümü için kullanılan yöntemler Şekil 5'te verilmiştir.

Şekil 5: ZPARP Çözüm Yöntemleri

Kesin Çözüm Yöntemleri

NP-zorlu optimizasyon problemlerini en uygun şekilde çözmek, bilgisayar tarihinin başlangıcından bu yana araştırmacıları zorlayan bir konu olmuştur. Son yıllarda önemli ilerleme kaydedilmesine rağmen çoğu problem tipi için yalnızca oldukça küçük durumlar çözülebilmektedir. Araç rotalama problemleri, çözülmesi zor olan kanıtlanmış problemler grubuna dahil olduğundan sadece orta büyüklükteki problemler tutarlı bir şekilde en uygun çözüme ulaşmaktadır. Bu da kesin çözüm yöntemleri ile sağlanmaktadır. Bu tam arama yöntemleri, en iyisini elde edene kadar çözümün her olasılığını hesaplamayı önerir. Bu nedenle son zamanlarda araç rotalama problemlerinin çözümü için kesin yaklaşımlar ileri sürülmüştür. Sonuç olarak, optimal olarak çözülebilen örnekler açısından önemli sonuçlar elde edilmiştir (Ropke, 2005: 146; Santana, 2016: 16). Hem uygulama alanının genişliği hem de NP-zor yapısı nedeniyle araç rotalama problemi oldukça dikkat çeken bir problem haline gelmiştir. ARP'nin bir genellemesi olan ZPARP de NP-zor sınıfında bir optimizasyon problemi olması sebebiyle araştırmacılar tarafından dikkat çekmiş ve literatürde önemli bir yere sahip olmuştur. Bu problemin çözümü için kullanılan kesin çözüm yöntemleri, ***hedef programlama*** (Ghoseiri ve Ghannadpour, 2010; Calvete vd., 2007; Aydemir, 2006), ***dal ve kesme*** (Brad vd., 2002), ***dal ve sınır*** (Tezer, 2009;), ***doğrusal programlama*** (Akca, 2015;), ***sütun türetme*** (Qureshi vd., 2009; Qureshi vd., 2010; Liberatore vd., 2011), ***düzlem kesme*** (Cook ve Rich, 1999), ***dinamik programlama*** (Kok vd., 2010) ve ***küme bölümleme algoritmasıdır*** (Tezer, 2009; Alvarenga vd., 2007).

Sezgisel Yöntemler

Genel olarak, kombinatoryal problemlerin formüle edilmesi kolaydır, ancak tamsayılı programlama problemleri olarak formüle edildikleri için çözülmesi zordur. Bu problemler NP-zor problemler sınıfına ait olduğundan, çözümleri için çoğu zaman sezgisel yöntemler kullanılır. Sezgisel bir arama yöntemi, makul bir hesaplama süresi içinde iyi bir çözümü belirlemek için problem yapısından faydalanan bir prosedür olarak görülebilir. Bu nedenle, bir sezgisel aramanın verimliliği, üretilen çözümün kalitesi ve gereken bilgisayar süresi açısından belirlenir. Özellikle büyük boyutlu gerçek yaşam problemlerinde çözüm üretmek için "kesin yöntemler" yerine sezgisel yöntemleri kullanmak daha uygundur. Gerçekte, bir gerçek dünya uygulamasıyla ilişkilendirilen matematiksel modelin boyutu ve verilerin kesinliği olmadığı düşünüldüğünde, "optimal" bir çözüm aramak tamamen gerçekçi olmayabilir. Bazı durumlarda, kesin yöntemlerin hızı, iyi bir sezgisel yöntemin hızından daha yavaştır. Buluşsal yöntemler genel olarak daha sezgiseldir ve son kullanıcı için daha erişilebilirdir. Parametrelerin ayarlanması gerekse bile, son kullanıcılar genel olarak parametrelerin değiştirilmesinin etkilerini öngörebilir (Ferland ve Costa, 2001: 2).

Yüksek zorluk derecesi ve gerçek hayatta karşılaşılan problemlerden biri olan ZPARP'de de en iyi çözümlerin ortaya konması için sezgisel yöntemler yoğun olarak kullanılmaktadır. Son yıllarda yapılan çalışmalar incelendiğinde, çalışmaların büyük bir kısmında ZPARP'nin çözümü için en çok sezgisel ve meta-sezgisellerin kullanıldığı görülmektedir. Tasarruf (saving), süpürme (sweep), en yakın komşu, ekleme sezgiselleri klasik sezgisel yöntemler arasında yer alırken; genetik, tabu arama, tavlama benzetimi, yerel arama, karınca kolonisi, yapay arı kolonisi, değişken komşu arama, ateş böceği, yarasa, parçacık sürü optimizasyonu, harmoni arama, kurbağa sıçrama, memetik algoritma, bakteriyel yiyecek arama, evrimsel algoritma ve yusufçuk gibi algoritmalar meta-sezgisel yöntemler sınıfında yer almaktadır.

Klasik Sezgiseller

ARP'nin çözümünde kullanılan klasik sezgiseller, rota oluşturucu yöntemler, rota geliştirici yöntemler ve iki aşamalı yöntemler olmak üzere 1992 yılında Laporte tarafından 3 gruba ayrılmıştır. Bu yöntemler, başlangıç çözümlerin oluşturulmasında kullanılan rota oluşturucu sezgiseller, bu sezgiseller yardımıyla oluşturulan çözümlerin niteliğini arttırmak için kullanılan rota geliştirici sezgiseller ve önce grupla sonra rotala veya önce rotala sonra grupla türü iki aşamalı yöntemlerden oluşmaktadır (Şahin, 2014: 71).

Tasarruf Algoritması

Araç rotalama problemini çözmek için kullanılan bir dizi sezgisel yöntem vardır. Araç rotalama problemini çözen en iyi bilinen sezgisel yöntemlerden biri Clarke ve Wright'in tasarruf algoritmasıdır (Jerabek vd., 2016: 117). Algoritma, değişken olarak araç sayısının kullanıldığı problemlerde ele alınmaktadır. Rotaların oluşturulması ve kaç adet araç kullanılması gerektiği bu yöntem ile bulunmaktadır. Paralel ve sıralı olmak üzere iki tip tasarruf algoritması söz konusudur. Bu yöntemin adımları aşağıda verilmiştir.

Adım 1. Her bir müşteri çifti için S_{ij} tasarrufları aşağıdaki gibi hesaplanır.

$$S_{ij} = d_{i0} + d_{0j} - d_{ij} \tag{2.21}$$

S_{ij} tasarruf değerleri büyükten küçüğe doğru sıralanır.

Adım 2. Paralel tasarruf algoritmasında S_{ij} tasarruf değerlerine göre büyükten küçüğe sıralanan (i, j) müşteri ikilileri; sıralamada xy ve yz müşteri çiftleri arka arkaya geliyorsa araç kapasitesi dikkate alınmak koşuluyla $0 - x - y - z - 0$ doğrultusunda bir rota oluşturularak, aksi halde xy ve tz olacak şekilde bu müşteri çiftleri arka arkaya geldiğinde bu müşteriler için ayrı ayrı rota olauşturulacak şekilde

birleştirilir. Sıralamada dikkate alınan müşteri çiftleri ile daha önce oluşturulan rotalar karşılaştırılır ve uygunsa rotaya eklenir. Burada birden fazla rota oluşturulur ve oluşturulan bu rotalar paralel versiyonda işlenerek müşteri çiftlerini bu rotalara atamak mümkündür. Tasarruf algoritmasının bir diğer çeşidi olan sıralı versiyonunda kapasite kısıtı gözönüne alınarak xy ve tz formatındaki müşteri çifti birleştirilerek $0-x-y-t-z-0$ doğrultusunda rotalar oluşturulur. Sıralamada dikkate alınan müşteri ikilileri aynı rota için karşılaştırılır ve kapasite kısıtına uygunsa rotaya eklenir. Bir rota tamamen oluşuncaya kadar tüm müşteri ikilisi kapasite kısıtı dikkate alınarak elden geçer. Her iki versiyonda da müşteri ikilileri toplam maliyet minimum olacak şekilde rotalara aktarılır. Tasarruf algoritmasının paralel versiyonu sıralı versiyonuna oranla daha iyi sonuçlar doğurur. Sıralı tasarruf aloritmasında yalnızca bir rota baz alınır ve bu rotayı elde etmek için S_{ij} tasarruf sıralaması dikkate alınır. Paralel versiyonda müşteri çiftinin birden fazla rota ile uygunluğunun S_{ij} sıralamasına bakılarak kontrol edilmesi ve bu sayede daha fazla alternatif gözden geçirilir (Erol, 2006: 30).

Backer ve Schaffer (1989), tasarruf algoritmasının bir uzantısını kullanmışlardır. Tasarruf algoritmasına ilişkin benzer bir sezgisel de Solomon tarafından 1987 yılında geliştirilmiştir. Ayrıca Landeghem (1988), iki kriterli tasarruf sezgiselini çalışmasında ele almıştır. İki kriterli sezgisel yöntemi, müşteriler arasında bir bağlantının zamanlama açısından ne kadar iyi olduğunu ölçmek için zaman pencerelerini kullanmıştır (El-Sherbeny, 2010: 127). Demircioğlu (2009), ZPARP ile ilgili çalışmasında tasarruf yöntemini geliştirmiş ve Mersin'deki bir dağıtım firmasına uygulama yapmıştır. Bu araştırma ile firmalara daha uygun dağıtım rotası belirlenerek maliyette tasarruf ve bundan sonra yapılacak olan çalışmalara fayda sağlaması açısından önemli sonuçlar elde edilmiştir.

En Yakın Komşu Algoritması

En yakın komşu sezgiseli, araç rotalama problemlerini çözmek için sıklıkla kullanılan bir diğer algoritmadır. Genellikle rota geliştirme sezgisellerini test etmek için başlangıç çözüm üretmek için kullanılır. Oldukça hızlı olmasına rağmen, çözümler neredeyse her zaman optimalden uzaktır. En yakın komşu algoritması, bir dizi şehirden rastgele bir başlangıç şehri seçer, onu bir rotaya ekler ve o şehri ziyaret edilen şehir olarak işaretleyerek şehir listesinden kaldırır. Daha sonra, algoritma bir dizi ziyaret edilmemiş şehirden rotaya daha önce eklenmiş olan şehre en yakın şehri bulur ve onu şehrin ziyaret edildiğini gösteren rotaya ekler. Bu süreç tüm şehirler ziyaret edilene kadar tekrar eder (Jakara vd., 2019: 458-459).

Göçken vd. (2018), ZPARP'nin çözümü için çok amaçlı genetik algoritmanın başlangıç popülasyonunun oluşturulmasında en yakın komşu algoritmasını kullanmışlardır.

Ekleme Sezgiseli

Solomon, ARP tur oluşturma algoritmalarını paralel ve sıralı olmak üzere ikiye ayırmıştır. Paralel prosedürler, rota sayısının önceden belirlenmiş bir sayı ile sınırlı olduğu veya serbestçe oluşturulduğu eş zamanlı rota inşası yoluyla oluşturulurken, sıralı prosedürler, tüm müşteriler planlanana kadar her seferinde bir rota oluşturur.

Sıralı ekleme sezgiseli, çözümün kalitesi ve çözümü bulmak için gereken hesaplama süresi açısından çok etkilidir. İlk çözümleri bulmak için başlatma kriterleri, rotaya eklenecek ilk müşteriyi seçme sürecini ifade eder. En yaygın kullanılan başlatma kriterleri, ya en uzak rotalanmamış müşteri ve en erken teslim tarihine sahip müşteri ya da en erken ve en son kabul edilebilir varış zamanı kriterleridir. Bir rotaya ilk eklenen müşteriye "çekirdek müşteri" adı verilir. Çekirdek müşteri seçilip bir rotaya yerleştirildiğinde, sıralı ekleme algoritması, yönlendirilmiş düğümler için, yakın zamanda ve kısmen oluşturulmuş rotaya bir müşteri eklemek için gereken ek mesafeyi ve zamanı en aza indiren yerleştirme yerini dikkate alır. Bir sonraki adımdaki seçim kriterleri, yeni bir doğrudan rota yerine müşteriyi mevcut kısmi rotaya

eklemekten kaynaklanan avantajı en üst düzeye çıkarmaya çalışır. Burada dikkat edilmesi gereken önemli noktanın düğümler ile müşteri terimlerinin birbirinin yerine kullanıldığıdır (Joubert ve Claasen, 2006: 107).

Jawarneh ve Abdullah (2015), ZPARP'ne uyarlamış oldukları arı kolonisi optimizasyon algoritmasında başlangıç yiyecek kaynaklarının oluşturulmasında sıralı ekleme sezgiselinden yararlanmışlardır.

Süpürme Algoritması

Gillet ve Miller (1974) tarafından geliştirilen bu algoritmada depo merkezli bir doğrunun döndürülmesi ile rotalarda yer alacak müşteriler elde edilir. Döndürme süresince doğru üzerinden geçen müşteriler iki sınıfa ayrılır. Grubun kapatılması kapasite veya mesafe kısıtına bağlıdır. Bu kısıtlardan biri aşıldığı zaman grup kapatılarak yeni bir grupla devam edilir. Merkez deponun da eklenmesi ile oluşturulan nokta grupları genel olarak gezgin satıcı problemi gibi çözülmesiyle rotalar oluşturulur. Müşteri koordinatları θ açı ve ρ doğru uzunluğu olmak üzere (θ_i, ρ_i) polar formatındadır. Talep noktalarının küçükten büyüğe doğru sıralanması θ açısına göre gerçekleşir. Ayrıca kapasite ve mesafe kısıtı gözönüne alınarak talep noktaları gruplanır. Süpürme algoritmasının genel işleyişi aşağıdaki gibidir (Erol, 2006: 37-38).

Adım 1: Depo ve müşteri noktalarının yeri harita üzerinden belirlenerek koordinatlar polar formatında (θ_i, ρ_i) yazılır. Rotaya tahsis edilmemiş herhangi bir araç tespit edilir.

Adım 2: Depodan saat yönünün tersinde 0° açı ile taranmaya başlanır. Bir müşteri ile karşılaşıldığında müşterinin talep miktarı aracın kapasitesini geçmezse müşteri araca atanır. Aksine bir durum söz konusu olduğunda saat yönünün tersinde hareket edilir. Diğer araca geçilmesi, her iki yönde de araca müşteri atanamadığında gerçekleşir.

Adım 3: Tarama, önceki aracın kaldığı yerden devam eder. Daha önce rotalama olmayan bir müşteri ile karşılaşılırsa, müşterinin araca atanması kapasite kısıtını aşmayacak şekilde gerçekleşir. Tüm noktaların rotalaması bitene kadar bu süreç devam eder.

Göçken vd. (2018), ZPARP'nin çözümü için çok amaçlı genetik algoritmanın başlangıç popülasyonunun oluşturulmasında süpürme algoritmasını kullanmışlardır. Diğer taraftan Hertrich vd. (2019), çalışmalarında basit ve etkili süpürme algoritmasını kullanarak probleme çözüm bulmuşlardır.

Meta Sezgiseller

Meta-sezgisel yöntemler, arama alanını araştırmak için zeki bir şekilde farklı kavramları birleştirerek, bir alt sezgisel çalışmayı yönlendiren yinelemeli bir çözüm süreci olarak tanımlanır. Öğrenme stratejileri, bilgiyi en uygun çözümlere ulaştırmak ve bilgileri yapılandırmak için kullanılır. Meta-sezgisel algoritmalar da, karmaşık problemleri çözmek için kullanılabilecek bu yaklaşık teknikler arasındadır (Said vd., 2014: 2). Genellikle çok sayıda probleme uygulanan güçlü teknikler ve çözüm kavramı olarak ifade edilir. Bir meta-sezgisel, tam veya eksik bir tek çözümü veya her bir yinelemede bir çözümler topluluğunu manipüle edebilir (El Sherbeny, 2010: 129). Meta-sezgiler, sezgisel tarama ve / veya geleneksel arama teknikleri de dahil olmak üzere diğer arama yöntemleri arasındaki işbirliğini koordine etmek için üst düzey problem çözme stratejileri olarak düşünülmüştür. Bu basit prosedür, genellikle kuş ve böcek sürülerinin davranışları, metallerdeki soğutma prosedürleri veya doğal evrim gibi doğal veya fiziksel olaylardan esinlenerek dizayn edilir.

Günümüzde çok sayıdaki gerçek dünya problemi meta-sezgisel teknikler kullanılarak çözülmektedir. Otomasyon ve robotik, biyoinformatik, ekonomi ve finans, mühendislik tasarımı, sağlık

ve tıp, görüntü işleme, bilgi işleme ve veri madenciliği, lojistik ve araç planlama, makine öğrenmesi, imalat ve endüstriyel uygulamalar, fizik uygulamalar, güç ağı optimizasyonu, rotalama, zamanlama, güvenlik ve güven, yazılım mühendisliği, stratejik ve askeri uygulamalar, telekomünikasyon, iş gücü planlama ve diğer alanlarda ortaya çıkan birçok gerçek dünya optimizasyon probleminde meta-sezgisel teknikler kullanılmış ve kullanılmaya devam edilmektedir (Nesmachnow, 2014: 322-329). Bu meta-sezgisel yöntemler arasında karınca kolonisi, tabu arama, tavlama benzetimi, yapay arı kolonisi, parçacık sürü optimizasyonu gibi yöntemler bu problemlerde sıkça kullanılan yöntemler arasındadır (Şahin, 2014: 73). ZPARP'nin çözümünde kullanılan metasezgisel yöntemlerin kullanıldığı çalışmalar takip eden bölümde incelenmiştir.

Genetik Algoritma

Genetik algoritmalar, arama ve optimizasyon problemlerini çözmek için kullanılabilecek adaptif yöntemlerdir. Biyolojik organizmaların genetik süreçlerine dayanmaktadır. Birçok nesil boyunca, doğal popülasyonlar, doğal seçilim ve "en uygun olanın hayatta kalması" ilkelerine göre evrimleşmektedir. Genetik algoritmalar bu süreci taklit ederek, eğer uygun şekilde kodlanmışlarsa, gerçek dünya problemlerine çözümler "geliştirebilir". Genetik algoritmaların gücü, tekniğin sağlam olması ve diğer metotların çözmesi için zorlayıcı olanlar da dâhil olmak üzere geniş bir problem alanıyla başarılı bir şekilde başa çıkabilmesinden kaynaklanmaktadır. Genetik algoritmaların bir probleme küresel optimum çözümü bulması garanti edilmez, fakat genellikle problemlere "kabul edilebilir derecede hızlı ve iyi" çözüm bulmakta iyidirler. Belirli problemleri çözmek için özel tekniklerin mevcut olduğu durumlarda, nihai sonucun hem hızında hem de doğruluğunda GA'lardan daha iyi performans göstermeleri muhtemeldir. O halde, GA'lar için temel zemin, bu tür tekniklerin olmadığı zor alanlardır. Mevcut tekniklerin iyi çalıştığı durumlarda bile, bunları bir GA ile hibritleyerek iyileştirmeler yapılmıştır (Beasley vd., 1993: 1-2). Kalıtım, mutasyon, seçim ve çaprazlama gibi doğal evrimden esinlenen teknikleri kullanarak optimizasyon problemlerine çözümler üreten daha geniş evrimsel algoritma sınıfına ait olan algoritmanın basit genel prosedürü aşağıdaki gibidir (Yadav ve Prajabati, 2012: 1-2).

Adım 1: Bireylerin başlangıç popülasyonunu seç.

Adım 2: Bu popülasyondaki her bireyin uygunluğunu hesapla.

Adım 3: Sona erene kadar bu nesli tekrarla (Zaman sınırı, yeterli uygunluk elde edilene kadar).

Adım 4: Üreme için en uygun bireyleri seç.

Adım 5: Çaprazlama ve mutasyon işlemleriyle yeni bireyler üret.

Adım 6: Yeni bireylerin bireysel uygunluğunu hesapla.

Adım 7: En az uygunluğa sahip popülasyonu yeni bireylerle değiştir.

Genetik algoritmalar biyoinformatik, filogenetik, hesaplamalı bilim, mühendislik, ekonomi, kimya, üretim, matematik, fizik, lojistik gibi alanlarda optimizasyon problemlerine uygulanmaktadır (Verma ve Verma, 2012: 5). Bu uygulama alanlarından biri olan lojistik dağıtımın özü ve odak noktası olan zaman pencereli araç rotalama problemi üzerine yapılan araştırmalar, temel olarak geliştirilmiş akıllı optimizasyon algoritmalarından olan genetik algoritmaya odaklanmaktadır. Bu araştırmalar aşağıda verilmiştir.

Braysy (2001), zaman pencereli araç rotalama problemini çözmek için geliştirilen genetik algoritma tabanlı yaklaşımları kısaca gözden geçirmiş ve saf genetik algoritmalarla elde edilen sonuçların, yayınlanmış olan en iyi sonuçlarla rekabetçi olmadığı, ancak farklılıkların çok büyük olmadığı kanısına varılmıştır. Berger ve Barkaoui (2004), yeni bir hibrit genetik algoritmanın paralel versiyonunu ele

almışlardır. Ele alınan yaklaşım, en iyi bilinen sezgisel rotalama yöntemleri olarak kullanılan tabu arama, çoklu karınca koloni sistemi, rota-komşuluk bazlı gibi algoritmalarla karşılaştırılmış ve çok rekabetçi olduğu gösterilmiştir. Alvarenga vd. (2007), etkin bir genetik algoritma ve küme bölme formülasyonu yoluyla seyahat uzaklığını temel amaç olarak kullanan sağlam bir sezgisel yaklaşım sunmuşlardır. Dursun (2009), ZPARP için rassal sayı kodlamalı melez olmayan bir genetik algoritma yaklaşımını ele almıştır. Diğer yöntemlerle rekabet edebilecek bir model ortaya koymak çalışmanın hedefi olmuştur.

Nazif ve Lee (2010), gereklilikleri karşılayan ve minimum toplam maliyet sağlayan, optimum bir teslimat rotası bulan, tam bir yönlendirilmemiş iki taraflı bir grafik tarafından tasarlanan optimize edilmiş bir çaprazlama operatörü kullanarak genetik bir algoritma önermişlerdir. Algoritma, bazı yaklaşımlarla karşılaştırılmış ve sonuçlar, algoritmanın bulunan çözümlerin kalitesi açısından rekabetçi olduğunu göstermiştir. Ghoseiri ve Ghannadpour (2010), hem gerekli toplam filo büyüklüğünün hem de toplam hareket mesafesinin en aza indirildiği ve kapasite ve zaman pencereleri kısıtlarının güvence altına alındığı çok amaçlı bir problemi ele almışlardır. Problemin formülasyonu için bir hedef programlama yaklaşımı ve bunu çözmek için uyarlanmış, etkin bir genetik algoritma kullanmışlardır. Yöntem, literatürde en iyi bilinen yaklaşımlarla rekabet edebilecek çözümler sağlamıştır. Kiremitçi vd. (2014), çok araçlı, dağıtım toplamalı, zaman pencereli rotalama problemine özgü gerçek değerli kodlamalı bir genetik algoritma geliştirerek çözüm aramışlardır. Ele alınan algoritma, Li ve Lim (2001)'in geliştirdiği tabu gömülü tavlama benzetimi meta-sezgiseli ve bilinen en iyi sonuçlar ile karşılaştırılmış, algoritmanın iyi sonuçlar verdiği ve bu problemler için iyi bir alternatif çözüm yöntemi olabileceği anlaşılmıştır.

Kuram (2016), zaman pencereli araç rotalama probleminde kullanılan yöntemleri detaylı bir şekilde inceleyerek, literatürde yer alan test problemlerini genetik algoritmalar ile çözmüştür. Gupta ve Diwaker (2017), karınca koloni optimizasyonu ve genetik algoritmanın birleşiminden oluşan yöntemi kullanmışlardır. Yöntemde, popülasyon çeşitliliğini arttırmak ve çözüm alanını tamamen genişletebilmek için genetik algoritmadaki mutasyon operatörleri yer almıştır. Algoritma, karınca koloni optimizasyonu, genetik algoritma gibi yöntemlerle karşılaştırılmış ve sonuç olarak yaklaşımın, problemi etkili bir şekilde çözebildiği, problemin amaçlarının yerine getirilmesinden anlaşılmıştır. Göçken vd. (2018), çok amaçlı genetik algoritma yaklaşımını ele almışlar ve bu yaklaşımla ZPARP'ne yönelik daha önce yapılan çalışmalardan daha etkin sonuçlar elde etmişlerdir.

Tabu Arama Algoritması

Tabu araması, yerel bir optimuma yakalanmaktan özellikle kaçınan, genişletilmiş komşulukları inşa etmek için bir yinelemeli meta-strateji prosedürüdür. Bu küresel optimizasyon meta-sezgiseli ilk olarak Glover tarafından ortaya atılmış, Glover ve arkadaşları tarafından geliştirilmiştir. Amaç fonksiyonu değerinde bir bozulmaya yol açsa bile, bir çözümden en iyi komşusuna hareket ederek arama uzayını araştırmaktan ibarettir. Bu şekilde yerel optimadan hareket etme olasılığı arttırılır. Bir çözümün ardışık komşuları üretilir ve amaç fonksiyon değerleri incelenir. Döngüden kaçınmak için, yakın zamanda incelenen çözümler belirli sayıda yineleme için yasaklanır veya tabu olarak ilan edilir (Barbarosoğlu ve Ozgur, 1999: 259). Tabu listesi, arama boyunca keşfedilen çözümleri veya daha genel olarak bu çözümlerin bazı ilgili özelliklerini bir listeye kaydederek belleği kullanma yollarından biridir.

Başarılı bir Tabu Arama uygulamasında, yoğunlaştırma ve çeşitlendirme arasında iyi bir dengenin olması önemlidir. Yoğunlaşma, genellikle iyi bir çözüm çevresinde, çözüm alanının bazı bölgelerinin ayrıntılı bir keşfidir. Çeşitlendirme, araştırmanın henüz keşfedilmemiş ve birbirinden ayrı, çözüm açısından umut verici bölgelere doğru yönlendirilmesini içerir. Operasyonel açıdan, Tabu Arama, her bir yinelemede, tek bir S çözümünden, en iyi N (S) çözümüne, tabu listesinde değil veya eğer öyleyse, bir miktar aspirasyon kriterini yerine getirerek başarılı bir şekilde hareket eder (Brandao, 2011: 141). Tabu arama prensibi, sürekli olarak mevcut en iyi çözümü geliştirmeye ve önceki hareketlerin listesini hafızaya

kaydederek, araştırmayı daha önce seyahat edilen bölgelerin dışında yönlendirmeye çalışırken, çözümlerin alanı üzerinde hareket etme yöntemine dayanmaktadır (Kaddouri ve Omary, 2017: 82-83). Mevcut en iyi çözümü geliştirme prensibine dayanan bu yaklaşım araç rotalama problemlerine de çok sayıda uygulanmıştır. Çözüm kalitesi ve süresi bakımından iyi sonuç verdiği için ARP'nin bir uzantısı olan ZPARP'de de en çok kullanılan yöntemler arasında olduğu aşağıda verilen çalışmalarca ispatlanmıştır.

Badeau vd. (1997), zaman pencereli araç rotalama probleminin çözümü için bir paralel tabu arama sezgiselini geliştirerek, iş istasyonları ağı üzerinde uygulamışlardır. Ampirik olarak, orijinal sıralı algoritmanın paralelleştirilmesinin, uygulamada önemli hızlandırmalar sağlarken aynı miktarda hesaplama için çözelti kalitesini düşürmediği gösterilmiştir. Schulze ve Fahle (1999), problem için yeni bir paralel tabu arama algoritmasını tanımlamışlardır. Basit müşteri değişimlerine dayanan ve mümkün olmayan geçici çözümleri düşünmeye olanak sağlayan bir komşuluk yapısı kullanılmıştır. İleri sürülen bu algoritma literatürdeki bazı sezgiseller ile karşılaştırılmış ve algoritma çözüm kalitesini koruyarak, önemli hızlandırmalar sağladığı sonucuna ulaşılmıştır. Braysy ve Gendreau (2002), tabu arama yaklaşımlarının çok etkili olduğu sonucuna yaptıkları kıyaslamalı analizlerden elde etmişlerdir. Chiang ve Russell (2004), karma bir komşuluk yapısı ve yüksek kalitede çözümler üretmek için gelişmiş bir iyileştirme prosedürü kullanarak bir tabu arama çözüm yöntemi geliştirmişlerdir. k-opt ve λ-değişim komşulukları birlikte ele alınarak karma bir komşuluk yapısı oluşturulmuştur. Hesaplamalı sonuçlardan algoritmanın etkili olduğu ispatlanmıştır.

Homberger ve Gehring (2005), zaman pencereli araç rotalama problemi için iki aşamalı hibrit bir meta-sezgiseli yöntem olarak kullanmışlardır. İlk aşamada, (μ;λ) evrim stratejisi ile araç sayısının en aza indirilmesi, ikinci aşamada ise tabu arama algoritması kullanılarak toplam uzaklığın en aza indirilmesi planlanmıştır. Komşuluk yapıları olarak insert, 2-opt ve change operatörleri kullanılmıştır. Ming-Yao vd. (2008), basit bir komşuluk yapısına sahip bir tabu arama algoritmasını tasarlamışlardır. Fu vd. (2008), zaman pencereli araç rotalama problemlerinin farklı türleri için birleşik bir ceza fonksiyonu ve birleşik bir tabu arama algoritması tasarlamışlardır. Algoritmada, 2-değişim üretim mekanizmasına dayanan karışık bir komşuluk yapısı kullanılmıştır. Yu vd. (2011), tabu arama algoritması ve karınca koloni algoritmasından oluşan melez bir yaklaşımı ele almışlardır.

Moccia vd. (2012), daha önce geliştirilen tabu arama sezgiselinin komşuluk yapısında değişiklikler yaparak, zaman pencereli araç rotalama problemi için bu algoritmanın etkili olduğuna hesaplamalı sonuçlardan ulaşmışlardır. Taş vd. (2014), esnek zaman pencereli araç rotalama problemi için tabu arama algoritması aracılığı ile bir çözüm prosedürü geliştirmişlerdir. Tabu arama algoritmasına dayanan çözüm prosedürü, literatürde bulunan diğer sezgisellerden daha iyi performans gösterdiği karşılaştırma sonucunda anlaşılmıştır. Kırcı (2016), google haritalarından gerçek dünya uygulamaları olarak kabul edilen zaman pencereli araç rotalama problemleri için tabu arama algoritmasını ve hopfield sinir ağlarını kullanmıştır.

Tavlama Benzetimi

ZPARP'nin çözümünde kullanılan bir diğer meta sezgisel de tavlama benzetimidir. Kavramsal olarak tavlama benzetimi; bir malzemenin bir sıvı stat içine ısıtıldığı ve tekrar kristalize edilmiş bir katı duruma geri soğutulduğu, tavlama olarak bilinen fiziksel bir işleme benzer olduğu gerçeğinden kaynaklanmaktadır. Geliştirilen ilk meta-sezgisellerden birisidir. Tavlama benzetimi kullanıldığında mevcut çözümün komşuluğunda en iyi çözüm aranmaz. Bunun yerine biri rastgele bir komşuluktan basitçe bir çözüm çizer. Çözüm daha iyiyse, her zaman yeni bir güncel çözüm olarak kabul edilir, ancak çözüm mevcut çözümden daha kötüyse, yalnızca belirli bir olasılıkla kabul edilir. Kabul olasılığı kademeli olarak düşürülen bir sıcaklık tarafından belirlenir. Sıcaklığın düşürülmesiyle, seçim yeni

çözümün kabulünde giderek daha seçici hale gelir. Tavlama benzetimi kavramı termodinamik ve metalürjiden gelir: füzyondaki bir metal yeterince yavaş bir şekilde soğutulduğunda, minimum enerjili bir yapıda katılaşma eğilimi gösterir (El-Sherbeny, 2010: 129).

Chiang ve Russell (1996), zaman pencereli araç rotalama problemi için 3 tavlama benzetimi meta-sezgiselini geliştirmişlerdir. Kullanılan yöntemlerde Osman'ın λ-değişim mekanizması, Christofides ve Beasley'in k-düğüm değişim süreci olmak üzere iki farklı komşuluk yapısı uygulanmıştır. Hesaplamalı deneyler, benzetilmiş tavlama uygulamalarının, temel benzetim tavlama algoritmalarının yavaş yakınsamasının üstesinden gelerek, bir 486DX2/66 kişisel bilgisayarı kullanarak makul CPU zamanında çok iyi sonuçlar elde edebileceğini göstermiştir. Bent ve Hentenryck (2004), bahsi geçen ulaşım problem için iki aşamalı melez bir algoritma önermişlerdir. İlk olarak tavlama benzetimi kullanılarak araç sayısı en aza indirilmiş, ardından, çok sayıda müşterinin yerini değiştirebilecek büyük geniş bir komşuluk araması kullanarak seyahat maliyetini en aza indirilmiştir. Deneysel sonuçlar, geliştiren algoritmanın etkinliğini ve algoritmanın çok sağlam olduğunu göstermiştir. Li ve Lim (2003), yerel aramaları çeşitlendirmek ve yoğunlaştırmak için yeniden başlatmalar gibi tavlama yöntemlerine dayanan bir meta-sezgiseli önermişlerdir. Yerel arama komşuluklarını oluşturmak için yöntemde 3 kenar değiştirme operatörü (shift, exchange, re-arrange) kullanılmıştır. Karşılaştırmalı deneylerden en iyi sonuçlar elde edilmiştir.

Chen ve Ting (2005), geliştirilmiş karınca koloni algoritması ile tavlama benzetimi algoritmasını bir araya getirerek melez bir algoritma oluşturmuşlardır. Geliştirilmiş karınca koloni sisteminde 2-opt ve ekleme olmak üzere iki farklı yerel arama operatörü kullanılmıştır. Tabu arama, çoklu karınca koloni sistemi, genetik algoritma gibi daha önceki meta-sezgisel yöntemlerle karşılaştırıldığında, en iyi performansı göstermiş ve toplam seyahat mesafesinin en düşük olmasını sağlamıştır. Lin vd. (2006), yerel arama ile tavlama benzetim algoritmasını birleştirmişlerdir. Optimal çözümü hızlı ve verimli bir şekilde bulmak için takas ve ekleme yerel aramaları kullanılmıştır. Geliştirilen yaklaşım, ortalama araç sayısını ve çoğu sınıftaki rota maliyetleri önceki araştırmalarınkinden daha iyi veya ona eşit olarak bulmuştur. De Oliveira vd. (2006), monoton olmayan tavlama benzetim tekniğini rastgele yeniden başlatmalı (Çok Başlamalı) bir tepeye tırmanma stratejisine bağlayan verimli bir hibrid sistemi uygulamışlardır. Toplam mesafenin minimuma indirilmesi, temel amaç olarak ele alınmıştır. İleri sürülen bu melez sistem, toplam seyahat mesafesine odaklanan yerel arama, tabu arama gibi yöntemlerle karşılaştırılmış ve bu sistemin üstün olduğu ortaya çıkmıştır.

Woch ve Lebkowski (2009), sıralı tavlama benzetimi algoritmasını kullanmışlardır. Ampirik kanıtlar, benzetilmiş tavlamanın iki kriterli optimizasyon problemlerine başarıyla uygulanabileceğini göstermiştir. De Oliveira ve Vasconcelos (2010), zaman pencereli araç rotalama probleminde toplam mesafenin en aza indirilmesi için tepe tırmanma ve rastgele yeniden başlama ile tavlama benzetiminin birleşiminden oluşan bir algoritma geliştirmişlerdir. Hibrit çözüm hakkında daha derin bir araştırma yapmak için, metoda dâhil olan farklı parametrelerin davranışını incelemek için istatistiksel bir analiz (varyans analizi ve lineer regresyon) uygulanmıştır. Taner vd. (2012), tavlama benzetimi ve yinelemeli yerel arama olmak üzere iki meta-sezgisel yöntemi geliştirmişlerdir. Algoritmalar, literatürde en iyi bilinen yöntemlerle karşılaştırılmış ve yinelemeli yerel arama algoritması optimizasyon sırasında bulunan en iyi çözümlerin etrafında daha yoğun çözümler aramasından dolayı tavlama benzetimi algoritmasından daha iyi genel sonuçlar vermiştir.

Moghaddam vd. (2011), yeni bir matematiksel modeli geliştirmişler ve problemin çözümü için benzetimli tavlama yöntemini kullanmışlardır. Mahmudy (2014), çelik veya cam gibi bir malzemenin ısıtıldığı ve daha sonra soğutulduğu tavlamanın fiziksel işleminden ilham alınarak oluşturulan tavlama benzetimi algoritmasını geliştirmişlerdir. Komşuluk çözümlerinin etkili bir şekilde araştırılması için özel

fonksiyonlar oluşturulmuştur. Önerilen yaklaşım, literatürde yer alan iyi bilinen kıyaslama problemleri ile karşılaştırılarak değerlendirilmiştir.

Yerel Arama Algoritması

Yerel Arama meta-sezgiselleri, son zamanlarda çok sayıda kombinatoryal sorun için çok etkili olduğu kanıtlanan kombinatoryal arama ve optimizasyon problemleriyle mücadelede ortaya çıkan bir yöntem sınıfıdır. Yerel Arama teknikleri, bir çözüm alanının yinelemeli araştırmasına dayanır: Her yinelemede, bir Yerel Arama algoritması bir çözümden "komşularından" birine, yani (bir anlamda) başlangıçtaki yakın olan çözümlere adım atar. Bu teknik ailesinin en büyük dezavantajı, çok çeşitli sorunlu durumlarda sağlamlık eksikliğidir. Aslında, çoğu durumda, bu yöntemler makul çalışma sürelerinde iyi sonuçlar elde etmeyi sağlarken, diğer durumlarda Yerel Arama teknikleri sözde yerel minimumda tutulur. Son zamanlarda literatürde bu sorunun çözümüne yönelik birkaç yaklaşım ortaya çıkmıştır. Bu yaklaşımlar, istatistiksel özelliklerin kullanılmasından (örneğin, çözüm alanının rastgele keşfedilmesinden), öğrenme yöntemlerinin veya melez tekniklerin uygulanmasına kadar uzanmaktadır (Di Gaspero, 2003: iv). Yapay zekâ uygulamaları, yöneylem araştırması, mühendislik ve bioenformatik gibi alanlarda geniş bir uygulamaya sahip olan yerel arama yaklaşımı aynı zamanda araç rotalama problemlerinde de yaygınca kullanılmaktadır (Şahin, 2014: 76).

Polacek vd. (2004), bir dizi komşu yapıları tanımlamak için çapraz değişim ve i-çapraz değişim operatörlerini kullanarak değişken komşuluk arama felsefesine dayanan bir algoritma tasarlamışlardır. Lin vd. (2006), yerel arama ile tavlama benzetim algoritmasını birleştirerek, optimal çözümü hızlı ve verimli bir şekilde bulmak için takas ve ekleme yerel aramaları kullanılmışlardır. Hashimoto vd. (2008), araçların rotalarını belirlemek için 2-opt, or-opt ve çapraz değişim olmak üzere standart komşuluklarda ufak değişiklikler yaparak bir yerel arama algoritmasını kullanmışlardır. Taner vd. (2012), tavlama benzetimi ve yinelemeli yerel arama olmak üzere iki meta-sezgisel yöntemi ele almışlardır. Dhahri vd. (2014), tamsayılı doğrusal programlama ve genel değişken komşuluk aramaya ilişkin melez bir meta-sezgisel geliştirmişlerdir. Miranda ve Conceiçao (2016), yinelenen yerel aramaya dayanan meta-sezgiselini, en düşük beklenen maliyetle en iyi rotayı bulmak ve belirli hizmet düzeylerinin karşılandığını garanti etmek için tasarlamışlardır.

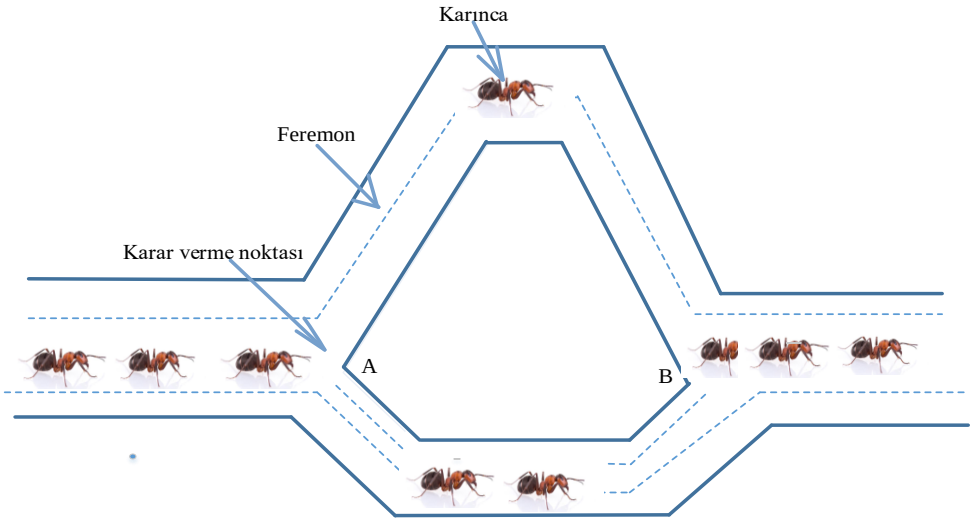
Karınca Kolonisi Algoritması

Çözülmesi zor kombinatoryal problemlerin çözümü için kullanılan meta-sezgisel tekniklerden birisi de karınca kolonisi algoritmasıdır. Karınca kolonisi algoritması, karınca kolonisinin, karınca yuvasından yiyecek kaynağına en kısa yolu bulma sürecindeki davranışını taklit eder. S. D. Shtovba'nın tanımına göre koloninin zeki davranışı sayesinde mümkün olan temel ilke, elementler arasındaki düşük seviyeli etkileşimin bir sonucu olarak küresel hedeflere ulaşmak için dinamik bir mekanizma kümesi olarak öz-örgütlenme ilkesidir. Bu etkileşim, herhangi bir merkezi etki ortadan kalkarken sistem unsurlarının yalnızca yerel bilgileri kullanması anlamına gelir. Bu gibi durumlarda, sürü zekası, farklı türden bir işbirlikçi davranış olarak kabul edilir. Bir karınca kolonisi, aslında, ajanları temel kurallara göre etkileşime giren çok ajanlı bir sistemdir. Bu nedenle, ajanların ilkel davranışlarına rağmen, sistemin davranışı son derece makul ve optimal şartlara yakındır.

Karınca kolonisi algoritmaları, gerçek karıncaların davranışlarına benzer yapay karıncaların kullanımına dayanır. Doğada karıncalar arasındaki etkileşim, doğrudan ve dolaylı bilgi aktarma yöntemleri kullanılarak gerçekleştirilir. Doğrudan yol, bir yiyeceğin yanı sıra görsel ve kimyasal temasların paylaşılmasıdır. Dolaylı yöntem, bazı kimyasal maddelerin (feromon), belirli bir alanda hapsolmuş karıncalar tarafından kullanılmasına dayanır; bu, diğer karıncaların hareketinden sonra bir

iz olarak kalır. Bu tür bir etkileşim zamanla dağılır ve stigmergy olarak adlandırılır. Stigmergy aşağıdaki gibi çalışır.

Bir karınca yiyecek ararken rastgele hareket eder ve sabit miktarda feromon bırakır. Başka bir karınca bu iz ile karşılaştığında ilerleyip ilerlememeye karar vermelidir. Eğer onunla birlikte hareket etmeye karar verirse, kendi feromonu ile mevcut izi güçlendirir, bu da bir sonraki karıncaların bu yolu seçmesi ihtimalini arttırır. Böylece bu yolda ne kadar karınca hareket ederse, o yol diğer karıncalar için o kadar çekici olur. Ek olarak, daha kısa bir rota kullanan karınca hızla karınca yuvasına geri döner ve feromonunu iki kez bırakır. Dolayısıyla feromon kısa yollarda birikir. Ayrıca, feromon zamanla buharlaşarak daha az arzu edilen yolların tespit edilmesini zorlaştırır ve dolayısıyla kullanımlarını azaltır. Ama yine de, rastgele yol seçimi karınca kolonisinin alternatif yolları tanımlamasına izin verir ve rotayı kesen engellerin başarılı bir şekilde atlanmasını sağlar. Bu, koloni davranışının adapte olabilirliği için koşullar yaratır (Zhikharevich vd., 2016: 86). Karıncaların bu yolculukları esnasındaki karar verme süreci Şekil 6'da ifade edilmiştir.



Şekil 6: Karıncaların Feromonlara Göre Yolculuklarını Seçmelerine Karar Verme Süreci
Kaynak: (Ostfeld, 2014: 4)

Karıncalar karar verme noktası olan A noktasında karşılaştıklarında bazıları bir tarafı bazıları ise diğer tarafı rastgele seçerler. Bu karıncaların aynı hızda gittiğini varsayalım, kısa kenarı seçenler karar verme noktasına B'ye uzun kenarı seçenlerden daha hızlı ulaşırlar. Tesadüfen kısa tarafı seçen karıncalar yuvaya ilk ulaşanlardır. Bu nedenle kısa taraf, uzun olandan daha önce feromon alır ve bu gerçek, karıncaların kısa yolu seçme olasılığını artırır. Sonuç olarak, feromon miktarı kısa tarafta uzun taraftan daha yüksek hızda bırakılır, çünkü karıncalar kısa tarafı uzun taraftan daha fazla seçerler. Şekil 6'daki kesik çizgi sayısı, yaklaşık olarak karınca sayısıyla doğrudan orantılıdır. Yapay karınca kolonisi sistemi de, çeşitli optimizasyon problemlerini çözmek için bu karınca kolonisi sistemi ilkesinden yapılır. Çünkü feremon karıncaların karar vermesinde bir kilit noktadır (Ostfeld, 2011: 4).

Karınca kolonisi algoritması ilk olarak gezgin satıcı problemine uygulanması için tanıtılmış ve o zamandan beri birçok ayrık optimizasyon problemi için uygulanmıştır. Atama problemleri, grafik renklendirme, maksimum klik problemi, çizelgeleme ve dahası araç rotalama problemleri gibi klasik problemlere de uygulanmaktadır (Katiyar vd., 2015: 4). Karınca kolonisi algoritmasını çözüm yöntemi olarak kullanan ZPARP ile ilgili çalışmalar aşağıda ifade edilmiştir.

Gambardella vd. (1999), çoklu amaç fonksiyonunu art arda optimize etmek için bir yapay karınca kolonileri hiyerarşisini tasarlamışlardır. Kuo vd. (2004), zaman pencereli araç rotalama problemi için yöntem olarak karınca kolonisi optimizasyon algoritmasını ve zaman değişkenlerini bulanık

değişkenlere dönüştürmek ve en kısa araç rotasını aramak için bulanık küme teorisini kullanmışlardır. Tan vd. (2005), karınca kolonisi algoritmasının temel işleyişini, iki karınca kolonisine talimat verecek şekilde iki adımda inşa etmişlerdir.

Chen ve Ting (2005), geliştirilmiş karınca koloni algoritması ile tavlama benzetimi algoritmasını bir araya getirerek melez bir algoritma oluşturmuşlardır. Geliştirilmiş karınca koloni sisteminde 2-opt ve ekleme olmak üzere iki farklı yerel arama operatörü kullanılmıştır. El Hassani vd. (2008), zaman pencereli araç rotalama problemi için tavlama benzetimi ile karınca koloni algoritmasının birleştirilmesiyle oluşturulan melez algoritmanın etkin olduğunu göstermeyi amaçlamışlardır. Yu ve Yang (2011), karınca koloni algoritmasında farklı günlerde sezgisel bilgi toplamak için çok boyutlu feromon matrisi ve algoritmanın performansını artırmak için ise iki çaprazlama operasyonu kullanılmıştır. Yeni turların yerel olarak optimize edilmesini sağlamak için 2-opt algoritmasından yararlanılmıştır.

Balseiro vd. (2011), problem için ekleme sezgiselleri ile hibritleştirilmiş karınca koloni sistemi algoritmasını sunmuşlardır. Ortaya çıkan algoritma, birkaç kıyaslama probleminde bilinen en iyi sonuçları eşleştiren veya geliştiren rekabetçi bir sonuç vermiştir. Yu vd. (2011), tabu arama algoritması ve karınca koloni algoritmasından oluşan melez bir yaklaşımı ele almışlardır. Karınca koloni algoritmasında kullanılan komşu arama, tabu arama işleminde de komşu çözümleri seçmek için kullanılmıştır. Ding vd. (2012), karınca koloni algoritmasının performansını artırmak için tasarruf algoritması ve λ-değişim mekanizmasından yararlanmışlardır. Ekleme sezgiseli ile melezleştirilmiş karınca koloni algoritması, yapay zeka sezgiselleri, geliştirilmiş genetik algoritma, tavlama benzetimi, tabu arama gibi yöntemlerle karşılaştırıldığında, algoritmanın rekabet edilebilir sonuçlar doğurduğu ortaya çıkmıştır. Gupta ve Diwaker (2017), karınca koloni optimizasyonu ve genetik algoritmanın birleşiminden oluşan yöntemi kullanmışlardır. Yöntemde, popülasyon çeşitliliğini arttırmak ve çözüm alanını tamamen genişletebilmek için genetik algoritmadaki mutasyon operatörleri yer almıştır.

Yapay Arı Kolonisi Algoritması

Yapay arı kolonisi algoritması, 2005 yılında Karaboğa'nın küresel nümerik fonksiyon optimizasyonu için geliştirdiği, bal arısı sürüsünün yiyecek ararkenki davranışını simüle eden, popülasyon bazlı bir meta-sezgisel yaklaşımdır. Basitliği ve uygulama kolaylığı nedeniyle, yapay arı koloni algoritması sürekli ve ayrık optimizasyon problemlerini çözmek için yaygın olarak kullanılmaktadır (Neelima vd., 2016: 1684).

Yapay arı koloni algoritmasındaki amaç, nektar miktarı maksimum olan çiçek parçalarını bulmaktır (Hussain vd., 2020: 795). Bu algoritma yiyecek kaynakları olarak adlandırılan bireylerin/ çözümlerin bir popülasyonunu korur. Popülasyon, işçi, gözcü ve kâşif arılar olmak üzere üç yapay arı grubu tarafından geliştirilen SN yiyecek kaynaklarından oluşmaktadır. İşçi arılar grubunda her arı belirli bir yiyecek kaynağına karşılık gelir. Bu arı yiyecek kaynağının konumunu hafızaya alır. İşçi arılar en iyi yiyecek kaynağını bulabilmek için yiyecek kaynağının komşuluğunda arama yaparlar. Daha sonra yeni yiyecek kaynakları işçi arılar tarafından güncellenip, bu yeni yiyecek kaynakları ile ilgili bilgiyi kovandaki gözcü arılarla paylaşırlar. Gözcü arılar işçi arılardan farklı bir şekilde çalışır. Sömürme, gözcü arılar tarafından rulet tekerleği seçimi ile gerçekleşir. Yani her gözcü arı, işlenecek kaliteye göre en iyi yiyecek kaynağını olasılıkla seçmektedir. Daha iyi bir konum bulmak için seçilen yiyecek kaynağı, gözcü arılar tarafından geliştirilmektedir. Kâşif arı olarak adlandırılan yeni bir tür yapay arı, arama uzayını araştırmak için ara sıra gönderilir. Bir yiyecek kaynağı, işçi ve gözcü arılar tarafından belli sayıda denemeden (bir kontrol parametresi olan limit) sonra geliştirilemezse bu yiyecek kaynağı terkedilecek fakir bir yiyecek kaynağı olarak düşünülür ve kâşif arı tarafından rastgele üretilen yeni bir yiyecek kaynağı ile bu yiyecek kaynağı yer değiştirir. Yapay arı koloni algoritmasında durdurma kriteri karşılanıncaya kadar tekrarlamalı bir şekilde üç çeşit yapay arı grubu çözüm alanını aramak için gönderilir. Başlangıçta işçi arıların sayısı ve

gözcü arıların sayısı, yiyecek kaynaklarının sayısına yani SN popülasyon büyüklüğüne eşittir (Li ve Yang, 2016: 363). Bu algoritma kullanılarak zaman pencereli araç rotalama problemi için yapılan çok sayıda çalışma aşağıda belirtilmiştir.

Iqbal (2012), bal arılarının yiyecek ararkenki zeki davranışlarından esinlenen yeni ve etkili meta-sezgisellerden yapay arı koloni algoritmasını esnek zaman pencereli araç rotalama problemine uyarlamıştır. Yaklaşımın performansı, literatürde bilinen en iyi yöntemler ile karşılaştırılmış ve algoritmanın, makul bir süre içinde daha kaliteli çözümler elde ettiği ortaya çıkmıştır. Shi vd. (2012), global arama kapasitesini artırmak için kullanılan turnuva seçim stratejisi ile yapay arı koloni algoritmasının birleşiminden oluşan bir sezgiseli ileri sürmüşlerdir. Nikolic vd. (2013), problemi çözmek için arı koloni optimizasyon meta-sezgiselinden yararlanmışlardır. Başlangıç çözüm (başlangıç rotalar) için basit ekleme sezgiseli kullanılmıştır. Rotadaki ilk düğüm rastgele bir şekilde belirlenmiş, bundan sonra, diğer tüm düğümler ekleme maliyetine göre rotaya yerleştirilmiştir. Sonuçlardan, yöntemin test edilmiş tüm iyi bilinen referans örnekleri için yüksek kaliteli çözüm üretebileceği ortaya çıkmıştır.

Jawarneh ve Abdullah (2015), bal arılarının sosyal iletişim modellerini taklit eden popülasyon tabanlı bir algoritma olan arı kolonisi optimizasyonunu probleme uyarlamışlardır. Algoritmanın performansı parametrelerine bağlıdır, bu yüzden etkinliğini ve sağlamlığını artırmak için çevrimiçi (kendinden uyarlamalı) parametre ayarlama stratejisi kullanılmıştır. Başlangıç çözüm greedy sezgiseli kullanılarak üretilmiştir. Ayrıca, çözümün kalitesi ve çözümü bulmak için gereken hesaplama süresi açısından çok etkili olan sıralı ekleme sezgiseli kullanılmıştır. Uyarlanan algoritma, temel arı kolonisi optimizasyonu algoritması ile karşılaştırılmış ve bu algoritma, araç sayısı ve ortalama 31 sürüş mesafesi bakımından daha iyi performans göstermiştir. Alzaqebah vd. (2016), temel yapay arı koloni algoritmasının kâşif arı aşaması boyunca güçlü bir keşif gerçekleştirme yeteneğinden yoksun olmasından dolayı keşfedilen arıların listesinin azami deneme sayısını (limit) aşan çözümleri ezberleyebilmesi için terkedilmiş çözümlerin bir listesinin tanımlandığı bir geliştirilmiş yapay arı koloni algoritmasını önermişlerdir. Hesaplamalı sonuçlar, geliştirilen algoritmanın, orijinal algoritmadan daha iyi performans gösterdiğini ve literatürdeki en iyi bilinen sonuçlarla karşılaştırıldığında iyi çözümler ürettiğini göstermiştir.

Yu vd. (2016), Çin'in Dalian kentinde, zaman pencereli araç yönlendirme problemi olarak tanımlanabilecek gerçek bir batı tarzı yiyecek dağıtım problemini ele almışlardır. Problem için bir tamsayılı doğrusal model ve problemi çözmek için bir çaprazlama işlemi ve bir mutasyon işlemi ve uyarlanabilir bir strateji olarak adlandırılan yeni bir stratejiye sahip bir yapay arı koloni algoritması geliştirilmiştir. Çaprazlama operatörü yeni daha iyi bir yiyecek kaynağı üretmek için, mutasyon operatörü ise yiyecek kaynağının çeşitliliğini korumak için kullanılmıştır. Toplam uzaklık ve hesaplama zamanı açısından orijinal yapay arı koloni algoritması ile karşılaştırıldığında en iyi sonuçları elde etmiştir. Mao vd. (2016), belirsiz zamana bağlı esnek zaman pencereli araç rotalama probleminde hem ulaşım maliyetlerini (toplam seyahat mesafesi ve araç sayısı) hem de servis maliyetlerini (erken ve geç gelenler) dikkate alan yeni bir matematiksel model geliştirmişlerdir. Problemi çözmek için yapay arı koloni algoritmasının bir çeşidi kullanılmıştır. İşçi ve gözcü arı evresinde sömürü araştırmasında komşuluk yapısı olarak bir insert operatörü tasarlanmıştır. Kâşif arı evresinde ise yeni çözüm elde etmek için swap-reverse operatörü kullanılmıştır. Hesaplamalı sonuçlar, bu yaklaşımın uygulanabilirliğini göstermiştir.

Yao vd. (2017), yapay arı koloni algoritmasının performansını, çaprazlama işlemine ve tarama stratejisine dayanan yerel bir optimizasyonla geliştirmişlerdir. Klasik zaman pencereli araç rotalama problemi deneyinde en iyi bilinen çözüm ile yapılan karşılaştırma, algoritmanın yeteneğini doğrulamıştır. Worawattawechai (2017), yapay arı koloni algoritmasını geliştirerek zaman pencereli geri dönüş yüklemeli araç rotalama problemine uygulamışlardır. Başlangıç çözümü (yiyecek kaynağı), rulet tekerleği seçimli en yakın komşu metodu tarafından oluşturulmuştur. Ayrıca, algoritmada gözcü arılar için sıralı arama, λ-değişim tekniği ve 1-hareketli rota içi değişimin birleşiminden oluşan yasaklı bir liste

olan üç strateji önerilmiştir. Küçük ve orta büyüklükteki problemler için algoritmanın performansının daha iyi olduğu hesaplamalı sonuçlardan ortaya çıkmıştır. Alzaqebah vd. (2018), zaman pencereli araç rotalama problemi için arılar algoritmasının kullanımını ve algoritmanın güçlü ve zayıf yönlerini incelemişlerdir. Ele aldıkları arılar algoritmasının kaliteli çözümler ürettiği, literatürdeki en modern yaklaşımlarla karşılaştırılabilir çözümler üretebileceği yapılan testlerce anlaşılmıştır.

Tuntitippawan ve Asawarungsaengkul (2018), zaman pencereli geri dönüş yüklemeli araç rotalama problemi için λ-değişimi ve 2-opt yerel aramaları ile birleştirilen yapay arı koloni algoritmasını kullanmışlardır. Başlangıç çözümler, en yakın komşu sezgisel odaklı, rasgele ağırlıklı zaman tarafından üretilmiştir. Komşuluk arama mekanizmaları olarak λ-değişim ve 2-opt kullanılmıştır. Yerel aramalı yapay arı koloni algoritması, genetik algoritma, diferansiyel bir evrim yaklaşımı ve hibrit bir meta-sezgisel gibi yöntemlerle karşılaştırıldığında algoritmanın performansının daha iyi olduğu ortaya çıkmıştır. Chen ve Zhou (2018), üç çeşit komşuluk arama yöntemi kullanılmışlardır. Lider arı ve takipçi arı arama evresinde, tekli arama modunu algoritmanın optimizasyon derinliğini artıran üç yönlü bir arama yöntemine dönüştürülmüştür. Kâşif arı tarafından üretilen yeni gıda kaynakları için çok sayıda komşuluk araştırması yapılması ve bir sonraki tekrarlamaya devam edilmesi, yeni gıda kaynaklarının hayatta kalmasını ve popülasyonların çeşitliliğini arttırmıştır. Simülasyon deneylerinden, geliştirilmiş ayrık yapay arı koloni algoritmasının büyük ölçekli zaman pencereli araç rotalama problemlerini çözmede belirgin avantajlara sahip olduğu anlaşılmıştır. Kantawonq ve Pravesjit (2020), çalışmalarında bulanık teknik, dağınık arama yöntemi ve SD-tabanlı seçim yöntemini yapay arı kolonisi algoritması ile birleştirerek probleme uygulamışlardır. Karşılaştırmalı sonuçlardan algoritmanın, diğer algoritmalara kıyasla iyi sonuçlar verdiği sonucu elde edilmiştir.

Parçacık Sürü Optimizasyon Algoritması

Parçacık sürü optimizasyonu, Eberhart ve Kennedy tarafından 1995 yılında ortaya atılan sürüyü temel alan stokastik bir optimizasyon tekniğidir (Wang vd., 2018: 387). Parçacık sürü optimizasyonunda bir çözüm, bir parçacık olarak temsil edilir ve çözümlerin oluşturduğu populasyona parçacık sürüsü denir. Her parçacığın konum ve hız olmak üzere iki önemli özelliği vardır. Her parçacık, hızı kullanarak yeni bir konuma hareket eder. Yeni bir konuma ulaşıldığında, her parçacığın en iyi konumu ve sürünün en iyi konumu gerektiği şekilde güncellenir. Her parçacığın hızı daha sonra parçacığın deneyimlerine dayanarak ayarlanır. İşlem, bir durdurma kriteri karşılanana kadar tekrarlanır.

Her parçacık rastgele bir konum ve hız ile başlatılır. Sonra her parçacık uygunluk değeri açışından değerlendirilir. Bir uygunluk değeri her hesaplandığında, parçacığın önceki en iyi uygunluk değeri ve tüm sürünün önceki en iyi uygunluk değeri ile karşılaştırılır ve uygun olduğunda kişisel en iyi ve küresel en iyi pozisyonlar güncellenir. Bir durdurma kriteri karşılanmazsa, hız ve konum yeni bir sürü oluşturmak için güncellenir (Kachitvichyanukul, 2012: 217). 1995'ten bu yana parçacık sürü optimizasyon algoritması, küresel optimizasyon problemlerini çözmek için en umut verici optimizasyon tekniklerinden biri olarak ortaya çıkmıştır (Pant vd., 2009: 101). Küresel optimizasyon problemlerinden biri olan zaman pencereli araç rotalama probleminin çözümünde bu yöntemi ele alan çalışmalar mevcuttur.

Liu vd. (2009), melez bir parçacık sürü optimizasyon algoritmasını ileri sürmüşlerdir. Yazarlar, algoritmayı geliştirmek için hem genetik algoritmadaki çaprazlama operatörünü hem de seviye küme teorisini kullanmışlardır. Deneysel karşılaştırma sonuçları, algoritmanın performansının, parçacık sürü optimizasyonu, genetik algoritma ve paralel parçacık sürü optimizasyon yöntemlerinden daha üstün olduğunu ve ayrık birleştirme problemlerini çözmek için etkili bir yaklaşım olacağını göstermiştir. Ai ve Kachitvichyanukul (2009), parçacık sürü optimizasyon algoritmasının performansını değerlendirmek için hesaplama süresi ve çözüm kalitesi olmak üzere iki kriter dahil etmişlerdir. Hesaplamalı deneylerden ileri sürülen algoritmanın zaman pencereli araç rotalama problemini çözmek için etkili

sonuçlar vereceğine ulaşılmıştır. Marinakis vd. (2019), ileri sürdükleri çoklu adaptif parçacık sürü optimizasyon algoritmasında; aç gözlü rastgele adaptif arama, adaptif kombinaoryal komşuluk topolojisi ve parçacık sürü optimizasyon algoritması olmak üzere 3 farklı adaptif strateji kullanmışlardır.

Ateş Böceği Algoritması

Ateşböceği algoritması, neredeyse optimizasyon ve mühendislik problemleri için hızlı gelişen evrimsel zekalardan biridir. Ateşböceği, bir biyolüminesans işlemi tarafından üretilen çoğunlukla kısa ve ritmik flaşlar üreten bir böcektir. Yanıp sönen ışığın işlevi, ortakları (iletişimi) çekmek ya da potansiyel avı ve avcıya karşı koruyucu bir uyarı almaktır. Dolayısıyla ışığın bu yoğunluğu ateşböceklerinin diğer ateşböceklerine doğru hareket etmesinin faktörüdür. Işık yoğunluğu, seyircinin gözlerinden uzakta değişmektedir. Uzaklık arttıkça ışık yoğunluğu azalmaktadır. Işık yoğunluğu aynı zamanda havanın çevreden etkilenmesinin etkisi, böylece uzaklık arttıkça yoğunluk daha az çekici hale gelir. Algoritmanın üç idealize kuralı aşağıdaki gibidir (Ali vd., 2014: 1732):

i. Ateşböcekleri cinsiyetten bağımsız olarak birbirlerine doğru çekilir.

ii. Ateşböceklerinin çekiciliği, ateşböceklerinin parlaklığı ile ilişkilidir, dolayısıyla daha az çekici ateşböcekleri daha çekici ateşböceklerine doğru ilerleyecektir.

iii. Ateşböceklerinin parlaklığı objektif fonksiyonuna bağlıdır.

2007 yılında Cambridge Üniversitesinde Dr. Xin-She Yang tarafından geliştirilen bu yöntem, parçacık sürü optimizasyonu, yapay arı koloni optimizasyonu ve bakteriyel yiyecek arama algoritmaları gibi sürü zekasına dayanan diğer algoritmalarla birçok benzerliğe sahip olsa da, aslında hem kavram hem uygulama açısından çok daha basittir. Ayrıca, çok verimlidir ve birçok optimizasyon problemini çözmek için genetik algoritmalar gibi diğer geleneksel algoritmalardan daha iyi performans gösterebilmektedir (Apostolopoulos ve Vlachos, 2011: 8-9). Örneğin; Pan vd. (2013), ateş böceği algoritmasının temel prensiblerini ve algoritma sürecini ayrıntılı bir şekilde inceleyerek, algoritmanın işleyişini ve çözüm basamaklarını zaman pencereli araç rotalama problemi için tasarlamışlardır. Kıyaslama testinden ve literatürdeki diğer test örneklerinden, zaman pencereli araç rotalama için ateş böceği algoritmasının geçerliliğini kanıtlayan iyi çıktılar elde edilmiştir. Aggarwal ve Kumar (2018), geliştirmiş oldukları ateş böceği algoritmasında iki ateş böceği arasındaki uzaklığı Chebyshev metodunu kullanarak hesaplamışlardır. İleri sürmüş oldukları bu metotla problem için en yakın sonuçları elde etmişlerdir.

Yarasa Algoritması

Yarasa algoritması, yeni sürü zeka temelli bir meta-sezgisel algoritmadır. Yang tarafından 2010 yılında geliştirilmiştir. Mikro yarasaların yüksek sesle ve değişen titreşim emisyonu ile yiyecek ararkenki davranışlarından esinlenilmiştir (Kongkaew, 2017: 642). Yarasadan ilham alan bu yaklaşım, optimum aramada mikro yarasaların gelişmiş gırtlak yankılarını taklit eder. Diğer yerel arama algoritmalarına kıyasla yarasa algoritması en belirgin özelliklere sahiptir:

- Çözüm çeşitliliğini genişletmek için bir frekans ayarlama stratejisi kullanılır;
- Arama hareketlerinde arama ve sömürme faktörleri güçlü bir şekilde ilişkili olduğundan bu iki faktör arasındaki dengeyi sağlamak için otomatik zoom kullanılır.

Yarasa algoritmasının bugüne kadar yapılan birkaç son derece başarılı uygulamaları vardır (Wang vd., 2018: 117). Bu uygulamalar problem özelliklerine göre; çizelgeleme, tahsis, tesis düzeni tasarımı, rotalama ve çoklu problem kombinasyonu olmak üzere beş gruba ayrılabilir (Kongkaew, 2017: 644). Bunlar arasında araç rotalama probleminin bir uzantısı olan zaman pencereli araç rotalama problemine yönelik yapılan üç çalışma aşağıda belirtilmiştir.

Taha vd. (2017), ZPARP için ileri sürdükleri geniş komşuluk aramalı yarasa algoritmasıyla, geniş komşuluk aramanın yok etme ve onarma paradigmasını kullanarak ayrık yarasa algoritmasının performansını arttırmış ve yarasanın çözüm alanının büyük bir bölümünü keşfetmesini sağlamışlardır. Osaba vd. (2018), yarasa algoritmasını rastgele yeniden yerleştirme operatörlerini kullanarak ZPARP'ne uyarlamışlardır. Geliştirmiş oldukları bu yöntemle, etkili ve elverişli sonuçlar elde edilmiştir. Pratiwi vd. (2018), yarasa algoritmasını tavlama benzetimi ile melezleştirmişlerdir. Toplam uzaklığın minimum olmasında ileri sürmüş oldukları algoritmanın elverişli olduğu yapılan testler sonucunda ortaya çıkmıştır.

Harmoni Arama Algoritması

Harmoni Arama yöntemi, müzisyenlerin harmoni doğaçlamalarının temel ilkelerinden ilham alınarak oluşturulan bu yöntem Geem vd. (2001) tarafından önerilmiştir. Algoritma basit olmakla beraber, arama verimliliğinin ayırt edici özelliklerine sahiptir. Harmoni arama algoritması, başlangıç popülasyonunu harmoni vektörlerinden rastgele olacak şekilde oluşturarak harmoni hafızasında depo eder. Hafıza çözümleri, ayar düzeltmesi ve rastgele seçim gibi yöntemler kullanılarak harmoni hafızasındaki çözümlerden yeni harmoni oluşturulur. Sonra, aday vektör, en kötü vektör ile güncelleme operatörü yardımıyla kıyaslanır ve aday vektör güncellenir. Bu süreç belirli bir iterasyon adedince tekrar edilir.

Basitliği ve uygulama kolaylığı nedeniyle son yıllarda, fonksiyon optimizasyonu, mekanik yapı tasarımı, boru ağı optimizasyonu ve veri sınıflandırma sistemlerinin optimizasyonu gibi alanlarda başarıyla kullanılmış ve birçok optimizasyon problemleri için dikkat çekici bir algoritma haline gelmiştir.(Akkoyunlu ve Engin, 2011: 142; Gao vd., 2015: 1). Bu optimizasyon problemlerinden olan zaman pencereli araç rotalama problemine de uygulanmış çalışmalar mevcuttur. Bunlardan Yassen vd. (2015), yerel arama algoritması ile melezleştirilen bir meta-harmoni algoritmasını ele almışlar ve problemin çözümünde kullanmışlardır. Yassen vd. (2017), uyarlanabilir bir melez harmoni arama algoritmasını tasarlamışlardır. Chen vd. (2017), değişken komşuluk arama sezgiseli ile harmoni arama algoritmasını birleştirerek dinamik zaman pencereli araç rotalama problemine uyarlamışlardır. Maleki vd. (2017), melez kendinden uyarlamalı küresel en iyi harmoni arama algoritmasını ele almışlardır. Sömürü kapasitesini arttırmak için taşıma, yer değiştirme, son müşteri değişimi, or-opt, 2-opt ve çapraz değişim operatörü olmak üzere 6 yerel arama komşuluk operatörlerini kullanılmıştır.

Kurbağa Sıçrama Algoritması

Kurbağa sıçrama algoritması, azami miktarda kullanılabilir gıdayı bulmak için yer bulma arayışındayken bir grup kurbağaların memetik evrimini taklit eden bir meta-sezgisel optimizasyon yöntemidir (Luo ve Chen, 2014: 2536). Kurbağa popülasyonu, topluluktaki kurbağaların tamamının bir araya gelmesiyle oluşur. Popülasyondaki kurbağaların her biri problem için olası bir çözüme karşılık gelir. Her kurbağanın uygunluk değeri algoritmada ifade edilen kısıt ve değişkenlere göre belirlenir. Kurbağa popülasyonunun rastgele oluşturulmasıyla algoritma başlar. Rastgele oluşturulan popülasyonun uygunluk değerleri hesaplanır ve sıralı bir şekilde gruplara ayrılır. Her bir grup birbirinden bağımsız bir şekilde memetik evrime belirli bir iterasyon adedince tabi tutulur. İyi değerlere sahip bireylerin etkisinin kötü bireylerden daha fazla olması bu evrimde amaçlanır. Bu nedenle memetik evrim aşamasında seçme işlemi yapılmadan önce her kurbağaya bir şeçilme katsayısı verilir. Gruplar kendi içlerinde sonuçları bulur ve bu sonuçlar baz alınarak popülasyonun yeni ve farklı gruplara ayrılma işlemi gerçekleştirilir. Böylece kurbağaların sahip oldukları memetik bilgi, global seviyede paylaşılır ve en iyi çözüme yavaş yavaş yaklaşılmaktadır (Karakoyun, 2015: 24-25).

Luo ve Chen (2014), çok depolu zaman pencereli araç rotalama problemi ve çok depolu kapasiteli araç rotalama probleminin çözümünde çok aşamalı kurbağa sıçrama algoritmasını kullanmışlardır.

Deney sonuçları algoritma ile kısa sürede yüksek nitelikli sonuçların elde edildiğini ortaya çıkarmıştır. Kombinatoryal optimizasyon problemlerinden ZPARP'ne yönelik bu yöntemi kullanarak çözüm bulmayı amaçlayan bir diğer çalışma Luo vd., (2015) tarafından yapılmıştır. Problemi etkili bir şekilde ele almak için melez bir kurbağa sıçrama algoritmasından faydalanmışlardır. Çözümlerin kalitesini artırmak ve nüfusa daha fazla çeşitlilik getirmek için değiştirilen klon seçim prosedürü kullanılmıştır. İleri sürülen yaklaşım, daha önce kullanılan algoritmalarla karşılaştırılmış ve algoritmanın hem hesaplama verimliliği hem de çözüm kalitesi açısından etkili performansı gösterilmiştir.

Memetik Algoritma

Memetik algoritmalar, popülasyon tabanlı arama (evrimsel tekniklerde olduğu gibi) ve yerel arama (tepe-tırmanma tekniklerinde olduğu gibi) gibi farklı algoritmik çözücülerden alınan fikirlerin sinerjistik kombinasyonuna dayanan optimizasyon teknikleridir. "Memetik algoritmalar" ın (MA) genel değeri, geniş bir meta-sezgisel sınıfını (yani, altta yatan bir sezgisel yöntemi yönlendirmeyi amaçlayan genel amaçlı yöntemler) kapsayacak şekilde kullanılır. Metot bir ajan popülasyonuna dayanır ve çeşitli problem alanlarında ve özellikle de NP-zor optimizasyon problemlerinin yaklaşık çözümü için pratik başarısı olduğu kanıtlanmıştır. Geleneksel evrimsel hesaplama yöntemlerinden farklı olarak, Memetik Algoritma'lar çalışılan problemle ilgili mevcut tüm bilgileri kullanmaktan doğal olarak sorumludur. MA'ların başarısı muhtemelen dâhil ettikleri farklı arama yaklaşımlarının sinerjisinin doğrudan bir sonucu olarak açıklanabilir (Moscato ve Cotta, 2010: 141-142).

Nagata vd. (2010), etkili bir memetik algoritmayı ileri sürmüşlerdir. Burada, çapraz kenar montajı, zaman penceresi kısıtlaması ile başa çıkmak için özellikler dahil edilerek zaman pencereli araç rota problemine adapte edilmiştir. Komşuluk operatörleri olarak 2-opt, or-değişim, yer değiştirme ve değişim operatörlerinin alt komşuluklar için farklı versiyonları kullanılmıştır. Bu yaklaşımın daha etkili olduğu yapılan testler sonucunda anlaşılmıştır. Nalepa ve Blocho (2016), coğrafi olarak dağınık bir dizi müşteriye hizmet vermek için rota planında harcanan toplam mesafeyi en aza indirme amacına sahip problemin çözümü için uyarlanabilir bir memetik algoritmayı sunmuşlardır. Çözüm alanının keşfedilmesini ve kullanılmasını dengelemek için yeni bir uyarlanabilir seçim planı önerilmiştir. Karşılaştırmalı sonuçlardan algoritmanın çok rekabetçi olduğu anlaşılmıştır.

Bakteriyel Yiyecek Arama Algoritması

Bakteriler yiyecek arama programı, program yürütme ilerledikçe ve gittikçe daha iyi bir uygunluğa (daha az maliyet işlevi) yol açtıkça, programın her yinelenen adımından sonra maliyet fonksiyonunu tahmin eden evrimsel bir algoritmadır. Optimize edilecek parametreler bakterilerin koordinatlarını (konumunu) temsil eder. Parametreler istenen aralıkta ayrıştırılır, bu ayrık değerlerin her biri uzay koordinatlarında bir noktayı temsil eder. Sonra her noktada bir bakteri konumlandırılır (yaratılır). Her aşamalı adımdan sonra bakteriler yeni pozisyonlara hareket eder (yeni koordinat değerleri) ve her pozisyonda maliyet fonksiyonu hesaplanır ve daha sonra bu hesaplanan maliyet fonksiyonu değeri ile, bakterilerin daha fazla hareketine maliyet fonksiyonunun yönünün azaltılmasıyla karar verilir. Bu nihayet bakterileri en yüksek uygunlukta bir pozisyona (optimizasyon parametreleri kümesi) götürür (Sharma vd., 2012: 9).

Niu vd. (2012), doğrusal olmayan azalan bir üstel modülasyon modeli ile geliştirilmiş bakteriyel yiyecek arama algoritmasını kullanmışlardır. Algoritmanın üstünlüğü diğer bakteriyel yiyecek arama algoritmaları ile karşılaştırıldığında ortaya çıkmıştır. Tan vd. (2015), zamanla değişen kemotaksis adım uzunluğu ve adaptif kapsamlı öğrenme bakteriyel yem arama optimizasyonu olarak adlandırılan kapsamlı öğrenme stratejisi ile birlikte bakteriyel yem arama optimizasyonu algoritmasının bir varyantını önermişlerdir. Algoritmanın araştırılması ve kullanılması arasında iyi bir denge sağlamak için

uyarlamalı, doğrusal olmayan bir şekilde azalan modülasyon modeli kullanılmıştır. Yöntem, çoklu mod problemlerini çözmede önemli ölçüde daha iyi performans göstermiştir.

Evrimsel Algoritma

Evrim, çevreye uyum sağlama ve genetik bilgilerin sonraki nesillere aktarılması olayıdır. Darwin (1859) doğal evrimi yönlendiren üreme, doğal seçilim ve bireylerin çeşitliliği olmak üzere üç temel ilke belirlemiştir. Doğal evrimin bu özellikleri, biyolojik evrimi ve doğal seçilimi taklit eden geniş bir evrimsel algoritma sınıfına giriş bulmuştur. Evrimsel algoritma (EA) terimi, doğal evrim sürecini simüle eden bir stokastik optimizasyon yöntemleri sınıfını ifade eder. Genel olarak, bir Evrimsel Algoritma (EA), bir seçim sürecinden geçen ve genetik operatörler tarafından manipüle edilen bir dizi çözüm adayının muhafaza edilmesiyle karakterize edilir. Doğal evrime benzer şekilde, çözüm adaylarına birey, çözüm adayları kümesine de popülasyon adı verilir. Her birey, problemdeki olası bir çözümü temsil eder. Bununla birlikte, bir birey bir karar vektörü değildir, bunun yerine optimizasyon probleminin çözümünü uygun bir yapıya, örneğin gerçek değerli bir vektöre dayalı bir karar vektörüne kodlar. Kodlanmış çözümü (kromozom) tutan veri yapısının her bir alt bölümüne gen denir ve genellikle tek bir parametrenin değerini kodlar. Seçim, adayların (ebeveynlerin) uygunluk değerlerine göre yeniden birleştirme için seçildiği bir süreçtir. Burada uygunluk, çözüm uzayını keşfederken maksimize edilecek kâr, fayda veya iyilik ölçüsünü ifade eder. Rekombinasyon (veya çaprazlama) ve mutasyon, mevcut olanlardan arama uzayında yeni çözümler üretmeyi amaçlayan genetik operatörlerdir. Çaprazlama operatörü, belirli sayıda çocuk (yavru) oluşturmak için belirli sayıda ebeveynden gelen bilgileri birleştirir. Buna karşılık, mutasyon operatörü, belirli bir olasılığa (mutasyon hızı) göre ilişkili karar vektörlerindeki (tipik olarak) küçük parçaları rastgele değiştirerek bireyleri değiştirir.

Yukarıdaki kavramlara dayalı olarak, doğal evrim yinelemeli bir hesaplama süreci ile simüle edilir. Başlangıçta, eldeki bir problem için bir aday çözüm popülasyonu oluşturulur. Bu genellikle çözüm uzayından rastgele örnekleme ile gerçekleştirilir. Daha sonra ebeveyn değerlendirmesi (uygunluk ataması), seçim, rekombinasyon ve/veya mutasyondan oluşan bir döngü belirli sayıda yürütülür. Her döngü yinelemeye bir nesil denir ve bazı yakınsama kriterleri veya koşulları karşılandığında arama durdurulur. Bu tür kriterler, örneğin, maksimum nesil sayısına veya benzer bireylerden oluşan homojen bir popülasyona yakınsamaya karşılık gelir.

EA'ların kökenleri 1950'lerin sonlarına kadar uzanabilir ve 1970'lerden bu yana başta genetik algoritmalar, evrimsel programlama ve evrim stratejileri olmak üzere çeşitli evrimsel metodolojiler önerilmiştir. Tüm bu yaklaşımlar bir dizi aday çözüm üzerinde çalışır. Ayrıca, EA'lar çok amaçlı optimizasyon için özellikle uygun görünmektedir, çünkü tek bir simülasyon çalışmasında birden çok pareto-optimal çözümü yakalayabilmektedirler ve rekombinasyonla çözümlerin benzerliklerinden faydalanabilmektedirler.

Çok amaçlı optimizasyonda evrimsel algoritmaların (EA) uygulanması, çeşitli geçmişlere sahip araştırmacılar tarafından artan ilgi görmektedir (Bhargava, 2013: 31; Braysy vd., 2004: 5-6). Bunlar arasında Tan vd. (2006), ZPARP için özel genetik operatörler ve değişken uzunluklu kromozom temsili ile melezleştirilen çok amaçlı evrimsel algoritmayı ele almışlardır. Bu yaklaşımın literatürde yer alan mevcut en iyi yaklaşımlardan daha iyi sonuç verdiği yapılan testlerce anlaşılmıştır. Creput vd. (2007), evrimsel yaklaşımda kendi kendini organize eden haritalarla probleme çözüm bulmaya çalışmışlardır. Najera ve Bullinaria (2011), çok amaçlı evrimsel algoritmayı ZPARP'nin çözümü için geliştirmişlerdir. Elde edilen sonuçlar geliştirilen bu yaklaşımın iyi bilinen evrimsel algoritmalardan daha iyi olduğunu ortaya çıkarmıştır.

Yusufçuk Algoritması

Yusufçuk algoritması, 2016 yılında Griffith Üniversitesi'nde Mirjalili tarafından geliştirilmiştir. Sürü zekâsına dayanan bir meta-sezgisel algoritma olan bu teknik, doğadaki yusufçukların statik ve dinamik davranışlarından esinlenmiştir. Arama ve sömürme, optimizasyonun iki ana aşamasıdır. Bu iki aşama, dinamik olarak veya statik olarak yiyecek arayan veya düşmandan kaçınan yusufçuklar tarafından modellenmiştir. Besleme, optimizasyonda statik bir sürü olarak modellenir; göç dinamik bir sürü olarak modellenmiştir. Craig ve Hart'a göre, sürülerin üç özel davranışı vardır: ayırma, hizalama ve uyum. Burada, ayırma kavramı, sürülerdeki bir bireyin komşusu ile statik çarpışmayı önlediği anlamına gelir. Hizalama, ajanların komşu bireylerle eşleşme hızını ifade eder. Son olarak, uyum kavramı bireylerin sürünün merkezine doğru eğilimlerini gösterir. Yusufçuk algoritmasındaki bu üç temel davranışa iki ek davranış eklenir: yiyeceğe doğru hareket etmek ve düşmandan kaçınmak. Bu davranışları algoritmaya eklemenin nedeni, her sürünün temel amacının hayatta kalmasıdır. Bu nedenle, tüm bireyler gıda kaynaklarına doğru ilerlerken, aynı zamanda düşmandan kaçınmalıdırlar (Acı ve Gülcan, 2019: 2-3).

En son sürü tabanlı algoritmalardan olan bu yaklaşım, makine öğrenimi, görüntü işleme, kablosuz ve ağ uygulamaları ve diğer bazı alanlarda birçok problemi optimize etmek için kullanılmıştır (Rahman ve Rashid, 2019: 2-9). Bu optimizasyon problemlerinden olan ZPARP için yapılan çalışmadan biri Liu vd., (2019) tarafından yapılan çalışmadır. Yusufçuk algoritmasını probleme uyarlamışlardır. Bu uyarladıkları algoritmanın etkili olduğu yapılan simülasyon deneylerinden anlaşılmıştır. Gunawan (2020), yusufçuk algoritmasının çözüm kalitesi açısından fil sürüsü optimizasyonundan daha iyi sonuç verdiğini yaptığı çalışmada ispatlamıştır.

Guguk Kuşu Arama Algoritması

Guguk kuşu arama algoritması, guguk kuşlarının doğal davranışlarından ilham alan Yang ve Deb tarafından 2009 yılında geliştirilen sürü zeka temelli bir meta-sezgisel algoritmadır (Fister Jr. vd., 2013: 390). Guguk kuşları, kuluçka parazitleri (kuluçka asalaklığı) kuşlarıdır. Asla kendi yuvasını inşa etmez ve yumurtalarını başka bir konakçı kuşun yuvasına koyar. Guguk kuşları en iyi bilinen bir kuluçka parazitidir. Bazı ev sahibi kuşlar, davetsiz misafir guguk kuşu ile doğrudan bağlantı kurabilir. Ev sahibi kuş, yumurtaları olmayan yumurtaları tanımlarsa, o yumurtaları yuvalarından uzağa fırlatır veya yuvalarından kurtulur ve yeni bir yuva oluşturur. Bir yuvada, her yumurta bir çözümü temsil ederken, guguk kuşu yumurtası ise yeni ve iyi bir çözümü temsil eder. Elde edilen çözüm, var olanı ve bazı özelliklerin değişikliklerini temel alan yeni bir çözümdür. En basit formda, her yuvada birden fazla yumurta vardır ve bunların bir tanesi bir dizi çözümü temsil eden guguk kuşu yumurtasıdır (Pentapalli ve Varma P., 2016: 556). Guguk kuşu araması, aşağıda üç idealize edilmiş kural kullanılarak tanımlanabilir:

- Her guguk kuşu her seferinde bir yumurta yumurtlar ve yumurtasını rastgele seçilen yuvaya bırakır;
- Yüksek kalitede yumurta içeren en iyi yuvalar gelecek nesillere taşınır;
- Mevcut ev sahibi yuvalarının sayısı sabittir ve bir guguk kuşu tarafından yumurtlanan yumurta, ev sahibinin doğurma olasılığı $pa \in [0,1]$ ile bulunur.

Her guguk kuşu rastgele seçilen bir yuvaya ayrılan bir zamanda tek bir yumurta bırakır. Mükemmel kalitede yumurtaları olan optimum yuva gelecek nesillere taşınır. Ev sahibi yuvalarının sayısı sabittir ve bir ev sahibi, varlığı ya yumurtadan atılmasına ya da ev sahibi kuş tarafından yuvayı terk etmesine sebeb olan $pa \in [0,1]$ olasılıklı yabancı bir yumurta bulabilir. Bir yuvadaki her bir yumurta bir çözümü ve bir guguk kuşu yumurtası ise yeni bir çözümü temsil eder. Buradaki amaç en zayıf uygunluğa sahip çözüm ile yeni çözümü değiştirmektir (Dash ve Mohanty, 2014: 3541-3542).

Çizelgeleme problemi, yapı mühendisliğinde tasarım optimizasyon problemleri ve global optimizasyon problemleri gibi birçok uygulamada guguk kuşu arama algoritması kullanılmaktadır

(Pentapalli ve Varma P., 2016: 556). Ayrıca ZPARP gibi dağıtım problemlerinde de bu yaklaşımın kullanıldığı Lei vd. (2018) tarafından yapılan bir çalışmada ispatlanmıştır. Çalışmalarında zaman pencereli lojistik araçların yol planlamasında guguk kuşu arama algoritmasından faydalanmışlardır. Simülasyonla yapılan deneyler sonucunda bu yöntemin bu tarz problemler için etkili olabileceği sonucuna varılmıştır. Bir diğer çalışma ise Rezaeipanah vd. (2019) tarafından yapılmıştır. Çalışmalarında problemin çözümü için açgözlü algoritma ile melezleştirilmiş guguk kuşu arama algoritmasını kullanmışlardır. Karşılaştırmalı sonuçlardan algoritmanın üstün performans sergilediği ortaya çıkmıştır.

ZPARP'nin çözümünde kesin, sezgisel ve meta-sezgisel yöntemlerin kullanıldığı çalışmalar yayın yıllarına göre Tablo 2'de listelenmiştir.

Tablo 2: ZPARP'nin Çözümünde Kesin, Sezgisel ve Meta-Sezgisel Yöntem Kullanılan Çalışmalar

		KULLANILAN YÖNTEM															
Yazar(lar)	**Yıl**	**Düzlem K.**	**Dal K.**	**Hedef P.**	**Küme B.**	**Dal S.**	**Sütun Ü.**	**Dinamik P.**	**Doğrusal P.**								
Cook ve Rich	1999	✓															
Brad vd.	2002		✓														
Aydemir	2006			✓													
Calvete vd.	2007			✓													
Alvarenga vd.	2007				✓												
Tezer	2009				✓	✓											
Qureshi vd.	2009						✓										
Kok vd.	2010							✓									
Ghoseiri ve Ghannadpour	2010			✓													
Qureshi vd.	2010						✓										
Liberatore vd.	2011						✓										
Akca	2015								✓								
Yazarlar	**Yıl**	**Tasarruf**	**En Yakın Komşu**	**Ekleme**	**Süpürme**												
Landeghem	1988	✓															
Backer ve Schaffer	1989	✓															
Demircioğlu	2009	✓															
Jawarneh ve Abdullah	2015			✓													
Göçken vd.	2018		✓		✓												
Hertrich vd.	2019				✓												
Yazarlar	**Yıl**	**GA**	**TA**	**TB**	**YAA**	**KK**	**YAKA**	**PSO**	**ABA**	**YA**	**HAA**	**KSA**	**MA**	**BYA**	**EA**	**YUA**	**GKA**
Chiang ve Russell	1996			✓													
Badeau vd.	1997		✓														
Schulze ve Fahle	1999		✓														
Gambardella vd.	1999					✓											
Braysy	2001	✓															
Braysy ve Gendreau	2002		✓														

Tablo 2'nin Devamı

Yazarlar	Yıl	GA	TA	TB	YAA	KK	YAKA	PSO	ABA	YA	HAA	KSA	MA	BYA	EA	YUA	GKA
Li ve Lim	2003			✓													
Berger ve Barkaoui	2004	✓															
Chiang ve Russell	2004		✓														
Bent ve Hentenryck	2004			✓													
Polacek vd.	2004				✓												
Kuo vd.	2004					✓											
Homberger ve Gehring	2005		✓														
Chen ve Ting	2005			✓		✓											
Tan vd.	2005					✓											
Lin vd.	2006			✓													
De-Oliveira vd.	2006			✓													
Lin vd.	2006				✓												
Tan vd.	2006														✓		
Alvarenga vd.	2007	✓															
Creput vd.	2007														✓		
Ming-Yao vd.	2008		✓														
Fu vd.	2008		✓														
Hashimoto vd.	2008				✓												
El Hassani vd.	2008					✓											
Dursun	2009	✓															
Woch ve Lebkowski	2009			✓													
Liu vd.	2009							✓									
Ai ve Kachitvichyanukul	2009							✓									
Nazif ve Lee	2010	✓															
Ghoseiri ve Ghannadpour	2010	✓															
De Oliveira ve Vasconcelos	2010			✓													
Nagata vd.	2010												✓				
Yu vd.	2011		✓			✓											
Moghaddam vd.	2011			✓													
Yu ve Yang	2011					✓											
Balseiro vd.	2011					✓											
Najera ve Bullinaria	2011														✓		
Moccia vd.	2012		✓														
Taner vd.	2012			✓													
Taner vd.	2012				✓												
Ding vd.	2012					✓											
Iqbal	2012						✓										
Shi vd.	2012						✓										
Nikolic vd.	2013						✓										
Pan vd.	2013								✓								
Kiremitçi vd.	2014	✓															
Taş vd.	2014		✓														
Mahmudy	2014			✓													
Dhahri vd.	2014				✓												
Jawarneh ve Abdullah	2015						✓										
Yassen vd.	2015										✓						

Tablo 2'nin Devamı

Yazarlar	Yıl	GA	TA	TB	YAA	KK	YAKA	PSO	ABA	YA	HAA	KSA	MA	BYA	EA	YUA	GKA
Luo ve Chen	2014											✓					
Luo vd.	2015											✓					
Niu vd.	2012													✓			
Tan vd.	2015													✓			
Kuram	2016	✓															
Kırcı	2016		✓														
Miranda ve Conceiçao	2016				✓												
Alzaqebah vd.	2016						✓										
Yu vd.	2016						✓										
Mao vd.	2016						✓										
Nalepa ve Blocho	2016												✓				
Gupta ve Diwaker	2017	✓				✓											
Yao vd.	2017						✓										
Worawattawechai	2017						✓										
Taha vd.	2017									✓							
Yassen vd.	2017										✓						
Chen vd.	2017										✓						
Maleki vd.	2017										✓						
Göçken vd.	2018	✓															
Alzaqebah vd.	2018						✓										
Tuntitippawan ve Asawarungsaengkul	2018						✓										
Chen ve Zhou	2018						✓										
Aggarwal ve Kumar	2018								✓								
Osaba vd.	2018									✓							
Pratiwi vd.	2018									✓							
Lei vd.	2018																✓
Rezaeipanah vd.	2019																✓
Marinakis vd.	2019							✓									
Liu vd.	2019															✓	
Gunawan	2020															✓	
Kantawonq ve Pravesjit	2020						✓										

ÜÇÜNCÜ BÖLÜM

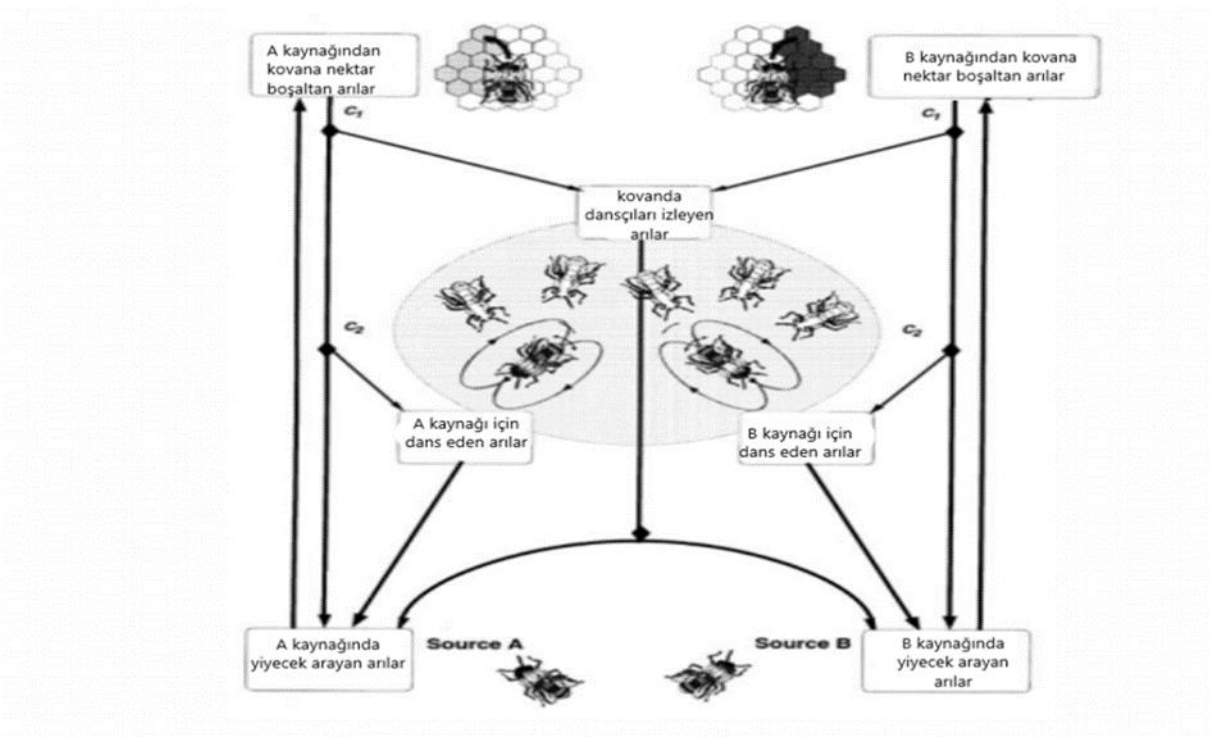
ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMİNİN META-SEZGİSEL YÖNTEMLERLE OPTİMİZASYONU

Bu bölümde, sürü zekâsının genel özellikleri, doğadaki bal arılarının yiyecek ararkenki davranışlarına, yiyecek kaynaklarına, arıların işçi, işsiz arılar olarak gruplandırılmasına, arıların yiyecek arama davranışlarına, sergiledikleri danslar ile yiyecek kaynakları hakkında diğer arılara bilgi vermelerine, yapay arı kolonisinin algoritmik yapısına ve evrelerine ve zaman pencereli araç rotalama problemi için geliştirilmiş yapay arı kolonisi algoritmasının evrelerine ve ZPARP için geliştirilen yapay arı kolonisi algoritmasının küçük boyutlu bir örnek üzerinde, başlangıç yiyecek kaynaklarının en yakın komşu sezgiseli ile oluşturulması ve oluşturulan bu yiyecek kaynaklarının uygunluk değerlerinin hesaplanışı, ekleme, yer değiştirme ve alt diziyi rastgele ekleme komşuluk operatörleri ile komşu yiyecek kaynaklarının oluşturulmasının nasıl olduğu ele alınmıştır.

Sürü Zekâsının Genel Özellikleri

Dünyada pek çok tür sürü vardır. Bunların hepsinin zekâsını aramak mümkün değildir veya zekâ seviyeleri sürülerden sürülere değişebilmektedir. Kendi kendine organize olabilme, basit ajanlar arasındaki yerel etkileşimler yoluyla kolektif davranışa neden olan bir sürü sisteminin kilit bir özelliğidir. Bonabeau vd. (1999), sürü örgütündeki kendi kendine organize olmayı, dört özellik ile yorumlamıştır:

Olumlu Geribildirim: Uygun yapıların oluşturulmasını teşvik eder. Bazı karınca türlerinde veya arılardaki danslarda iz bırakma ve takip etme gibi işe alım ve güçlendirme, olumlu geri bildirim örneği olarak gösterilebilir. Bir arı bir nektar kaynağı bulduğunda, kovana geri döner ve nektarını bir kovan arıya bırakır. Daha sonra ya diğer arılara yiyecek kaynağının yönünü ve mesafesini belirtmek için dans etmeye başlar ya da bulunduğu yiyecek kaynağını terk ederek kendi başına sıradan bir takipçi olup yiyecek kaynağında yiyecek aramaya devam eder. Koloniye yuvadan aynı mesafede iki özdeş yiyecek kaynağı sunulursa arılar iki kaynağı simetrik olarak kullanır. Bununla birlikte, bir kaynak diğerinden daha iyi ise, arılar daha iyi kaynaktan faydalanabilir veya daha sonra keşfedilse bile bu daha iyi kaynağa geçebilirler. Deneysel olarak da bir arının iyi bir yiyecek kaynağı için dans etme ve zayıf bir gıda kaynağını terk etme olasılığının yüksek olduğu gösterilmiştir. Bu basit davranış kuralları koloninin daha iyi kaliteli kaynağını seçmesini sağlar. Seeley vd. (1991), Camazine ve Sneyd (1991), yiyecek arayıcıları nektar kaynak kalitesine dayalı olarak farklı dans ve terk oranları ile oluşturulan olumlu bir geri bildirim yoluyla en iyi gıda kaynağına ev sahipliği yapabilecekleri bu gözlemlere dayanan basit bir matematiksel model ile doğrulamıştır. Şekil 7'de yiyecek arama aktivitesinin şematik bir temsili gösterilmektedir. (C1: takipçi mi?) ve (C2: dansçı mı?) karar noktaları siyah elmaslarla gösterilmiştir (Bonabeau vd., 1999: 9).



Şekil 7: Yiyecek Arama Faaliyetinin Şematik Gösterimi

Kaynak: (Bonabeau vd., 1999: 11)

Olumsuz Geribildirim: Olumlu geribildirimlerin dengelenmesi ve kollektif düzenin dengelenmesine yardımcı olur. Mevcut yiyecek arayıcılarda meydana gelebilecek doygunluğu önlemek için negatif geri besleme mekanizmasına ihtiyaç vardır.

Dalgalanmalar: Rastgele yürüyüşler, hatalar, yaratıcılık için hayati olan sürü bireyleri arasında rastgele görev değiştirme. Rastgele, yeni çözümlerin keşfedilmesini sağladığı için ortaya çıkan yapılar için önemlidir.

Çoklu Etkileşimler: Sürüdeki ajanlar, diğer ajanlardan gelen bilgileri kullanır; böylece bilgiler ağ boyunca yayılır.

Bu karakteristiklere ek olarak, iş bölümü olarak adlandırılan uzman ajanlar tarafından eş zamanlı olarak görev yapmak, zekânın oluşması için kendi kendine organize olabilmenin yanı sıra bir sürünün önemli bir özelliğidir.

Millonas'a (1994) göre bir sürü zekâsını tanımlamak için sürünün aşağıdaki prensipleri karşılaması gerekir:

- ✓ Sürü basit alan ve zaman hesaplamaları yapabilmelidir (yakınlık prensibi).
- ✓ Sürü, çevredeki kalite faktörlerine cevap verebilmelidir (kalite prensibi).
- ✓ Sürü, faaliyetlerini aşırı sınırlı kanallar boyunca gerçekleştirmemelidir (farklı tepki ilkesi).
- ✓ Sürü, çevrenin her dalgalanması üzerine davranış şeklini değiştirmemelidir (stabilite prensibi).
- ✓ Sürü gerektiğinde davranış modunu değiştirebilmelidir (uyarlanabilirlik ilkesi) (Karaboga vd., 2014: 24; Bonabeau vd., 1999: 10-11).

Bal Arısının Doğadaki Davranışı

Yapay arı kolonisi algoritması, bal arılarının yiyecek arama sırasında sergilemiş oldukları davranışlarından esinlenerek oluşturulmuştur. Doğada var olan sürülerden biri, yiyecekleri ararken kollektif zekâ davranışını takip eden bal arısı sürüsüdür. Bu sürü, bilgileri iletme, çevreyi tanıma, bilgileri muhafaza ederek paylaşabilir duruma getirme ve buna dayanarak kararlar alabilme gibi birçok özelliğe

sahiptir. Ortamdaki değişikliklere göre, sürü kendini günceller, görevleri dinamik olarak atar ve sosyal öğrenme ve öğretme ile daha ileriye doğru hareket eder. Arıların bu zekice davranışı, araştırmacıları arı sürüsünün yukarıdaki yiyecek arama davranışını taklit etmeye motive etmiştir. Gerçek bal arılarının davranışı başlarda, yiyecek kaynakları, işçi arılar, işsiz arılar, yiyecek arama davranışı ve danslar gibi başlıklarda aşağıda özetlenmiştir (Bansal vd., 2013: 125-126).

Yiyecek Kaynakları:

Arı, yiyecek ararken kendine uygun bir çiçek (yiyecek kaynağı) seçip, yiyecek kaynağının içerdiği nektarın miktarı, bu nektarın çiçekten ne kadar kolay elde edebileceği, kovandan ne kadar uzakta olduğu konusundaki bilgiyi bu seçmiş olduğu yiyecek kaynağından toplar. Arı, bu gerçekleri kolaylık ve basitlik adına tek bir nitelik (bu belirli yiyecek kaynağı için toplam kârlılık olarak adlandırılan) olarak saklar.

İşçi Arılar:

Mevcut yiyecek kaynakları, belirli bir grup arı tarafından kullanılmaktadır. Bu arılara işçi arılar denir ve bunların her biri ilişkili olduğu yiyecek kaynağının kârlılığını (zenginliği, kovandan uzaklığını ve yönünü) korur.

İşsiz Arılar:

İşçi arılar, bilgilerini belli bir olasılıkla işsiz arı olarak adlandırılan başka bir grup arıyla paylaşırlar. İşsiz arılar, işçi arılardan elde ettikleri bilgileri özetlemekten ve sömürülecek gıda kaynağını seçmekten sorumludur. Bu işsiz arılar, gözcü ve kâşif arılar olarak ikiye ayrılır. Gözcü arılar, kovandaki işçi arılardan bilgi toplayan arılardır ve verileri analiz ettikten sonra, kendileri için bir yiyecek kaynağı oluştururlarken kâşif arı kovanın çevresindeki yeni besin kaynaklarını bulmaktan sorumludur. Mevcut yiyecek kaynaklarından bazıları tükendiğinde, bu arılar kovan etrafındaki çevreyi aramaya başlar ve yeni besin kaynaklarını rastgele bulur. Bir bal arısı sürüsü, genellikle ortalama % 50 işçi arılar, % 50 işsiz arılar ve toplam arıların% 5 ile % 10'u kâşif arılardan oluşur.

Yiyecek Arama Davranışı:

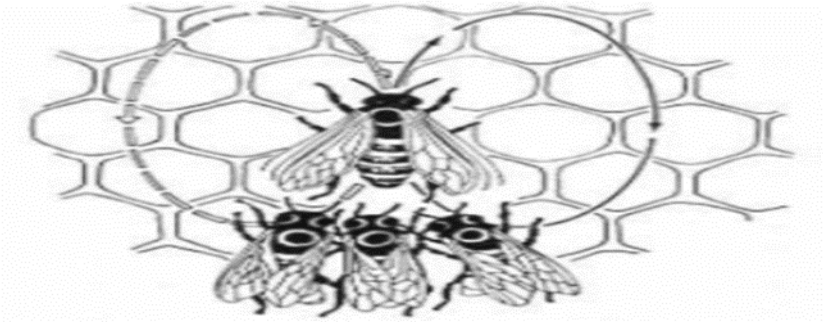
Bal arısı sürüsünün en önemli özelliği, yiyecek arama davranışıdır. Yiyecek arama sürecinde, arı kovandan ayrılır ve yiyecekleri aramaya başlar. Arı bir yiyecek kaynağına ulaştığında nektarı oradan çıkarır ve midesine depolar. Nektarı; zenginlik, besin kaynağının kovandan uzaklığı gibi koşullara göre 30-120 dakikaya kadar çıkarır. Daha sonra midesinde enzimlerin salgılanmasıyla bal yapım süreci başlar ve kovana ulaştıktan sonra boş hücrelerde nektarı boşaltır. Sonunda kovandaki bilgilerini, bir sonraki bölümde tanımlanan çeşitli dans türleri biçiminde paylaşır.

Dans:

Kovanda yaşayan diğer arılara, besin kaynağının ne kadar bol olduğunu, kovandan ne kadar uzakta ve hangi yönde olduğunu, işçi arı kovanın farklı bölgelerinde, dans adı verilen belirli türde adımlar uygular. Von Frisch (1967) (1973 Nobel Ödülü sahibi) arıların dans dilini çözmüş ve bir arı dansının yön bilgisinin, bir besin kaynağının güneşe göre konumunu gösterdiğini ve besin kaynağının mesafesinin farklı dans türleriyle işaretlendiğini belirtmiştir. Tarpy (2009) ve Wenner ve Wells (1990), bir yiyecek kaynağı gövdesindeki çiçek kokularının, işçi arıların yeni yiyecek kaynakları bulmalarını sağlayan ana işaretler olduğunu savunmuşlardır.

Dans dilleri veya çiçek kokuları, yiyecek arama davranışını yerine getiren arılar arasında iletişim olduğunu gösterir. Dans aracılığıyla, başkalarına yiyecek kaynaklarını takip edip etmemeleri gerektiğini bildirmek ister. Dans hareketleri kovanın farklı alanlarında yapılır, böylece onunla ilişkilendirilen gıda kaynakları hakkında daha fazla arı bilgilendirilebilir. Dans ederken diğer arılar, besin kaynağının nektarını tatmak için antenleriyle ona dokunurlar. Besin kaynağının kârlılığına bağlı olarak, işçi arı aşağıdaki dans formlarından birini gerçekleştirir:

- ✓ ***Dairesel Dans:*** Bu dans türü besin kaynağının yönü hakkında bilgi vermez, ancak arı bu dansı besin kaynağının yakınına geldiğinde (yaklaşık 100 metreden fazla olmayan) kovana yaklaştığında yapar.
- ✓ ***Kuyruk Dansı:*** Bu dans formu, diğer arılara, güneş ışığına göre besin kaynağının yönü hakkında bilgi verir ve kaynak kovandan uzaksa, işçi arılar bu dans formunu seçer. Dansın hızı, besin kaynağının kovandan olan uzaklığı ile orantılıdır. Bu dans Şekil 8'de gösterilmiştir.



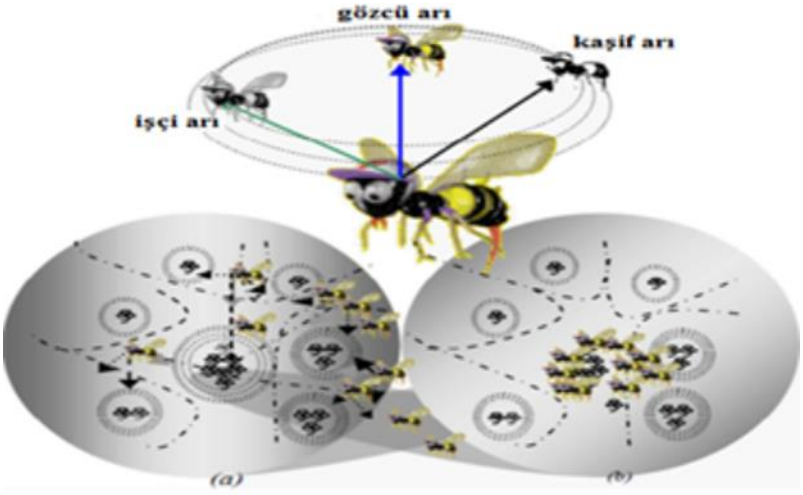
Şekil 8: Arıların Kuyruk Dansı
Kaynak: (Mogaka vd., 2016: 16)

- ✓ ***Titreme Dansı:*** Bir arının nektarı boşaltması daha uzun zaman aldıysa titremeye başlar ve başkalarını bilgilendirmeden önce çok zaman aldığı için gıda kaynağının mevcut kârlılığını (zenginliğini) bilmediğini belirtir.

Yapay Arı Kolonisinin Algoritmik Yapısı

Gerçek bal arılarının minimal yiyecek seçimi modelinde olduğu gibi yapay arı kolonisi algoritmasında yapay arı kolonisi üç grup arı içerir. Bunlar; belirli yiyecek kaynakları ile ilişkilendirilmiş işçi arılar, kovan içindeki işçi arıların dansını izleyerek yiyecek seçen gözcü arılar ve yiyecek kaynaklarını rastgele arayan kâşif arılardır. İşçi veya gözcü arıların sayısı yiyecek kaynaklarının sayısına eşittir. Yiyecek kaynağı tükenen işçi arılardan biri kâşif arısı olur ve yeni besin kaynağını rastgele arar. Hem gözcüler hem de kâşifler, işsiz arılar olarak adlandırılırlar. Başlangıçta tüm yiyecek kaynağı konumları kâşif arılar tarafından keşfedilir. Sonrasında yiyecek kaynaklarının nektarı, işçi ve gözcü arılar tarafından sömürülür ve bu sürekli sömürü eninde sonunda yiyecek kaynaklarının tükenmesine sebep olur. Daha sonra kaynağı tükenmiş işçi arı bir kez daha başka yiyecek kaynaklarını aramak için kâşif arı olur. Başka bir deyişle, yiyecek kaynağı tükenmiş olan işçi arı, kâşif arı haline gelir. Yapay arı kolonisinde, bir yiyecek kaynağının konumu, problem için olası bir çözümü temsil eder ve bir yiyecek kaynağının nektar miktarı, ilgili çözümün kalitesine (uygunluğuna) karşılık gelir. Temel yapay arı kolonisinde, işçi arıların sayısı yiyecek kaynaklarının (çözümlerin) sayısına eşittir. Yani, her işçi arı sadece ve sadece bir yiyecek kaynağı ile ilişkilendirilir (Karaboga vd., 2014: 25-26).

Yapay arı kolonisi algoritması da diğer sürü tabanlı algoritmalara benzer şekilde tekrar eden bir süreçtir. Algoritmada, arama alanının farklı alanlarını keşfetmeyi sağlayan çeşitlilik süreci ve önceki deneyimlerin sömürülmesini sağlayan seçim süreci olmak üzere yapay arı kolonisi popülasyonunun evrimini türeten iki temel süreç vardır. Bununla birlikte popülasyonun yerel bir optimuma yakınsamaması durumu olsa da algoritmanın zaman zaman küresel optimuma doğru ilerlemeyi bırakabileceği gösterilmiştir (Bansal vd., 2013: 127). Yapay arı kolonisi algoritmasının temel arama süreci Şekil 9'da gösterilmiştir. Burada şeklin a) ile gösterilen kısmı arama sürecinin başlangıç durumu b) ile gösterilen kısmı ise arama sürecinin son durumudur (Mogaka vd., 2016: 17)



Şekil 9: YAKA'nın Temel Arama Süreci

Kaynak: (Mogaka vd., 2016: 17)

Yapay arı kolonisi algoritmasında süreç; başlangıç evresi, işçi arı evresi, gözcü arı evresi ve kâşif arı evresi olmak üzere dört aşamalı bir döngüyü gerektirir. Bu evrelerin her biri aşağıda ifade edilmiştir (Bansal vd., 2013: 127-130).

Başlangıç Popülasyonu

Başlangıçta, yapay arı kolonisi SN çözümlerinin düzgün dağılmış bir popülasyonunu oluşturur. Burada, her $x_i (i = 1,2, \dots, SN)$ çözümü, D −boyutlu bir vektördür. Ayrıca, D; optimizasyon problemindeki değişkenlerin sayısını ve x_i ise popülasyondaki i. yiyecek kaynağını temsil eder. Her bir yiyecek kaynağı aşağıdaki (3.1) denklemindeki gibi oluşturulur:

$$x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right), \qquad \forall j = 1,2, \dots, D \tag{3.1}$$

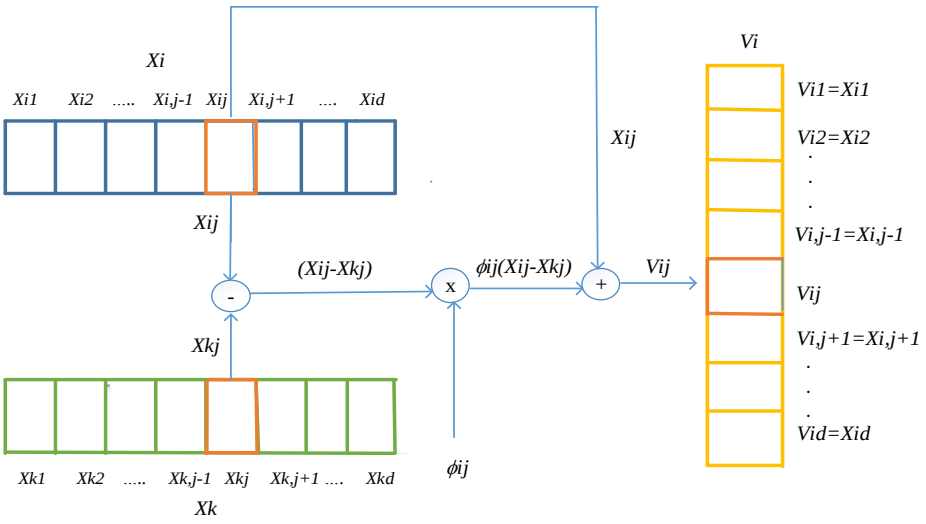
Burada, $x_{min}^j \; ve \; x_{max}^j$ sırasıyla j. yöndeki x_i'nin alt ve üst sınırlarıdır.

İşçi Arı Evresi

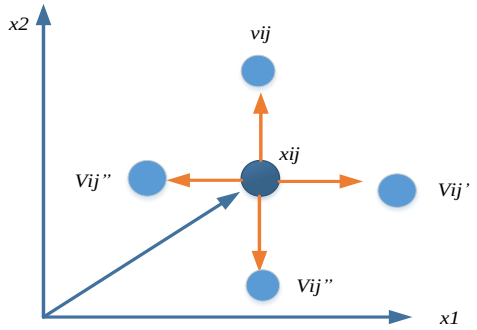
Bu evrede, işçi arılar, bireysel deneyimlerinin bilgisine ve yeni çözümün uygunluk değerine (nektar miktarı) dayanarak mevcut çözümü değiştirirler. Yeni yiyecek kaynağının uygunluk değeri, eski yiyecek kaynağının uygunluk değerinden yüksekse arı konumunu eski yiyecek kaynağından yeni yiyecek kaynağına getirerek günceller ve eskisini atar. $i.$ adayın j. boyutu için konum güncelleme denklemi (3.2) denkleminde gösterilmiştir:

$$v_{ij} = x_{ij} + \varphi_{ij}\left(x_{ij} - x_{kj}\right) \tag{3.2}$$

Burada, $\varphi_{ij}\left(x_{ij} - x_{kj}\right)$, hareket büyüklüğü, $k \in \{1,2, \dots, SN\}, j \in \{1,2, \dots, D\}$ keyfi seçilmiş iki indekstir. k, i'den farklı olmalı ki, hareket büyüklüğü bazı önemli katkıya sahip olsun. φ_{ij}, $[-1,1]$ aralığındaki keyfi bir sayıdır. İşçi arı evresinde konum güncelleme süreci Şekil 10'da gösterilmiştir. Burada x_i, bir arının mevcut konumunu temsil eder. Vurgulanan kutu, rastgele seçilen j yönünü temsil eder. x_k, rastgele seçilen bir arıdır. Bu adımda, keyfi bir $k \neq i$ arının j yönü $i.$ arının j yönünden çıkarılır. Bu fark, keyfi bir sayı olan $\varphi_{ij} \in [-1,1]$'den çarpılır. Son olarak bu sayı, yeni yiyecek konumu v_{ij}'nin j. boyutunu elde etmek için x_i'nin j. boyutuna eklenir. Şekildeki dikey vektör v_{ij} tarafından temsil edilir ve x_i'nin komşuluğunda oluşturulur. Diğer tüm boyutları, x_i ile aynıdır. Arama uzayı 2 −boyutlu olarak düşünülürse bu yeni yiyecek kaynağı v_{ij} için olası konumlar Şekil 11'deki gibi olur.



Şekil 10: Basit bir konum güncelleme uygulamasının gösterimi



Şekil 11: 2-boyutlu arama alanındaki konum güncelleme denklemi nedeniyle x_{ij}'nin komşuluğunda oluşabilecek farklı yeni vektörler

Gözcü Arı Evresi

İşçi arı evresinin tamamlanmasından sonra gözcü arı evresine başlanır. Bu evrede, işçi arıların tümü, güncel çözümlerin (yiyecek kaynakları) uygunluk (nektar) değerlerini ve yiyecek kaynakların konum bilgisini kovanda bekleyen gözcü arılarla paylaşırlar. Gözcü arılar, mevcut bilgileri analiz eder ve uygunluğu ile ilgili olan p_i olasılıklı bir çözümü seçerler. p_i olasılığı aşağıdaki ifade kullanılarak hesaplanabilir:

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{3.3}$$

Burada $fitness_i$, i. çözümün uygunluk değeridir. İşçi arı evresinde olduğu gibi, gözcü arı hafızasındaki konumunda bir değişiklik yapar ve aday kaynağın uygunluğunu kontrol eder. Eğer uygunluk öncekinden yüksekse, arı yeni konumu hafızaya alır ve eskisini unutur.

Kâşif Arı Evresi

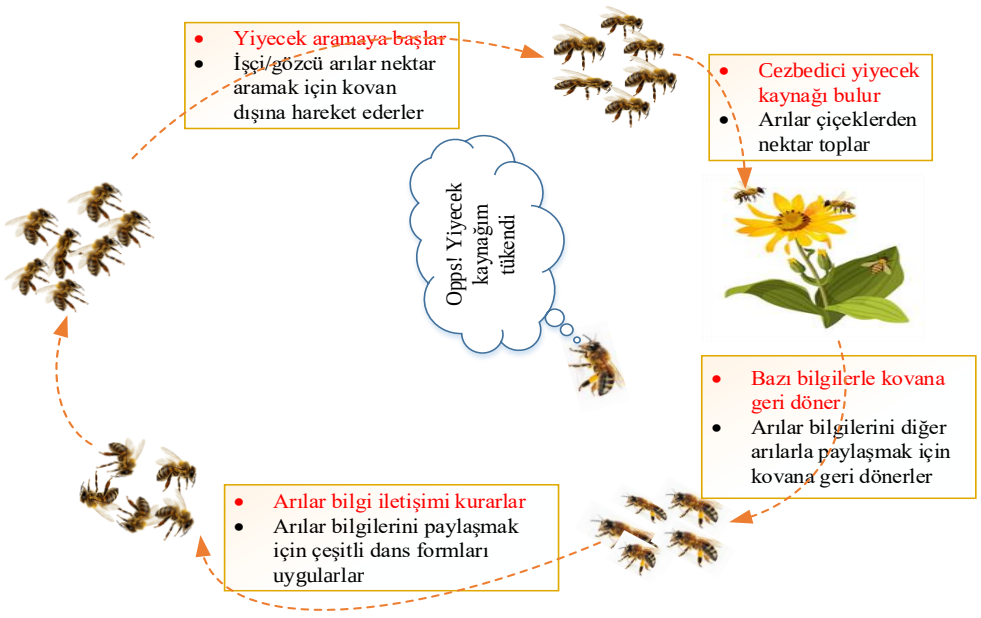
Bir yiyecek kaynağının konumu, önceden tespit edilmiş bir döngü sayısı içinde güncellenmezse yiyecek kaynağının terkedildiği varsayılır ve kâşif arı evresi başlatılır. Bu evrede, terk edilmiş yiyecek kaynağı ile ilişkili arı, kâşif arı haline gelir ve yiyecek kaynağı, arama alanındaki rastgele seçilen yiyecek kaynağı ile değiştirilir. Yapay arı kolonisinde önceden belirlenmiş döngü sayısı, vazgeçme limiti olarak adlandırılan önemli bir kontrol parametresidir.

Kabul edelim ki terkedilen kaynak x_i olsun. Bu durumda, kâşif arı bu yiyecek kaynağını yeni yiyecek kaynağı olan x_i ile aşağıdaki gibi yer değiştirir:

$$x_i^j = x_{min}^j + rand[0,1](x_{max}^j - x_{min}^j), \qquad \forall j = 1,2,\dots,D \tag{3.4}$$

Burada, $x_{min}^j \; ve \; x_{max}^j$ sırasıyla x_i'nin j. yönündeki alt ve üst sınırlarıdır.

Şekil 12'de yapay arı kolonisinde arıların yiyecek ararkenki davranışları gösterilmiştir. Şekil 12'den de anlaşılacağı üzere, her aşamada arıların sadece yarısının yiyecek kaynaklarını araştırdığı ve toplanan bilgileri çeşitli danslar yaparak kovanda kalan diğer arılara ilettiği açıktır. Besin kaynağı tükenen arı, kâşif arı olur ve rastgele yeni bir çözüm arar.



Şekil 12: Bal Arılarının Sosyal Davranışları

Kaynak: (Bansal vd., 2013: 130)

Yapay Arı Kolonisi Algoritmasının Temel Adımları

Yapay arı kolonisinde arama sürecinin üç önemli kontrol parametresi vardır:

- ✓ Yiyecek kaynaklarının sayısı, ***SN*** (işçi veya gözcü arıların sayısına eşit)
- ✓ Tükenen yiyecek kaynağının tespitinde kullanılan önemli bir kontrol parametresi olan **limit** değeri
- ✓ Yapay arı kolonisi algoritmasının döngüsünün sonlanmasında durdurma kriteri olarak kullanılan **maksimum iterasyon (yineleme) sayısı**

Bu parametreler algoritmanın başlangıç evresinde belirlenir. Daha sonra işçi arı, gözcü arı ve kâşif arı evresi olarak adlandırılan bu üç evre yinelemeli olarak durdurma kriteri sağlanana kadar devam eder. Yapay arı kolonisi algoritmasının tekrar eden bu döngüsü aşağıda ifade edilmiştir (Gothania vd., 2014: 64-66).

Algoritma 1: YAKA (Gothania vd., 2014)

Başlangıç Parametreleri;

Durdurma Kriteri Sağlanmadıysa

Adım 1: Yeni yiyecek kaynaklarını oluşturmak için işçi arı evresi;

Adım 2: Nektar miktarlarına bağlı olarak yiyecek kaynaklarını güncellemek için gözcü arı evresi;

Adım 3: Terkedilen yiyecek kaynaklarının yerine yeni yiyecek kaynaklarını keşfetmek için kâşif arı evresi;

Adım 4: Şimdiye kadar bulunan en iyi yiyecek kaynağını hafızaya al;

Sonlanana Kadar

Şimdiye kadar bulunan en iyi çözümün çıktısını al.

Yapay arı kolonisi algoritması aslında dört farklı seçim süreci kullanır (Karaboga ve Basturk, 2007: 463):

- ✓ Denklem (3.3)'te ifade edildiği gibi gelecek vaat eden bölgeleri keşfetmek için gözcü arıların kullandığı bir küresel seçim süreci
- ✓ Yerel bilgilere bağlı olarak yapay işçi ve gözcü arılar tarafından bir bölgede gerçekleştirilen bir yerel seçim süreci (gerçek arılar söz konusu olduğunda, bu bilgi çiçeklerin rengini, şeklini ve kokusunu içerir) (arılar, nektar kaynağının türünü doğru yere gelinceye kadar tanımlayamayacaklar ve Denklem (3.2)'de tanımlandığı gibi, kaynağın etrafındaki bir komşu yiyecek kaynağını belirlemek için kokularına göre orada yetişen kaynaklar arasında ayrım yapamayacaklardır.
- ✓ Komşu kaynağın nektar miktarının mevcut olandan daha iyi olması durumunda arıların mevcut olanı unutması ve komşu kaynağı hafızaya alması için tüm arılar tarafından gerçekleştirilen açgözlü seçim süreci adı verilen yerel bir seçim süreci. Aksi takdirde, arı mevcut olanı hafızada tutar.
- ✓ Kâşif arılar tarafından gerçekleştirilen rastgele bir seçim süreci

ZPARP için Önerilen YAKA

Çalışmanın bu bölümünde ZPARP'nin çözümü için geliştirilmiş yapay arı kolonisi algoritmasının evrelerine değinilmiştir. Başlangıç evresinde yiyecek kaynaklarının en yakın komşu algoritması ile oluşturulmasına, oluşturulan yiyecek kaynaklarının uygunluk fonksiyonlarının nasıl hesaplanması gerektiğine, işçi ve gözcü arı evrelerinde komşu yiyecek kaynaklarının oluşturulmasında ekleme, yer değiştirme ve alt diziyi rastgele ekleme operatörlerinin kullanılmasına ve kâşif arı evresine ve bu evrede yiyecek kaynaklarının rastgele ve en yakın komşu sezgiseli olmak üzere iki şekilde oluşturulmasına yer verilmiştir. Optimum çözümlere ulaşmak için başlangıç ve kâşif arı evresindeki yiyecek kaynaklarının (çözümlerinin) oluşturulmasında en yakın komşu sezgiseli kullanılmıştır. Burada ele alınan en yakın komşu sezgiseli ile yiyecek kaynakları oluşturulurken her bir yiyecek kaynağındaki her bir rotanın depodan sonraki ilk müşterisi rastgele diğer müşteriler EYK yöntemine göre seçilerek oluşturulmuştur. EYK yöntemine göre eklenecek müşterilerin maliyet ve süre değerleri hesaplanarak bu değerlerden büyük olanı bu müşterilerin seçim değerleri olarak dikkate alınmıştır. Seçim değeri küçük olan müşteri rotaya eklenmiştir. Her bir müşterinin birbirlerine ve depoya olan uzaklıkları Öklid uzaklığı ile hesaplanmış, oluşturulan yiyecek kaynağının uygunluk değeri ise bu yiyecek kaynağındaki (çözüm satırındaki) müşterilerin birbirlerine ve depoya olan uzaklıkları toplamı olarak hesaplanmıştır. Diğer taraftan işçi ve gözcü arı evresinde komşu yiyecek kaynaklarının (yeni çözümlerin) oluşturulmasında rota geliştirici sezgiseller olarak bilinen ekleme, yer değiştirme ve alt diziyi rastgele ekleme operatörleri kullanılmıştır. İşçi arı evresinde rastgele seçilen yiyecek kaynağı üzerinde önceden oluşturulmuş rastgele sayıya göre eşit olasılıkla ekleme, yer değiştirme ve alt diziyi rastgele ekleme operatörü uygulanarak

komşu yiyecek kaynakları oluşturulurken; gözcü arı evresinde ise komşu yiyecek kaynakları gözcü arı evresine kadar ki oluşturulan yiyecek kaynaklarından seçilen en iyi yiyecek kaynağı üzerinde bu üç operatör önceden oluşturulmuş rastgele bir sayıya göre eşit olasılıkla uygulanarak oluşturulmuştur.

Başlangıç Evresi

İlk olarak yiyecek sayısı kadar en yakın komşu sezgiseli ile oluşturulan rotaların yer aldığı yiyecek kaynakları oluşturulur. Temel yapay arı kolonisi algoritmasında belirtildiği gibi burada da yiyecek kaynaklarının sayısı işçi arı sayısına eşit olup, yiyecek kaynaklarında yer alan her bir yiyecek satırı bir çözümü temsil etmektedir. Bu yiyecek kaynaklarına bağlı olarak işçi arı sayısı ve gözcü arı sayısı belirlenir. Maksimum iterasyon sayısı, ilk populasyon aday bulma deneme sayısı, işçi/gözcü arı aşaması deneme sayısı ve alt diziyi rastgele ekleme iç deneme sayısı tespit edilir. Kontrol amaçlı sayaç değişkeni oluşturulur.

ZPARP için en yakın komşu algoritması Bölüm 3.1.4.1.1.'de ifade edilmiştir.

En Yakın Komşu Algoritması (EYK)

En yakın komşu sezgiseli, bir rotadaki en son müşterinin ardına bu müşteriye mesafe açısından en yakın olan müşterinin eklenmesidir. Tabi burada problemin zaman pencereli araç rotalama problemi olması sebebiyle sadece mesafe açısından değil aynı zamanda zaman penceresi kısıtını da göz önünde bulundurarak çözümler oluşturulacaktır. Bu sebepten dolayı, rotaya eklenen son müşterinin müşteri listesindeki gidilmeyen müşterilere uzaklıkları ile o müşterinin servise hazır olma süresi karşılaştırılır. Tabi burada karşılaştırmalar bir birim uzaklığın bir birim zaman diliminde gidildiği varsayımı ile yapılır. Servise hazır olma süresi ve uzaklık değerlerinden büyük olanı o müşterinin seçim değeri olarak tespit edilir. Başlangıç noktası (depo) ile bir rota oluşturulur. Tüm müşteriler arasından minimum seçim değerine sahip müşteri ilk rotanın ilk müşterisi olarak seçilerek listeden kaldırılır ve rotaya ilave edilir. Araç kapasitesinin doluluğunu hesaplanır. Rotadaki müşterilerin toplam talebi araç kapasitesine eşit olana kadar müşteriler aynı şekilde rotaya alınmaya devam ettirilir. Talepleri araç kapasitesini geçtiği durumda aracın depoya dönmesi için rota sonuna depo eklenir ve yeni bir rota oluşturulur. Bu süreç müşteri listesinde müşteri kalmayana kadar tekrar ettirilir. (Göçken vd., 2018: 778-779). Bu çalışmada da her bir yiyecek kaynağı en yakın komşu sezgiseli ile oluşturulmuştur. Tabi burada her bir rotanın ilk müşterisi yukarıda anlatılandan farklı olarak rastgele seçilip rotaya eklenmiştir. Rotaların geri kalan müşterileri ise en yakın komşu yöntemine göre seçilip rotaya ilave edilmiştir. Problemimiz için kullanılan en yakın komşu algoritmasının aşamaları ayrıntılı biçimde aşağıda ifade edilmiştir. Başlangıç yiyecek kaynaklarını oluşturma aşamasında, populasyondaki her bir yiyecek kaynağı için yiyecek kaynağı tüm müşterileri içeren bir çözüm olana kadar aşağıdaki adımlar takip edilerek bireye rotalar eklenmiştir.

Adım 1: Depo (0) rotaya eklenir. Gidilmeyen müşteriler bulunur.

Adım 2: Gidilmeyen müşteriler arasından rastgele bir müşteri seçilir. Seçilen müşteri rotaya eklenir (bu müşteri M_2 olarak anılacaktır).

Adım 3: M_2, gidilmeyenler listesinden çıkarılır.

Adım 4: Gidilmeyenler arasında cezalı olmayanların her biri için aşağıdaki adımlar tekrar edilir.

- **i.** Gidilmeyen bir müşteri seçilir (M_x).
- **ii.** Rotaya seçilen müşteri eklenir ve rotanın Mesafe ve Süre değeri hesaplanır. Mesafe değeri depodan başlanarak rotadaki son müşteriye ulaşana kadar gidilmesi gereken uzaklık. Süre değeri depodan başlanarak rotadaki son müşteriye ulaşıp müşteriden yükün alınarak (servis süresi dâhil) ayrılma anına kadar geçen süredir.
- **iii.** Maliyet ve Süre değerinin en büyüğü seçilerek M_x'in maliyeti bulunur.

Adım 5: Gidilmeyenler arasında en düşük maliyet değerine sahip müşteri seçilir (M_{best}).

Adım 6: M_{best} rotaya eklenir.

Adım 7: Rotada kapalı müşteri sorunu olup olmadığı kontrol edilir. Rotada kapalı müşteri sorunu varsa M_{best} rotadan çıkarılır ve cezalı olarak belirlenir. Rotanın depoya dönülerek sonlandırılması gerekip-gerekmediği kontrol edilir. Depoya dönülecekse, rotaya Depo eklenerek rota kapatılır. Depoya dönülmeyecekse Adım-4'e dönülür.

Adım 8: Rotada kapalı müşteri sorunu yoksa M_{best} gidilmeyenler listesinden çıkarılır.

Adım 9: Gidilmeyenlerin cezaları silinir.

Adım 10: Aracın dolu olup olmadığı kontrol edilir. Araç yük olarak dolu ise (kapasitesi aşılmışsa) M_{best} rotadan çıkarılır ve Depo rotaya eklenerek rota kapatılır.

Adım 11: Gidilmeyen müşteri varsa Adım-4'e dönülür.

Adım 12: Gidilmeyen müşteri kalmamışsa rota sonlandırılır.

Uygunluk Fonksiyonu

Her yiyecek kaynağı (çözüm satırları) için uygunluk fonksiyonu, oluşturulan yiyecek kaynaklarındaki (çözüm satırlarındaki) müşterilerin birbirlerine olan uzaklıkları ile müşterilerin depoya olan uzaklıklarının toplamı şeklinde ifade edilmiştir. En iyi çözüm, bütün iterasyonlar içerisinde en kısa uzaklığa sahip olan çözümdür. Bu durumda, çözüm yöntemlerinin uygunluk fonksiyonu Denklem 3.5'te belirtilmiştir.

$$f_i = \sum_{j=1}^{k} d_j \qquad (3.5)$$

f_i: i yiyecek kaynağının uygunluk değeri

d_j: oluşturulan yiyecek kaynağının toplam yol uzunluğu ($j = 1,2,3, \dots, k$)

Her bir yiyecek kaynağına ait uygunluk değerleri Denklem (3.5)'e göre hesaplanır.

İşçi Arı Evresi

Her bir yiyecek kaynağı en yakın komşu sezgiseli ile oluşturulup uygunluk değerleri hesaplandıktan sonra işçi arılar devreye girer. İşçi arılar rastgele bir yiyecek kaynağı seçerek bu çözüm satırındaki rotaları manüple ederler. Daha önce de belirtildiği üzere, işçi arıların sayısı yiyecek kaynaklarının sayısına eşittir. Dolayısıyla, her bir işçi arı popülasyondaki her bir çözüme karşılık gelir. Temel yapay arı kolonisi algoritmasına göre her işçi arı mevcut kaynağın komşuluğunda bir yiyecek kaynağı üretir. Bir çalışmada bu yiyecek kaynaklarının üretiminde yer değiştirme, ekleme ve ters çevirme operatörleri önceden oluşturulmuş rastgele bir sayıya göre eşit olasılıkla kullanılmıştır (Özdemir, 2013: 88). Biz de burada işçi arıların komşu yiyecek kaynağı oluşturmaları için yer değiştirme, ekleme ve alt dizileri rastgele ekleme komşuluk operatörlerini önceden oluşturulmuş rastgele bir sayıya göre eşit olasılıkla kullanacağız. İşçi arılar rastgele bir besin seçip besin satırında yer alan rotaların yerlerini değiştirerek komşu yiyecek kaynaklarını bahsedilen bu üç operatörü kullanarak oluştururlar. Oluşturulan yiyecek kaynaklarının (komşu çözümlerin) nektar miktarı (uygunluk değeri) problem için tanımlanan amaç fonksiyon değerine göre hesaplanır. Elde edilen çözüm değeri mevcut çözüm değerinden daha iyi ise hafızaya alınır, aksi takdirde deneme sayacı bir arttırılır.

Gözcü Arı Evresi

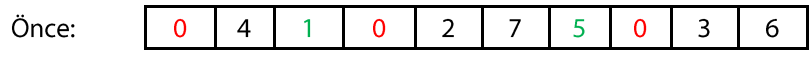
İşçi arı evresinden sonra gözcü arı evresi başlar. Bu evrede gözcü arılar oluşturulan her bir çözüm değerinin uygunluk değerine (amaç fonksiyonuna göre) göre bir yiyecek satırı seçer ve işçi arı evresinde yapıldığı gibi bu besin satırındaki rotaların yerini yer değiştirme, ekleme ve alt diziyi rastgele ekleme

operatörleri ile değiştirerek yeni çözüm oluştururlar. Amaç minimizasyon olması sebebiyle önceden tespit edilmiş rastgele sayıya bağlı olarak en düşük amaç değerine sahip olan besinde karar kılınır. Elde edilen yeni çözüm değeri mevcut çözümden daha iyi ise hafızaya alınır, aksi takdirde sayaç değişkeni bir arttırılır.

Yukarıda bahsi geçen komşuluk operatörleri aşağıda açıklanmıştır.

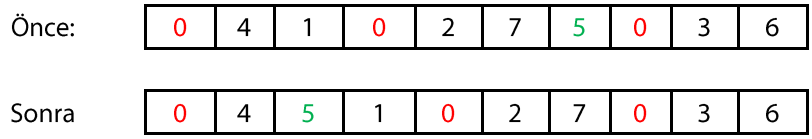
Komşuluk Operatörleri

Komşuluk operatörlerinden rastgele yer değiştirme operatörü; çözüm vektöründe $i \neq j$ olmak üzere i ve j konumlarını rastgele seçer ve i ve j konumunda bulunan müşterilerin yerlerini değiştirir. Örneğin $i = 3$ ve $j = 7$ olmak üzere verilen bir çözüm vektörünün bu konumlarda bulunan müşterilerin yer değiştirilmesi Şekil 13'teki gibidir (Szeto vd., 2011: 128-129):



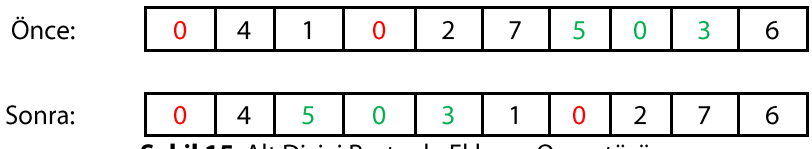
Şekil 13: Yer Değiştirme Operatörü

Diğer taraftan rastgele ekleme operatörü, $i \neq j$ olmak üzere i ve j konumlarını rastgele seçerek, müşteriyi i konumundan j konumuna taşıma işleminden oluşur. Örneğin 5 müşterisini 7 konumundan 3 konumuna yeniden konumlandırma işlemi Şekil 14'te verilmiştir:



Şekil 14: Ekleme Operatörü

Alt dizileri rastgele ekleme operatörü, ekleme operatörünün bir uzantısıdır. i konumundan başlayarak rastgele uzunluktaki bir depo ve müşteri dizisinin j konumuna yeniden konumlandırılmasıdır. Bu operatörün bir örneği Şekil 15'te verilmiştir:



Şekil 15: Alt Diziyi Rastgele Ekleme Operatörü

Kâşif Arı Evresi

Gözcü arı evresinden sonra kâşif arı evresi başlar. Kâşif arı evresinde, ilk adımda üretilmiş olan yiyecek kaynakları yeniden oluşturulur. Bu yiyecek kaynakları ilk aşamadan farklı olarak rastgele ve en yakın komşu algoritması oluşturulur. Yiyecek kaynağındaki her bir çözüm satırının uygunluk değerleri hesaplanır. Elde edilen bu değerler hafızadaki en iyi sonuçlarla karşılaştırılır. Bu çözüm değerleri mevcut çözüm değerlerinden daha iyi ise hafızaya alınıp, ilgili yiyecek kaynağındaki çözüm satırı ile yer değiştirir. Aksi halde, sayaç değişkeni bir arttırılır.

Yukarıda detayları anlatılan algoritmanın adımları aşağıda belirtilmiştir:

Adım 1: Kullanıcının program arayüzünden belirlediği değerler ile "PopulasyonBoyutu, İşçi Arı Sayısı, Gözcü Arı Sayısı, Maksimum İterasyon Sayısı, Maksimum Aday Deneme Sayısı, İşçi Arı Deneme Sayısı, Alt Dizi Ekleme Deneme Sayısı, Kâşif Arı Rastgele-EnYakın" değerleri belirlenir. Kâşif arı sayısı 1 olarak alınmıştır.

Adım 2: "En Yakın Komşu" yöntemi ile aşağıdaki adımlar gerçekleştirilerek ilk popülasyon (yiyecek kaynakları) oluşturulur.

Adım 2.1: Depo (0) rotaya eklenir. Gidilmeyen müşteriler bulunur.

Adım 2.2: Gidilmeyen müşteriler arasından rastgele bir müşteri seçilir. Seçilen müşteri rotaya eklenir (bu müşteri M_2 olarak anılacaktır).

Adım 2.3: M2, gidilmeyenler listesinden çıkarılır.

Adım 2.4: denemeSayisi = 0 olarak belirlenir.

Adım 2.5: "denemeSayisi > maksimum aday deneme sayısı" şartı gerçekleşirse depoya dönülerek rota sonlandırılır.

Adım 2.6: Gidilmeyenler arasında cezalı olmayanların her biri için aşağıdaki adımlar tekrar edilir.

Adım 2.6.1: Gidilmeyen bir müşteri seçilir (M_i).

Adım 2.6.2: Rotaya seçilen müşteri eklenir ve rotanın Mesafe ve Süre değeri hesaplanır. Mesafe değeri depodan başlanarak rotadaki son müşteriye ulaşana kadar gidilmesi gereken uzaklık. Süre değeri depodan başlanarak rotadaki son müşteriye ulaşıp müşteriden yükün alınarak (servis süresi dâhil) ayrılma anına kadar geçen süredir.

Adım 2.6.3: Maliyet ve Süre değerinin en büyüğü seçilerek M_i'nin maliyeti bulunur.

Adım 2.7: Gidilmeyenler arasında en düşük maliyet değerine sahip müşteri seçilir (M_{best}).

Adım 2.8: M_{best} rotaya eklenir.

Adım 2.9: Rotada kapalı müşteri sorunu olup olmadığı kontrol edilir. Rotada kapalı müşteri sorunu varsa M_{best} rotadan çıkarılır ve cezalı olarak belirlenir. Rotanın depoya dönülerek sonlandırılması gerekip-gerekmediği kontrol edilir. Depoya dönülecekse, rotaya Depo eklenerek rota kapatılır. Depoya dönülmeyecekse Adım-2.6'ya dönülür.

Adım 2.10: Rotada kapalı müşteri sorunu yoksa, M_{best} gidilmeyenler listesinden çıkarılır.

Adım 2.11: Gidilmeyenlerin cezaları silinir.

Adım 2.12: Aracın dolu olup olmadığı kontrol edilir. Araç yük olarak dolu ise (kapasitesi aşılmışsa) M_{best}rotadan çıkarılır ve depo rotaya eklenerek rota kapatılır.

Adım 2.13: Gidilmeyen müşteri varsa Adım-2.6'ya dönülür.

Adım 2.14: Gidilmeyen müşteri kalmamışsa rota sonlandırılır.

Adım 3: İlk popülasyon oluşturulduktan sonra iterasyon = 1 değer ataması ile başlanarak makIterasyonSayisi değerine ulaşana kadar aşağıdaki adımlar gerçekleştirilir.

Adım 3.1: İterasyonun "İşçi Arı" evresi için aşağıdaki adımlar gerçekleştirilir.

Adım 3.1.1: İşçi Arı Sayısı kadar aşağıdaki adımlar tekrar edilir

Adım 3.1.1.1: İşçi Arı Deneme Sayısı kadar aşağıdaki adımlar tekrar edilir

Adım 3.1.1.1.1: Popülasyondan rastgele bir çözüm seçilir

Adım 3.1.1.1.2: İşlem seçim $= Rand(0,1]$ aralığında rastgele bir değer üretilir

Adım 3.1.1.1.3: İşlem seçim değeri $0 < r \leq \frac{1}{3}$ aralığında ise "Ekleme Operatörü" gerçekleştirilir.

Adım 3.1.1.1.4: İşlem seçim değeri $\frac{1}{3} < r \leq \frac{2}{3}$ aralığında ise "Yer Değiştirme Operatörü" gerçekleştirilir.

Adım 3.1.1.1.5: İşlem seçim değeri $\frac{2}{3} < r \leq 1$ aralığında ise "Alt Dizi Ekleme Operatörü" gerçekleştirilir.

Adım 3.1.1.1.6: İşlem sonucu elde edilen aday çözüm Adım 3.1.1.1.1'deki (üzerinde operatör çalışmadan önceki) çözümden daha iyi ise popülasyondaki ilgili çözüm ile yer değiştirilir.

Adım 3.2: İterasyonun "Gözcü Arı" evresi için aşağıdaki adımlar gerçekleştirilir.

Adım 3.2.1: Gözcü Arı Sayısı kadar aşağıdaki adımlar tekrar edilir.

Adım 3.2.1.1: Gözcü Arı Deneme Sayısı kadar aşağıdaki adımlar tekrar edilir.

Adım 3.2.1.1.1. Popülasyonun en iyi çözümü seçilir, Bbest.

Adım 3.2.1.1.2: İşlem Seçim= $Rand(0,1]$ aralığında rastgele bir r değeri üretilir.

Adım 3.2.1.1.3: İşlem seçim değeri $0 < r \leq \frac{1}{3}$ aralığında ise "Ekleme Operatörü" gerçekleştirilir.

Adım 3.2.1.1.4: İşlem seçim değeri $\frac{1}{3} < r \leq \frac{2}{3}$ aralığında ise "Yer Değiştirme Operatörü" gerçekleştirilir.

Adım 3.2.1.1.5: İşlem seçim değeri $\frac{2}{3} < r \leq 1$ aralığında ise "Alt Dizi Ekleme Operatörü" gerçekleştirilir.

Adım 3.2.1.1.6: İşlem sonucu elde edilen aday çözüm Adım 3.2.1.1.1'deki (üzerinde operatör çalışmadan önceki) çözümden daha iyi ise popülasyondaki ilgili çözüm ile yer değiştirilir.

Adım 3.3: İterasyonun "Kâşif Arı" evresi için aşağıdaki adımlar gerçekleştirilir

Adım 3.3.1: Kâşif Arı Rastgele-EnYakın Komşu değerinin seçimine göre Rastgele veya En Yakın Komşu yöntemi ile Aday Popülasyon oluşturulur.

Adım 3.3.2: Aday popülasyon ile algoritmanın ana popülasyon birleştirilerek çözümler uygunluk değerine göre sıralanır ve en iyi popülasyon boyutu adet birey yeni ana popülasyon olarak belirlenir.

Adım 4: Popülasyonun en iyi yiyecek kaynağı algoritmanın nihai sonucunu barındıran çözüm olarak belirlenir.

ZPARP için Geliştirilen YAKA'nın Küçük Boyutlu Bir Örnek Üzerinde Gösterimi

Bölüm 3.1.4.'te anlatılanların daha kapsamlı anlaşılabilmesi açısından 10 müşteri ve bu müşterilerin her birine ait X, Y koordinatı, servis süresi, talep miktarı, açılış zamanı ve kapanış zamanı bilgileri aşağıdaki tabloda verilmiştir. Bu bilgiler doğrultusunda 10 araç ve her bir aracın kapasitesi 70 adet olacak şekilde problemin çözümü anlatılacaktır. Müşteriler ile ilgili bilgiler Tablo 3'te ifade edilmiştir. En yakın komşu sezgiseli kullanılarak oluşturulmuş olan başlangıç çözümlerden biri Şekil 16'da gösterilmiştir. Başlangıç çözümünde oluşturulan rotalarda yer alan "0" (sıfır) ile depo ifade edilmektedir.

Tablo 3: Depo ve Müşteri Bilgileri

No	X Koordinatı	Y Koordinatı	Servis Süresi	Talep Miktarı	Açılış Zamanı	Kapanış Zamanı
0(Depo)	40	50	0	0	0	1236
1	45	68	90	10	912	967
2	45	70	90	30	825	870
3	42	66	90	10	65	146
4	42	68	90	10	727	782
5	42	65	90	10	15	67
6	40	69	90	20	621	702
7	40	66	90	20	170	225
8	38	68	90	20	255	324

9	38	70	90	10	534	605
10	35	66	90	10	357	410

Müşterilerin depoya ve birbirlerine olan uzaklığın hesaplanmasında öklidyen uzaklık kullanılmıştır. Söz konusu uzaklıklar Tablo 4'te verilmiştir.

Tablo 4: Uzaklık Matrisi

	Depo	**1.M**	**2. M**	**3. M**	**4. M**	**5. M**	**6. M**	**7. M**	**8.M**	**9.M**	**10.M**
Depo	0,00	18,68	20,62	16,12	18,11	15,13	19,00	16,00	18,11	20,10	16,76
1.M	18,68	0,00	2,00	3,61	3,00	4,24	5,10	5,39	7,00	7,28	10,20
2.M	20,62	2,00	0,00	5,00	3,61	5,83	5,10	6,40	7,28	7,00	10,77
3.M	16,12	3,61	5,00	0,00	2,00	1,00	3,61	2,00	4,47	5,66	7,00
4.M	18,11	3,00	3,61	2,00	0,00	3,00	2,24	2,83	4,00	4,47	7,28
5.M	15,13	4,24	5,83	1,00	3,00	0,00	4,47	2,24	5,00	6,40	7,07
6.M	19,00	5,10	5,10	3,61	2,24	4,47	0,00	3,00	2,24	2,24	5,83
7.M	16,00	5,39	6,40	2,00	2,83	2,24	3,00	0,00	2,83	4,47	5,00
8.M	18,11	7,00	7,28	4,47	4,00	5,00	2,24	2,83	0,00	2,00	3,61
9.M	20,10	7,28	7,00	5,66	4,47	6,40	2,24	4,47	2,00	0,00	5,00
10.M	16,76	10,20	10,77	7,00	7,28	7,07	5,83	5,00	3,61	5,00	0,00

Örneğin depo ile 2 numaralı müşterinin birbirine olan uzaklığı şu şekilde hesaplanacaktır:

$$d(Depo, Müşteri\ 2) = \sqrt{(x_0 - x_2)^2 + (y_0 - y_2)^2}$$
$$d(Depo, Müşteri\ 2) = \sqrt{(40 - 45)^2 + (50 - 70)^2}$$
$$= \sqrt{425}$$
$$\cong 20{,}62$$

YK_1:	0	4	2	1	0	3	7	8	10	9	0	5	6	0

Şekil 16: EYK ile Oluşturulmuş Başlangıç Yiyecek Kaynaklarından (Çözüm) Biri

EYK ile oluşturulan YK_1'de rota 1'in nasıl oluşturulduğuna bakalım.

En yakın komşu sezgiseli ile oluşturulmuş başlangıç çözümde 1'inci araç hareket ettirilir ve bu araca ait bilgiler şu şekildedir: Başlangıç noktası depo (0) ile bir rota oluşturulur, kullanılan kapasite miktarı ve şimdiki zamanı 0'dır. Müşteri listesinden 4 nolu müşteri rastgele seçilir ve bu müşterinin herhangi bir araca tahsis edilmemiş olmasından dolayı kısıtların sağlanıp sağlanmadığı sırasıyla kontrol edilir. Söz konusu müşteri 1. aracın rotasında olursa, müşterinin talebini karşılamak için aracın taşıması gereken yük miktarı 10 adet olacaktır ve bu miktar aracın kapasitesi 70'dan küçüktür, yani ilk kısıt sağlanmıştır. Diğer taraftan araç bulunduğu depodan bu müşteriye müşterinin kapanış zamanından önce varabilmelidir. Depo ve müşteri 4 arasındaki mesafe 18,11 uzaklık birimi ve süre olarak alındığında ise 18,11 zaman birimine karşılık gelecektir. Yani araç müşteriye ulaştığında şimdiki zamanı 18,11 olacaktır, bu da müşterinin kapanış zamanı olan 782 zaman biriminden öncedir, ikinci kısıt da sağlanmaktadır. Fakat araç talebinin karşılanacağı müşterinin açılış zamanına kadar beklemek durumundadır. Böylece müşteriye olan teslimat en erken 727 zaman biriminde başlayacaktır. Son olarak deponun son kapanış zamanı kısıtının sağlanması gerekir. Eğer müşteri aracın son müşterisi ise araç teslimat yaptıktan sonra depoya dönecek olduğundan bu işlem deponun kapanış zamanından önce yapılmalıdır. 4 nolu müşteriye teslimat 727 zaman biriminde olacaktır, 90 zaman birim teslimat süresinden sonra 817 zaman biriminde depoya geri dönmek üzere yola çıkacaktır. Aralarındaki uzaklık simetrik olduğundan dönüşte de 18,11 zaman birimlik mesafe olacağından ve araç depoya dönecekse 835,11 zaman biriminde depoda

olacaktır ki bu da deponun kapanış zamanı olan 1236 zaman biriminden öncedir. Son kısıt da böylece sağlanmış oldu. Dolayısıyla birinci aracın rotasında depodan sonraki tahsis edilen ilk müşteri 4 no'lu müşteri olmuş oldu. Mevcut konumu 4 nolu müşteri olan aracın kullanılan kapasitesi 10 adet ve şimdiki zamanı 817 zaman birimi olacaktır.

Bir sonraki müşterinin sorgulanmasında depoya dönüş süresi dâhil edilmeyecektir. Aracın rotasındaki ikinci sıraya uygunluğu kontrol edilecektir. Müşteri 4 ise bir araca atanmış olarak müşteri listesinden kaldırılacaktır. 4 nolu müşterinin zaman kısıtı ve aracın kapasite kısıtı göz önünde bulundurularak, bir sonraki müşterinin seçiminde rotanın mesafe değeri ve süre değeri hesaplanacak ve bu değerlerden büyük olanı bir sonraki müşterinin seçim değeri(maliyet değeri) olacaktır. Seçim değeri(maliyet değeri) en küçük olan müşteri rotaya eklenir. Rotanın kapalı müşteri sorununun olup olmadığına bakılır. Rotada kapalı müşteri sorunu varsa müşteri rotadan çıkartılır ve cezalı olarak belirlenir. Müşteri listesindeki 4 nolu müşteriden sonraki gidilmeyen 1, 2, 3, 5, 6, 7, 8, 9 ve 10 nolu müşteriler için rotanın mesafe ve süre değeri aşağıdaki gibi hesaplanır. Minimum maliyete sahip müşteri rotaya eklenir.

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{7,00\ [357,410]}{\rightarrow} \underbrace{10} \leftrightarrow \begin{cases} md = 25,11 \\ sd = 914 \end{cases} \rightarrow maliyet\ değeri = max\{25,11; 914\} = 914$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{4,47\ [534,605]}{\rightarrow} \underbrace{9} \leftrightarrow \begin{cases} md = 22,58 \\ sd = 911,47 \end{cases} \rightarrow maliyet\ değeri = max\{22,58; 911,47\} = 911,47$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{4,00\ [255,324]}{\rightarrow} \underbrace{8} \leftrightarrow \begin{cases} md = 22,11 \\ sd = 911 \end{cases} \rightarrow maliyet\ değeri = max\{22,11; 911\} = 911$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{2,83\ [170,225]}{\rightarrow} \underbrace{7} \leftrightarrow \begin{cases} md = 20,94 \\ sd = 909,83 \end{cases} \rightarrow maliyet\ değeri = max\{20,94; 909,83\} = 909,83$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{2,24\ [621,702]}{\rightarrow} \underbrace{6} \leftrightarrow \begin{cases} md = 30,35 \\ sd = 909,24 \end{cases} \rightarrow maliyet\ değeri = max\{20,35; 909,24\} = 909,24$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{3,00\ [15,67]}{\rightarrow} \underbrace{5} \leftrightarrow \begin{cases} md = 21,11 \\ sd = 910 \end{cases} \rightarrow maliyet\ değeri = max\{21,11; 910\} = 910$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{2,00\ [65,146]}{\rightarrow} \underbrace{3} \leftrightarrow \begin{cases} md = 20,11 \\ sd = 909 \end{cases} \rightarrow maliyet\ değeri = max\{20,11; 909\} = 909$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{3,61\ [825,870]}{\rightarrow} \underbrace{2} \leftrightarrow \begin{cases} md = 21,72 \\ sd = 915 \end{cases} \rightarrow maliyet\ değeri = max\{21,72; 915\} = 915$

$0 \underset{18,11\ [727,782]}{\rightarrow} \underbrace{4} \underset{3,00\ [912,967]}{\rightarrow} \underbrace{1} \leftrightarrow \begin{cases} md = 21,11 \\ sd = 1002 \end{cases} \rightarrow maliyet\ değeri = max\{21,11; 1002\} = 1002$

Yapılan hesaplamalar sonrasında müşterilerin maliyet değerleri küçükten büyüğe doğru aşağıdaki gibi sıralanmıştır.

$$M_3 < M_6 < M_7 < M_5 < M_8 < M_9 < M_{10} < M_2 < M_1$$

Burada minimum maliyet değerine sahip olan 3 nolu müşteri rotaya eklenir. Ancak rotanın kapalı müşteri sorunu olduğundan dolayı 3 nolu müşteri rotadan çıkartılır ve cezalı olarak belirlenir. 3 nolu müşteriden sonraki en küçük maliyet değerine sahip olan müşteriler sırasıyla 6, 7, 5, 8, 9, 10 nolu müşteriler olup, bu müşterilerde sırasıyla rotaya eklenmiş ve bu işlemler sırasında yine rotanın kapalı müşteri sorunu ile karşılaşıldığından seçilen her bir müşteri rotadan çıkartılmış ve cezalı olarak tespit edilmiştir.

Müşteri listesinde kalan 1 ve 2 nolu müşterilerin maliyet değerleri sırasıyla 1002 ve 915 birim zaman olarak hesaplanmıştır. Burada seçim (maliyet) değeri küçük olan müşteri 2 nolu müşteri olduğundan 4 nolu müşteriden sonraki müşteri 2 nolu müşteridir. Kısıtlara sırasıyla bakıldığında aracın kapasite kısıtı

sağlanmaktadır. Araç 2 nolu müşteriye geldiğinde 3.61 birim zaman geçecek olduğundan aracın yeni zamanı 820,61 birim zamandır ki bu, müşterinin zaman penceresi olan (825,870)'i geçmemiştir. Dolayısıyla zaman penceresi kısıtı da sağlanmıştır. 2 nolu müşterinin açılış zamanı 825 birim olduğu için araç bu zamana kadar bekleyecektir. Bu müşterinin servise başlama zamanı 825 birim olup, 90 birimlik teslimat süresinden sonra aracın şimdiki zaman birimi 915 ve aracın toplam kapasitesi 40 adet olacaktır. Dolayısıyla kısıtlar sağlandığından aracın rotasındaki ikinci müşteri 2 no'lu müşteri olarak listeden silinerek rotaya eklenecektir. 2 nolu müşterinin zaman kısıtı göz önünde bulundurularak en yakın komşu sezgiseli ile rotaya eklenecek bir sonraki müşteri 1 nolu müşteridir.

$$0 \underset{18{,}11\ [727{,}782]}{\rightarrow} \underset{}{4} \underset{3{,}61\ [825{,}870]}{\rightarrow} \underset{}{2} \underset{2{,}00\ [912{,}967]}{\rightarrow} \underset{}{1} \leftrightarrow \begin{cases} md = 23{,}72 \\ sd = 1007 \end{cases} \rightarrow maliyet\ değeri = 1007$$

Araç 2 nolu müşteriden 1 nolu müşteriye gittiğinde 2,00 birim zaman geçecek ve yeni zaman 917 birim olacaktır. Aracın şimdiki zaman birimi 1 nolu müşterinin (912, 967) zaman aralığında olduğu için müşterinin talebi karşılanacaktır. 90 birimlik teslimat süresinden sonra aracın şimdiki zaman birimi 1007 ve aracın mevcut kullanılan kapasitesi 60 adet olduğundan, aracın 2 nolu müşteriden sonraki rotası 1 nolu müşteri olmuş oldu. 1 nolu müşteri de böylece müşteri listesinden silinmiş oldu. Listede kalan 5, 3, 7, 8, 10, 9, 6 numaralı müşteriler sırasıyla kapasite ve zaman penceresi kısıtlarını sağlamadıklarından dolayı ilk aracın rotasındaki ilgili sıraya atanamamışlardır. Sonuç olarak araç 1'in rotasında sırasıyla 4, 2, 1 nolu müşteriler yer almış olup, araç depodan sırasıyla müşteri 4, müşteri 2, müşteri 1'e oradan da tekrar depoya dönecektir.

$$0 \underset{18{,}11\ [727{,}782]}{\rightarrow} \underset{}{4} \underset{3{,}61\ [825{,}870]}{\rightarrow} \underset{}{2} \underset{2{,}00\ [912{,}967]}{\rightarrow} \underset{}{1} \underset{18{,}68}{\rightarrow} 0$$

5, 3, 7, 8, 10, 9, 6 numaralı müşteriler henüz bir rotaya atanmamış olduklarından sıradaki araçlara problemin kısıtlarını sağlayacak şekilde rotaların ilk müşterisi rastgele sonraki müşteriler en yakın komşu yöntemi ile seçilerek, 1. aracın rotasında yapılan işlemlere benzer şekilde tahsis edilirler. Söz konusu araçların başlangıç noktaları depodur ve mevcut kullanılan kapasiteleri ve şimdiki zamanları 0'dır.

Bu örnek için 3 araç aktif olarak kullanılmış ve 10 müşterinin bu üç araca tahsis edilmesi sonrasında 3 rota oluşturulmuştur ve rotalar Şekil 17'de gösterilmiştir. Oluşturulan rotalar ve toplam mesafe şu şekildedir:

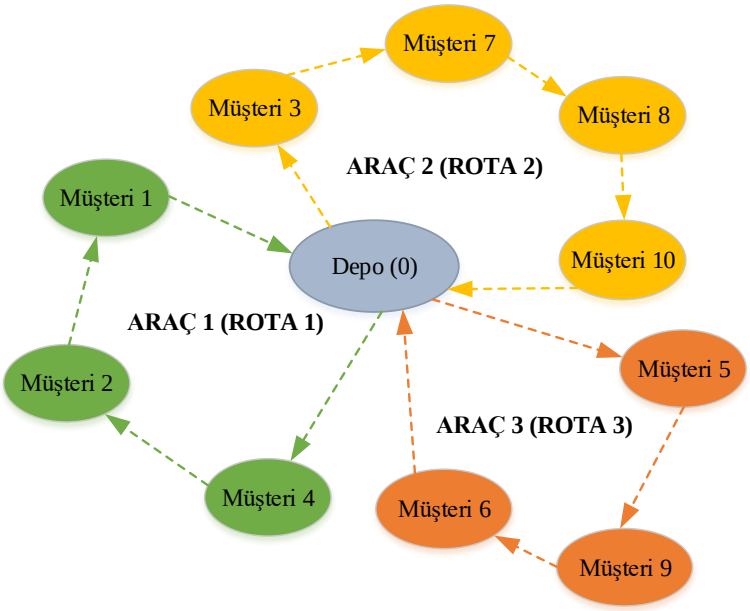
Araç 1 (Rota 1): $Depo\ (0) \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow Depo\ (0)$

Araç 2 (Rota 2): $Depo\ (0) \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 10 \rightarrow 9 \rightarrow Depo\ (0)$

Araç 3 (Rota 3): $Depo\ (0) \rightarrow 5 \rightarrow 6 \rightarrow Depo\ (0)$

$$\begin{aligned} f_1 = Toplam\ Mesafe = {} & d(Depo, Müşteri\ 4) + d(Müşteri\ 4, Müşteri\ 2) \\ & + d(Müşteri\ 2, Müşteri\ 1) + d(Müşteri\ 1, Depo) \\ & + d(Depo, Müşteri\ 3) + d(Müşteri\ 3, Müşteri\ 7) \\ & + d(Müşteri\ 7, Müşteri\ 8) + d(Müşteri\ 8, Müşteri\ 10) \\ & + d(Müşteri\ 10, Müşteri\ 9) + d(Müşteri\ 9, Depo) \\ & + d(Depo, Müşteri\ 5) + d(Müşteri\ 5, Müşteri\ 6) \\ & + d(Müşteri\ 6, Depo) \end{aligned}$$

$$f_1 = 18{,}11 + 3{,}61 + 2{,}00 + 18{,}68 + 16{,}12 + 2{,}00 + 2{,}83 + 3{,}61 + 5 + 20{,}10 + 15{,}13 + 4{,}47 + 19{,}00 = 130{,}66$$



Şekil 17: Örnek Rotaları

Başlangıç yiyecek kaynaklarından birisi yukarıda belirtildiği gibi en yakın komşu sezgiseli ile oluşturulmuş ve bu yiyecek kaynağının uygunluk değeri problemin amaç fonksiyonuna göre hesaplanmıştır. Oluşturulan başlangıç yiyecek kaynağında (çözüm) 3 araç aktif olarak kullanılmış ve 10 müşteri 3 araca tahsis edilmiştir. Böylelikle 3 rota oluşturulmuştur. Diğer iki yiyecek kaynağı da EYK ile YK_1'e benzer şekilde oluşturulmuş ve uygunluk değerleri amaç fonksiyonuna göre hesaplanmıştır. Bu yiyecek kaynakları ve uygunluk değerleri aşağıda ifade edilmiştir.

YK_2:	0	1	0	2	0	5	3	7	8	10	0	4	0	9	6	0

f_2:197,49

YK_3:	0	5	3	7	8	10	0	9	6	4	2	0	1	0

f_3:127,50

En yakın komşu algoritması sonrası oluşturulan yiyecek kaynakları

YK_1: 0 4 2 1 0 3 7 8 10 9 0 5 6 0 $\rightarrow f_1 = 130{,}66,$ Araç Sayısı = 3

YK_2: 0 1 0 2 0 5 3 7 8 10 0 4 0 9 6 0 $\rightarrow f_2 = 197{,}49,$ Araç Sayısı = 5

YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 $\rightarrow f_3 = 127{,}50,$ Araç Sayısı = 3

1. İşçi arı 1. Deneme çalışıyor

Seçilen yiyecek kaynağı YK_2 olsun. İşlem seçim değeri $r = 0{,}25$ olsun. Ekleme operatörü çalışıyor. Rastgele seçilen rota 1 den 1 nolu müşteri zaman ve kapasite kısıtını sağlayacak şekilde 2 nolu rotadaki 2 nolu müşterinin arkasına eklenir. Yeni yiyecek kaynağı ve bu yiyecek kaynağına ait uygunluk değeri aşağıdaki gibidir:

$$Rota\ 1: 0 \underbrace{\overset{uzaklık}{\rightarrow}}_{18{,}68} \overbrace{1}^{10}\ \underbrace{\overset{uzaklık}{\rightarrow}}_{18{,}68} 0; \quad Rota\ 2: 0 \underbrace{\overset{uzaklık}{\rightarrow}}_{20{,}62} \overbrace{2}^{30}\ \underbrace{\overset{uzaklık}{\rightarrow}}_{20{,}62} 0$$

[912,967] [825,870]

YK_2^1: 0 2 1 0 5 3 7 8 10 0 4 0 9 6 0 $\rightarrow f_2^1 = 160{,}19$, Araç Sayısı = 4

Burada, $f_2^1 < f_2$ olduğundan YK_2'nin yerine YK_2^1 geçerek yiyecek kaynağı güncellenir.

YK_1: 0 4 2 1 0 3 7 8 10 9 0 5 6 0 $\rightarrow f_1 = 130{,}66$, Araç Sayısı = 3

YK_2^1: 0 2 1 0 5 3 7 8 10 0 4 0 9 6 0 $\rightarrow f_2 = 160{,}19$, Araç Sayısı = 4

YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 $\rightarrow f_3 = 127{,}50$, Araç Sayısı = 3

2. İşçi arı 1. Deneme çalışıyor.

Seçilen yiyecek kaynağı YK_2^1 ve işlem seçim değeri $r = 0{,}70$ olsun. Alt diziyi rastgele ekleme operatörü çalışıyor.

YK_2^1:

0	2	1	0	5	3	7	8	10	0	4	0	9	6	0

Bu yiyecek satırında bir depo ve müşterilerden oluşan dizi problemin kısıtlarını sağlayacak şekilde

4	0

$\overset{10}{\underset{[727,782]}{4}}$ $\underset{18{,}11}{\overset{uzaklık}{\rightarrow}}$ 0

olarak rastgele seçilsin. Seçilen bu dizi 4. rotaya zaman ve kapasite kısıtlarını sağlayacak şekilde 6 nolu müşteriden sonraki konuma yerleştirilir.

$$0 \underset{20{,}10}{\overset{uzaklık}{\rightarrow}} \overset{10}{\underset{[534,605]}{9}} \underset{2{,}24}{\overset{uzaklık}{\rightarrow}} \overset{20}{\underset{[621,702]}{6}} \underset{2{,}24}{\overset{uzaklık}{\rightarrow}} \overset{10}{\underset{[727,782]}{4}} \underset{18{,}11}{\overset{uzaklık}{\rightarrow}} 0\ 0$$

Uygulanan bu alt diziyi rastgele ekleme operatörü sonrası oluşturulan yeni çözüm satırı;

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32$, Araç Sayısı = 3

olarak elde edilmiştir. $f_2^{1^1} < f_2^1$ olduğundan yiyecek kaynağı $YK_2^{1^1}$ olarak güncellenmiştir.

Yiyecek Kaynakları (Seçilen yiyecek kaynağı iyileştirildi, popülasyon güncellendi)

YK_1: 0 4 2 1 0 3 7 8 10 9 0 5 6 0 $\rightarrow f_1 = 130{,}66$, Araç Sayısı = 3

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32$, Araç Sayısı = 3

YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 $\rightarrow f_3 = 127{,}50$ Araç Sayısı = 3

3. İşçi arı 1. Deneme çalışıyor.

Seçilen yiyecek kaynağı YK_1 ve işlem seçim değeri $r = 0{,}50$ olsun. Yer değiştirme operatörü çalışıyor.

YK_1:

0	4	2	1	0	3	7	8	10	9	0	5	6	0

Yiyecek kaynağında rastgele seçilen 2. rotadaki 3 nolu müşteri ile 3. rotadaki 5 nolu ilgili rotalardan çıkarılır ve yer değiştirerek rotalara eklenir. Buradaki çıkarma ve ekleme işlemleri rotalardaki araç kapasitesi, müşterilerin ve deponun zaman penceresi kısıtları göz önüne alınarak yapılır. Tüm bu işlemler sonucunda oluşturulan yiyecek kaynağı, bu kaynağa ilişkin uygunluk değeri ve araç sayısı;

$$Rota\ 2: 0 \underset{16{,}12}{\overset{uzaklık}{\rightarrow}} \overset{10}{\underset{[65,146]}{3}} \underset{2{,}00}{\overset{uzaklık}{\rightarrow}} \overset{20}{\underset{[170,225]}{7}} \underset{2{,}83}{\overset{uzaklık}{\rightarrow}} \overset{20}{\underset{[255,324]}{8}} \underset{3{,}61}{\overset{uzaklık}{\rightarrow}} \overset{10}{\underset{[357,410]}{10}} \underset{5}{\overset{uzaklık}{\rightarrow}} \overset{10}{\underset{[534,605]}{9}} \underset{20{,}10}{\overset{uzaklık}{\rightarrow}} 0$$

$$Rota\ 3: 0 \underset{15{,}13}{\overset{uzaklık}{\rightarrow}} \overset{10}{\underset{[15,67]}{5}} \underset{4{,}47}{\overset{uzaklık}{\rightarrow}} \overset{20}{\underset{[621,702]}{6}} \underset{19{,}00}{\overset{uzaklık}{\rightarrow}} 0$$

YK_1^1: 0 4 2 1 0 5 7 8 10 9 0 3 6 0 $\rightarrow f_1^1 = 128{,}8$, Araç Sayısı = 3

olarak elde edilmiştir. $f_1^1 < f_1$ olduğundan yeni yiyecek kaynağı YK_1^1 bir önceki yiyecek kaynağının yerini almıştır.

İşçi Arı Evresi Sonrası Yiyecek Kaynakları (Seçilen yiyecek kaynağı iyileştirildi, popülasyon güncellendi)

YK_1^1: 0 4 2 1 0 5 7 8 10 9 0 3 6 0 → $f_1^1 = 128{,}8$, Araç Sayısı = 3

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 → $f_2^{1^1} = 125{,}32$, Araç Sayısı = 3

YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 → $f_3 = 127{,}50$, Araç Sayısı = 3

1. Gözcü Arı 1. Deneme çalışıyor.

Seçilen en iyi yiyecek kaynağı en düşük amaç fonksiyonuna sahip $YK_2^{1^1}$ ve işlem seçim değeri $r = 0{,}24$ olsun. Ekleme operatörü çalışıyor.

$YK_2^{1^1}$:	0	2	1	0	5	3	7	8	10	0	9	6	4	0

Bu yiyecek kaynağında 2 nolu müşteri zaman ve kapasite kısıtını sağlayacak şekilde birinci rotadan rastgele seçilerek çıkartılır. 2 nolu müşteri çözüm satırında yer alan bu ve diğer rotalardaki müşterilerin zaman kısıtlarını ve her bir aracın kapasite kısıtını sağlayacak şekilde rastgele üçüncü rotaya 4 nolu müşteriden sonra eklenerek yeni çözüm satırı

$$\overset{30}{\underbrace{2}}_{[825{,}870]} \leftrightarrow 0 \overset{uzaklık}{\underbrace{\rightarrow}}_{20{,}10} \overset{10}{\underbrace{9}} \overset{uzaklık}{\underbrace{\rightarrow}}_{2{,}24} \overset{20}{\underbrace{6}} \overset{uzaklık}{\underbrace{\rightarrow}}_{2{,}24} \overset{10}{\underbrace{4}} \overset{uzaklık}{\underbrace{\rightarrow}}_{3{,}61} \overset{30}{\underbrace{2}} \overset{uzaklık}{\underbrace{\rightarrow}}_{20{,}62} 0$$

$YK_2^{1^{11}}$: 0 1 0 5 3 7 8 10 0 9 6 4 2 0 → $f_2^{1^{11}} = 127{,}50$ Araç Sayısı = 3

şeklinde elde edilir. Burada, $f_2^{1^1} < f_2^{1^{11}}$ olduğundan ekleme operatörü ile oluşturulan çözüm bir önceki çözümden iyi olmadığından dolayı yeni çözüm eski çözüm ile yer değiştiremez.

2.Gözcü arı 1. Deneme çalışıyor.

Seçilen en iyi yiyecek kaynağı en düşük amaç fonksiyonuna sahip $YK_2^{1^1}$ ve işlem seçim değeri $r = 0{,}44$ olsun. Yer değiştirme operatörü çalışıyor.

$YK_2^{1^1}$:	0	2	1	0	5	3	7	8	10	0	9	6	4	0

Yiyecek kaynağında rastgele seçilen 3. rotadaki 9 nolu müşteri ile 2. rotadaki 10 nolu ilgili rotalardan çıkarılır ve yer değiştirerek rotalara eklenir. Buradaki çıkarma ve ekleme işlemleri rotalardaki araç kapasitesi, müşterilerin ve deponun zaman penceresi kısıtları göz önüne alınarak yapılır. Tüm bu işlemler sonucunda yeni çözüm satırı

$$0 \overset{uzaklık}{\underbrace{\rightarrow}}_{15{,}13} \overset{10}{\underbrace{5}}_{[15{,}67]} \overset{uzaklık}{\underbrace{\rightarrow}}_{1{,}00} \overset{10}{\underbrace{3}}_{[65{,}146]} \overset{uzaklık}{\underbrace{\rightarrow}}_{2{,}00} \overset{20}{\underbrace{7}}_{[170{,}225]} \overset{uzaklık}{\underbrace{\rightarrow}}_{2{,}83} \overset{20}{\underbrace{8}}_{[255{,}324]} \overset{uzaklık}{\underbrace{\rightarrow}}_{3{,}61} \overset{10}{\underbrace{10}}_{[357{,}410]} \overset{uzaklık}{\underbrace{\rightarrow}}_{16{,}76} 0$$

$$0 \overset{uzaklık}{\underbrace{\rightarrow}}_{20{,}10} \overset{10}{\underbrace{9}}_{[534{,}605]} \overset{uzaklık}{\underbrace{\rightarrow}}_{2{,}24} \overset{20}{\underbrace{6}}_{[621{,}702]} \overset{uzaklık}{\underbrace{\rightarrow}}_{2{,}24} \overset{10}{\underbrace{4}}_{[727{,}782]} \overset{uzaklık}{\underbrace{\rightarrow}}_{18{,}11} 0$$

$YK_2^{1^{12}}$: 0 2 1 0 5 3 7 8 9 0 10 6 4 0 → $f_2^{1^{12}} = 127{,}30$ Araç Sayısı = 3

olarak bulunur. Burada, $f_2^{1^1} < f_2^{1^{12}}$ olduğundan yer değiştirme operatörü ile oluşturulan çözüm bir önceki çözümden iyi olmadığından dolayı hafızaya alınmaz.

3. Gözcü arı 1. Deneme çalışıyor

Seçilen en iyi yiyecek kaynağı en düşük amaç fonksiyonuna sahip $YK_2^{1^1}$ ve işlem seçim değeri $r = 0{,}83$ olsun. Alt diziyi rastgele ekleme operatörü çalışıyor.

$YK_2^{1^1}$:	0	2	1	0	5	3	7	8	10	0	9	6	4	0

Bu yiyecek satırında bir depo ve müşterilerden oluşan dizi problemin kısıtlarını sağlayacak şekilde

2	1	0

olarak rastgele seçilsin. Seçilen bu dizi 3. rotaya zaman ve kapasite kısıtlarını sağlayacak şekilde 6 nolu müşteriden sonraki konuma yerleştirilir. Yeni yiyecek kaynağı

$$0 \underbrace{\overset{uzaklık}{\rightarrow}}_{20,10} \overset{10}{\underbrace{9}}_{[534,605]} \underbrace{\overset{uzaklık}{\rightarrow}}_{2,24} \overset{20}{\underbrace{6}}_{[621,702]} \underbrace{\overset{uzaklık}{\rightarrow}}_{5,10} \overset{30}{\underbrace{2}}_{[825,870]} \underbrace{\overset{uzaklık}{\rightarrow}}_{2,00} \overset{10}{\underbrace{1}}_{[912,967]} \underbrace{\overset{uzaklık}{\rightarrow}}_{18,68} 0$$

$YK_2^{1^{13}}$: 0 5 3 7 8 10 0 9 6 2 1 0 4 0 $\rightarrow f_2^{1^{13}} = 125{,}67,$ Araç Sayısı = 3

şeklinde olur. Burada, $f_2^{1^1} < f_2^{1^{13}}$ olduğundan alt diziyi rastgele ekleme operatörü ile oluşturulan çözüm bir önceki çözümden daha iyi olmadığından dolayı yeni çözüm eski çözüm ile yer değiştiremez.

Gözcü Arı Evre Sonrası Yiyecek Kaynakları

YK_1^1: 0 4 2 1 0 5 7 8 10 9 0 3 6 0 $\rightarrow f_1^1 = 128{,}8,$ Araç Sayısı = 3

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32,$ Araç Sayısı = 3

YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 $\rightarrow f_3 = 127{,}50,$ Araç Sayısı = 3

✓ Kâşif Arı Evresi

Kâşif arı evresinde yiyecek kaynakları başlangıç evresindeki yiyecek kaynaklarının sayısına eşit olacak şekilde rastgele ve en yakın komşu algoritması ile ayrı ayrı ele alınarak oluşturulacaktır. Başlangıç yiyecek kaynağı evresinde EYK ile yiyecek kaynaklarının nasıl oluşturulduğu belirtildiğinden bu evrede yiyecek kaynaklarının rastgele bir şekilde oluşturulması ele alınacaktır.

Durum 1: Kâşif arı evresindeki yiyecek kaynaklarının rastgele oluşturulması

RYK_1:

0	7	9	4	1	0	2	0	5	3	8	10	6	0

Rastgele oluşturulan Yiyecek kaynağı 1 de rota 1'in nasıl oluşturulduğuna bakalım.

$$0 \underbrace{\overset{uzaklık}{\rightarrow}}_{16,00} \overset{20}{\underbrace{7}}_{[170,225]} \underbrace{\overset{uzaklık}{\rightarrow}}_{4,47} \overset{10}{\underbrace{9}}_{[534,605]} \underbrace{\overset{uzaklık}{\rightarrow}}_{4,47} \overset{10}{\underbrace{4}}_{[727,782]} \underbrace{\overset{uzaklık}{\rightarrow}}_{3,00} \overset{10}{\underbrace{1}}_{[912,967]} \underbrace{\overset{uzaklık}{\rightarrow}}_{18,68} 0$$

Rastgele oluşturulmuş başlangıç çözümde 1'inci araç hareket ettirilir ve bu araca ait bilgiler şu şekildedir: Başlangıç noktası depo (0) ile bir rota oluşturulur, kullanılan kapasite miktarı ve şimdiki zamanı 0'dır. Müşteri listesinden rastgele seçilmiş 7 no'lu müşteriye bakılır, söz konusu müşteri herhangi bir araca atanmadığından kısıtların sağlanıp sağlanmadığına bakılır. Eğer müşteri bu aracın rotasında yer alırsa, aracın taşıması gereken yük miktarı 20 adet olacaktır ve bu miktar aracın kapasitesi 70'dan küçüktür, yani ilk kısıt sağlanmıştır.

Araç depodan bu müşteriye müşterinin kapanış zamanından önce varabilmelidir. Depo ve müşteri 7 arasındaki uzaklık 16,00 uzaklık birimidir ve süre olarak alındığında ise 16,00 zaman birimine karşılık gelecektir. Yani araç müşteriye vardığında şimdiki zamanı 16,00 olacaktır, bu da müşterinin kapanış zamanı olan 225 zaman biriminden öncedir, ikinci kısıt da yerine getirilmiştir. Fakat araç teslimatın gerçekleşebilmesi için müşterinin açılış zamanına kadar beklemek zorundadır. Bu durumda teslimat en erken 170 zaman biriminde başlayacaktır. Üçüncü ve son kısıt deponun son kapanış zamanı kısıtıdır, eğer müşteri bu araca tahsis edilen son müşteri ise araç müşterinin talebini yerine getirdikten sonra depoya geri gidecek olduğundan bu işlem deponun kapanış zamanından önce gerçekleşmelidir. Bu müşteriye teslimat 170 zaman biriminde olacaktır, 90 zaman birim teslimat süresinden sonra 260 zaman biriminde depoya gitmek üzere yola çıkacaktır. Aralarındaki uzaklık simetrik olduğundan dönüşte de 16,00 zaman birimlik mesafe olacağından ve araç depoya dönecekse 276 zaman biriminde depoda

olacaktır ki bu da deponun kapanış zamanı olan 1236 zaman biriminden öncedir. Son kısıt da böylece sağlanmış oldu. Dolayısıyla birinci aracın rotasında depodan sonraki tahsis edilen ilk müşteri 7 no'lu müşteri olmuş oldu. Mevcut konumu 7 nolu müşteri olan aracın, kullanılan kapasitesi 20 adet ve şimdiki zamanı 260 zaman birimi olacaktır.

Bir sonraki müşteri sorgulanırken depoya gidiş süresi ilave edilmeyecektir. Müşterinin rotadaki ikinci sıraya uygunluğu kontrol edilecektir. Müşteri 7 ise bir araca atanmış olarak müşteri listesinden kaldırılacaktır. Bir sonraki sırada 9 numaralı müşteri bulunmaktadır ve herhangi bir aracın rotasına tahsis edilmemiştir. Kısıtlara sırasıyla bakıldığında aracın kapasite kısıtı sağlanmaktadır. Aracın şimdiki zamanı 260, müşterinin zaman penceresi olan (534,605)'i geçmemiştir. Dolayısıyla zaman penceresi kısıtı da sağlanmıştır. Birinci araç 7 numaralı müşteriden 9 numaralı müşteriye gittiğinde 4,47 birim zaman geçecek ve yeni zaman 264,47 birim olacaktır. 9 nolu müşterinin açılış zamanı 534 birim olduğu için araç bu zamana kadar bekleyecektir. Bu müşterinin servise başlama zamanı 534 birim olup, 90 birimlik teslimat süresinden sonra aracın şimdiki zaman birimi 624 ve aracın toplam kapasitesi 30 adet olacaktır. Dolayısıyla kısıtlar sağlandığından dolayı aracın rotasındaki ikinci müşteri 9 no'lu müşteri olarak listeden silinerek rotaya eklenecektir.

Araç 9 nolu müşteriden 4 nolu müşteriye gittiğinde 4,47 birim zaman geçecek ve yeni zaman 628,47 birim olacaktır. Aracın şimdiki zaman birimi 4 nolu müşterinin (727, 782) zaman aralığından önce olduğu için araç müşterinin açılış zamanı 727 ye kadar bekleyecektir. Açılış zamanı geldiğinde aracın şimdiki zamanı 727 birim olacaktır. 90 birimlik teslimat süresinden sonra aracın şimdiki zaman birimi 817 ve aracın mevcut kullanılan kapasitesi 40 adet olduğundan, aracın 9 nolu müşteriden sonraki rotası 4 nolu müşteri olmuş oldu. Araç 4 nolu müşteriden 1 nolu müşteriye gittiğinde 3,00 birimlik zaman geçecek ve yeni zaman 820 birim olacaktır. 1 nolu müşterinin açılış zamanı 912 birim olduğundan dolayı araç bu saate kadar bekleyecek olup, aracın o saat geldiğinde şimdiki zamanı 912 birim olacaktır. 1 nolu müşteri böylece müşteri listesinden silinmiş oldu. Listede kalan 2, 3, 5, 6, 8, 10 numaralı müşteriler sırasıyla kapasite ve zaman penceresi kısıtlarını sağlamadıklarından dolayı ilk aracın rotasındaki ilgili sıraya atanamamışlardır. Sonuç olarak araç 1'in rotasında sırasıyla 7, 9, 4 ve 1 nolu müşteriler yer almış olup, araç depodan sırasıyla müşteri 7, müşteri 9, müşteri 4 ve müşteri 1'e oradan da tekrar depoya dönecektir. 2, 3, 5, 6, 8, 10 numaralı müşteriler henüz bir rotaya atanmamış olduklarından sıradaki araçlara problemin kısıtlarını sağlayacak şekilde rastgele seçilerek, 1. aracın rotasında yapılan işlemlere benzer şekilde tahsis edilirler. Söz konusu araçların başlangıç noktaları depodur ve mevcut kullanılan kapasiteleri ve şimdiki zamanları 0'dır. Diğer yiyecek kaynakları da benzer şekilde rastgele oluşturulur ve uygunluk değerleri amaç fonksiyonuna göre hesaplanır. Rastgele oluşturulan yiyecek kaynaklarına ait çözümler, uygunluk değerleri ve araç sayıları aşağıda ifade edilmiştir.

Yiyecek Kaynakları (Rastgele Kaşif Arı Yiyecek Kaynakları)

RYK_1: 0 7 9 4 1 0 2 0 5 3 8 10 6 0 $\rightarrow f_{ryk1} = 136{,}9,$ Araç Sayısı = 3

RYK_2: 0 9 1 0 3 7 6 4 0 10 2 0 8 0 5 0 $\rightarrow f_{ryk2} = 202{,}16,$ Araç Sayısı = 5

RYK_3: 0 1 0 4 2 0 5 3 7 8 9 0 6 0 10 0 $\rightarrow f_{ryk3} = 194{,}28,$ Araç Sayısı = 5

Yiyecek Kaynakları (Gözcü Arı ve Kâşif Arı Birleştirilmiş Yiyecek Kaynakları)

YK_1^1: 0 4 2 1 0 5 7 8 10 9 0 3 6 0 $\rightarrow f_1^1 = 128{,}8,$ Araç Sayısı = 3
$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32,$ Araç Sayısı = 3
YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 $\rightarrow f_3 = 127{,}50,$ Araç Sayısı = 3

RYK_1: 0 7 9 4 1 0 2 0 5 3 8 10 6 0 $\rightarrow f_{ryk1} = 136{,}9,$ Araç Sayısı = 3

RYK_2: 0 9 1 0 3 7 6 4 0 10 2 0 8 0 5 0 $\rightarrow f_{ryk2} = 202{,}16,$ Araç Sayısı = 5

RYK_3: 0 1 0 4 2 0 5 3 7 8 9 0 6 0 10 0 $\rightarrow f_{ryk3} = 194{,}28,$ Araç Sayısı = 5

En İyi Sonuç

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32$, Araç Sayısı = 3

Durum 2: Kâşif arı evresindeki yiyecek kaynaklarının EYK ile oluşturulması

Yiyecek kaynakları başlangıç evresinde olduğu gibi EYK ile oluşturulmuş, uygunluk değerleri ve araç sayıları ile birlikte bu yiyecek kaynakları aşağıda belirtilmiştir.

Yiyecek Kaynakları (EYK Kâşif Arı Yiyecek Kaynakları)

$EYKYK_1$: 0 6 4 2 1 0 5 3 7 8 10 0 9 0 $\rightarrow f_{eykyk1} = 127{,}6$, Araç Sayısı = 3

$EYKYK_2$: 0 10 9 6 4 1 0 2 0 7 8 0 3 0 5 0 $\rightarrow f_{eykyk2} = 188{,}6$, Araç Sayısı = 5

$EYKYK_3$: 0 4 2 1 0 7 8 10 9 0 5 3 6 0 $\rightarrow f_{eykyk3} = 128{,}68$, Araç Sayısı = 3

Yiyecek Kaynakları (Gözcü Arı ve Kâşif Arı Birleştirilmiş Yiyecek Kaynakları)

YK_1^1: 0 4 2 1 0 5 7 8 10 9 0 3 6 0 $\rightarrow f_1^1 = 128{,}8$, Araç Sayısı = 3

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32$, Araç Sayısı = 3

YK_3: 0 5 3 7 8 10 0 9 6 4 2 0 1 0 $\rightarrow f_3 = 127{,}50$, Araç Sayısı = 3

$EYKYK_1$: 0 6 4 2 1 0 5 3 7 8 10 0 9 0 $\rightarrow f_{eykyk1} = 127{,}6$, Araç Sayısı = 3

$EYKYK_2$: 0 10 9 6 4 1 0 2 0 7 8 0 3 0 5 0 $\rightarrow f_{eykyk2} = 188{,}6$, Araç Sayısı = 5

$EYKYK_3$: 0 4 2 1 0 7 8 10 9 0 5 3 6 0 $\rightarrow f_{eykyk3} = 128{,}68$, Araç Sayısı = 3

En İyi Sonuç

$YK_2^{1^1}$: 0 2 1 0 5 3 7 8 10 0 9 6 4 0 $\rightarrow f_2^{1^1} = 125{,}32$, Araç Sayısı = 3

Ateş Böceği Algoritması (ABA)

Bu bölümde ateş böceklerinin biyolojik temellerine, geceleri yanıp sönen ışıkları sayesinde partnerlerini çekmeleri veya olası tehditlere karşı önlem aldıklarına, temel ateş böceği algoritmasına, ayrık ateş böceği algoritmasına, zaman pencereli araç rotalama problemi için geliştirilen ateş böceği algoritmasına değinilmiş ve geliştirilen algoritmanın işleyişi küçük boyutlu bir örnek üzerinde gösterilmiştir. Diğer taraftan iki ateş böceği arasındaki uzaklığın hesaplanması iki ayrı formülle ele alınmıştır. Temel ateş böceği algoritmasında bu uzaklık Öklid uzaklığı ile ele alınırken ayrık ateş böceği algoritmasında ise Hamming uzaklığı ile hesaplanmıştır. Problem için geliştirilen ateş böceği algoritmasında başlangıç ateş böceği popülasyonu rastgele ve en yakın komşu algoritması olmak üzere iki farklı şekilde oluşturulmuştur. Yeni ateş böceklerinin oluşturulmasında ekleme, yer değiştirme ve $2-opt^*$ operatörlerinden faydalanılmıştır.

Ateş Böceklerinin Biyolojik Temelleri

Ateş böcekleri, tüm böceklerin en karizmatikleri arasındadır ve muhteşem kur gösterileri şairlere ve bilim adamlarına ilham kaynağı olmuştur. Günümüzde dünya çapında 2000'den fazla türü bulunmaktadır. Genellikle, ateş böcekleri çeşitli sıcak ortamlarda yaşar ve en çok yaz gecelerinde aktiftirler. Birçok araştırmacı, doğada ateşböceği olaylarını incelemiş olup, ateş böceklerini araştıran çok sayıda makale literatürde yer almaktadır.

Ateş böcekleri, biyokimyasal proses biyolüminesansı tarafından üretilen yanıp sönen ışıklarıyla karakterize edilir. Bu tür yanıp sönen ışık, çiftleşme için birincil kur sinyalleri olarak işlev görebilir. Yanıp sönen ışık çiftleşme partnerlerini çekmenin yanı sıra, potansiyel yırtıcıları uyarmaya da yarayabilir. Bazı ateş böceği türlerinde bazı yetişkinler biyolüminesanstan yoksundur. Bu türler karıncalara benzer şekilde feromon sayesinde eşlerini çekerler.

Ateş böceklerinde, biyolüminesan reaksiyonlar fener adı verilen ışık üreten organlardan meydana gelir. En biyolüminesan organizmalar sadece yavaş modüle edilmiş flaşlar sağlar (ayrıca parıltılar). Buna karşılık, birçok ateş böceği türündeki yetişkinler, yüksek ve ayrık flaşlar yaymak için biyolüminesanslarını kontrol edebilirler. Fenerlerin ışık üretimi, ateşböceğinin merkezi sinir sisteminden kaynaklanan sinyallerle başlatılır.

Çoğu ateş böceği türleri, biyolüminesans kur sinyallerine güvenir. Tipik olarak, ilk işaretçiler zeminde uçamayan dişileri çekmeye çalışan uçan erkeklerdir. Bu sinyallere yanıt olarak, dişiler sürekli veya yanıp sönen ışıklar yayar. Her iki çiftleşme ortağı da tür kimliği ve cinsiyet gibi bilgileri kodlamak için kesin olarak zamanlanmış farklı flaş sinyal örnekleri üretir. Dişiler, kur sinyalindeki davranış farklılıklarına göre cezbedilmektedir. Tipik olarak, dişiler daha parlak erkek flaşları tercih eder. Flaş yoğunluğunun kaynaktan uzaklığa göre değiştiği iyi bilinmektedir. Neyse ki, bazı ateş böceği türlerinde dişiler daha güçlü ışık kaynağı tarafından üretilen daha uzak flaşlar ile zayıf ışık kaynakları tarafından üretilen daha yakın flaşlar arasında ayrım yapamazlar.

Ateş böceği flaş sinyalleri oldukça dikkat çekicidir ve bu nedenle çok çeşitli potansiyel yırtıcıları caydırabilir. Sadece en güçlü bireyin hayatta kalabileceği doğal seleksiyon anlamında, flaş sinyalleri potansiyel yırtıcıları uyarmaya yarayan savunma mekanizmaları olarak gelişir (Fister vd., 2013: 35-36).

Temel Ateş Böceği Algoritması

Ateş böceği, çoğunlukla biyolüminesans işlemiyle üretilen kısa ve ritmik flaşlar üreten bir böcektir. Ateş böcekleri, yanıp sönen ışıkları sayesinde, partnerlerini ve potansiyel avları kendilerine doğru çekerlerken aynı zamanda bu ışıkları yırtıcıya karşı koruyucu bir önlem olarak kullanırlar. Böylece, ışık yoğunluğu ateş böceklerinin diğer ateş böceklerine doğru hareket etmesi olarak tanımlanabilir.

Işık yoğunluğu, seyredenin gözünden uzaklığa göre değişiklik göstermektedir. Uzaklık arttıkça ışık yoğunluğunun azaldığını söylemek güvenlidir. Işık yoğunluğu aynı zamanda çevre tarafından emilen havanın etkisidir, böylece uzaklık arttıkça yoğunluk daha az çekici hale gelir (Ali vd., 2014: 1732).

İlk olarak Yang (2008, 2009) tarafından geliştirilen temel ateş böceği algoritması, ateş böceklerinin yanıp sönen özelliklerinin ideal davranışlarına dayanır. Algoritmayı doğru bir şekilde anlamak için aşağıdaki üç ideal kuralı belirtmek önemlidir:

- ✓ Sürüdeki tüm ateş böcekleri cinsiyetsiz olarak kabul edilir ve böylece bir ateş böceği cinsiyeti ne olursa olsun diğer ateş böceklerini etkileyebilir.
- ✓ Çekicilik, parlaklık ile orantılıdır, yani herhangi iki ateşböceği için daha parlak olanın daha az parlak olanı çekeceği anlamına gelir. Ateş böcekleri arasındaki uzaklık arttıkça çekicilik azalır. Ayrıca, bir ateş böceği sürünün en parlak olanı ise, rastgele hareket eder.
- ✓ Bir ateş böceğinin parlaklığı doğrudan söz konusu problemin amaç fonksiyonu tarafından belirlenir. Bu şekilde, bir maksimizasyon problemi için parlaklık amaç fonksiyon değeri ile orantılı olabilir. Öte yandan, bir minimizasyon problemi için, amaç fonksiyon değerinin karşıtı olabilir. Yani, daha parlak olan ateş böceği amaç fonksiyon değeri en küçük olan ateş böceğidir.

Algoritma 2'de, Yang (2008) tarafından ileri sürülen temel ateş böceği algoritmasının sözde kodu ifade edilmiştir. Bu doğrultuda, ateş böceği algoritmasında çekicilik, uzaklık ve hareket olmak üzere dikkate alınması gereken üç önemli faktör vardır. Ateş böceği algoritmasının temel modelinde bu faktörler aşağıdaki gibi ele alınmaktadır. Her şeyden önce, bir ateş böceğinin çekiciliği ışık yoğunluğu ile belirlenir ve aşağıdaki formül kullanılarak hesaplanabilir:

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{3.6}$$

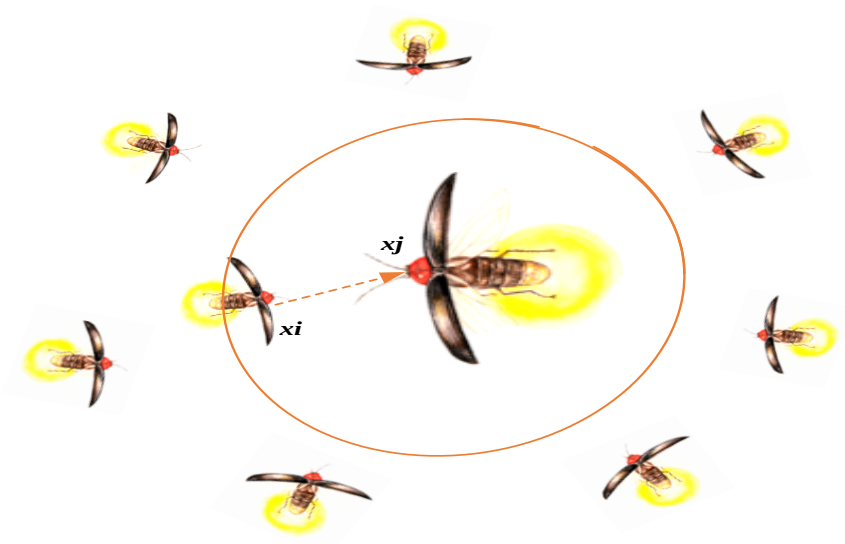
Diğer taraftan, temel ateş böceği algoritmasında herhangi bir iki ateş böceği i ve j arasındaki uzaklık r_{ij}, Kartezyen uzaklığı kullanılarak hesaplanır ve aşağıdaki denklemle ifade edilir:

$$r_{ij} = \|X_i - X_j\| = \sqrt{\sum_{k=1}^{d} (X_{i,k} - X_{j,k})^2} \quad (3.7)$$

Burada $X_{i,k}$, i. ateş böceğinin X_i uzaysal koordinatının k. bileşenidir. Son olarak, bir i ateş böceğinin daha parlak olan j ateş böceğine doğru hareketi ise aşağıdaki denklem ile belirlenir:

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}(X_j - X_i) + \alpha(rand - 0{,}5) \quad (3.8)$$

Burada α, keyfi bir parametre ve $rand$'da [0,1] aralığındaki keyfi bir sayıdır. Öte yandan, denklemin ikinci terimi çekim varsayımından kaynaklanmaktadır (Osaba vd., 2017: 5300-5301). Bir i. ateş böceğinin kendisinden daha parlak olan j. ateş böceğine olan sistematik hareketi Şekil 18'de gösterilmiştir.



Şekil 18: Ateş Böceğinin Sistematik Hareketi
Kaynak: (Yesodha ve Amudha, 2019:165)

Algoritma 2. Temel Ateş Böceği Algoritmasının Kodu	
1	Amaç Fonksiyonu $f(x)$'i tanımla;
2	Ateş böceği popülasyonunu başlat $X = x_1, x_2, \ldots x_n$;
3	Işık emilim katsayısı olan γ'ı tanımla;
4	**for** popülasyondaki her x_i ateş böceği **do**
5	Işık yoğunluğu I_i'yi başlat;
6	**end**
7	**repeat**
8	**for** sürüdeki her x_i ateş böceği **do**
9	**for** sürüdeki diğer her x_j ateş böceği **do**
10	**if** $I_j > I_i$ **then**
11	x_i ateş böceğini x_j ateş böceğine doğru hareket ettir;
12	**end**
13	Çekicilik $e^{-\gamma r}$ üzerinden r uzaklığıyla değişir;
14	Yeni çözümleri değerlendir ve ışık yoğunluğunu güncelle;
15	**end**
16	**end**
17	Ateş böceklerini sırala ve mevcut en iyisini bul;
18	Durdurma kriterine ulaşılana kadar;
19	Ateş böceklerini sırala ve mevcut en iyisine dön;
Kaynak: (Osaba vd., 2017: 5301)	

Ayrık Ateş Böceği Algoritması (AABA)

Klasik ateş böceği algoritması, sürekli optimizasyon problemleri için geliştirilmiştir. Bu sebepten dolayı, çizelgeleme ve araç rotalama gibi farklı ayrık problemlerin çözümünde bu algoritmanın temel versiyonu doğrudan kullanılamaz (Qamhan vd., 2019: 5). Bu sebepten algoritmanın ayrık versiyonunda birtakım değişiklikler yapılmıştır.

İlk olarak, ileri sürülen ayrık ateş böceği algoritmasında sürüdeki her ateş böceği ayrık bir problem için olası ve uygun bir çözümü temsil eder. Ateş böceklerinin tümü rastgele oluşturulur. Problemin amaç fonksiyonuna göre ateş böceklerinin çekiciliği belirlenir. Problem, bir maksimizasyon problemi olduğunda en yüksek amaç fonksiyon değerine sahip ateş böceği en çekici ateş böceği iken problem, bir minimizasyon problemi olduğunda ise en düşük amaç fonksiyon değerine sahip ateş böceği en çekici ateş böceğidir. Ayrıca, ışık emilim kavramı ateş böceği algoritmasının bu versiyonunda da ifade edilmektedir. Bu durumda, $\gamma = 0{,}95$ ışık emilim katsayı parametresi denklem (3.8)'de de görüldüğü gibi kullanılır. Bu parametre, literatürün çeşitli çalışmalarında önerilen ilkelere göre ayarlanmıştır. Bunun yanı sıra, iki farklı ateş böceği arasındaki uzaklık, dizideki karşılık gelmeyen elemanların sayısı olarak ifade edilen Hamming Uzaklığı ile hesaplanır. Örneğin; rastgele iki ateş böceği ve bu ateş böceklerinin 8 düğümden oluşan bir keyfi k dizisi aşağıdaki gibi verilsin:

$$x_1(dizi - k): \{0,1,2,3,4,5,6,7\}$$

$$x_2(dizi - k): \{0,1,3,2,5,4,6,7\}$$

Bu durumda k dizisi için x_1 ve x_2 ateş böceği arasındaki Hamming uzaklığı 4'tür. Bu aynı karşılaştırma her dizi için yapılır. Böylece i ve j ateş böceği arasındaki toplam uzaklık, her dizi için hesaplanan tüm uzaklıkların toplamıdır.

Son olarak, başka bir daha parlak j ateş böceğine çekilen bir i ateş böceğinin hareketi aşağıdaki gibi hesaplanır:

$$n = Random\left(2, r_{ij}.\gamma^{g}\right) \tag{3.9}$$

$$x_i = Ekleme\ Fonksiyonu(x_i, n) \tag{3.10}$$

Burada r_{ij}, i ateş böceği ile j ateş böceği arasındaki Hamming uzaklığı ve g, iterasyon sayısıdır. Bu durumda, bir ateş böceğinin hareket uzunluğu 2 ile $r_{ij}.\gamma^{g}$ arasında rastgele bir sayı olacaktır. Hareket fonksiyonuna gelince, ekleme fonksiyonu kullanılmıştır. Bu fonksiyon, rastgele bir rotadan rastgele bir düğüm seçer ve çıkarır. Sonra, bu düğüm seçilen düğümün dizisinin içinde rastgele bir konuma eklenir. Bu fonksiyon, mümkün olmayan çözümler yaratmamak için kapasite kısıtlamasını dikkate alır. Daha önce gezgin satıcı problemi için geliştirilen ve yayınlanan ateş böceği algoritması ile aynı felsefeye takiben, önerilen ayrık ateş böceği algoritmasındaki ateş böceklerinin hareket yönleri yoktur. Bunun yerine ateş böcekleri evrim stratejilerini kullanarak hareket ederler. Bu şekilde, her bir ateş böceği, n tane varis üreterek ve ekleme fonksiyonunu n kere kullanarak hareket ederler. Bu n hareketten sonra yeni ateş böcekleri üretilerek en iyisi bulunmaya çalışılır.

İleri sürülen ayrık ateş böceği algoritmasının sözde kodu Algoritma 3'te gösterilmektedir. 1-3 arası satırlarda, ateş böceklerinin başlatılması ve değerlendirilmesi aşaması olan başlangıç evresi gerçekleştirilir. Ayrıca, γ parametresi belirlenir. Satır 4'te popülasyondaki her ateş böceği için amaç fonksiyon değeri yani ışık yoğunluğu hesaplanır. Ek olarak, 10-12 satırlarında hareket süreci başlatılır. 10. Satırda seçilen x_i ile x_j arasındaki uzaklık Hamming Uzaklığı ile hesaplanır. Uzaklık elde edildiğinde, 2 ile $r_{ij}.\gamma^{g}$ arasında rastgele bir sayı olan n parametresi hesaplanır. Son olarak hareket daha önce de açıklandığı gibi Ekleme Fonksiyonu kullanılarak 12. satırda gerçekleştirilir. Bu hareket sürecinden sonra ateş böcekleri satır 14'te değerlendirilir ve 17. satırda sıralanır. Bu iterasyon süreci durdurma kriteri karşılanana kadar devam eder. (Osaba vd., 2017: 5301-5302).

Algoritma 3. Ayrık Ateş Böceği Algoritmasının Kodu	
1	Başlangıç ateş böceği popülasyonunu oluştur $X = x_1, x_2, \dots x_n$;
2	$\gamma = 0.95$;
3	**for** popülasyondaki her x_i ateş böceği **do**
4	$I_i = fitness(x_i)$;
5	**end**
6	**repeat**
7	**for** sürüdeki her x_i ateş böceği **do**
8	**for** sürüdeki diğer her x_j ateş böceği **do**
9	**if** $I_j < I_i$ **then**
10	$r_{ij} = HammingDistance(x_i, x_j)$;
11	$n = Random(2, r_{ij}.\gamma^g)$;
12	$x_i = Ekleme\ Fonksiyonu(x_i, n)$;
13	**end**
14	Yeni çözümleri değerlendir ve ışık yoğunluğunu güncelle;
15	**end**
16	**end**
17	Ateş böceklerini sırala ve mevcut en iyisini bul;
18	Durdurma kriterine ulaşılana kadar;
19	Ateş böceklerini sırala ve mevcut en iyisine dön;
Kaynak: (Osaba vd., 2017: 5302)	

ZPARP için Geliştirilmiş Ayrık Ateş Böceği Algoritması

Zaman pencereli araç rotalama problemini çözmek için önerilen ayrık ateş böceği algoritması aşağıdaki gibidir.

Başlangıç Popülasyonunun Oluşturulması

ZPARP için önerilen ateş böceği algoritmasının parametreleri; popülasyon boyutu, maksimum iterasyon sayısı, ışık emilim katsaysısı γ, ilk popülasyon aday bulma deneme sayısı ve $2-opt^*$ iç deneme sayısı algoritmanın performansını kontrol etmek için belirlenir. Başlangıç ateş böceği popülasyonu 2 farklı şekilde belirlenmiştir. Bunlar sırasıyla rastgele başlangıç popülasyonu ve en yakın komşu algoritması yöntemleridir. En yakın komşu sezgiselinin detayları daha önce Bölüm 3.1.4.1.1'de ifade edilmiştir. Başlangıç ateş böceği popülasyonunun rastgele oluşturulması ise aşağıda açıklanmıştır.

Başlangıç Popülasyonunun Rastgele Oluşturulması

Başlangıç bireylerin rastgele oluşturulmasının meta-sezgisel algoritmaların optimizasyon kalitesini kanıtlamada en uygun yolu olduğunu belirtmekte yarar vardır (Osaba vd., 2018: 4). Kapasite kısıtlı araç rotalama problemini ele alan bir çalışmada, ateş böceği algoritmasında sürüdeki ateş böcekleri rastgele olacak şekilde aşağıdaki adımlara göre oluşturulmuştur (Altabeeb vd., 2019: 3):

i. Başlangıç noktası (depo) ile bir rota oluşturulur;

ii. Müşteri listesinden rastgele bir müşteri seçilir ve müşterinin tekrar seçilmemesini sağlamak için aynı listeden seçilen müşteri silinir;

iii. Seçilen müşteri aşağıdaki şartı sağlayacak şekilde rotaya eklenir;

- Rotanın Kapasitesi ≤ Aracın Kapasitesi ise;

Adım **ii.** ve **iii.** tekrar edilir, Aksi halde son nokta (depo) rotaya eklenir.

iv. Rota sayısı 1 arttırılır;

v. Tüm müşterilere hizmet verilene kadar i'den iv'ye kadar olan adımları tekrarla.

Bizim çalışmamızda da sürüdeki her bir ateş böceği yukarıdaki adımlara göre rastgele oluşturulacaktır. Tabi burada iii. adımdaki

- Rotanın Kapasitesi ≤ Aracın Kapasitesi ise;

kısıtına ek olarak problemin zaman penceresi kısıtı da dikkate alınarak seçilen müşteri rotaya eklenecektir.

Işık Yoğunluğunun Hesaplanması ve En Çekici (En İyi) Ateş Böceğinin Tespit Edilmesi

Bir çalışmada, her bir ateş böceğinin ışık yoğunluğu, ateş böceğinin amaç fonksiyon değeri ile belirlenmiştir (Qamhan vd., 2019: 5). Bir diğer çalışmada ise, ateş böceği oluşturulduktan sonra, amaç fonksiyon değerinin hesaplanması, ilgili ateş böceğindeki tüm rotaların toplam uzaklığı olarak yapılmıştır. Her bir rotanın uzaklığı, başlangıçtaki depo ve bitiş noktaları dahil olmak üzere tüm müşterileri arasındaki Öklid uzaklıklarının toplamı ile hesaplanmıştır (Altabeeb vd., 2019: 3).

Bizim çalışmamızda her bir ateş böceğinin ışık yoğunluğunu, ateş böceğinin amaç fonksiyon değeri olarak ele alınıp, amaç fonksiyon değeri de ilgili ateş böceğindeki tüm rotaların toplam uzaklığı olarak hesaplanacaktır. Yine burada da her bir rotanın uzaklığını hesaplamada Öklid uzaklığı kullanılacaktır. Ele aldığımız problem, bir minimizasyon problemi olduğundan en düşük amaç fonksiyon değerine sahip ateş böceği en çekici (en iyi) ateş böceği olarak kabul edilecektir. Bu durumda her bir ateş böceğinin ışık yoğunluğu fonksiyonu Denklem (3.11)'de ifade edilmiştir.

$$l_i = \sum_{j=1}^{k} d_j \, , j = 1,2,3, \dots , k \qquad (3.11)$$

l_i : i. Ateş böceğinin ışık yoğunluğu

d_j: i. Ateş böceğinde düğümler arası uzaklık

Yeni Ateş Böceklerinin Oluşturulması

Yeni ateş böcekleri aşağıdaki adımlara göre oluşturulur.

Adım 1: Her bir ateş böceğinin en iyi ateş böceğine olan hamming uzaklığı hesaplanır. (HD=Depolar hariç karşılık gelmeyen müşterilerin sayısı)

Sürekli optimizasyon problemlerinde iki ateş böceği arasındaki uzaklığı hesaplamada yaygın bir şekilde Öklid uzaklığı kullanılır (Qamhan vd., 2019: 5). Bizim ele aldığımız ayrık problem için bu uzaklık kullanılamaz. Bölüm 3.2.3.'te belirtildiği gibi, ayrık problemler için yaygın olarak kullanılan Hamming uzaklığı ile her bir i ateş böceğinin en iyi ateş böceğine olan uzaklığı aşağıdaki örnekteki gibi hesaplanacaktır.

3 araç ve 10 müşterinin bu araçlara tahsis edilmesiyle oluşturulan x_i ateş böceği (çözüm) ile bu müşterilerin toplam uzaklık minimum olacak şekilde araçlara tahsis edilmesiyle oluşturulan en iyi ateş böceği (çözüm) x_j aşağıdaki gibi verilsin. Burada depolar 0 ile ifade edilmektedir. Bu durumda i. ateş böceğinin en iyi ateş böceğine olan hamming uzaklığı depolar hariç karşılık gelmeyen müşterilerin sayısına eşittir.

x_i: 0 → 1 → 2 → 3 → 0 → 4 → 5 → 6 → 0 → 7 → 8 → 9 → 10 → 0

$x_{EN\ \text{İYİ}}$: 0 → 10 → 2 → 8 → 0 → 7 → 6 → 5 → 4 → 0 → 3 → 9 → 1 → 0

$$HD_{i,EN\ \text{İYİ}} = \begin{cases} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10 \\ 10\ 2\ 8\ 7\ 6\ 5\ 4\ 3\ 9\ 1 \end{cases} \rightarrow HD_{i,EN\ \text{İYİ}} = 8$$

Adım 2: Her bir ateş böceğinden oluşturulacak yeni ateş böceklerinin sayısı (aday sayısı) belirlenir.

Bir i ateş böceğinin en iyi (en çekici) ateş böceğine doğru yapılan hareket uzunluğu aşağıdaki gibi hesaplanır. Bu hareket uzunluğu ile bu i ateş böceğinden oluşturulacak yeni ateş böceklerinin sayısı belirlenir.

$$k = Random\left(2, HD_{i,EN\ \text{İYİ}} . \gamma^{g}\right) \qquad (3.11)$$

Burada $HD_{i,EN\ \text{İYİ}}$, i ateş böceği ile en iyi ateş böceği arasındaki Hamming uzaklığı, g iterasyon sayısı ve γ, ışık emilim katsayısıdır. Bu durumda, oluşturulacak aday sayısı 2 ile $HD_{i,EN\ \text{İYİ}} . \gamma^{g}$ arasında rastgele bir sayı olacaktır.

Adım 3: İşlem seçim değeri r tespit edilir.

İşlem seçim değeri r, (0,1] arasında rastgele bir sayı olarak belirlenir. Bu değer için üç durum söz konusudur.

i. $0 < r \leq \frac{1}{3}$
ii. $\frac{1}{3} < r \leq \frac{2}{3}$
iii. $\frac{2}{3} < r \leq 1$

Adım 4: İşlem seçim değerine göre ateş böceklerine uygulanacak operatörler belirlenir ve bu operatörler aday sayısı (k) kadar ateş böceklerine uygulanarak yeni ateş böcekleri oluşturulur. Oluşturulan ateş böcekleri arasından en iyisi bulunmaya çalışılır.

Ateş böceklerine uygulanacak operatör fonksiyonu aşağıda ifade edilmiştir.

$$x_i = \begin{cases} Ekleme(x_i, k), & 0 < r \leq \frac{1}{3} \\ Yer\ Değiştirme(x_i, k), & \frac{1}{3} < r \leq \frac{2}{3} \\ 2 - opt^{*}(x_i, k), & \frac{2}{3} < r \leq 1 \end{cases} \qquad (3.12)$$

Burada,

i. $0 < r \leq \frac{1}{3}$ ise x_i ateş böceğine k kadar ekleme operatörü uygulanarak yeni ateş böcekleri oluşturulur ve ışık yoğunlukları hesaplanır.

ii. $\frac{1}{3} < r \leq \frac{2}{3}$ ise x_i ateş böceğine k kadar yer değiştirme operatörü uygulanır ve bu ateş böceğinden k tane yeni ateş böceği oluşturulur ve ışık yoğunlukları hesaplanır.

iii. $\frac{2}{3} < r \leq 1$ ise x_i ateş böceğine k kadar $2 - opt^{*}$ operatörü uygulanarak bu ateş böceğinden k tane yeni ateş böceği oluşturulur ve ışık yoğunlukları hesaplanır.

Oluşturulan yeni ateş böceklerinin ışık yoğunluğu amaç fonksiyonuna göre hesaplanır ve ateş böcekleri sıralanarak mevcut en iyisi bulunur.

Yeni ateş böceklerinin oluşturulmasında kullanılan operatörler aşağıdaki bölümde ifade edilmiştir.

Yeni Ateş Böceklerinin Oluşturulmasında Kullanılan Operatörler

i. Ekleme operatörü, rastgele bir rotadan bir müşteri seçer ve çıkarır. Sonra, bu müşteri seçilen müşteri dizisinin içinde rastgele bir konuma eklenir. Bu operatör, mümkün olmayan çözümler yaratmamak için kapasite kısıtlamasını ve zaman penceresi kısıtını dikkate alır (Osaba vd., 2018: 14).

ii. Yer değiştirme operatörü, rastgele iki rotadan iki müşteri seçer ve seçilen bu iki müşterinin yerlerini değiştirir. Bu operatör ekleme operatöründe olduğu gibi mümkün olmayan çözümler oluşturmamak için kapasite ve zaman penceresi kısıtlamalarını göz önünde bulundurmalıdır (Osaba vd., 2018: 15).

iii. $2 - opt^{*}$ operatör, ZPARP için tanıtılmıştır. $2 - opt$'in bir uzantısıdır. Rotaların yönünü korumayı amaçlar. Başlangıçta 2-opt * prosedürü, her rotadaki bir bağlantıyı silerek iki rotayı keser.

Daha sonra, ilk rotadaki ilk müşteri grubu ikinci rotadaki son müşteri grubuna bağlanır ve ikinci rotadaki ilk müşteri grubu ilk rotadaki son müşteri grubuna bağlanır. Yapılan tüm bu işlemler, mümkün olmayan çözümler yaratmamak için kapasite ve zaman penceresi kısıtları dikkate alınarak yapılır (Braysy ve Gendreau, 2005: 111; Wang ve Zhou, 2021: 10).

Durdurma Kriterinin Karşılanıp Karşılanmadığının Kontrolü:

Bu kriter sürecin hangi durumlarda sona ereceğini ifade eder. Bu çalışmada maksimum iterasyon sayısı durdurma kriteri olarak kullanılmıştır.

Geliştiriren ayrık ateş böceği algoritmasına ait adımlar aşağıda ifade edilmiştir.

Adım 1: Kullanıcının program arayüzünden belirlediği değerler ile "Populasyon Boyutu p, Maksimum İterasyon Sayısı g_{max}, Işık Emilim Katsayısı γ, Maksimum Aday Deneme Sayısı, Maksimum $2 - opt^*$ Deneme" değerleri belirlenir.

Adım 2: Kullanıcı ilk popülasyonun oluşumunda "En Yakın Komşu" yöntemini seçmişse 3. Adımdan, "Rastgele" yöntemini seçmişse 4. Adımdan devam edilir.

Adım 3: "En Yakın Komşu" yöntemi ile aşağıdaki adımlar gerçekleştirilerek ilk popülasyon oluşturulur.

Adım 3.1: Depo (0) rotaya eklenir. Gidilmeyen müşteriler bulunur.

Adım 3.2: Gidilmeyen müşteriler arasından rastgele bir müşteri seçilir. Seçilen müşteri rotaya eklenir (bu müşteri M_2 olarak anılacaktır).

Adım 3.3: M_2, gidilmeyenler listesinden çıkarılır.

Adım 3.4: denemeSayisi = 0 olarak belirlenir.

Adım 3.5: "denemeSayisi > maksimum aday deneme sayısı" şartı gerçekleşirse depoya dönülerek rota sonlandırılır.

Adım 3.6: Gidilmeyenler arasında cezalı olmayanların her biri için aşağıdaki adımlar tekrar edilir.

Adım 3.6.1: Gidilmeyen bir müşteri seçilir (M_x).

Adım 3.6.2: Rotaya seçilen müşteri eklenir ve rotanın Mesafe ve Süre değeri hesaplanır. Mesafe değeri depodan başlanarak rotadaki son müşteriye ulaşana kadar gidilmesi gereken uzaklıktır. Süre değeri depodan başlanarak rotadaki son müşteriye ulaşıp müşteriden yükün alınarak (servis süresi dâhil) ayrılma anına kadar geçen süredir.

Adım 3.6.3: Maliyet ve Süre değerinin en büyüğü seçilerek M_x'in maliyeti bulunur.

Adım 3.7: Gidilmeyenler arasında en düşük maliyet değerine sahip müşteri seçilir (M_{best}).

Adım 3.8: M_{best} rotaya eklenir.

Adım 3.9: Rotada kapalı müşteri sorunu olup olmadığı kontrol edilir. Rotada kapalı müşteri sorunu varsa Mbest rotadan çıkarılır ve cezalı olarak belirlenir. Rotanın depoya dönülerek sonlandırılması gerekip-gerekmediği kontrol edilir. Depoya dönülecekse, rotaya Depo eklenerek rota kapatılır. Depoya dönülmeyecekse Adım 3.4'e dönülür.

Adım 3.10: Rotada kapalı müşteri sorunu yoksa, M_{best} gidilmeyenler listesinden çıkarılır.

Adım 3.11: Gidilmeyenlerin cezaları silinir.

Adım 3.12: Aracın dolu olup olmadığı kontrol edilir. Araç yük olarak dolu ise (kapasitesi aşılmışsa) M_{best} rotadan çıkarılır ve Depo rotaya eklenerek rota kapatılır.

Adım 3.13: Gidilmeyen müşteri varsa Adım 3.4'e dönülür.

Adım 3.14: Gidilmeyen müşteri kalmamışsa rota sonlandırılır.

Adım 4: "Rastgele" yöntemi ile aşağıdaki adımlar gerçekleştirilerek ilk popülasyon oluşturulur.

Adım 4.1: Süre Deneme Sayısı= 0 olarak belirlenir.

Adım 4.2: Depo (0) rotaya eklenir. Gidilmeyen müşteriler bulunur.

Adım 4.3: Süre Deneme Sayısı değeri 1 artırılır.

Adım 4.4: "Süre Deneme Sayısı > Maksimum Aday Deneme Sayısı" şartı sağlanıyorsa depo rotaya eklenerek rota kapatılır.

Adım 4.5: Gidilmeyen müşteriler arasından rastgele bir müşteri seçilir. Seçilen müşteri rotaya eklenir (bu müşteri M_2 olarak anılacaktır).

Adım 4.6: Rotada kapalı müşteri sorunu olup olmadığı kontrol edilir. Rotada kapalı müşteri sorunu varsa M_2 rotadan çıkarılır. Rotanın depoya dönülerek sonlandırılması gerekip-gerekmediği kontrol edilir. Depoya dönülecekse, rotaya Depo eklenerek rota kapatılır. Depoya dönülmeyecekse Adım 4.3'e dönülür. Kapalı müşteri sorunu yoksa Süre Deneme Sayısı değerine 0 (sıfırlanır) atanır ve bir sonraki adımdan devam edilir.

Adım 4.7: M_2, gidilmeyenler listesinden çıkarılır.

Adım 4.8: Aracın dolu olup olmadığı kontrol edilir. Araç yük olarak dolu ise (kapasitesi aşılmışsa) M_2 rotadan çıkarılır ve Depo rotaya eklenerek rota kapatılır.

Adım 4.9: Gidilmeyen müşteri kalmamışsa rota sonlandırılır.

Adım 4.10: Gidilmeyen müşteri varsa Adım 4.3'e dönülür.

Adım 5: İlk popülasyon oluşturulduktan sonra iterasyon = 1 değer ataması ile başlanarak makIterasyonSayisi değerine ulaşana kadar aşağıdaki adımlar gerçekleştirilir.

Adım 5.1: Popülasyonun en iyi (uygunluk değeri en düşük) böceği seçilir (B_{best})

Adım 5.2: Popülasyonda en iyi böcek dışındaki tüm böcekler için aşağıdaki adımlar tekrar edilir.

Adım 5.2.1: Popülasyonda sıradaki böcek seçilir (B_i)

Adım 5.2.2: En iyi böcek (B_{best}) ile B_i arasındaki uzaklık (Hamming Distance) hesaplanır (HD)

Adım 5.2.3: $makDeger = HD.\left(\gamma^{iterasyon\ sayısı}\right)$ değeri hesaplanır

Adım 5.2.4: Aday Sayısı= Rand[2, makDeger) aralığından rastgele bir değer üretilir

Adım 5.2.5: İşlem seçim $= Rand(0,1]$ aralığında rastgele bir değer üretilir

Adım 5.2.6: İşlem seçim değeri $0 < r \leq \frac{1}{3}$ aralığında ise "Ekleme Operatörü" gerçekleştirilir.

Adım 5.2.7: İşlem seçim değeri $\frac{1}{3} < r \leq \frac{2}{3}$ aralığında ise "Yer Değiştirme Operatörü" gerçekleştirilir.

Adım 5.2.8: İşlem seçim değeri $\frac{2}{3} < r \leq 1$ aralığında ise "$2-opt^*$Operatörü" gerçekleştirilir.

Adım 5.2.9: Aday Sayısı âdetince üretilen adaylardan en iyisi bulunarak popülasyonda B_i böceği ile yer değiştirilir.

Adım 6: Popülasyonun en iyi böceği algoritmanın nihai sonucunu barındıran çözüm olarak belirlenir.

ZPARP için Geliştirilen ABA'nın Küçük Boyutlu Bir Örnek Üzerinde Gösterimi

Bölüm 3.2.4.'te anlatılanların daha kapsamlı anlaşılabilmesi açısından 10 müşteri ve bu müşterilerin her birine ait X, Y koordinatı, servis süresi, talep miktarı, açılış zamanı ve kapanış zamanı bilgileri aşağıdaki tabloda verilmiştir. Bu bilgiler doğrultusunda 10 araç ve her bir aracın kapasitesi 60 adet olacak şekilde problemin çözümü anlatılacaktır. Müşteriler ile ilgili bilgiler Tablo 5'te ifade edilmiştir. Rastgele oluşturulmuş başlangıç çözümlerden biri Şekil 19'da gösterilmiştir. Başlangıç çözümünde oluşturulan rotalarda yer alan "0" (sıfır) ile depo ifade edilmektedir. Ateş böceği poülasyonu 4, Işık emilim katsayısı $\gamma = 1$, Maksimum iterasyon sayısı $g = 10$ olsun.

Tablo 5: Depo ve Müşteri Bilgileri

No	X Koordinatı	Y Koordinatı	Servis Süresi	Talep Miktarı	Açılış Zamanı	Kapanış Zamanı
0(Depo)	40	50	0	0	0	1236
1	45	68	90	10	912	967
2	45	70	90	30	825	870
3	42	66	90	10	65	146
4	42	68	90	10	727	782
5	42	65	90	10	15	67
6	40	69	90	20	621	702
7	40	66	90	20	170	225
8	38	68	90	20	255	324
9	38	70	90	10	534	605
10	35	66	90	10	357	410

Müşterilerin depoya ve birbirlerine olan uzaklığın hesaplanmasında öklidyen uzaklık kullanılmıştır. Söz konusu uzaklıklar Tablo 6'da verilmiştir.

Tablo 6: Uzaklık Matrisi

	Depo	**1.M**	**2. M**	**3. M**	**4. M**	**5. M**	**6. M**	**7. M**	**8.M**	**9.M**	**10.M**
Depo	0,00	18,68	20,62	16,12	18,11	15,13	19,00	16,00	18,11	20,10	16,76
1.M	18,68	0,00	2,00	3,61	3,00	4,24	5,10	5,39	7,00	7,28	10,20
2.M	20,62	2,00	0,00	5,00	3,61	5,83	5,10	6,40	7,28	7,00	10,77
3.M	16,12	3,61	5,00	0,00	2,00	1,00	3,61	2,00	4,47	5,66	7,00
4.M	18,11	3,00	3,61	2,00	0,00	3,00	2,24	2,83	4,00	4,47	7,28
5.M	15,13	4,24	5,83	1,00	3,00	0,00	4,47	2,24	5,00	6,40	7,07
6.M	19,00	5,10	5,10	3,61	2,24	4,47	0,00	3,00	2,24	2,24	5,83
7.M	16,00	5,39	6,40	2,00	2,83	2,24	3,00	0,00	2,83	4,47	5,00
8.M	18,11	7,00	7,28	4,47	4,00	5,00	2,24	2,83	0,00	2,00	3,61
9.M	20,10	7,28	7,00	5,66	4,47	6,40	2,24	4,47	2,00	0,00	5,00
10.M	16,76	10,20	10,77	7,00	7,28	7,07	5,83	5,00	3,61	5,00	0,00

Örneğin depo ile 2 numaralı müşterinin birbirine olan uzaklığı şu şekilde hesaplanacaktır:

$$d(Depo, Müşteri\ 2) = \sqrt{(x_0 - x_2)^2 + (y_0 - y_2)^2}$$

$$d(Depo, Müşteri\ 2) = \sqrt{(40 - 45)^2 + (50 - 70)^2}$$

$$= \sqrt{425}$$

$$\cong 20{,}62$$

Adım 1: Ateş Böceklerinin Rastgele Oluşturulması ve Işık Yoğunluklarının Hesaplanması

Ateş böceği algoritmasının yapısında ateş böcekleri EYK yöntemi ve rastgele olmak üzere iki şekilde oluşturulacağına değinilmişti. Bunlardan EYK yöntemi'nin örnek üzerinde işleyişi Bölüm 3.1.5'te başlangıç yiyecek kaynaklarını oluştururken anlatılmıştı. Burada ise sadece başlangıç ateş böceklerinden birinin rastgele nasıl oluşturulduğu örnek üzerinde ifade edilmiştir.

x_1:

0	6	2	1	0	5	3	7	8	0	10	9	4	0

Şekil 19: Rastgele Oluşturulmuş Başlangıç Ateş Böceklerinden (Çözüm) Biri

Rastgele oluşturulmuş başlangıç çözümde 1'inci araç aktif edilir ve bu aracın bilgileri şu şekildedir: Başlangıç noktası depo (0) ile bir rota oluşturulur, kullanılan kapasite miktarı ve şimdiki zamanı 0'dır. Müşteri listesinden rastgele seçilmiş 6 no'lu müşteriye bakılır, bu müşteri herhangi bir araca atanmadığından kısıtların kontrolü yapılır. Eğer müşteri bu aracın rotasında yer alırsa, aracın şu anki kapasitesi 20 adet olacaktır ve bu miktar aracın maksimum kapasitesi 60'dan küçüktür, yani ilk kısıt sağlanmıştır. Araç depodan bu müşteriye müşterinin kapanış zamanından önce varabilmelidir. Depo ve müşteri 6 arasındaki mesafe 19,00 uzaklık birimidir ve süre olarak alındığında ise 19,00 zaman birimine karşılık gelecektir. Yani araç müşteriye vardığında şu anki zamanı 19,00 olacaktır, bu da müşterinin kapanış zamanı olan 702 zaman biriminden öncedir, ikinci kısıt da sağlanmaktadır. Fakat müşterinin talebini karşılamak için araç müşterinin açılış zamanına kadar beklemelidir. Bu durumda teslimat en erken 621 zaman biriminde başlayacaktır.

Üçüncü ve son kısıt deponun son kapanış zamanı kısıtıdır, eğer müşteri bu araca tahsis edilen son müşteri ise araç teslimat yaptıktan sonra depoya dönecek olup, bu deponun kapanış zamanından önce gerçekleşmelidir. Bu müşteriye teslimat 621 zaman biriminde olacaktır, 90 zaman birim teslimat süresinden sonra 711 zaman biriminde depoya geri dönmek üzere yola çıkacaktır. Aralarındaki uzaklık simetrik olduğundan dönüşte de 19,00 zaman birimlik mesafe olacağından ve araç depoya dönecekse 730 zaman biriminde depoda olacaktır ki bu da deponun kapanış zamanı olan 1236 zaman biriminden öncedir. Son kısıt da böylece sağlanmış oldu. Dolayısıyla birinci aracın rotasında depodan sonraki tahsis edilen ilk müşteri 6 no'lu müşteri olmuş oldu. Aracın mevcut konumu müşteri 6, mevcut kullanılan kapasitesi 20 adet ve şimdiki zamanı 711 zaman birimi olacaktır. Bir sonraki müşteri sorgulanırken depoya dönüş süresi eklenmeyecektir. Aracın rotasındaki ikinci sıraya uygunluğu kontrol edilecektir. Müşteri 6 ise bir araca atanmış olarak müşteri listesinden kaldırılacaktır.

Bir sonraki sırada 2 numaralı müşteri bulunmaktadır ve herhangi bir aracın rotasına tahsis edilmemiştir. Kısıtlara sırasıyla bakıldığında aracın kapasite kısıtı sağlanmaktadır. Aracın şimdiki zamanı 711, müşterinin zaman penceresi olan (825,870)'i geçmemiştir. Dolayısıyla zaman penceresi kısıtı da sağlanmıştır. Birinci araç 6 numaralı müşteriden 2 numaralı müşteriye gittiğinde 5,10 birim zaman geçecek ve yeni zaman 716,10 birim olacaktır. 2 nolu müşterinin açılış zamanı 825 birim olduğu için araç bu zamana kadar bekleyecektir. Bu müşterinin servise başlama zamanı 825 birim olup, 90 birimlik teslimat süresinden sonra aracın şimdiki zaman birimi 915 ve aracın toplam kapasitesi 50 adet olacaktır. Dolayısıyla kısıtlar sağlandığından dolayı aracın rotasındaki ikinci müşteri 2 no'lu müşteri olarak listeden silinerek rotaya eklenecektir. Araç 2 nolu müşteriden 1 nolu müşteriye gittiğinde 2,00 birim zaman geçecek ve yeni zaman 917 birim olacaktır. Aracın şimdiki zaman birimi 1 nolu müşterinin (912, 967) zaman aralığında olduğu için müşterinin talebi karşılanacaktır. 90 birimlik teslimat süresinden sonra aracın şimdiki zaman birimi 1007 ve aracın mevcut kullanılan kapasitesi 60 adet olduğundan, aracın 2 nolu müşteriden sonraki rotası 1 nolu müşteri olmuş oldu. 1 nolu müşteri de böylece müşteri listesinden silinmiş oldu.

Listede kalan 5, 3, 7, 8, 10, 9, 4 numaralı müşteriler sırasıyla kapasite ve zaman penceresi kısıtlarını sağlamadıklarından dolayı ilk aracın rotasındaki ilgili sıraya atanamamışlardır. Sonuç olarak araç 1'in

rotasında sırasıyla 6, 2, 1 nolu müşteriler yer almış olup, araç depodan sırasıyla müşteri 6, müşteri 2, müşteri 1'e oradan da tekrar depoya dönecektir. 5, 3, 7, 8, 10, 9, 4 numaralı müşteriler henüz bir rotaya atanmamış olduklarından sıradaki araçlara problemin kısıtlarını sağlayacak şekilde rastgele seçilerek, 1. aracın rotasında yapılan işlemlere benzer şekilde tahsis edilirler. Söz konusu araçların başlangıç noktaları depodur ve mevcut kullanılan kapasiteleri ve şimdiki zamanları 0'dır.

Bu ateş böceği (çözüm) için 3 araç aktif olarak kullanılmış ve 10 müşterinin bu üç araca tahsis edilmesi sonrasında 3 rota oluşturulmuştur ve rotalar Şekil 19'da gösterilmiştir. Oluşturulan ateş böceğinin ışık yoğunluğu Tablo 6'dan yararlanarak aşağıdaki gibi hesaplanmıştır.

x_1:

0	6	2	1	0	5	3	7	8	0	10	9	4	0

$$0 \underbrace{\rightarrow}_{19{,}00} 6 \underbrace{\rightarrow}_{5{,}10} 2 \underbrace{\rightarrow}_{2{,}00} 1 \underbrace{\rightarrow}_{18{,}68} 0$$
$$0 \underbrace{\rightarrow}_{15{,}13} 5 \underbrace{\rightarrow}_{1{,}00} 3 \underbrace{\rightarrow}_{2{,}00} 7 \underbrace{\rightarrow}_{2{,}83} 8 \underbrace{\rightarrow}_{18{,}11} 0$$
$$0 \underbrace{\rightarrow}_{16{,}76} 10 \underbrace{\rightarrow}_{5{,}00} 9 \underbrace{\rightarrow}_{4{,}47} 4 \underbrace{\rightarrow}_{18{,}11} 0$$

$$l_1 = d_1 = toplam\ uzaklık(x_1)$$
$$= 19{,}00 + 5{,}10 + 2{,}00 + 18{,}68 + 15{,}13 + 1{,}00 + 2{,}00 + 2{,}83 + 18{,}11 + 16{,}76 + 5{,}00 + 4{,}47 + 18{,}11 = 128{,}19$$

x_1 ateş böceğinin rastgele oluşturulması ve oluşturulduktan sonra bu ateş böceğinin ışık yoğunluğunun hesaplanması yukarıda ifade edilmiştir. Geriye kalan 3 ateş böceği de benzer şekilde oluşturulmuş ve ışık yoğunlukları hesaplanmıştır.

x_2:

0	6	4	2	0	5	3	7	8	0	10	9	1	0

$$l_2 = d_2 = toplam\ uzaklık(x_2)$$
$$= 19{,}00 + 2{,}24 + 3{,}61 + 20{,}62 + 15{,}13 + 1{,}00 + 2{,}00 + 2{,}83 + 18{,}11 + 16{,}76 + 5{,}00 + 7{,}28 + 18{,}68 = 132{,}26$$

x_3:

0	8	10	9	6	0	2	1	0	3	4	0	5	7	0

$$l_3 = d_3 = toplam\ uzaklık(x_3)$$
$$= 18{,}11 + 3{,}61 + 5{,}00 + 2{,}24 + 19{,}00 + 20{,}62 + 2{,}00 + 18{,}68 + 16{,}12 + 2{,}00 + 18{,}11 + 15{,}13 + 2{,}24 + 16{,}00 = 158{,}86$$

x_4:

0	5	3	10	9	4	1	0	7	8	6	0	2	0

$$l_4 = d_4 = toplam\ uzaklık(x_4)$$
$$= 15{,}13 + 1{,}00 + 7{,}00 + 5{,}00 + 4{,}47 + 3{,}00 + 18{,}68 + 16{,}00 + 2{,}83 + 2{,}24 + 19{,}00 + 20{,}62 + 20{,}62 = 135{,}59$$

Adım 2: En İyi Ateş Böceğinin (Çözümün) Bulunması

Problem bir minimizasyon problemi olduğundan en düşük ışık yoğunluğuna (amaç fonksiyon değeri) sahip ateş böceği (çözüm) en çekici, en parlak (en iyi çözüm) ateş böceğidir.

$$l_1 = d_1 = toplam\ uzaklık(x_1) = 128{,}19$$

$$l_2 = d_2 = toplam\ uzaklık(x_2) = 132{,}26$$
$$l_3 = d_3 = toplam\ uzaklık(x_3) = 158{,}86$$
$$l_4 = d_4 = toplam\ uzaklık(x_4) = 135{,}59$$

$l_1 < l_2 < l_4 < l_3$ olduğundan en iyi (en parlak) ateş böceği 1. ateş böceği olan

x_1:	0	6	2	1	0	5	3	7	8	0	10	9	4	0

dir.

Adım 3: Yeni Ateş Böceklerinin Oluşturulması

i. Her ateş böceğinin en iyi ateş böceğine doğru olan hamming uzaklığı hesaplanır. (HD=depolar hariç karşılık gelmeyen müşterilerin sayısı)

ii. Her bir ateş böceğinden oluşturulacak yeni ateş böceklerinin sayısı belirlenir.

$$k = Random[2, HD_{i,EN\ \text{İYİ}}.\gamma^g)$$

iii. İşlem seçim değeri (r) tespit edilir.

iv. İşlem seçim değerine göre ateş böceklerine uygulanacak operatörler belirlenir.

- $0 < r \leq \frac{1}{3}$ →Ekleme Operatörü
- $\frac{1}{3} < r \leq \frac{2}{3}$ →Yer Değiştirme Operatörü
- $\frac{2}{3} < r \leq 1 \rightarrow 2-opt^*$ Operatörü

1. Ateş böceği en iyiye yaklaşıyor

$x_{en\ iyi}$: 0 6 2 1 0 5 3 7 8 0 10 9 4 0 $HD_{1,EN\ \text{İYİ}} = 0$

x_1: 0 6 2 1 0 5 3 7 8 0 10 9 4 0

Birinci böcek en iyi ateş böceği olduğundan dolayı bu ateş böceğinden yeni ateş böcekleri oluşturulamaz.

2. Ateş böceği en iyiye yaklaşıyor

$x_{en\ iyi}$: 0 6 2 1 0 5 3 7 8 0 10 9 4 0 $HD_{2,EN\ \text{İYİ}} = 3$

x_2: 0 6 4 2 0 5 3 7 8 0 10 9 1 0

Oluşturulacak aday sayısı

$$k = Random\left[2, HD_{2,EN\ \text{İYİ}}.\gamma^g\right) = Random[2{,}3.1^0) = 2$$

işlem seçim değeri $r = 0{,}25$ olsun. Bu durumda 2. Ateş böceğine 2 kez ekleme operatörü uygulanarak yeni ateş böcekleri oluşturulur. Oluşturulan ateş böceklerinin ışık yoğunluğu hesaplanır.

x_2:	0	6	4	2	0	5	3	7	8	0	10	9	1	0

Rastgele seçilen 2 nolu rotadaki 7 nolu müşteri seçilerek rotadan çıkarılır. 3 nolu rotadaki 10 nolu müşterinin önüne problemin kısıtlarını bozmadan eklenir.

Rota 2: 0 $\underset{15{,}13}{\overset{uzaklık}{\rightarrow}}$ $\underset{[15{,}67]}{\overset{10}{5}}$ $\underset{1{,}00}{\overset{uzaklık}{\rightarrow}}$ $\underset{[65{,}146]}{\overset{10}{3}}$ $\underset{2{,}00}{\overset{uzaklık}{\rightarrow}}$ $\underset{[170{,}225]}{\overset{20}{7}}$ $\underset{2{,}83}{\overset{uzaklık}{\rightarrow}}$ $\underset{[255{,}324]}{\overset{20}{8}}$ $\underset{18{,}11}{\overset{uzaklık}{\rightarrow}}$ 0

Rota 3: 0 $\underset{16{,}00}{\overset{uzaklık}{\rightarrow}}$ $\underset{[170{,}225]}{\overset{20}{7}}$ $\underset{5{,}00}{\overset{uzaklık}{\rightarrow}}$ $\underset{[357{,}410]}{\overset{10}{10}}$ $\underset{5{,}00}{\overset{uzaklık}{\rightarrow}}$ $\underset{[534{,}605]}{\overset{10}{9}}$ $\underset{7{,}28}{\overset{uzaklık}{\rightarrow}}$ $\underset{[912{,}967]}{\overset{10}{1}}$ $\underset{18{,}68}{\overset{uzaklık}{\rightarrow}}$ 0

x_2^1:	0	6	4	2	0	5	3	8	0	7	10	9	1	0

$$l_2^1 = d_2^1 = toplam\ uzaklık(x_2^1)$$
$$= 19{,}00 + 2{,}24 + 3{,}61 + 20{,}62 + 15{,}13 + 1{,}00 + 4{,}47 + 18{,}11 + 16{,}00 + 5{,}00 + 5{,}00$$
$$+ 7{,}28 + 18{,}68 = 136{,}14$$

Rastgele seçilen 1 nolu rotadan 2 nolu müşteri seçilerek rotadan çıkarılır. 3 nolu rotadaki 9 nolu müşteri ile 1 nolu müşterinin arasına problemin kısıtlarını sağlayacak şekilde aşağıdaki gibi eklenir ve bu işlem sonrasında yeni ateş böceği elde edilir.

Rota 1: 0 $\underbrace{\overset{uzaklık}{\rightarrow}}_{19{,}00}$ $\underbrace{\overset{20}{\overbrace{6}}}_{[621,702]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{2{,}24}$ $\underbrace{\overset{10}{\overbrace{4}}}_{[727,782]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{3{,}61}$ $\underbrace{\overset{30}{\overbrace{2}}}_{[825,870]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{20{,}62}$ 0

Rota 3: 0 $\underbrace{\overset{uzaklık}{\rightarrow}}_{16{,}76}$ $\underbrace{\overset{10}{\overbrace{10}}}_{[357,410]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{5{,}00}$ $\underbrace{\overset{10}{\overbrace{9}}}_{[534,605]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{7{,}00}$ $\underbrace{\overset{30}{\overbrace{2}}}_{[825,870]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{2{,}00}$ $\underbrace{\overset{10}{\overbrace{1}}}_{[912,967]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{18{,}68}$ 0

x_2^2:

0	6	4	0	5	3	7	8	0	10	9	2	1	0

$$l_2^2 = d_2^2 = toplam\ uzaklık\ (x_2^2)$$
$$= 19{,}00 + 2{,}24 + 18{,}11 + 15{,}13 + 1{,}00 + 2{,}00 + 2{,}83 + 18{,}11 + 16{,}76 + 5{,}00 + 7{,}00$$
$$+ 2{,}00 + 18{,}68 = 127{,}86$$

2. Ateş böceğine iki kez ekleme operatörü uygulanması sonucu ortaya çıkan ateş böceklerinin ışık yoğunlukları hesaplanmış ve $l_2^2 < l_2^1$ olduğundan en iyi ateş böceğinin x_2^2 olduğu tespit edilmiştir.

3. Ateş böceği en iyiye yaklaşıyor

$x_{en\ iyi}$: 0 6 2 1 0 5 3 7 8 0 10 9 4 0 $\quad HD_{3,EN\ İYİ} = 10$

x_3: 0 8 10 9 6 0 2 1 0 3 4 0 5 7 0

Oluşturulacak aday sayısı

$$k = Random\left[2, HD_{3,EN\ İYİ}.\gamma^g\right) = Random\,[2, 10.1^0) = 2$$

işlem seçim değeri $r = 0{,}35$ olsun. Bu durumda 3. Ateş böceğine 2 kez yer değiştirme operatörü uygulanarak yeni ateş böcekleri oluşturulur. Bu ateş böceklerinin ışık yoğunlukları hesaplanır.

x_3:

0	8	10	9	6	0	2	1	0	3	4	0	5	7	0

Rastgele seçilen iki rota 1. ve 4. rota olsun. 1. rotadaki 8 nolu müşteri ile 4. rotadaki 7 nolu müşteri problemin kısıtlarını bozmadan yer değiştirirse yeni ateş böceği aşağıdaki gibi olur.

Rota 1: 0 $\underbrace{\overset{uzaklık}{\rightarrow}}_{18{,}11}$ $\underbrace{\overset{20}{\overbrace{8}}}_{[255,324]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{3{,}61}$ $\underbrace{\overset{10}{\overbrace{10}}}_{[357,410]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{5{,}00}$ $\underbrace{\overset{10}{\overbrace{9}}}_{[534,605]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{2{,}24}$ $\underbrace{\overset{20}{\overbrace{6}}}_{[621,702]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{19{,}00}$ 0

Rota 4: 0 $\underbrace{\overset{uzaklık}{\rightarrow}}_{15{,}13}$ $\underbrace{\overset{10}{\overbrace{5}}}_{[15,67]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{2{,}24}$ $\underbrace{\overset{20}{\overbrace{7}}}_{[170,225]}$ $\underbrace{\overset{uzaklık}{\rightarrow}}_{16{,}00}$ 0

x_3^1:

0	7	10	9	6	0	2	1	0	3	4	0	5	8	0

$$l_3^1 = d_3^1 = toplam\ uzaklık(x_3^1)$$
$$= 16{,}00 + 5{,}00 + 5{,}00 + 2{,}24 + 19{,}00 + 20{,}62 + 2{,}00 + 18{,}68 + 16{,}12 + 2{,}00 + 18{,}11$$
$$+ 15{,}13 + 5{,}00 + 18{,}11 = 163{,}01$$

Rastgele seçilen iki rota 2 ve 3 olsun. 2. Rotadaki 2 nolu müşteri ile 3. Rotadaki 4 nolu müşteri yer değiştirerek yeni ateş böceği aşağıdaki gibi elde edilir ve ışık yoğunluğu hesaplanır.

Rota 2: 0 —(uzaklık: 20,62)→ 2 [825,870] —(uzaklık: 2,00)→ 1 [912,967] —(uzaklık: 18,68)→ 0

Rota 3: 0 —(uzaklık: 16,12)→ 3 [65,146] —(uzaklık: 2,00)→ 4 [727,782] —(uzaklık: 18,11)→ 0

x_3^2:

0	8	10	9	6	0	4	1	0	3	2	0	5	7	0

$$l_3^2 = d_3^2 = toplam\ uzaklık(x_3^2)$$
$$= 18{,}11 + 3{,}61 + 5{,}00 + 2{,}24 + 19{,}00 + 18{,}11 + 3{,}00 + 18{,}68 + 16{,}12 + 5{,}00 + 20{,}62$$
$$+ 15{,}13 + 2{,}24 + 16{,}00 = 162{,}86$$

3. Ateş böceğine iki kez yer değiştirme operatörü uygulanması sonucu ortaya çıkan ateş böceklerinin ışık yoğunlukları hesaplanmış ve $l_3^2 < l_3^1$ olduğundan en iyi ateş böceğinin x_3^2 olduğu tespit edilmiştir.

4. Ateş böceği en iyiye yaklaşıyor

$x_{en\ iyi}$: 0 6 2 1 0 5 3 7 8 0 10 9 4 0 $HD_{4,EN\ İYİ} = 10$

x_4: 0 5 3 10 9 4 1 0 7 8 6 0 2 0

Oluşturulacak aday sayısı

$$k = Random\left[2, HD_{4,EN\ İYİ}.\gamma^g\right) = Random\left[2, 10.1^0\right) = 2$$

işlem seçim değeri $r = 0{,}80$ olsun. Bu durumda 4. Ateş böceğine 2 kez $2-opt^*$ operatörü uygulanarak yeni ateş böcekleri oluşturulur. Oluşturulan ateş böcekleri ve bunlara ait ışık yoğunluğu aşağıdaki gibidir.

x_4:

0	5	3	10	9	4	1	0	7	8	6	0	2	0

Rastgele iki rota 1 ve 2 olsun. Rota 1 de 3 nolu müşteri ile 10 nolu müşteri arasındaki bağlantı ile rota 2 de 7 nolu müşteri ile 8 nolu müşteri arasındaki bağlantı kesilsin. İlk rotadaki ilk müşteri grubu 2. rotadaki son müşteri grubuna, 2. rotadaki ilk müşteri grubu 1. rotadaki son müşteri grubuna bağlanarak yeni ateş böceği oluşturulur ve ışık yoğunluğu hesaplanır. Tüm bu yapılan işlemler aşağıda belirtilmiştir.

0 —(uzaklık: 15,13)→ 5 [15,67] —(uzaklık: 1,00)→ 3 [65,146] —(uzaklık: 7,00)→ 10 [357,410] —(uzaklık: 5,00)→ 9 [534,605] —(uzaklık: 4,47)→ 4 [727,782] —(uzaklık: 3,00)→ 1 [912,967] —(uzaklık: 18,68)→ 0

0 —(uzaklık: 16,00)→ 7 [170,225] —(uzaklık: 2,83)→ 8 [255,324] —(uzaklık: 2,24)→ 6 [621,702] —(uzaklık: 19,00)→ 0

x_4^1:

0	5	3	8	6	0	7	10	9	4	1	0	2	0

$$l_4^1 = d_4^1 = toplam\ uzaklık(x_4^1)$$
$$= 15{,}13 + 1{,}00 + 4{,}47 + 2{,}24 + 19{,}00 + 16{,}00 + 5{,}00 + 5{,}00 + 4{,}47 + 3{,}00 + 18{,}68$$
$$+ 20{,}62 + 20{,}62 = 135{,}23$$

Rastgele seçilen iki rota 1 ve 2 olsun. Rota 1'deki 9 ile 4 nolu müşteri arasındaki bağlantı ile Rota 2'deki 8 ile 6 nolu müşteri arasındaki bağlantı kesilsin. İlk rotadaki ilk müşteri grubu ikinci rotadaki son müşteri grubuna, ikinci rotadaki ilk müşteri grubu ilk rotadaki son müşteri grubuna bağlanarak oluşturulan ateş böceği ve bu ateş böceğine ait ışık yoğunluğu aşağıdaki gibidir.

$$0 \xrightarrow[15{,}13]{uzaklık} \overset{10}{5}\ [15{,}67] \xrightarrow[1{,}00]{uzaklık} \overset{10}{3}\ [65{,}146] \xrightarrow[7{,}00]{uzaklık} \overset{10}{10}\ [357{,}410] \xrightarrow[5{,}00]{uzaklık} \overset{10}{9}\ [534{,}605] \xrightarrow[4{,}47]{uzaklık} \overset{10}{4}\ [727{,}782] \xrightarrow[3{,}00]{uzaklık} \overset{10}{1}\ [912{,}967] \xrightarrow[18{,}68]{uzaklık} 0$$

$$0 \xrightarrow[16{,}00]{uzaklık} \overset{20}{7}\ [170{,}225] \xrightarrow[2{,}83]{uzaklık} \overset{20}{8}\ [255{,}324] \xrightarrow[2{,}24]{uzaklık} \overset{20}{6}\ [621{,}702] \xrightarrow[19{,}00]{uzaklık} 0$$

x_4^2:	0	5	3	10	9	6	0	7	8	4	1	0	2	0

$$l_4^2 = d_4^2 = toplam\ uzaklık\ (x_4^2)$$
$$= 15{,}13 + 1{,}00 + 7{,}00 + 5{,}00 + 2{,}24 + 19{,}00 + 16{,}00 + 2{,}83 + 4{,}00 + 3{,}00 + 18{,}68$$
$$+ 20{,}62 + 20{,}62 = 135{,}12$$

4. Ateş böceğine iki kez $2-opt^*$ operatörü uygulanması sonucu ortaya çıkan ateş böceklerinin ışık yoğunlukları hesaplanmış ve $l_4^2 < l_4^1$ olduğundan en iyi ateş böceğinin x_4^2 olduğu tespit edilmiştir.

Buraya kadar yapılan tüm bu işlemler 1. iterasyona kadar ki olan işlemlerdir. 0. iterasyon sonucunda her bir operatörün uygulanmasıyla ortaya çıkan ateş böcekleri arasından en iyileri seçilmiş ve aşağıda ifade edilmiştir.

0. İterasyon sonucunda uygulanan her bir operatör sonucu ortaya çıkan en iyi ateş böcekleri

x_1:	0	6	2	1	0	5	3	7	8	0	10	9	4	0	l_1:128,19
x_2^2:	0	6	4	0	5	3	7	8	0	10	9	2	1	0	l_2^2: 127,86
x_3^2:	0	8	10	9	6	0	4	1	0	3	2	0	5	7	l_3^2:162,86
x_4^2:	0	5	3	10	9	6	0	7	8	4	1	0	2	0	l_4^2: 135,12

Bu ateş böceklerinden en iyisi ekleme operatörü sonucunda ortaya çıkan x_2^2 ateş böceğidir. Geriye kalan diğer ateş böceklerinin seçilen bu ateş böceğine olan hamming uzaklığı hesaplanır. Oluşturulacak aday sayısı (k), işlem seçim değeri (r) ve işlem seçim değerine göre ateş böceklerine uygulanacak operatörler tespit edilerek yeni ateş böcekleri oluşturulur. Oluşturulan ateş böcekleri sıralanarak en iyisi bulunur. Süreç maksimum iterasyon sayısı 10'a ulaşılıncaya kadar devam eder.

DÖRDÜNCÜ BÖLÜM

GELİŞTİRİLEN YAZILIM VE UYGULAMA

Program ve Özellikleri

Önceki bölümlerde detayları anlatılan yöntemlerin geliştirilmesinin ardından kullanılacak olan yazılımın kodlanmasına geçilmiştir. Yazılım, Microsoft tarafından geliştirilmiş bir uygulama geliştirme ortamı olan Visual Studio ortamından açık kaynak kodlu kütüphane topluluğu olan .Net Framework kütüphaneleri kullanılarak C# programlama dilinde, Windows uygulama tipinde geliştirilmiştir. C#, derleme ve yürütme süresi hızlı, mevcut güncelleme imkânı olan, birçok işlevsel özelliği sayesinde daha anlaşılır ve kolay değiştirilebilir bir basit programlama dili olmasından dolayı tercih edilmiştir.

Algoritmalar Visual Studio geliştirme ortamında C# programlama dili kullanılarak Windows Forms proje türünde geliştirilmiştir. Algoritma aşamasındaki her işlem kullanıcı tarafından adı verilen log dosyasında ayrıntılı olarak kayıt altına alınmıştır. Kullanıcı dostu arayüze sahip olması ve üzerinde çalıştığı donanımın işlemci ve bellek kaynaklarını rahatlıkla kullanabilmesi için Windows Forms proje türü seçilmiştir. Solomon'un 25 ve 50'lilik veri setleri programın işleyebileceği şekilde ön işlemden geçirilerek uygulamanın .exe uzantılı çalıştırılabilir dosyası ile aynı klasöre taşınmıştır. Programda olması gereken özellikler çözümleme çalışmaları ile tespit edilmiş ve program aşağıdaki kısımlardan oluşacak şekilde YAKA ve FA için ayrı ayrı ele alınmıştır.

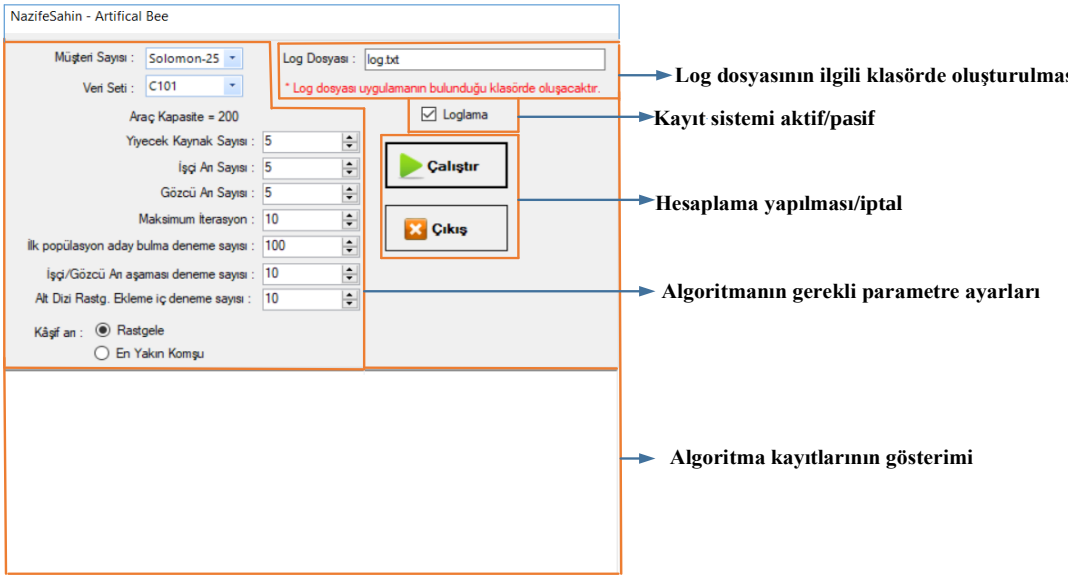
- ✓ **Parametre Bölümü:** YAKA için parametre bölümünde müşteri sayısı (25, 50, 100), veri seti (C101, C102, … ,RC108), araç kapasitesi, yiyecek kaynak sayısı=işçi arı sayısı, gözcü arı sayısı, maksimum iterasyon sayısı, ilk popülasyon aday bulma deneme sayısı, işçi/gözcü arı aşaması deneme sayısı, alt diziyi rastgele ekleme iç deneme sayısı ve kâşif arı evresinde yiyecek kaynaklarının seçilecek butona göre rastgele/en yakın komşu algoritması olmak üzere iki şekilde oluşturulması için gerekli butonlara parametre ekranında yer verilmiştir.

ABA için rastgele/en yakın komşu algoritması seçeneğine göre ateş böceklerinin oluşturulması, ateş böceği sayısı, maksimum iterasyon sayısı, ışık emilim katsayısı, ilk popülasyon aday bulma deneme sayısı, $2 - opt^*$ iç deneme sayısı ve problem için gerekli veri setlerinin ayarının yapıldığı müşteri sayısı (25, 50, 100), araç kapasitesi ve veri seti (C101, C102, …, RC108) butonları parametre ekranında yer almaktadır. Ayrıca her iki yöntem için yapılan her bir deney sonucunun log dosyalarına ayrıntısı ile kaydedilmesi için gerekli buton yine parametre bölümünde tanımlanmıştır.

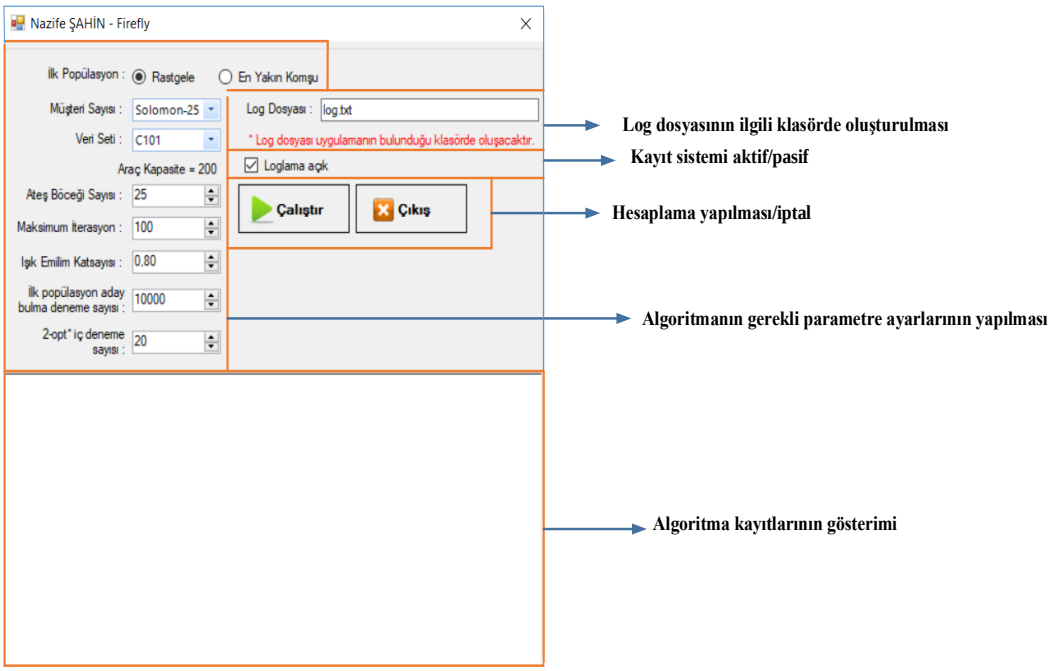
- ✓ **En İyi Sonuç ve Çözüm Detayı:** Her iki yöntem için gerekli parametre değerleri girişlerinin yapılmasının ardından yapılan her bir deney için elde edilen en iyi sonuç ve bu sonuca ait çözüm detayı programın ara yüzünün alt kısmında ve log dosyalarında gösterilmektedir.

✓ **Veri Seti:** İleri sürülen algoritmalar Solomon'un 25 ve 50 müşterili veri setlerinden C101, …, C109, R101, …, R112, ve RC101, …, RC108 veri grubuna ait toplam 58 problem üzerinde test edilmiştir. Her bir veri setinde bir depo, toplam 25 homojen filolu araç, veri setine göre şekillenen müşteri sayısı, (x, y) koordinatları ile belirtilen müşterilerin ve depoların konumları, her müşteriye ve depolara ait açılış ve kapanış zamanları, müşterilerin talep miktarları, servis süreleri ve araç kapasiteleri bulunmaktadır.

"Nazife Sahin-Artificial Bee" ve "Nazife Sahin-Firefly" programlarının arayüzleri sırasıyla Şekil 20 ve Şekil 21'de gösterilmektedir. Arayüz; problem ve problemin çözümü için gerekli olan parametre ayarlarının yapıldığı kısım, programın her bir çalıştırılmasında çözüm detaylarının kaydedilip kaydedilmediği yani loglama kayıt sisteminin aktif ya da pasif olduğu kısım, log dosyasının uygulamanın bulunduğu klasörde verilecek isme göre oluşturulacağını belirten log dosyası ve algoritma kayıtlarının gösterildiği kısımdan oluşmaktadır.



Şekil 20: YAKA için Yazılımın Arayüzü



Şekil 21: ABA için Yazılımın Arayüzü

En İyi Parametre Setinin Belirlenmesi

YAKA için En Uygun Parametre Setinin Belirlenmesi

Bu bölümde, Zaman Pencereli Araç Rotalama probleminin çözümü için kullanılacak olan YAKA için en uygun parametre setinin belirlenmesine ilişkin yapılan deneysel çalışmalar ve sonuçları sunulmaktadır. Populasyon büyüklüğü, işçi arı sayısı, gözcü arı sayısı, maksimum iterasyon sayısı, maksimum aday deneme sayısı, alt dizi rastgele ekleme deneme sayısı ve kâşif arı rastgele-en yakın komşu gibi parametreler ileri sürülen metodun performansını önemli derecede etkiler. Kâşif arı sayısı 1 olarak alınmıştır.

Bu tür problemlerde en iyi parametre setini elde etmek için çeşitli çalışmalar yapılmasına rağmen, parametrelerin tespitinin yapılmasında en uygun yöntem varyans analizidir. Diğer taraftan Varyans Analizi (ANOVA) testleri ile bir dizi tekrarlı deney sonucunda elde edilen uygunluk değerlerinin ortalamaları arasında anlamlı bir farkın olup olmadığı belirlenebilir (Şahin, 2014: 133). Yapılacak deneylerde kullanılacak parametre değerleri aşağıdaki Tablo 7'de ifade edilmektedir.

Tablo 7: Parametre Değerleri

Zaman Pencereli Araç Rotalama Problemi ($YAKA_{ZPARP}$)			Seviyeler	
	PAR_N0	**Parametre**	**Seviye 1**	**Seviye 2**
	1	Popülasyon Büyüklüğü=İşçi Arı Sayısı	20	25
	2	Gözcü Arı Sayısı	15	20
	3	Maksimum İterasyon Sayısı	20	25
	4	İlk Popülasyon Aday Bulma Deneme Sayısı	90	100
	5	İşçi/Gözcü Arı Aşaması Deneme Aşaması	20	30
	6	Alt Dizi Ekleme Deneme Sayısı	50	60
	7	Kâşif Arı Rotalama Yöntemi	Rastgele	En Yakın Komşu

İstatistiksel deney tasarımında tam faktöriyel, kesirli faktöriyel ve taguchi olmak üzere farklı yöntemler kullanılmaktadır. Tam faktöriyel deney tasarımı, iki veya daha fazla parametrenin söz konusu olduğu durumlarda bu parametrelere ait iki veya daha fazla seviye olması halinde, seviyelerin birbiri ile çarpımı sonucu oluşan kombinasyonların tümünün dikkate alınması ile yapılırken; kesirli faktöriyel deney tasarımı, deneye tabi tutulacak kombinasyon sayısının kesirli olarak azaltılması halinde elde edilir. Değişkenliği oluşturan ve kontrol edilemeyen faktörlere karşı, kontrol edilebilen faktörlerin düzeylerinin en uygun kombinasyonunu seçerek, değişkenliği en aza indirmeye çalışan Taguchi yöntemi, Genichi Taguchi tarafından ileri sürülmüş bir deneysel tasarım yöntemidir (Gökçe ve Taşgetiren, 2009: 75-76; Şahin, 2014: 134). Burada Taguchi deney tasarımı yöntemi kullanılarak Tablo 7'de ifade edilen parametre değerleri ile deneyler gerçekleştirilmiştir. Deneysel çalışmada toplam 7 parametre ve her bir parametreye ait 2 seviye bulunduğu için L16 ortogonal dizisi kullanılmış ve toplamda 16 farklı parametre değerleri elde edilmiştir. Bu elde edilen farklı parametre değerleri ile 5 tekrarlı deney yapılarak en uygun parametre seti belirlenmeye çalışılmıştır. L16 ortogonal dizisi ve parametre karşılıkları Tablo 8 ve Tablo 9'da ifade edilmektedir.

Tablo 8: Taguchi L16 Ortogonal Dizisi

SIRA	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	2	1	2	2
3	1	1	2	1	2	1	2
4	1	1	2	2	2	2	1
5	1	2	1	1	2	2	1
6	1	2	1	2	2	1	2
7	1	2	2	1	1	2	2

8	1	2	2	2	1	1	1
9	2	1	1	1	2	2	2
10	2	1	1	2	2	1	1
11	2	1	2	1	1	2	1
12	2	1	2	2	1	1	2
13	2	2	1	1	1	1	2
14	2	2	1	2	1	2	1
15	2	2	2	1	2	1	1
16	2	2	2	2	2	2	2

Tablo 9: Deneyler İçin Kullanılacak Parametreler

SIRA	1	2	3	4	5	6	7
1	20	15	20	90	20	50	Rastgele
2	20	15	20	100	20	60	En Yakın
3	20	15	25	90	30	50	En Yakın
4	20	15	25	100	30	60	Rastgele
5	20	20	20	90	30	60	Rastgele
6	20	20	20	100	30	50	En Yakın
7	20	20	25	90	20	60	En Yakın
8	20	20	25	100	20	50	Rastgele
9	25	15	20	90	30	60	En Yakın
10	25	15	20	100	30	50	Rastgele
11	25	15	25	90	20	60	Rastgele
12	25	15	25	100	20	50	En Yakın
13	25	20	20	90	20	50	En Yakın
14	25	20	20	100	20	60	Rastgele
15	25	20	25	90	30	50	Rastgele
16	25	20	25	100	30	60	En Yakın

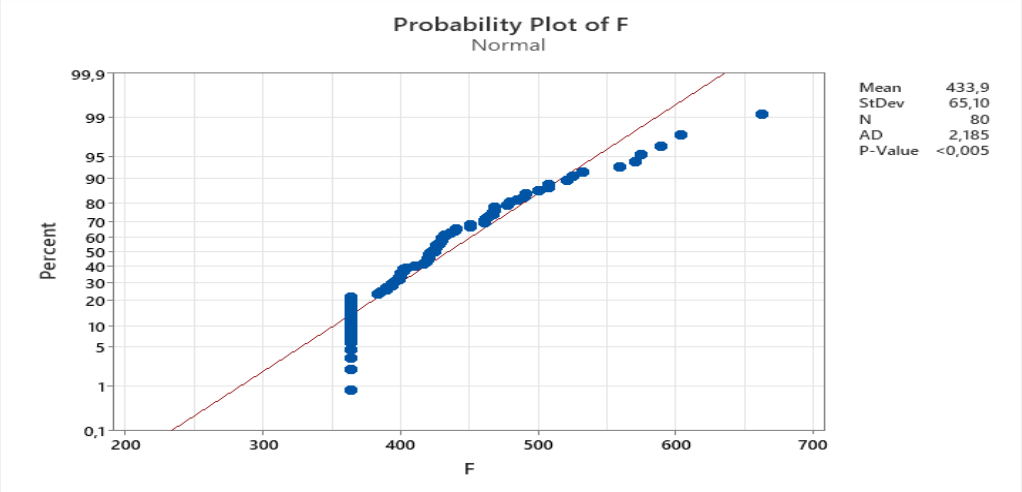
Deneyler, 16 GB RAM belleği olan 2.60 Ghz işlemcili standart bir bilgisayar yardımıyla Solomon'un 50 müşterili veri setlerinden olan C101 üzerinde gerçekleştirilmiştir. Farklı parametre değerleri dikkate alınarak toplamda $16x5 = 80$ adet deney yapılmış ve deneylerden elde edilen uygunluk değerleri Tablo 10'da gösterilmiştir.

Tablo 10: Deneylerden Elde Edilen Uygunluk Değerleri

P Seti	*F*	P Seti	*F*	P Seti	*F*	P Seti	*F*
1	491,18	**5**	401,74	**9**	393,33	**13**	363,25
1	507,87	**5**	389,99	**9**	419,19	**13**	401,74
1	662,22	**5**	477,45	**9**	404,17	**13**	363,25
1	363,25	**5**	499,84	**9**	419,47	**13**	399,24
1	603,21	**5**	363,25	**9**	425,89	**13**	363,25
2	485,13	**6**	363,25	**10**	416,68	**14**	468,21
2	398,72	**6**	363,25	**10**	439,45	**14**	507,35
2	467,31	**6**	422,06	**10**	450,80	**14**	461,87
2	420,91	**6**	393,37	**10**	532,21	**14**	467,75
2	363,25	**6**	383,25	**10**	570,91	**14**	424,29
3	435,93	**7**	389,87	**11**	461,01	**15**	363,25
3	431,58	**7**	428,70	**11**	574,92	**15**	450,24

3	363,25	**7**	363,25	**11**	479,31	**15**	396,01
3	400,31	**7**	489,49	**11**	460,99	**15**	440,47
3	363,25	**7**	363,25	**11**	363,25	**15**	400,31
4	559,15	**8**	420,85	**12**	428,14	**16**	430,28
4	466,70	**8**	432,12	**12**	463,84	**16**	425,89
4	429,48	**8**	521,29	**12**	409,84	**16**	363,25
4	525,42	**8**	363,25	**12**	385,41	**16**	363,25
4	418,42	**8**	589,13	**12**	424,50	**16**	363,25

Elde edilen verilerin istatistiksel analizine başlamadan önce veri setinin parametrik mi yoksa nonparametrik mi olup olmadığının tespitinin yapılması gerekmektedir. Verilerin normal dağılıma uyması, varyansların homojen olması, verilerin aralıklı ölçekleme yapısına sahip olması ve bağımsızlık gibi parametrik veri varsayımları dikkate alınarak hangi istatistiksel testin yapılacağına karar verilir. Bu varsayımlardan biri veya bir kaçı sağlanmıyorsa parametrik yöntemler (bağımsız t testi, tek yönlü ANOVA vb.) yerine parametrik olmayan yöntemler (Mann Whitney, Kruskall-Wallis vb.) tercih edilir (Durmuş, 2019: 360). Bu çalışmada verilerin normal dağılım gösterip göstermediği araştırılırken Anderson-Darling testinden yararlanılmıştır. Ortaya çıkan sonuçlar, 0,05 anlamlılık düzeyine göre; $p >$ 0,05 olduğunda verilerin normal dağılıma uyduğu, $p < 0{,}05$ olduğunda ise verilerin normal dağılıma uymadığı şeklinde yorumlanmıştır. Normallik testi gerçekleştirilmiş ve deneysel verilerin normallik testi sonuçları Şekil 22'de gösterilmiştir.



Şekil 22: Deneysel Verilerin Analizi İçin Normallik Testi

Şekil 22 incelendiğinde p değeri $p < 0{,}005$ olarak elde edilmiştir. p değeri 0,05 ten küçük olduğundan veriler normal dağılıma uygun değildir. Dolayısıyla elde edilen uygunluk değerleri parametrik testlerden olan Tek Yönlü ANOVA'nın parametrik olmayan karşılığı olan Kruskall-Wallis testine tabi tutulmuş ve nihai test sonuçları Tablo 11'de verilmiştir. p değerinin 0,05'ten küçük olması grup medyanlar arasında fark olduğu anlamına gelmektedir.

Tablo 11: Kruskal-Wallis Testi Sonuçları

ZAMAN PENCERELİ ARAÇ ROTALAMA				
Par_No	**N**	**Median**	**Ave**	**Z**
1	5	507,87	61,50	2,09
2	5	420,91	40,30	-0,02
3	5	400,31	29,50	-1,09
4	5	466,70	58,00	1,74
5	5	401,74	39,00	-0,15
6	5	383,25	20,40	-2,00

7	5	389,87	30,60	-0,98
8	5	432,12	49,50	0,89
9	5	419,19	34,30	-0,62
10	5	450,80	58,20	1,76
11	5	461,01	52,90	1,23
12	5	424,50	39,80	-0,07
13	**5**	**363,25**	**17,20**	**-2,32**
14	5	467,75	58,80	1,82
15	5	400,31	34,00	-0,65
16	**5**	**363,25**	**24,00**	**-1,64**
Overall	**80**		**40,50**	
DF=15	H=29,18	P=0,0015		
DF=15	H=29,52	P=0,014		

13 ve 16 numaralı parametre setleri elde edilen deney sonuçları içerisinde en düşük medyan değerlerine sahip parametre setleridir. Bu parametre setlerinin medyan değerleri birbirine eşit olduğundan deney sonucu ortaya çıkan uygunluk değerlerinin ortalamaları hesaplanarak en düşük ortalamaya sahip parametre setinin yöntem için kullanılması uygun görülmüştür. 13. ve 16. parametre setinin uygunluk değerlerinin ortalaması Tablo12'de gösterilmiştir.

Tablo 12: 13. ve 16. Parametre Setinin Ortalama Değerleri

P_Seti	***F***	**P_Seti**	***F***
13	363,25	**16**	430,28
13	401,74	**16**	425,89
13	363,25	**16**	363,25
13	399,24	**16**	363,25
13	363,25	**16**	363,25
Sum	1890,73	**Sum**	1945,92
Mean	**378,15**	**Mean**	389,18

13 nolu parametre seti Zaman Pencereli Araç Rotalama problemi için en düşük ortalama değerine sahip olduğundan bu parametre seti Yapay Arı Kolonisi Algoritması yönteminin en uygun parametre seti olarak belirlenmiştir. Parametrelere ait değerler Tablo 13'te ifade edilmektedir.

Tablo 13: $\text{YAKA}_{\text{ZPARP}}$ İçin En Uygun Parametre Seti

Parametre	**Parametre Değeri**	**Parametre Seviyesi**
Yiyecek Kaynağı Sayısı	25	2
İşçi Arı Sayısı	25	2
Gözcü Arı Sayısı	20	2
Maksimum İterasyon Sayısı	20	1
İlk Populasyon Aday Bulma Deneme Sayısı	90	1
İşçi/Gözcü Arı Aşaması Deneme Sayısı	20	1
Alt Dizi Ekleme Deneme Sayısı	50	1
Kâşif Arı Rotalama Yöntemi	En Yakın Komşu	2

ABA için En Uygun Parametre Setinin Belirlenmesi

Bu kısımda, Zaman Pencereli Araç Rotalama probleminin çözümünde kullanılacak olan Ateş Böceği Algoritması (ABA) için en uygun parametre setinin belirlenmesine yönelik yapılan deneysel çalışmalar ve sonuçları sunulmaktadır. Ateş böceği sayısı (popülasyon), maksimum iterasyon sayısı, ışık emilim

katsayısı, ilk popülasyon aday bulma deneme sayısı, ilk popülasyon oluşturma yöntemi (Rastgele/En Yakın) ve $2-opt^*$ iç deneme sayısı gibi parametreler ileri sürülen yöntemin akışını etkileyecek parametrelerdir. Yine bu kısımda algoritmanın performansını olumlu yönde etkileyen en uygun parametre seti ANOVA testleri ile belirlenmiştir. ANOVA testleri için yapılacak deneylerde gerekli parametreler ve değerleri Tablo 14'te ifade edilmektedir.

Tablo 14: Ateş Böceği Algoritması İçin Parametre Değerleri

PAR_NO	Parametre	Seviyeler		
		Seviye 1	Seviye 2	Seviye 3
1	İlk Popülasyon Oluşturma Yöntemi	Rastgele	En Yakın	-
2	Ateş Böceği (Popülasyon) Sayısı	50	75	100
3	Maksimum İterasyon Sayısı	100	130	160
4	Işık Emilim Katsayısı	0,70	0,75	0,80
5	İlk Popülasyon Aday Bulma Deneme Sayısı	9998	9999	10000
6	$2-opt^*$ İç Deneme Sayısı	30	60	90

YAKA'da olduğu gibi burada da Taguchi deneysel tasarım yöntemi ile parametre analizinin yapılması uygun görülmüştür. Algoritma için gerekli olan 6 farklı parametreden 5'i üç seviye 1'i 2 seviye olarak dikkate alındığı için L18(2^13^5) ortogonal dizisi aracılığıyla toplamda 18 farklı parametre değerleri ile 5 tekrarlı deney yapılarak en uygun parametre seti tespit edilmeye çalışılmıştır. Deney için gerekli L18(2^13^5) ortogonal dizisi ve parametre karşılıkları Tablo 15 ve Tablo 16'da belirtilmiştir.

Tablo 15: Taguchi L18 (2^13^5) Ortogonal Dizisi

SIRA	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	1	2	2	2	2
3	1	1	3	3	3	3
4	1	2	1	1	2	2
5	1	2	2	2	3	3
6	1	2	3	3	1	1
7	1	3	1	2	1	3
8	1	3	2	3	2	1
9	1	3	3	1	3	2
10	2	1	1	3	3	2
11	2	1	2	1	1	3
12	2	1	3	2	2	1
13	2	2	1	2	3	1
14	2	2	2	3	1	2
15	2	2	3	1	2	3
16	2	3	1	3	2	3
17	2	3	2	1	3	1
18	2	3	3	2	1	2

Tablo 16: Deneyler İçin Kullanılacak Parametreler

SIRA	1	2	3	4	5	6
1	Rastgele	50	100	0,70	9998	30
2	Rastgele	50	130	0,75	9999	60
3	Rastgele	50	160	0,80	10000	90
4	Rastgele	75	100	0,70	9999	60
5	Rastgele	75	130	0,75	10000	90
6	Rastgele	75	160	0,80	9998	30

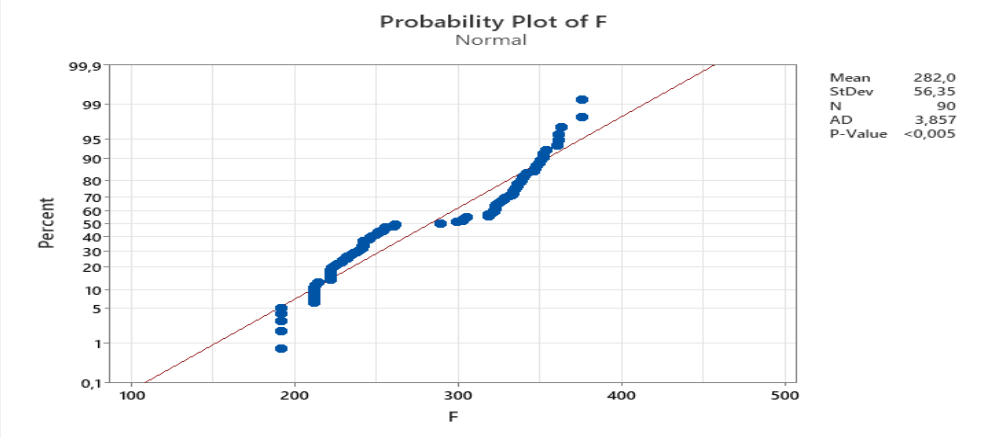
7	Rastgele	100	100	0,75	9998	90
8	Rastgele	100	130	0,80	9999	30
9	Rastgele	100	160	0,70	10000	60
10	En Yakın	50	100	0,80	10000	60
11	En Yakın	50	130	0,70	9998	90
12	En Yakın	50	160	0,75	9999	30
13	En Yakın	75	100	0,75	10000	30
14	En Yakın	75	130	0,80	9998	60
15	En Yakın	75	160	0,70	9999	90
16	En Yakın	100	100	0,80	9999	90
17	En Yakın	100	130	0,70	10000	30
18	En Yakın	100	160	0,75	9998	60

Deneyler, 16 GB RAM belleği olan 2.60 Ghz işlemcili standart bir bilgisayar yardımıyla Solomon'un 25 müşterili veri setlerinden olan C101 üzerinde gerçekleştirilmiştir. Farklı parametre değerleri dikkate alınarak toplamda $18x5 = 90$ adet deney yapılmış ve deneyler sonucunda elde edilen uygunluk değerleri Tablo 17'de gösterilmiştir.

Tablo 17: Deneylerden Elde Edilen Uygunluk Değerleri

P_Seti	*F*	P_Seti	*F*	P_Seti	*F*	P_Seti	*F*	P_Seti	*F*	P_Seti	*F*
1	345,81	**4**	335,95	**7**	339,51	**10**	254,38	**13**	231,58	**16**	249,58
1	375,10	**4**	362,74	**7**	360,48	**10**	240,76	**13**	241,00	**16**	191,81
1	346,71	**4**	353,40	**7**	331,08	**10**	241,64	**13**	239,72	**16**	222,49
1	338,83	**4**	341,91	**7**	375,44	**10**	241,64	**13**	237,87	**16**	241,99
1	350,36	**4**	361,34	**7**	351,99	**10**	221,94	**13**	222,07	**16**	236,31
2	347,65	**5**	336,34	**8**	325,89	**11**	211,81	**14**	253,35	**17**	231,58
2	327,89	**5**	333,77	**8**	322,75	**11**	250,50	**14**	222,07	**17**	211,58
2	319,71	**5**	351,64	**8**	324,87	**11**	191,81	**14**	224,41	**17**	251,25
2	361,17	**5**	333,39	**8**	322,96	**11**	229,14	**14**	212,18	**17**	222,07
2	340,54	**5**	337,57	**8**	298,95	**11**	191,81	**14**	228,42	**17**	211,81
3	317,98	**6**	304,95	**9**	328,43	**12**	211,81	**15**	246,10	**18**	245,66
3	334,17	**6**	318,47	**9**	323,44	**12**	214,37	**15**	261,51	**18**	211,81
3	332,75	**6**	302,40	**9**	322,85	**12**	241,99	**15**	255,55	**18**	225,98
3	349,15	**6**	321,49	**9**	335,01	**12**	260,66	**15**	234,81	**18**	191,81
3	288,98	**6**	321,82	**9**	303,19	**12**	247,00	**15**	221,94	**18**	191,81

Anova testlerine geçilmeden önce Anderson Darling testi ile Tablo 17'deki uygunluk değerlerinin normalliği test edilmiş ve sonuçlar Şekil 23'te ifade edilmiştir.



Şekil 23: Deneysel Verilerin Analizi İçin Normallik Testi

Şekil 23'te $p < 0{,}005$ olduğundan 0,05 anlamlılık düzeyine göre veriler normal dağılıma uymamaktadır. Bu nedenle Kruskal Wallis testi ile uygunluk değerleri test edilmiş ve elde edilen test sonuçları Tablo 18'de verilmiştir. p değerlerinin 0,05'ten küçük çıkması, ortanca uygunluk değerlerinin en az bir parametre seti için farklı olduğu anlamına gelmektedir.

Tablo 18: Kruskal – Wallis Testi Sonuçları

PAR_NO	N	Median	Mean	Z-Value
1	5	346,71	79,60	3,00
2	5	340,54	71,00	2,25
3	5	332,75	62,00	1,45
4	5	353,40	81,00	3,13
5	5	336,34	71,60	2,30
6	5	318,47	51,80	0,55
7	5	351,99	79,20	2,97
8	5	322,96	56,40	0,96
9	5	323,44	59,40	1,22
10	5	241,64	29,50	-1,41
11	**5**	**211,81**	**15,10**	**-2,68**
12	5	241,99	27,00	-1,63
13	5	237,87	24,90	-1,81
14	5	224,41	21,60	-2,10
15	5	246,10	32,50	-1,14
16	5	236,31	23,70	-1,92
17	5	222,07	18,80	-2,35
18	**5**	**211,81**	**13,90**	**-2,78**
Overall	90		45,50	
DF=17	H=76,10	P=0,000		
DF=17	H=76,13	P=0,000		

Elde edilen deney sonuçları içerisinde en düşük medyan değerine sahip parametre setleri 11 ve 18'dir. Bu parametre setlerinin medyan değerleri birbirine eşit olduğundan deney sonucu ortaya çıkan uygunluk değerlerinin ortalamaları hesaplanmış ve en düşük ortalamayı veren parametre seti en uygun parametre seti olarak belirlenmiştir. 11. ve 18. parametre setinin uygunluk değerlerinin ortalaması Tablo 19'da gösterilmiştir.

Tablo 19: 11. ve 18. Parametre Setinin Ortalama Değerleri

P_Seti	F	P_Seti	F
11	211,81	18	245,66
11	250,50	18	211,81
11	191,81	18	225,98
11	229,14	18	191,81
11	191,81	18	191,81
Sum	1075,07	Sum	1067,07
Mean	215,014	Mean	213,414

Böylece Ateş Böceği Algoritması için en uygun parametre setinin en düşük ortalama değerine sahip 18 nolu parametre seti olduğu ortaya çıkmıştır. Her bir parametrenin değerleri ve seviyeleri Tablo 20'de gösterilmektedir.

Tablo 20: $\mathrm{ABA}_{\mathrm{ZPARP}}$ için En Uygun Parametre Seti

Parametre	Parametre Değeri	Parametre Seviyesi
İlk Ppopülasyon Oluşturma Yöntemi	En Yakın	2
Ateş Böceği (Popülasyon) Sayısı	100	3
Maksimum İterasyon Sayısı	160	3
Işık Emilim Katsayısı	0,75	2
İlk Populasyon Aday Bulma Deneme Sayısı	9998	1
$2-opt^*$ İç Deneme Sayısı	60	2

Geliştirilen Yöntemlerin Etkinliğinin Araştırılması

Problem Veri Seti

Algoritmalar, 25 ve 50 müşterili üç farklı problem türünden (C101, …,C109, R101, …,R112 ve RC101,…, RC108) oluşan klasik bir dizi kıyaslama probleminin 58'i (Solomon, 1987) üzerinde test edilmiştir.

Veri setleri http://www.bernabe.dorronsoro.es/vrp/index.html?/results/resultsSolom.htm adresinden alınmıştır. Bu problem verilerinin genel olarak özellikleri aşağıda tanımlanmaktadır.

Müşterilerin rastsal olarak coğrafi bölgelere dağıldığı R1 tipi problemlerde, zaman pencereleri sıkı ve araç kapasiteleri küçüktür. R2 tipi problemlerde müşteriler R1'de olduğu gibi rastsal olarak coğrafi bölgelere dağılmıştır. Fakat bu problem verilerinde her bir müşteriye hizmet verilmesi gereken zaman aralığı geniş ve araç kapasiteleri büyüktür. C1 tipi problemlerde müşteriler coğrafi bölgelere kümelenmiş bir biçimde dağılım göstermiştir. Her bir müşteriye hizmet verilmesi gereken zaman aralıkları sıkı ve araç kapasiteleri küçüktür. C2 tipi problemlerde her bir müşteriye hizmet verilmesi gereken zaman aralığı geniş ve araç kapasiteleri büyük iken yine C1'de olduğu gibi müşteriler coğrafi bölgelere kümelenmiş bir şekilde dağılmışlardır. RC1 ve RC2 tipi problemlerde müşteriler hem rastsal hem de kümelenmiş bir biçimde coğrafi bölgelere dağılım göstermişlerdir. RC1'de araç kapasitesi küçük ve zaman aralığı dar olup her rotada çok fazla müşteriye hizmet verilememektedir. RC2 tipi problemlerde ise zaman aralıkları geniş olmakla birlikte araç kapasitelerinin büyük olması çok az rotaya ihtiyaç olduğunu gösterir (Kuram, 2016: 73-74).

25, 50 ve 100 müşterili veri setleri sırasıyla, birbirinden ayrılmış farklı coğrafi bölgelerde yer alan 25, 50 ve 100 müşteri ve bir depodan oluşmaktadır. Hizmet süresi ve araç kapasiteleri veri setine göre farklılık göstermektedir. R ve RC tipi problemlerde müşterilerin hizmet süresi 10 birim iken C tipi problemlerde ise 90 birim olarak ele alınmıştır. R1, C1 ve RC1 tipi problemlerde araç kapasiteleri 200

birim, R2 ve RC2 tipi problemlerde 1000 birim iken C2 tipi problemlerde ise araç kapasiteleri 700 birim olarak ele alınmış ve her araç homojen filolu bir yapıya sahiptir.

Test Problemlerinin Geliştirilen Yöntemler İle Çözümü

Çalışma kapsamında ileri sürülen YAKA ve ABA yöntemleri literatürde ilk defa zaman pencereli araç rotalama probleminin eş zamanlı çözümünü sağlamaktadır. Daha önce ikisini birlikte ele alan bir çalışma olmadığı için ileri sürülen yöntemlerin etkinliği Solomon'un (1987) 25 ve 50 müşterili veri setleri kullanılarak test edilmiştir. 25 ve 50 müşterili veri setlerinden sadece C1, R1 ve RC1 grubuna ait problem verileri kullanılmıştır. Tablo 21'de karşılaştırma için kullanılan yöntemler, Tablo 22'de ise 25 müşterili problem setlerinin geliştirilen yöntemlerle çözümünden elde edilen uygunluk değerleri ve en iyi çözüme uzaklıkları gösterilmektedir.

Tablo 21: Karşılaştırma İçin Kullanılan Yöntemler

KISALTMA	YIL	YAZARLAR	ALGORİTMA
LP	2002	Bard vd.	Doğrusal Programlama
PSO1	2009	Ai and Kachitvichyanukul	Parçacık Sürü Optimizasyon Algoritması
PSO2	2011	Gong vd.	Parçacık Sürü Optimizasyon Algoritması
WOA	2022	Yodwangjai and Malampong	Balina Optimizasyon Algoritması

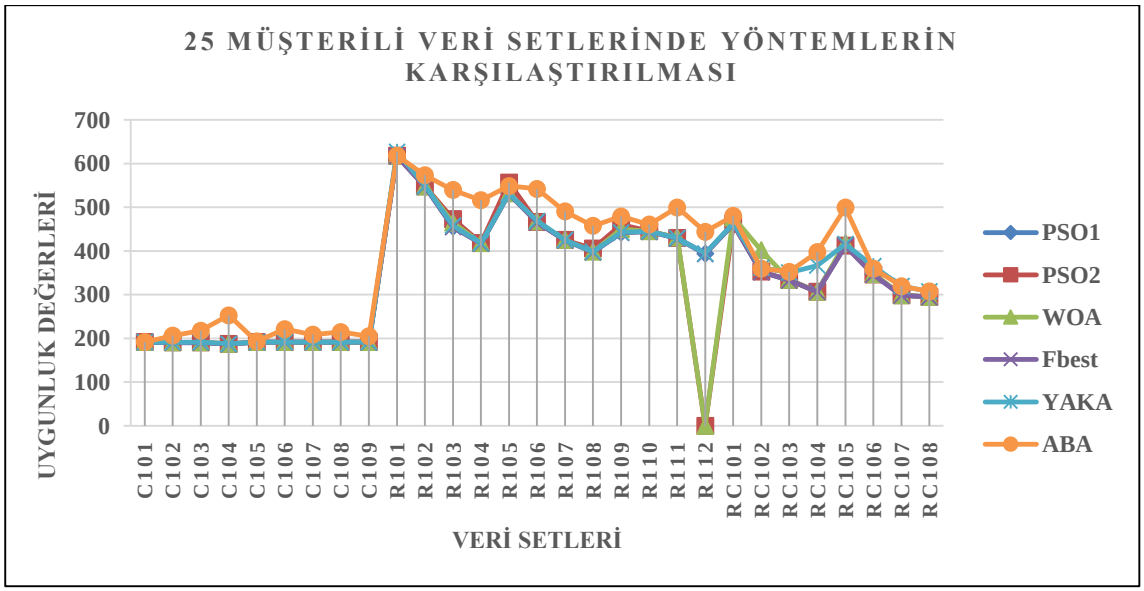
Tablo 22: 25 Müşterili Problem Setlerinin Geliştirilen Yöntemler İle Çözümü

	Veri	PSO1	PSO2	WOA	F_{best}	YAKA	Fark(%)	ABA	Fark(%)
C10_ VERİ SETLERİ	**C101**	191,8	191,81	191,81	**191,3**	191,81	0,27	**191,81**	0,27
	C102	190,7	190,74	190,74	**190,3**	190,74	0,23	206,12	8,31
	C103	190,7	190,74	190,74	**190,3**	**191,81**	0,79	217,66	14,38
	C104	187,4	187,45	187,45	**186,9**	187,45	0,29	252,74	35,23
	C105	191,8	191,81	191,81	**191,3**	191,81	0,27	193,04	0,91
	C106	191,8	191,81	191,81	**191,3**	191,81	0,27	221,42	15,74
	C107	191,8	191,81	191,81	**191,3**	191,81	0,27	208,56	9,02
	C108	191,8	191,81	191,81	**191,3**	191,81	0,27	214,76	12,26
	C109	191,8	191,81	191,81	**191,3**	191,81	0,27	204,76	7,04
R10_ VERİ SETLERİ	**R101**	618,3	618,33	618,33	**617,1**	**627,13**	**1,63**	**618,33**	**0,20**
	R102	548,1	548,11	548,11	**547,1**	548,95	0,34	573,34	4,80
	R103	455,7	473,39	464,83	**454,6**	458,37	0,83	539,80	18,74
	R104	418,0	418,30	417,96	**416,9**	417,96	0,25	516,30	23,84
	R105	531,5	556,72	531,54	**530,5**	531,54	0,20	548,82	3,45
	R106	466,5	466,48	467,85	**465,4**	**468,96**	0,77	542,13	16,49
	R107	425,3	425,27	425,27	**424,3**	425,27	0,23	490,43	15,59
	R108	398,3	405,39	398,29	**397,3**	398,29	0,25	457,75	15,22
	R109	442,6	460,52	450,26	**441,3**	442,63	0,30	479,03	8,55
	R110	445,9	445,80	445,18	**444,1**	445,18	0,24	460,48	3,69
	R111	429,7	429,70	431,12	**428,8**	429,70	0,21	499,84	16,57
	R112	394,1	-	-	**393,0**	394,10	0,28	443,88	12,95
RC10_ VERİ	**RC101**	462,2	462,16	476,96	**461,1**	463,60	0,54	480,40	4,19
	RC102	352,7	352,74	401,79	**351,8**	**361,89**	**2,87**	**359,73**	**2,25**
	RC103	333,9	333,92	333,92	**332,8**	**350,78**	5,40	352,57	5,94

	RC104	307,1	307,14	307,14	**306,6**	**366,81**	19,64	397,33	29,59
	RC105	412,4	412,38	418,52	**411,3**	415,70	1,07	499,50	21,44
	RC106	346,5	346,51	347,31	**345,5**	**365,27**	**5,72**	**359,93**	**4,18**
	RC107	298,9	298,95	298,95	**298,3**	**319,99**	**7,27**	**319,44**	**7,09**
	RC108	295,0	294,99	294,99	**294,5**	**307,22**	4,32	307,22	4,32

Not: Bard vd., (2002) C10_, R10_ ve RC10_ veri setleri için deney yapmamıştır. Gong vd., (2011) ve Yodwangjai and Malampong (2022) R112 veri seti için deney yapmamıştır.

Tablo 22'de ifade edilen veri setleri üzerinde YAKA yöntemi ile %0,20 ile %19,64 arasında; ABA yöntemi ile %0,20 ile %35,23 arasında değişen oranlarda çözümler elde edilmiştir. Birçok veri setinde bilinen en iyi çözüme yakın ve karşılaştırılan diğer yöntemlerden daha iyi sonuçlar YAKA yöntemi ile sağlanmıştır. RC104 hariç diğer deney setleri üzerinde YAKA ile %0,20 ile %7,27 arasında değişen oranlarda en iyi çözüme yakın iyi sonuçlar elde edilirken; ABA yöntemi ile ise birçok problem verisinde, %12,26 ile %35,23 arasında karşılaştırılan diğer yöntemler ve YAKA'dan uzakta sonuçlar elde edilmiştir. YAKA, ABA ile karşılaştırıldığında 29 test probleminin 23'ünde daha iyi sonuçlar ortaya çıkarmıştır. 14 problem verisinde ABA yöntemi ile %0,20 ile %9,02 arasında değişen oranlarda bilinen en iyi çözüme yakın sonuçlar elde edilmiştir. Diğer taraftan R101, RC102, RC106 ve RC107 problemlerinde ABA ile elde edilen uygunluk değerlerinin YAKA ile elde edilen uygunluk değerlerinden daha küçük çıkması bu veri setleri için ABA yönteminin YAKA yönteminden daha iyi performans gösterdiğini ortaya koymuştur. C101 ve RC108 problemleri için kullanılan her iki yöntemde de sırasıyla bilinen en iyi çözümden %0,27 ve %4,32 farkla aynı sonuçlar elde edilmiştir. Karşılaştırma neticesinde ele alınan 25 müşterili veri setlerinin çoğunda YAKA yönteminin ABA ve diğer yöntemlere kıyasla daha iyi sonuçlar ortaya çıkardığı tespit edilmiştir. Genel olarak 25 müşterili deney setleri üzerinde sonuçlar incelendiğinde, problem için geliştirilen YAKA yönteminin performans açısından yeterli bulunmuş olması ile birlikte ABA için aynı durumun söz konusu olmadığı ortaya çıkmıştır. Elde edilen uygunluk değerlerine ilişkin grafik Şekil 24'te belirtilmektedir. Grafikte C10_ veri setlerinde yöntemler değerlendirilecek olursa; YAKA yöntemi, bu veri grubundaki tüm problemler üzerinde bilinen en iyi çözüme ve karşılaştırılan diğer yöntemlere eşit seviyede sonuçlar ortaya çıkarırken, ABA yöntemi, sadece C101 ve C105 problemi üzerinde bilinen en iyi çözüme ve karşılaştırılan diğer yöntemlere oldukça yakın sonuçlar sağlamıştır.



Şekil 24: Elde Edilen Uygunluk Değerlerinin Grafik Gösterimi

50 müşterili veri setlerinin geliştirilen yöntemler ile çözümünden elde edilen uygunluk değerleri ve en iyi çözüme uzaklıkları, Tablo 23'te ifade edilmektedir. Tabloda gösterilen veri setleri üzerinde YAKA

yöntemi ile %0,21 ile %15,86 arasında; ABA yöntemi ile %15,61 ile %65,96 arasında değişen oranlarda çözümler elde edilmiştir. Birçok veri setinde karşılaştırmada kullanılan yöntemlerin sonucuna eşit ve bu yöntemlerden daha iyi sonuçlar YAKA yöntemi ile ortaya konmuştur. Ele alınan veri setinin tamamında ABA yöntemi ile elde edilen uygunluk değerleri, YAKA ve karşılaştırılan diğer yöntemlerden elde edilen uygunluk değerlerinden çözüm kalitesi açısından olumsuz sonuçlanmıştır. RC106 hariç RC10_veri grubundaki tüm problemler ve C104 problemi için YAKA yöntemi ile elde edilen çözüm değeri, karşılaştırılan diğer yöntemlerden elde edilen çözüm değerlerinden büyük çıkmıştır. 50 müşterili veri setleri üzerinde genel değerlendirme yapılacak olursa; toplam 29 test probleminin tamamında YAKA yöntemi ile ortaya çıkan uygunluk değerlerinin %0,21 ile %15,86 değişen oranlarda bilinen en iyi çözüme yakın çıkması geliştirilen bu yöntemin zaman pencereli araç rotalama problemi için yeterli olduğunu göstermiştir. ABA yöntemi ise performans açısından problemin çözümünde yetersiz kalmıştır. Müşteri sayısındaki artış yükseldikçe yöntemlerin sonuçları arasındaki fark daha net olarak görülmektedir.

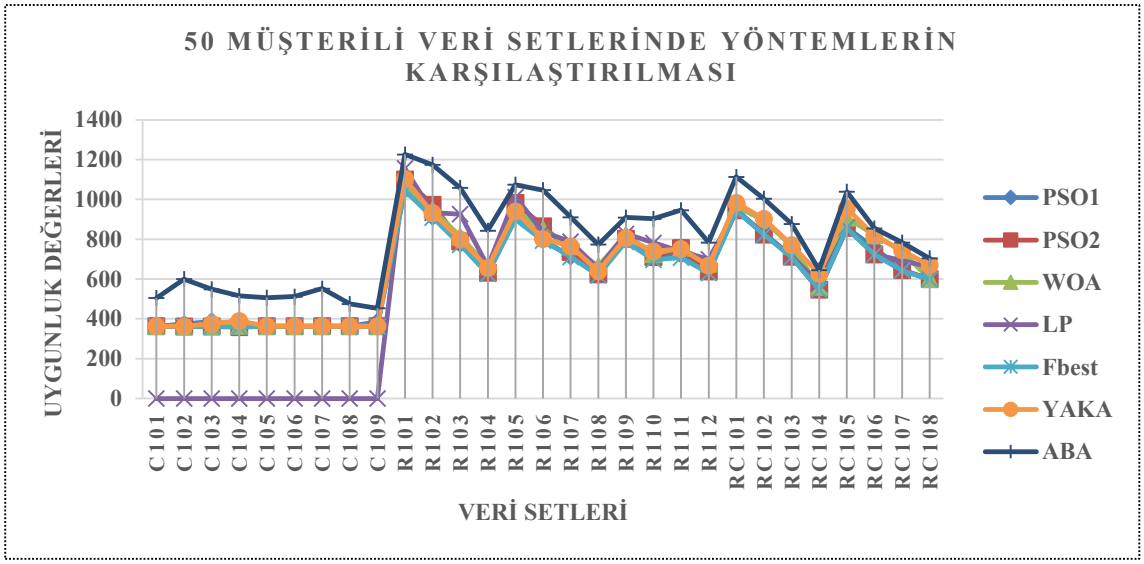
Tablo 23: 50 Müşterili Problem Setlerinin Geliştirilen Yöntemler İle Çözümü

	Veri	PSO1	PSO2	WOA	LP	F_{best}	YAKA	Fark(%)	ABA	Fark(%)
C10_ VERİ SETLERİ	C101	363,2	363,25	363,25	-	**362,4**	363,25	0,23	504,51	39,21
	C102	373,5	362,17	362,17	-	**361,4**	362,17	0,21	599,78	65,96
	C103	387,4	362,17	363,34	-	**361,4**	**375,45**	3,89	548,87	51,87
	C104	366,7	358,88	365,38	-	**358,0**	**389,79**	8,88	514,99	43,85
	C105	363,2	363,25	363,25	-	**362,4**	363,25	0,23	505,97	39,62
	C106	363,2	363,25	363,25	-	**362,4**	363,25	0,23	512,25	41,35
	C107	363,2	363,25	363,25	-	**362,4**	363,25	0,23	552,51	52,46
	C108	363,2	363,25	363,92	-	**362,4**	363,25	0,23	475,14	31,11
	C109	385,4	363,25	363,25	-	**362,4**	363,25	0,23	452,96	24,99
R10_ VERİ SETLERİ	R101	1053,9	1100,72	1054,88	1156,0	**1044,0**	1103,08	5,66	1226,09	17,44
	R102	913,6	973,71	946,67	933,6	**909,0**	931,02	2,42	1173,42	29,09
	R103	778,5	790,17	813,94	925,8	**772,9**	799,45	3,44	1057,69	36,85
	R104	632,2	631,58	663,64	662,9	**625,4**	656,36	4,95	843,63	34,89
	R105	932,9	983,49	943,95	1012,1	**899,3**	940,05	4,53	1074,46	19,48
	R106	797,3	865,93	848,63	837,0	**793,0**	801,29	1,05	1046,70	32,00
	R107	713,9	737,10	763,97	788,0	**711,1**	764,14	7,46	910,58	28,05
	R108	620,3	624,29	665,13	652,6	**617,7**	639,92	3,60	773,01	25,14
	R109	803,8	801,97	821,64	827,5	**786,8**	806,58	2,51	909,62	15,61
	R110	708,4	710,40	725,85	781,9	**697,0**	738,84	6,00	902,39	29,47
	R111	724,2	756,35	756,21	738,0	**707,2**	751,84	6,31	946,58	33,85
	R112	637,8	638,49	677,33	699,9	**630,2**	667,15	5,86	783,84	24,38
RC10_VERİ SETLERİ	RC101	945,6	945,58	966,12	944,6	**944,0**	983,22	4,16	1113,50	17,96
	RC102	828,0	823,97	893,56	828,4	**822,5**	902,07	9,67	1003,16	21,96
	RC103	712,6	712,91	766,70	714,4	**710,9**	770,99	8,45	876,97	23,36
	RC104	546,5	546,51	561,72	579,2	**545,8**	626,52	14,79	645,19	18,21
	RC105	857,7	856,97	908,28	858,8	**855,3**	954,22	11,57	1038,69	21,44
	RC106	757,2	724,65	820,47	724,4	**723,2**	818,44	13,17	854,56	18,16
	RC107	645,4	645,70	738,68	693,1	**642,7**	744,63	15,86	784,04	21,99

	RC108	599,2	599,70	600,23	662,0	**598,1**	669,07	11,87	704,30	17,76

Not: Bard vd., (2002) C10_50 veri setleri için deney yapmamıştır.

50 müşterili veri setleri üzerinde geliştirilen yöntemler ile elde edilen uygunluk değerlerine ilişkin grafik Şekil 25'te ifade edilmektedir.



Şekil 25: 50 Müşterili Veri Setlerinden Elde Edilen Uygunluk Değerlerinin Grafik İle Gösterimi

YAKA ve ABA yöntemleri kullanılarak 25 ve 50 müşterili problem setleri üzerinde 5 tekrarlı deneyler yapılmıştır. Deneyler sonucunda zaman pencereli araç rotalama problemi için elde edilen minimum ve maksimum([min, max]) uygunluk değerleri ve bu uygunluk değerlerine ilişkin kullanılan araç sayıları (NV) ve ne kadar sürede elde edildiklerine ilişkin çözüm süreleri sırasıyla Tablo 24 ve Tablo 25'te ifade edilmektedir.

Tablo 24: 25 Müşterili Veri Setleri İçin Elde Edilen Sonuçlar

	Deney Seti	YAKA			ABA		
		F_{ZPARP}	NV	T(dk)	F_{ZPARP}	NV	T(dk)
C10_ VERİ SETLERİ	**C101**	[191,81 – 217,33]	[3 – 3]	[13,32-14,13]	[191,81 – 235,47]	[3 – 4]	[8,50-8,48]
	C102	[190,74 – 216,26]	[3 – 3]	[6,03-6,08]	[206,12 – 249,02]	[3 – 3]	[5,13-5,27]
	C103	[191,81 – 217,33]	[3 – 3]	[4,25-4,73]	[217,66 – 262,90]	[3 – 5]	[4,07-3,92]
	C104	[187,45 – 218,71]	[3 – 3]	[4,00-3,95]	[252,74 – 265,92]	[3 – 5]	[3,32-3,42]
	C105	[191,81 – 191,81]	[3 – 3]	[8,05-8,68]	[193,04 – 222,86]	[3 – 3]	[6,47-6,58]
	C106	[191,81 – 191,81]	[3 – 3]	[12,32-13,75]	[221,42 – 241,00]	[4 – 4]	[7,07-6,83]
	C107	[191,81 – 217,33]	[3 – 3]	[7,92-7,73]	[208,56 – 264,80]	[3 – 4]	[5,37-5,35]
	C108	[191,81 – 217,33]	[3 – 3]	[6,30-5,85]	[214,76 – 254,06]	[3 – 3]	[4,80-4,90]
	C109	[191,81 – 194,40]	[3 – 3]	[5,10-4,97]	[204,76 – 233,31]	[3 – 3]	[5,08-4,93]
R10_ VERİ SETLERİ	**R101**	[627,13 – 639,65]	[9 – 8]	[13,28-10,01]	[618,33 – 630,80]	[8 – 8]	[17,00-17,20]
	R102	[548,95 – 569,65]	[7 – 7]	[6,60-5,97]	[573,34 – 617,85]	[7 – 7]	[7,30-7,48]
	R103	[458,37 – 475,84]	[5 – 6]	[8,23-4,85]	[539,80 – 559,94]	[6 – 6]	[6,27-6,35]
	R104	[417,96 – 455,56]	[4 – 5]	[5,07-5,42]	[516,30 – 537,39]	[5 – 5]	[5,47-6,22]
	R105	[531,54 – 553,73]	[6 – 6]	[8,13-9,15]	[548,82 – 576,03]	[6 – 7]	[15,03-10,82]
	R106	[468,96 – 503,36]	[5 – 6]	[6,48-8,82]	[542,13 – 554,62]	[5 – 6]	[9,87-7,42]
	R107	[425,27 – 460,64]	[4 – 4]	[6,27-5,72]	[490,43 – 526,88]	[4 – 5]	[7,83-8,17]
	R108	[398,29 – 436,94]	[4 – 4]	[6,80-7,15]	[457,75 – 507,53]	[4 – 4]	[9,17-8,30]
	R109	[442,63 – 460,43]	[5 – 5]	[8,33-9,32]	[479,03 – 496,20]	[5 – 5]	[12,10-12,68]
	R110	[445,18 – 461,38]	[5 – 5]	[6,70-8,55]	[460,48 – 504,16]	[5 – 5]	[8,30-8,40]

	R111	[429,70 – 463,64]	[4 – 4]	[7,58-6,93]	[499,84 – 517,94]	[5 – 5]	[7,57-8,10]
	R112	[394,10 – 413,39]	[4 – 4]	[8,42-7,12]	[443,88 – 459,96]	[4 – 4]	[9,02-8,25]
RC10_VERİ SETLERİ	**RC101**	[463,60 – 514,03]	[4 – 5]	[2,80 –19,60]	[480,40 – 487,35]	[4 – 4]	[19,27-18,03]
	RC102	[361,89 – 412,44]	[3 – 4]	[6,65-5,01]	[359,73 – 431,70]	[3 – 4]	[6,73-7,19]
	RC103	[350,78 – 408,47]	[3 – 4]	[6,52-7,02]	[352,57 – 410,52]	[3 – 4]	[6,53-6,94]
	RC104	[366,81 – 441,00]	[4 – 4]	[6,58-5,37]	[397,33 – 422,97]	[4 – 4]	[7,26-8,21]
	RC105	[415,70 – 527,76]	[4 – 6]	[4,97-7,79]	[499,50 – 536,82]	[5 – 5]	[7,46-6,21]
	RC106	[365,27 – 421,55]	[3 – 4]	[6,73-7,03]	[359,93 – 426,04]	[3 – 4]	[6,71-7,08]
	RC107	[319,99 – 376,60]	[3 – 4]	[7,43-6,51]	[319,44 – 387,81]	[3 – 4]	[7,32-7,45]
	RC108	[307,22 – 379,94]	[3 – 4]	[7,04-8,53]	[307,22 – 386,47]	[3 – 4]	[7,56-9,08]

- Yapılan 5 tekrar sonucunda elde edilen minimum maksimum uygunluk değerleri ve bu değerlere ilişkin araç sayıları ve hesaplama süreleri tabloda verilmektedir.

Tablo 25: 50 Müşterili Veri Setleri İçin Elde Edilen Sonuçlar

	D_Seti	**YAKA**			**ABA**		
		F_{ZPARP}	**NV**	**T(dk)**	F_{ZPARP}	**NV**	**T(dk)**
C10_ VERİ SETLERİ	**C101**	[363,25 – 435,26]	[5 – 6]	[34,72-34,28]	[504,51 – 650,52]	[7 – 10]	[16,05-16,63]
	C102	[362,17 – 546,95]	[5 – 7]	[18,20-15,20]	[599,78 – 667,38]	[7 – 7]	[12,10-11,52]
	C103	[375,45 – 452,84]	[5 – 7]	[33,03-34,73]	[548,87 – 636,58]	[6 – 8]	[7,48-7,58]
	C104	[389,79 – 441,35]	[5 – 6]	[26,95-27,05]	[514,99 – 522,01]	[6 – 7]	[6,20-6,14]
	C105	[363,25 – 455,49]	[5 – 6]	[30,67-30,22]	[505,97 – 576,94]	[7 – 8]	[17,13-14,20]
	C106	[363,25 – 440,30]	[5 – 7]	[37,57-27,45]	[512,25 – 568,44]	[6 – 9]	[16,57-15,87]
	C107	[363,25 – 430,08]	[5 – 7]	[29,53-20,95]	[552,51 – 610,58]	[7 – 7]	[14,85-13,78]
	C108	[363,25 – 459,95]	[5 – 7]	[18,91-18,77]	[475,14 – 541,13]	[7 – 6]	[12,68-14,55]
	C109	[363,25 – 448,87]	[5 – 6]	[52,85-15,63]	[452,96 – 553,17]	[6 – 7]	[11,73-11,85]
R10_ VERİ SETLERİ	**R101**	[1103,08 – 1298,36]	[13 – 15]	[31,62-31,94]	[1226,09 – 1277,48]	[14 – 16]	[33,53-31,35]
	R102	[931,02 – 973,67]	[12 – 11]	[118,08-95,08]	[1173,42 – 1254,25]	[13 – 13]	[17,70-6,55]
	R103	[799,45 – 886,14]	[9 –11]	[40,33-42,45]	[1057,69 – 1120,22]	[11 – 12]	[13,43-11,57]
	R104	[656,36 – 695,71]	[7 – 7]	[40,42-49,87]	[843,63 – 894,91]	[8 – 8]	[11,43-19,38]
	R105	[940,05 – 981,93]	[10 – 11]	[29,93-30,63]	[1074,46 – 1121,62]	[10 – 12]	[21,57-21,10]
	R106	[801,29 – 853,67]	[9 – 9]	[55,23-59,30]	[1046,70 – 1082,76]	[10 – 11]	[16,52-15,90]
	R107	[764,14 – 802,09]	[9 – 9]	[54,85-38,50]	[910,58 – 1006,98]	[9 – 10]	[13,12-13,67]
	R108	[639,92 – 708,35]	[6 – 7]	[34,77-39,10]	[773,01 – 853,80]	[7 – 7]	[14,68-14,60]
	R109	[806,58 – 850,52]	[8 – 9]	[80,63-57,05]	[909,62 – 986,84]	[9 – 11]	[17,82-17,43]
	R110	[738,84 – 802,32]	[8 – 9]	[25,75-25,52]	[902,39 – 938,81]	[9 – 9]	[21,25-19,92]
	R111	[751,84 – 817,67]	[8 – 9]	[17,12-16,10]	[946,58 – 981,75]	[8 – 10]	[16,80-16,52]
	R112	[667,15 – 710,79]	[7 – 7]	[18,18-19,22]	[783,84 – 801,44]	[7 – 7]	[28,92-38,93]
RC10_VERİ SETLERİ	**RC101**	[983,22 – 1101,73]	[10 – 12]	[15,55-2,50]	[1113,50 – 1144,07]	[11 – 11]	[21,47-15,13]
	RC102	[902,07 – 989,80]	[9 – 10]	[22,10-11,73]	[1003,16 – 1048,64]	[9 – 9]	[18,33-17,65]
	RC103	[770,99 – 892,75]	[7 – 9]	[44,08-37,70]	[876,97 – 981,51]	[7 – 9]	[26,55-28,42]
	RC104	[626,52 – 677,49]	[6 – 6]	[4,83-5,88]	[645,19 – 729,56]	[6 – 7]	[29,90-27,90]
	RC105	[954,22 – 969,77]	[10 – 10]	[64,03-24,82]	[1038,69 – 1101,03]	[10 – 11]	[24,33-24,68]
	RC106	[818,44 – 861,80]	[7 – 8]	[191,98-188,78]	[854,56 – 902,06]	[8 – 8]	[27,68-44,72]
	RC107	[744,63 – 780,59]	[7 – 7]	[40,73-38,96]	[784,04 – 819,41]	[7 – 7]	[29,47-32,97]
	RC108	[669,07 – 701,68]	[6 – 6]	[6,57-18,23]	[704,30 – 751,96]	[6 – 7]	[28,40-27,62]

- Yapılan 5 tekrar sonucunda elde edilen minimu maksimum uygunluk değerleri ve bu değerlere ilişkin araç sayıları ve hesaplama süreleri tabloda verilmektedir.

Deney Sonuçlarının Karşılaştırılması

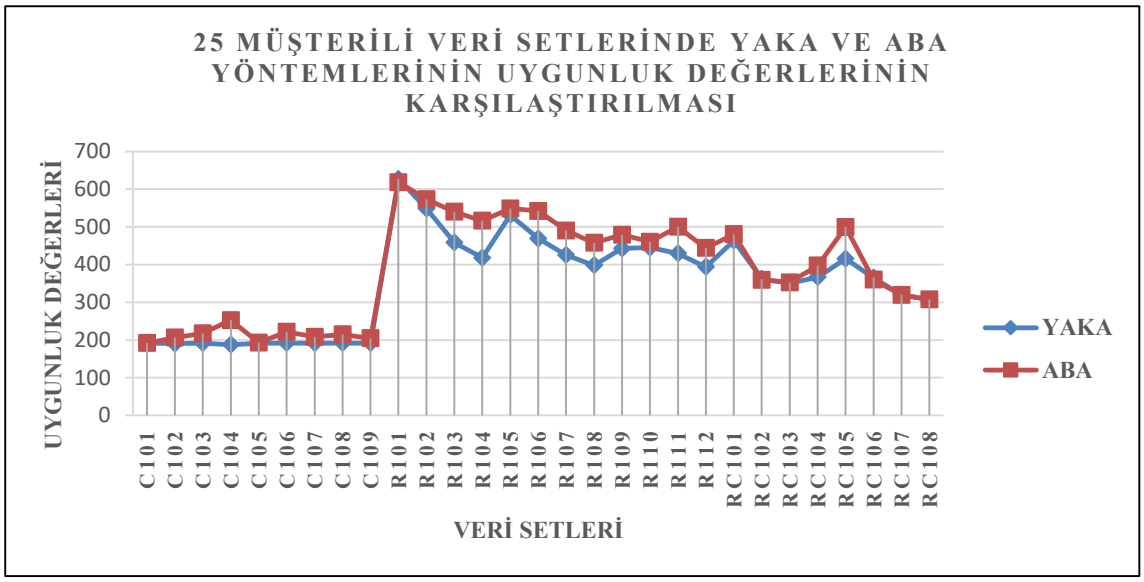
Tablo 24'te yer alan klasik zaman pencereli araç rotalama problemleri için 25 müşterili veri setleri üzerinde yapılan deneylerden her iki yöntem ile elde edilen minimum uygunluk değerleri Tablo 26'da özetlenmiştir. Tabloda yer alan en iyi uygunluk değerlerinin grafik üzerinde gösterimi ise Şekil 26'da yer almaktadır.

Tablo 26: C10_25, R10_25 ve RC10_25 Veri Grubuna Ait Uygunluk Değerleri

	Veri Seti	YAKA	En İyi Çözümden Sapma(%)	ABA	En İyi Çözümden Sapma(%)
C10_VERİ SETLERİ	**C101**	**191,81**	**0**	**191,81**	**0**
	C102	190,74	0	206,12	8,06
	C103	191,81	0	217,66	13,48
	C104	187,45	0	252,74	34,83
	C105	191,81	0	193,04	0,64
	C106	191,81	0	221,42	15,44
	C107	191,81	0	208,56	8,73
	C108	191,81	0	214,76	11,96
	C109	191,81	0	204,76	7,04
R10_VERİ SETLERİ	**R101**	**627,13**	**1,42**	**618,33**	**0,00**
	R102	548,95	0	573,34	4,44
	R103	458,37	0	539,8	17,77
	R104	417,96	0	516,3	23,53
	R105	531,54	0	548,82	3,25
	R106	468,96	0	542,13	15,6
	R107	425,27	0	490,43	15,32
	R108	398,29	0	457,75	14,93
	R109	442,63	0	479,03	8,22
	R110	445,18	0	460,48	3,44
	R111	429,7	0	499,84	16,32
	R112	394,1	0	443,88	12,63
RC10_VERİ SETLERİ	**RC101**	463,6	0	480,4	3,62
	RC102	**361,89**	**0,6**	**359,73**	**0**
	RC103	350,78	0	352,57	0,51
	RC104	366,81	0	397,33	8,32
	RC105	415,7	0	499,5	19,58
	RC106	**365,27**	**1,48**	**359,93**	**0**
	RC107	**319,99**	**0,17**	**319,44**	**0**
	RC108	**307,22**	**0**	**307,22**	**0**

Tablo 26'da C10_25 grubu için en iyi uygunluk değerleri incelendiğinde C101 verisi dışında kalan diğer verilerde YAKA yönteminin ABA yöntemine göre %0,64 ile %34,83 arasında değişen oranlarda daha iyi çözüm sağladığı görülmektedir. C101 problemi için her iki yöntem ile elde edilen uygunluk değerleri eşit çıkmıştır. R10_25 veri grubuna ait uygunluk değerleri karşılaştırıldığında ise YAKA yöntemi R101 problemi dışında kalan tüm problemler için ABA yöntemine göre %3,25 ile %23,53 arasında değişen oranlarda en iyi çözümü sağlamıştır. ABA yöntemi YAKA yöntemine göre %1,42 farkla daha iyi sonucu R101 probleminde göstermiştir. RC102, RC106, RC107 ve RC108 veri setleri dışındaki

veriler için YAKA yönteminin ABA yöntemine göre %0,51 ile %19,58 arasında değişen oranlarda daha iyi çözümler sağlamıştır. RC10_25 veri grubuna ait RC108 probleminde her iki yöntem ile yapılan deneyler sonucu ortaya çıkan uygunluk değerleri eşit iken; RC102, RC106 ve RC107 deney verileri üzerinde ise ABA yöntemi, YAKA yöntemine göre %0,6 ile %1,48 arasında değişen oranlarda daha iyi sonuçlar vermiştir.



Şekil 26: C10_25, R10_25 ve RC10_25 Veri Grubuna Ait Uygunluk Değerleri

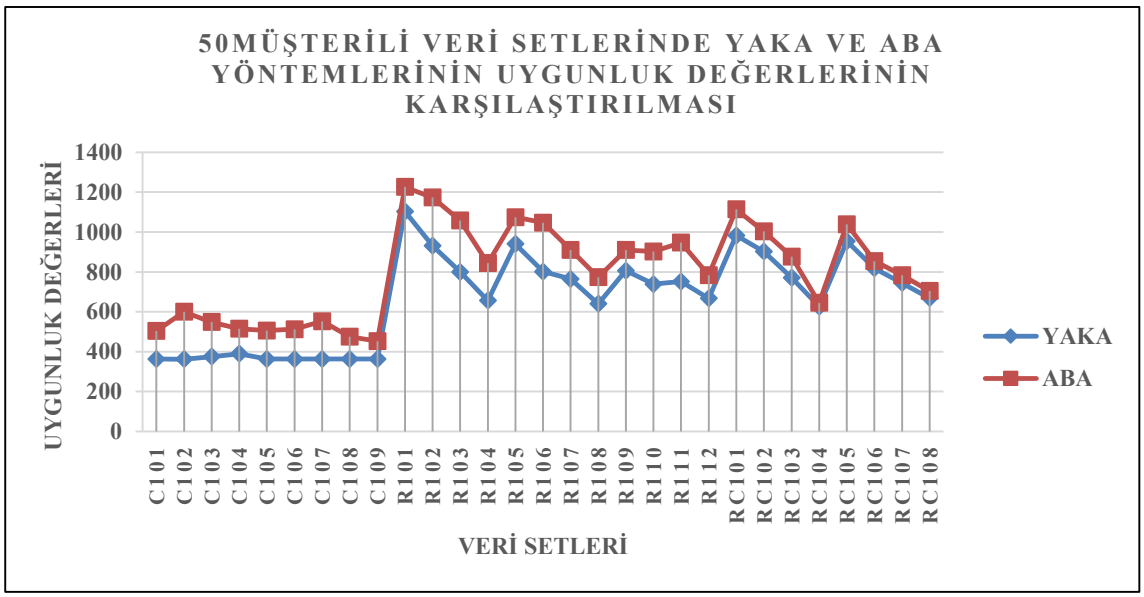
50 müşterili veri setleri üzerinde yapılan deneylerden elde edilen sonuçlar Tablo 25'te gösterilmişti. Bu tabloda yer alan değerlerden her bir veri grubuna ait problem verileri için bulunan en iyi uygunluk değerleri Tablo 27'da özetlenmiştir.

Tablo 27: C10_50, R10_50 ve RC10_50 Veri Grubuna Ait Uygunluk Değerleri

	Veri Seti	YAKA	En İyi Çözümden Sapma(%)	ABA	En İyi Çözümden Sapma(%)
C10_VERİ SETLERİ	**C101**	363,25	0	504,51	38,89
	C102	362,17	0	599,78	65,61
	C103	375,45	0	548,87	46,19
	C104	389,79	0	514,99	32,12
	C105	363,25	0	505,97	39,29
	C106	363,25	0	512,25	41,02
	C107	363,25	0	552,51	52,10
	C108	363,25	0	475,14	30,80
	C109	363,25	0	452,96	24,70
R10_VERİ SETLERİ	**R101**	1103,08	0	1226,09	11,15
	R102	931,02	0	1173,42	26,04
	R103	799,45	0	1057,69	32,30
	R104	656,36	0	843,63	28,53
	R105	940,05	0	1074,46	14,30
	R106	801,29	0	1046,7	30,63
	R107	764,14	0	910,58	19,16
	R108	639,92	0	773,01	20,80
	R109	806,58	0	909,62	12,78

	R110	738,84	0	902,39	22,14
	R111	751,84	0	946,58	25,90
	R112	667,15	0	783,84	17,49
RC10_VERİ SETLERİ	**RC101**	983,22	0	1113,5	13,25
	RC102	902,07	0	1003,16	11,21
	RC103	770,99	0	876,97	13,75
	RC104	626,52	0	645,19	2,98
	RC105	954,22	0	1038,69	8,85
	RC106	818,44	0	854,56	4,41
	RC107	744,63	0	784,04	5,29
	RC108	669,07	0	704,3	5,27

Tablo 27'den C10_50 problem verilerinin tamamında YAKA yönteminin, ABA yöntemine göre %24,70 ile %65,61 arasında değişen oranlarda en iyi çözümü sağlarken, R10_50 veri grubuna ait tüm problemlerde ise YAKA yöntemi ABA yöntemine göre %11,15 ile %32,30 arasında değişen oranlarda daha iyi performans göstermiştir. Son olarak uygunluk değerleri açısından RC10_50 veri grubuna ait sonuçlar değerlendirilecek olursa en iyi sonuç YAKA yöntemi ile elde edilmiştir. ABA yöntemi ise en iyi sonucu sağlayan YAKA yönteminden %2,98 ile %13,75 arasında değişen oranlarda uzaklıkta çözümler sağlamıştır. Şekil 27'da elde edilen uygunluk değerlerine ilişkin grafik gösterilmektedir.



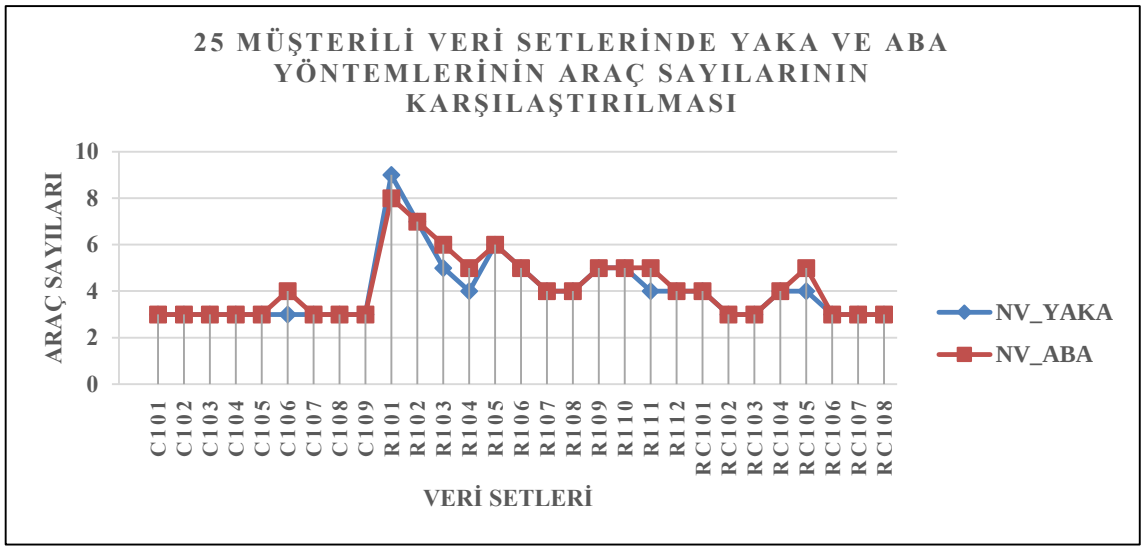
Şekil 27: C10_50, R10_50 ve RC10_50 Veri Grubuna Ait Uygunluk Değerleri

Çalışmada çözüm kalitesinin değerlendirilmesinde ele alınan bir diğer parametre ise kullanılan araç sayısı olmuştur. Tablo 24'te yer alan deneylerden C10_25, R10_25 ve RC10_25 veri gruplarına ait problemler için en iyi uygunluk değerlerini veren araç sayıları alınmış ve geliştirilen YAKA ve ABA yöntemi için karşılaştırmaya tabi tutulmuş ve Tablo 28'de belirtilmiştir.

Tablo 28: C10_25, R10_25 ve RC10_25 Veri Grubuna Ait Araç Sayılarının Karşılaştırılması

Veri Seti	NV_{YAKA}	En İyi Çözümden Sapma(%)	NV_{ABA}	En İyi Çözümden Sapma(%)
C101	3	0	3	0
C102	3	0	3	0
C103	3	0	3	0
C104	3	0	3	0
C105	3	0	3	0
C106	**3**	**0**	**4**	**33,33**
C107	3	0	3	0
C108	3	0	3	0
C109	3	0	3	0
R101	**9**	**12,5**	**8**	**0**
R102	7	0	7	0
R103	**5**	**0**	**6**	**20**
R104	**4**	**0**	**5**	**25**
R105	6	0	6	0
R106	5	0	5	0
R107	4	0	4	0
R108	4	0	4	0
R109	5	0	5	0
R110	5	0	5	0
R111	**4**	**0**	**5**	**25**
R112	4	0	4	0
RC101	4	0	4	0
RC102	3	0	3	0
RC103	3	0	3	0
RC104	4	0	4	0
RC105	**4**	**0**	**5**	**25**
RC106	3	0	3	0
RC107	3	0	3	0
RC108	3	0	3	0

Tabloda araç sayıları açısından yapılan incelemede her iki yöntemde de C106 dışında kalan bütün C10_25 deney setleri üzerinde aynı sonuçlar elde edilmiştir. R101, R103, R104 ve R111 problemleri dışındaki tüm R10_25 problemlerinde YAKA ve ABA yöntemi ile eşit sayıda araç kullanılarak sonuçlara ulaşılmıştır. R101 probleminde ABA yönteminde YAKA yöntemine göre %12,50 oranla daha iyi çözüm sağlanırken; R103, R104 ve R111 veri grubunda ise YAKA yöntemi ABA yöntemine göre %20 ile %25 oranlarda daha iyi sonuçlar elde etmiştir. Son olarak RC10_25 grubuna ait problem verilerinde ise her iki yöntem ile RC105 verisi dışındaki tüm verilerde eşit sayıda araç kullanılarak sonuçlara ulaşılmıştır. RC105 verisinde en uygun çözüm YAKA yöntemi ile 4 araç kullanılarak elde edilirken; ABA yöntemi ile ise 5 araçla en uygun sonuca ulaşılmıştır. Elde edilen araç sayılarına ilişkin grafik Şekil 28'de gösterilmektedir.



Şekil 28: C10_25, R10_25 ve RC10_25 Grubuna Ait Araç Sayılarının Karşılaştırılması

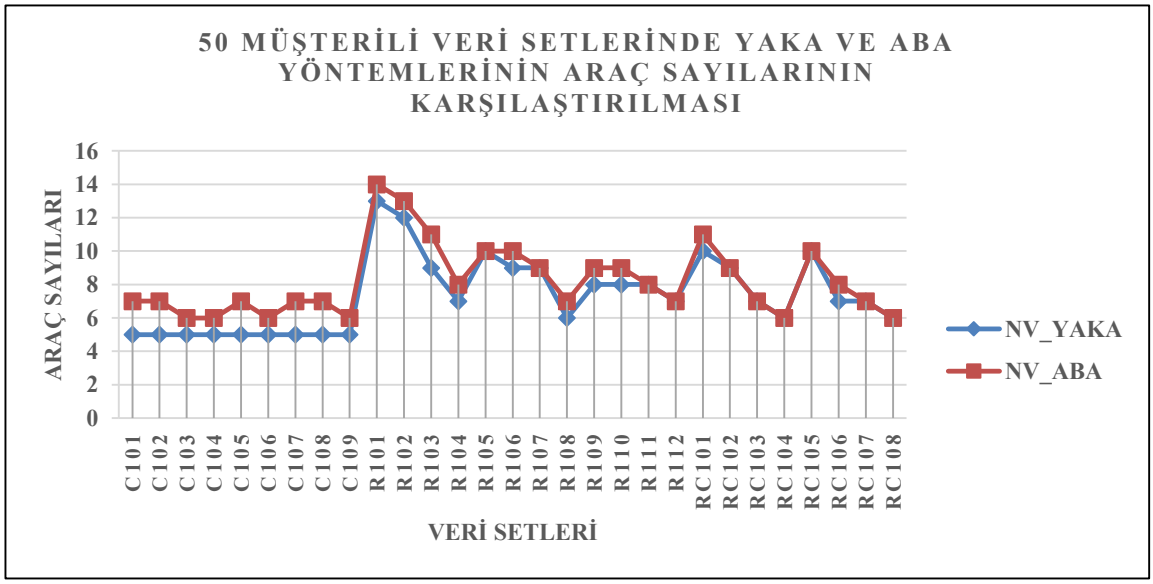
Tablo 25'ten 50 müşterili 29 test problemleri üzerinde her iki yöntem kullanılarak yapılan 5 tekrarlı deneylerin sonucunda ortaya çıkan en iyi sonuçların kaç araçla elde edildiğine ilişkin bilgiler veri grublarına göre Tablo 29'da ele alınmıştır.

Tablo 29: C10_50, R10_50 ve RC10_50 Veri Grubuna Ait Araç Sayılarının Karşılaştırılması

	Veri Seti	NV_{YAKA}	En İyi Çözümden Sapma(%)	NV_{ABA}	En İyi Çözümden Sapma(%)
C10_VERİ SETLERİ	**C101**	5	0,00	7	40,00
	C102	5	0,00	7	40,00
	C103	**5**	0,00	6	20,00
	C104	**5**	0,00	6	20,00
	C105	5	0,00	7	40,00
	C106	5	0,00	6	20,00
	C107	5	0,00	7	40,00
	C108	5	0,00	7	40,00
	C109	5	0,00	6	20,00
R10_VERİ SETLERİ	**R101**	13	0,00	14	7,69
	R102	12	0,00	13	8,33
	R103	9	0,00	11	22,22
	R104	7	0,00	8	14,29
	R105	**10**	**0,00**	**10**	**0,00**
	R106	9	0,00	10	11,11
	R107	**9**	**0,00**	**9**	**0,00**
	R108	6	0,00	7	16,67
	R109	8	0,00	9	12,50
	R110	8	0,00	9	12,50
	R111	**8**	**0,00**	**8**	**0,00**
	R112	**7**	**0,00**	**7**	**0,00**
RC10_VERİ SETLERİ	**RC101**	10	0,00	11	10,00
	RC102	**9**	**0,00**	**9**	**0,00**
	RC103	**7**	**0,00**	**7**	**0,00**

	RC104	**6**	**0,00**	**6**	**0,00**
	RC105	**10**	**0,00**	**10**	**0,00**
	RC106	7	0,00	8	14,29
	RC107	**7**	**0,00**	**7**	**0,00**
	RC108	**6**	**0,00**	**6**	**0,00**

C10_50 grubuna ait problemlerin tamamında YAKA ile 5 araç kullanılarak en iyi çözüm elde edilirken; ABA yöntemi ile veri grubuna göre kullanılan araç sayısı 6 ile 7 arasında değişkenlik göstermiştir. Diğer bir ifade ile YAKA yöntemi ABA yöntemine göre %20 ile %40 farkla daha iyi performans sergilemiştir. R105, R107, R111 ve R112 problemleri haricindeki tüm R10_50 veri grubundaki problemlerde YAKA yöntemi ABA yöntemine göre kullanılan araç sayısı bakımından %7,69 ile %22,22 arasında değişen oranlarda daha iyi performans sergilediği Tablo 29'dan görülmektedir. R105, R107, R111 ve R112 problemlerinde ise her iki yöntem ile aynı sayıda araç kullanılarak en iyi sonuçlara ulaşılmıştır. Bu problem verileri için aynı sayıda araç kullanılması ortaya çıkan uygunluk değerlerinin her iki yöntem için aynı olacağı anlamına gelmemektedir. Uygunluk değerleri açısından yapılan kıyaslamada YAKA yönteminin ABA yönteminden daha iyi sonuç ortaya çıkardığı Tablo 27'de belirtilmişti. RC101 ve RC106'de YAKA yöntemi ile daha az sayıda araç kullanılarak en iyi sonuca ulaşılmıştır. Her iki yöntem ile aynı sayıda araç kullanılarak en iyi çözümün elde edildiği veri setlerinin ise RC102 RC103, RC104, RC105, RC107 ve RC108 olduğu Tablo 29'dan anlaşılmaktadır. Şekil 29'da bu çözümlere ait grafik gösterilmektedir.



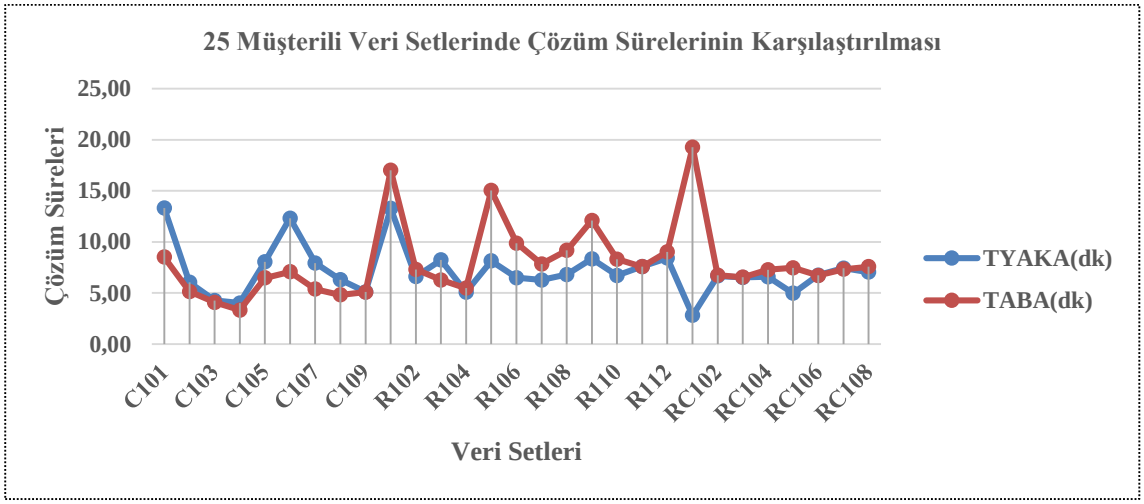
Şekil 29: C10_50, R10_50 ve RC10_50 Grubuna Ait Araç Sayılarının Karşılaştırılması

Çözüm kalitesinin değerlendirilmesinde kullanılan son parametre ise çözüm süresidir. Tablo 24'te yer alan deneylerden C10_25, R10_25 ve RC10_25 veri gruplarına ait problemler için en iyi uygunluk değerlerini veren çözüm süreleri alınmış ve geliştirilen YAKA ve ABA yöntemi için karşılaştırmaya tabi tutulmuş ve Tablo 30'da belirtilmiştir. Çözüm süreleri dakika olarak verilmiştir.

Tablo 30: C10_25, R10_25 ve RC10_25 Veri Grubuna Ait Uygunluk Değerlerinin Hesap Sürelerinin Karşılaştırılması

	Veri Seti	$T_{YAKA(dk)}$	En İyi Çözümden Sapma(%)	$T_{ABA(dk)}$	En İyi Çözümden Sapma(%)
C10_ VERİ SETLERİ	**C101**	13,32	56,71	**8,50**	0,00
	C102	6,03	17,54	**5,13**	0,00
	C103	4,25	4,42	**4,07**	0,00
	C104	4,00	20,48	**3,32**	0,00
	C105	8,05	24,42	**6,47**	0,00
	C106	12,32	74,26	**7,07**	0,00
	C107	7,92	47,49	**5,37**	0,00
	C108	6,30	31,25	**4,80**	0,00
	C109	5,10	0,39	**5,08**	0,00
R10_ VERİ SETLERİ	**R101**	**13,28**	0,00	17,00	28,01
	R102	**6,60**	0,00	7,30	10,61
	R103	8,23	31,26	**6,27**	0,00
	R104	**5,07**	0,00	5,47	7,89
	R105	**8,13**	0,00	15,03	84,87
	R106	**6,48**	0,00	9,87	52,31
	R107	**6,27**	0,00	7,83	24,88
	R108	**6,80**	0,00	9,17	34,85
	R109	**8,33**	0,00	12,10	45,26
	R110	**6,70**	0,00	8,30	23,88
	R111	7,58	0,13	**7,57**	0,00
	R112	**8,42**	0,00	9,02	7,13
RC10_VERİ SETLERİ	**RC101**	**2,80**	0,00	19,27	588,21
	RC102	**6,65**	0,00	6,73	1,20
	RC103	**6,52**	0,00	6,53	0,15
	RC104	**6,58**	0,00	7,26	10,33
	RC105	**4,97**	0,00	7,46	50,10
	RC106	6,73	0,30	6,71	0,00
	RC107	7,43	1,50	7,32	0,00
	RC108	7,04	0,00	7,56	7,39

Tablo 30 incelendiğinde, ABA yöntemi hesap süresi bakımından YAKA yöntemine göre C10_25 veri grubunda %0,39 ile %74,26 arasında değişen oranlarda tasarruf sağlarken; R103 ve R111 problemleri hariç R10_25 veri grubundaki diğer problemlerde YAKA yöntemi ABA yöntemine göre %7,13 ile %84,87 arasında değişen oranlarda tasarruf sağlamıştır. RC10_25 veri grubuna ait RC106 ve RC107 problemlerinde ABA yöntemi YAKA yöntemine göre sırasıyla %0,30 ile %1,50 oranında çözüm süresi bakımından daha iyi sonuç sağlamışken; RC10_25 grubuna ait diğer problemlerde YAKA yöntemi %0,15 ile %588,21 arasında daha iyi sonuçlar vermiştir. Genel olarak her iki yöntemi çözüm süreleri açısından değerlendirecek olursak 29 test probleminin 13'ünde ABA, YAKA'ya göre daha kısa sürede en iyi sonucu verirken; geriye kalan 16 problem verisinde YAKA daha iyi performans göstermiştir. Tablo 30'da yer alan değerlere ilişkin grafik Şekil 30'da gösterilmektedir.



Şekil 30: 25 Müşterili Veri Setleri İçin Hesap Sürelerin Karşılaştırılması

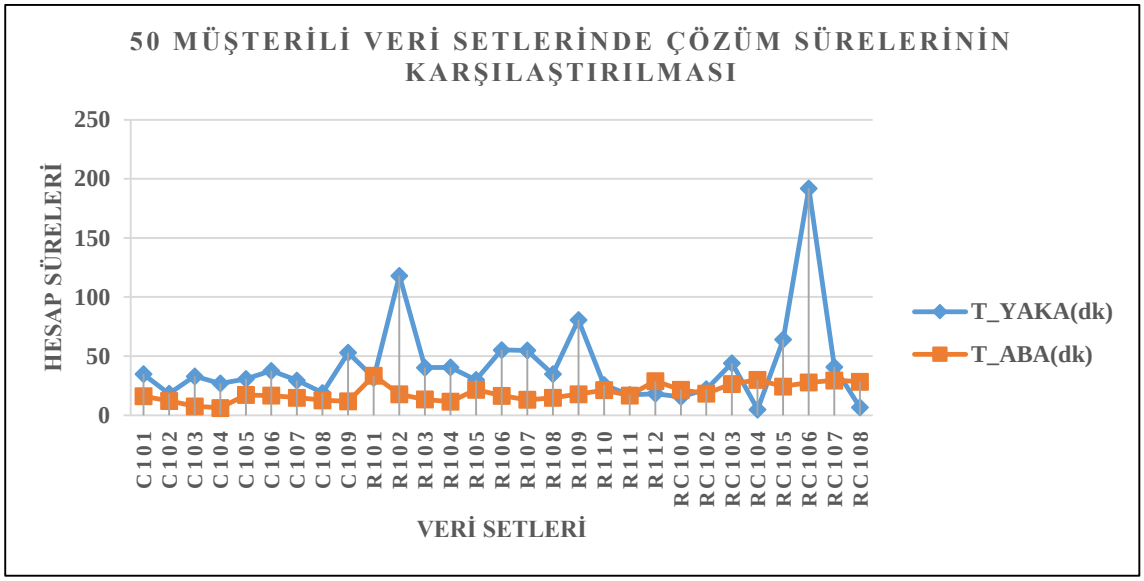
50 müşterili veri setleri üzerinde yapılan 5 tekrarlı deney sonucunda YAKA ve ABA yöntemi ile minimum ve maksimum uygunluk değerlerinin ne kadar sürede elde edildiğine ilişkin bilgiler Tablo 25'te verilmişti. Bu tablodan her iki yöntem ile elde edilen minimum uygunluk değerlerinin hesap süreleri karşılaştırmaya tabi tutulmuş ve Tablo 31'de gösterilmiştir.

Tablo 31: C10_50, R10_50 ve RC10_50 Veri Grubuna Ait Uygunluk Değerlerinin Hesap Sürelerinin Karşılaştırılması

	Veri Seti	$T_{YAKA(dk)}$	En İyi Çözümden Sapma(%)	$T_{ABA(dk)}$	En İyi Çözümden Sapma(%)
C10_ VERİ SETLERİ	**C101**	34,72	116,32	16,05	0,00
	C102	18,20	50,41	12,10	0,00
	C103	33,03	341,58	7,48	0,00
	C104	26,95	334,68	6,20	0,00
	C105	30,67	79,04	17,13	0,00
	C106	37,57	126,74	16,57	0,00
	C107	29,53	98,86	14,85	0,00
	C108	18,91	49,13	12,68	0,00
	C109	52,85	350,55	11,73	0,00
R10_ VERİ SETLERİ	**R101**	**31,62**	**0,00**	**33,53**	**6,04**
	R102	118,08	567,12	17,70	0,00
	R103	40,33	200,30	13,43	0,00
	R104	40,42	253,63	11,43	0,00
	R105	29,93	38,76	21,57	0,00
	R106	55,23	234,32	16,52	0,00
	R107	54,85	318,06	13,12	0,00
	R108	34,77	136,85	14,68	0,00
	R109	80,63	352,47	17,82	0,00
	R110	25,75	21,18	21,25	0,00
	R111	17,12	1,90	16,80	0,00
	R112	**18,18**	**0,00**	**28,92**	**59,08**
RC10_VER İ SETLERİ	**RC101**	15,55	0,00	21,47	38,07
	RC102	22,10	20,57	18,33	0,00
	RC103	44,08	66,03	26,55	0,00
	RC104	**4,83**	**0,00**	**29,90**	**519,05**

	RC105	64,03	163,17	24,33	0,00
	RC106	191,98	593,57	27,68	0,00
	RC107	40,73	38,21	29,47	0,00
	RC108	**6,57**	**0,00**	**28,40**	**332,27**

50 müşterili 29 test verisinin 25'inde ABA yönteminin hesap süresi bakımından %1,90 ile %593,57 arasında değişen oranlarda tasarruf sağladığı Tablo 31'den görülmektedir. R101, R112, RC104 ve RC108 problemlerinde ise YAKA yöntemi daha iyi sonuçlar vermiştir. Tablo 31'de belirtilen hesap süreline ilişkin grafik Şekil 31'de gösterilmektedir. Müşteri sayısındaki artış ile ABA yönteminin çözüm süresi açısından YAKA yöntemine oranla daha iyi performans gösterdiği ortaya çıkmıştır.



Şekil 31: 50 Müşterili Veri Setleri İçin Hesap Sürelerin Karşılaştırılması

SONUÇ VE DEĞERLENDİRME

Zaman pencereli araç rotalama problemine çözüm bulmak için kesin, sezgisel ve meta-sezgisel yöntemler kullanılmaktadır. Fakat bu tarz problemlerde büyük veri setlerinin kesin yöntemlerle çözülmesinin çok uzun zaman alması çoğu araştırmacıları sezgisel ve meta-sezgisel yöntemlere yöneltmiştir. Bu yöntemleri kullanarak çözüm arayan çok sayıda araştırmacı literatüre katkı sağlamıştır. Fakat incelemiş olduğumuz mevcut literatürde, probleme yapay arı kolonisi algoritması ve ateş böceği algoritması ile katkı sağlayan çalışmalar yok denecek kadar azdır. Bu çalışmada, zaman pencereli araç rotalama problemine yapay arı kolonisi algoritması ve ateş böceği algoritması ile eş zamanlı olarak çözüm aranmıştır. Probleme ilişkin geniş kapsamlı bir literatür taraması yapıldıktan sonra çözüm için kullanılacak olan yöntemler ayrıntılı bir şekilde ele alınmıştır.

Doğadaki bal arılarının yiyecek arama davranışlarından esinlenerek oluşturulan yapay arı kolonisi algoritmasına ve ateş böceklerinin yanıp sönen özelliklerinin ideal davranışlarına dayanan ateş böceği algoritmasına ayrıntılı bir şekilde yer verilmiştir. Sürü zekasının özellikleri, kendi kendine organize olan arıların doğadaki davranışlarına, yiyecek kaynaklarına, işçi ve işsiz arı olarak gruplandırılmasına, işçi arıların bulmuş oldukları yiyecek kaynağının niteliği ile ilgili bilgileri sergilemiş oldukları danslar ile kovanda bekleyen diğer arıları bilgilendirmesine, temel yapay arı kolonisi algoritmasının evrelerine ve parametrelerine, problem için geliştirilen yapay arı kolonisi algoritmasına, algoritmanın genel olarak işleyişinin küçük boyutlu bir örnek üzerinde gösterilmesine değinilmiştir.

Ateş böceklerinin biyolojik temellerine, geceleri yanıp sönen ışıkları sayesinde partnerlerini çekmeleri veya olası tehditlere karşı önlem aldıklarına, temel ateş böceği algoritmasına, ayrık ateş böceği algoritmasına, zaman pencereli araç rotalama problemi için geliştirilen ateş böceği algoritmasına değinilmiş ve geliştirilen algoritmanın işleyişi küçük boyutlu bir örnek üzerinde gösterilmiştir. İki ateş böceği arasındaki uzaklığın hesaplanması iki ayrı formülle ele alınmıştır. Temel ateş böceği algoritmasında bu uzaklık Öklid uzaklığı ile ele alınırken ayrık ateş böceği algoritmasında ise Hammington uzaklığı ile hesaplanmıştır.

ZPARP için geliştirilen YAKA ve ABA yöntemleri araç rotalama problemlerinde sıklıkla kullanılan en yakın komşu sezgiseli ile bütünleşik bir biçimde kullanılmıştır. En yakın komşu sezgiseli ile oluşturulan rota çözümlerinin kalitesini arttırabilmek adına ekleme, yer değiştirme, alt diziyi rastgele ekleme ve $2-opt^*$ operatörleri algoritmalara entegre edilmiştir.

Yöntemlere ait yazılımın C# programlama dilinde hazırlanmasının ardından, Yapay Arı Kolonisi Algoritması ve Ateş böceği Algoritması için en uygun parametre seti istatistiksel analizler ile tespit edilmiştir. Önerilen YAKA yönteminde problemin çözümü için 7 parametre ve her bir parametre 2 seviye olarak dikkate alınmış ve bu değerler bazında TAGUCHI deney tasarım yöntemi uygulanmıştır. TAGUCHI L16 ortogonal dizisi kullanılarak 16 deney yapılmış ve elde edilen sonuç verileri istatistiksel yöntemlerle analiz edilerek en uygun parametre seti belirlenmiştir.

Ateş böceği algoritmasında ise 1'i 2 seviye 5'i 3 seviye olmak üzere toplamda 6 parametre dikkate alınmıştır. TAGUCHI L18(2^13^5) ortogonal dizisi ile18 deney yapılmış ve yapılan istatistiksel testler ile en uygun parametre belirlenmiştir.

En uygun parametre setinin tespitinin yapılmasının ardından geliştirilen yöntemlerin etkinliği Solomon'un 25 ve 50 müşterili C10_25, R10_25, RC10_25, C10_50, R10_50 ve RC10_50 veri grubuna ait 29 test problemi kullanılarak test edilmiş ve elde edilen sonuçlar bilinen en iyi sonuçların yanında Bard vd., (2002), Ai e Kachitvichyanukul (2009), Gong vd., (2011), Yodwangjai ve Malampong (2022)'un sonuçları ile karşılaştırılmıştır. 25 müşterili veri setleri üzerinde önerilen YAKA yöntemi ile bilinen en iyi çözüme % 0,20 ile % 19,64 arasında değişen oranlarda yakın çözümler sağlanırken; aynı yöntem ile 50 müşterili 29 test problemleri üzerinde bilinen en iyi çözümden uzaklık % 0,21 ile % 15,86 arasında değişkenlik göstermiştir. YAKA yöntemi diğer dört yöntem ile kıyaslandığında, C103 hariç C10_25 grubuna ait tüm problemlerde aynı sonuçlar elde edilirken; R101, R106, RC102, RC103, RC104 ve RC106 problemleri dışındaki 25 müşterili test problemlerinin 14'ünde karşılaştırılan yöntemlerden daha iyi performans sağlanmıştır. ABA yöntemi ile ise 25 müşterili veri setleri üzerinde % 0,20 ile % 29,59 arasında değişen oranlarda çözümler bulunmuştur. C101, R101 ve RC102 haricindeki tüm veri setleri üzerinde ortaya çıkan sonuçlar bilinen en iyi çözüm ve karşılaştırılan diğer yöntemlerden daha yüksek çıkmıştır. C101 ile R101 problemlerinin ABA yöntemi ile çözümünden ortaya çıkan uygunluk değerleri ise karşılaştırılan diğer yöntemler ile aynı sonucu vermiştir. Yodgwangjai ve Malampong (2022)'ın sonucundan daha iyi sonucu RC102'de elde etmiştir. % 15,61 ile % 65,96 arasında değişen oranlarda bilinen en iyi çözümden uzaktaki sonuçlar 50 müşterili test problemleri üzerinde ABA yönteminin kullanılmasıyla gerçekleşmiştir.

Solomon'un test problemleri ile yöntemlerin etkinliğinin belirlenmesinin ardından ileri sürülen bu iki algoritma birbiri ile uygunluk değerleri, kullanılan araç sayıları ve hesap süreleri bakımından karşılaştırılmıştır. YAKA yönteminin ABA yöntemine göre uygunluk değerleri ve araç sayısı bakımından daha iyi çözümler sağladığı belirlenmiştir. Özellikle müşteri sayısındaki artış, her iki yöntem ile elde edilen en iyi uygunluk değerleri arasındaki farkın daha da net bir şekilde görülmesini sağlamıştır. Kullanılan araç sayıları bakımından değerlendirme neticesinde 25 müşterili test problemlerinin 23'ünde eşit sayıda araç kullanılarak uygunluk değerlerine ulaşılırken; diğer geriye kalan 5 problem verisinde YAKA ile ABA yöntemine göre daha az sayıda araç ile en uygun çözüm elde edilmiştir. Araç sayıları bakımından değerlendirmede ele alınan 50 müşterili problem setlerinin 19'unda daha iyi çözümler YAKA ile bulunmuş geriye kalan 10 problem verisinde en iyi çözümler ise her iki yöntemde de eşit sayıda araç kullanılarak ortaya çıkmıştır. Çözüm süreleri açısından yapılan karşılaştırmada 25 müşterili 29 test verisinin 16'sında YAKA yöntemi ile daha kısa sürede sonuçlara ulaşılırken; 50 müşterili 29 test verisinin 25'inde ABA yöntemi ile zamandan tasarruf sağlanarak minimum uygunluk değerleri elde edilmiştir.

25 müşterili veri setleri üzerinde yapılan deneylerde uygunluk değerleri bakımından 29 test probleminin 23'ünde YAKA yöntemi ABA yöntemine göre % 0,51 ile % 34,83 arasında değişen oranlarda daha iyi çözümler sağlamıştır. Deneye tabi tutulan 50 müşterili veri setlerinin 29'unun tamamında ise % 2,98 ile % 65,61 arasında değişen oranlarda ABA yöntemine göre en iyi çözüm YAKA ile elde edilmiştir. Araç sayısı ile ilgili olarak yapılan karşılaştırmalarda ise 25 müşterili veri setlerinden 5 problem verisi (C106, R103, R104, R111 ve RC105) üzerinde YAKA yöntemi ABA yöntemine göre % 20 ile % 33,33 arasında değişen oranlarda en az rota ile en iyi çözümü vermiştir. Bu oran 50 müşterili veri setlerinde ele alınan 29 problemin 19'unda % 7,69 ile % 40 arasında değişkenlik göstermiş ve ABA'ya oranla az sayıda araç kullanılarak oluşturulan çözümler YAKA ile elde edilmiştir. Hesap süresi ile ilgili olarak yapılan incelemede ise 25 müşterili veri setlerine ait 13 problem verisinde ABA yöntemi ile %0,13 ile %74,26 arasında değişen oranlarda daha iyi sonuçlar elde edilmiştir. Geri kalan diğer 16 problem verisinde ise YAKA yönteminin daha iyi çözümü %0,15 ile %588,21 arasında değişen oranlarda zamandan tasarruf sağlayarak elde ettiği görülmektedir. Müşteri sayısında ki artış ABA yöntemini çözüm süreleri açısından

olumlu etkilemiştir. 50 müşterili 26 test probleminde hesap süreleri açısından ABA yöntemi %1,90 ile %593,57 arasında değişen oranlarda iyi sonuçlar elde etmiştir.

Bazı problemlerin sonucunda yüksek oranda sapmalar ve varyasyon olduğu açıktır. Özellikle problem boyutu arttıkça ABA yöntemi ile uygunluk değerleri ve araç sayıları açısından bu sapmalar daha da ortaya çıkmıştır. Söz konusu bu sapmalar ve varyasyonlar problemin türünden ve geliştirilen ABA yönteminden kaynaklanabileceğinden önerilen yöntemin performansını ilerletmek için bu alanda daha fazla araştırma yapılması gerekmektedir. YAKA ile bazı problemlerde sonuçlardan sapmalar olsa da ABA'ya oranla uygunluk değerleri ve araç sayısı bakımından oldukça iyi sonuçlar elde edildiği açıktır. Müşteri sayısındaki artış ile birlikte YAKA yönteminin ABA yöntemine olan üstünlüğü uygunluk değerleri ve araç sayısı bakımından açıkça görülebilmektedir. Müşteri sayısının artması sadece ortaya çıkan uygunluk değerlerini ve araç sayılarını etkilememiş aynı zamanda hesap sürelerini de etkilemiştir. 58 problem verisinde genel olarak hesap süresi açısından yapılan kıyaslama da ABA yöntemi, YAKA yöntemine göre daha iyi performans göstermiştir. Fakat deney sonuçları, ZPARP için yapılan çalışmalar ile karşılaştırıldığında, YAKA yaklaşımının yüksek kaliteli çözümler bulabildiğini göstermiştir.

Gelecekteki çalışmalara gelince, önerilen YAKA ve FA'yı çoklu depo, heterojen filo ve bulanık zaman kısıtlamaları gibi diğer çeşitli ARP ile test etmek ilginç olabilir. Bununla birlikte, bazı problem durumlarının sonucunda ortaya çıkan yüksek sapma ve varyasyon ile başa çıkmak için ABA yönteminin performansını iyileştirme yönünde daha fazla araştırma yapılması önerilmektedir. YAKA yönteminin çözüm süresi açısından daha iyi performans göstermesi için değişken komşuluk arama sezgiseli, yarasa algoritması, guguk kuşu arama algoritması, yusufçuk algoritması gibi yöntemlerle entegre edilerek kullanılması sonraki yapılacak çalışmalar için tavsiye edilmektedir. Diğer taraftan ileri sürülen YAKA yönteminin diğer ARP varyantlarına veya problem tipine uygulanması da umut vericidir. Yeni etkili yöntemler ve bunlar arasında yeni işbirliği şemaları geliştirmek, gelecekte farklı hesaplama açısından zor problemleri çözmek için güçlü bir araç sağlayacaktır.

KAYNAKÇA

Acı, Ç. İ. ve Gülcan, H. (2019). A modified dragonfly optimization algorithm for single- and multiobjective problems using brownian motion. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2019/6871298

Aggarwal, D. ve Kumar, V. (2018). An improved firefly algorithm for the vehicle routing problem with time windows. *International Conference on Advances in Computing, Communications and Informatics.*

Ai, T. J. ve Kachitvichyanukul, V. (2009). A particle swarm optimization for vehicle routing problem with time windows. *International Journal Operational Research 6*(4).

Akca, K. (2015). *Hammadde tedarik aktivitesi için kesin zaman pencereli araç rotalama optimizasyonu* [Yayınlanmamış Yüksek Lisans Tezi]. Uludağ Üniversitesi. http://hdl.handle.net/11452/2759

Akkoyunlu, M. C. ve Engin, O. (2011). Kesikli harmoni arama algoritması ile optimizasyon problemlerinin çözümü: Literatür araştırması. *Selçuk Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi 26*(4).

Ali, N., Othman, M. A., Husain, M. N. ve Misran, M. H. (2014). A review of firefly algorithm. *ARPN Journal of Engineering and Applied Sciences 9*(10).

Altabeeb, A. M., Mohsen, A. M. ve Ghallab, A. (2019). An improved hybrid firefly algorithm for capacitated vehicle routing problem. *Applied Soft Computing 84*. https://doi.org/10.1016/j.asoc.2019.105728

Alvarenga, G. B., Mateus, G. R. ve de Tomi, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research 34*(6), 1561–1584. https://doi.org/10.1016/j.cor.2005.07.025

Alzaqebah, M., Abdullah, S. ve Jawarneh, S. (2016). Modified artificial bee colony for the vehicle routing problems with time windows. *Springer Plus 5* (1298). https://doi.org/10.1186/s40064-016-2940-8

Alzaqebah, M., Jawarneh, S., Sarim, H. M. ve Abdullah, S. (2018). Bees algorithm for vehicle routing problems with time windows. *International Journal of Machine Learning and Computing 8*(3). http://dx.doi.org/10.18178/ijmlc.2018.8.3.693

Apostolopoulos, T. ve Vlachos, A. (2011). Application of the firefly algorithm for solving the economic emissions load dispatch problem. *International Journal of Combinatorics*. https://doi.org/10.1155/2011/523806

Aydemir, E. (2006). *Esnek zaman pencereli araç rotalama problemi ve bir uygulama* (Yayın Nu. 184606) [Yüksek Lisans tezi, Gazi Üniversitesi].

Aydilek, İ. B. (2017). Değiştirilmiş ateşböceği optimizasyon algoritması ile kural tabanlı çoklu sınıflama yapılması. *Journal of the Faculty of Engineering and Architecture of Gazi University 32*(4), 1097-1107.

Badeau, P., Guertin, F., Gendreau, M., Potvin, J. Y. ve Taillard, E. (1997). A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies 5*(2), 109-122. https://doi.org/10.1016/S0968-090X(97)00005-3

Balseiro, S. R., Loiseau, I. ve Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers &Operations Research 38*(6), 954–966. https://doi.org/10.1016/j.cor.2010.10.011

Bansal, J. C., Sharma, H. ve Jadon, S. S. (2013). Artificial bee colony algorithm: a survey. *International Journal of Advanced Intelligence Paradigms 5*(1-2). https://doi.org/10.1504/IJAIP.2013.054681

Barbarosoglu, G. ve Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research 26*(3), 255-270. https://doi.org/10.1016/S0305-0548(98)00047-1

Bard, F. J., Kontoravdis, G. ve Yu, G. (2002). A branch and cut procedure for the vehicle routing problem with time windows. *Transportation Science 36*(2), 250-269.

Beasley, D., Bull, D. R. ve Martin, R. R. (1993). An overview of genetic algorithms: Part 1, fundamentals. *University Computing 15*(2), 56-69.

Bent, R. ve Hentenryck, P. V. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science 38*(4), 515–530. https://doi.org/10.1287/trsc.1030.0049

Berger, J. ve Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers Operations Research 31*(12), 2037-2053. https://doi.org/10.1016/S0305-0548(03)00163-1

Bhargava, S. (2013). A note on evolutionary algorithms and its applications. *Adults Learning Mathematics: An International Journal 8*(1), 31-45.

Bonabeau, E., Dorigo, M. ve Theraulaz, G. (1999). Swarm intelligence from natural to artificial systems (1. baskı). Oxford University Press.

Borcinova, Z. (2017). Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review (CRORR) 8*, 463-469. http://dx.doi.org/10.17535/crorr.2017.0029

Brad, J. B., Kontoravdis, G. ve Yu, G. (2002). A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science 36*(2), 250–269. http://dx.doi.org/10.1287/trsc.36.2.250.565

Bramel, J. ve Simchi-Levi, D. (1997). *The logic of logistics: Theory, algorithms, and applications for logistics management.* Springer.

Brandao, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research 38*(1), 140–151. https://doi.org/10.1016/j.cor.2010.04.008

Braysy, O. (2001). Genetic algorithms for the vehicle routing problem with time windows. *Arpakannus 1/2001 Special issue on Bioinformatics and Genetic Algorithms*.

Braysy, O. ve Gendreau, M. (2002). Tabu search heuristics for the vehicle routing problem with time windows. *Top 10* (2), 211-237. http://dx.doi.org/10.1007/BF02579017

Braysy, O. ve Gendreau, M. (2005). Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science 39*(1),104-118. http://dx.doi.org/10.1287/trsc.1030.0056

Braysy, O., Gendreau, M. ve Dullaert, W. (2004). Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics 10*(6), 587-611. http://dx.doi.org/10.1007/s10732-005-5431-6-

Calvete, H. I., Gale, C., Oliveros, M. J. ve Valverde, B. S. (2007). A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research 177*(3), 1720–1733. https://doi.org/10.1016/j.ejor.2005.10.010

Camazine, S. ve Sneyd, J. (1991). A model of collective nectar source selection by honey bees: self-organization through simple rules. *J. Theor. Biol. 149*, 547-571.

Chen, C. H. ve Ting, C. J. (2005). A hybrid ant colony system for vehicle routing problem with time windows. *Journal of the Eastern Asia Society for Transportation Studies 6*, 2822 – 2836. https://doi.org/10.11175/easts.6.2822

Chen, C. ve Zhou, K. (2018). Application of artificial bee colony algorithm in vehicle routing problem with time windows. *International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*. https://doi.org/10.1109/SDPC.2018.8664999

Chen, S., Chen, R. ve Gao, J. (2017). A modified harmony search algorithm for solving the dynamic vehicle routing problem with time windows. *Scientific Programming*, 1-13. http://dx.doi.org/10.1155/2017/1021432

Chiang, W. C. ve Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research 63*(1), 3-27. http://dx.doi.org/10.1007/BF02601637

Chiang, W.C. ve Russell, R.A. (2004). A metaheuristic for the vehicle routing problem with soft time windows. *Journal of the Operational Research Society 55*(12), 1298–1310. http://dx.doi.org/10.1057/palgrave.jors.2601791

Chiu, H. N. (1995) The integrated logistics management system: A framework and case study. *International Journal of Physical Distribution & Logistics Management 25*(6), 4-22. https://doi.org/10.1108/09600039510093249

Christopher, M. (2005). *Logistics and supply chain management: Creating value-adding networks* (3. baskı). FT Press.

Cook, W. ve Rich, J. L. (1999). A parallel cuttingplane algorithm for the vehicle routing problem with time windows. https://hdl.handle.net/1911/101910

Creput, J. C., Koukam, A. ve Hajjam, A. (2007). Self-organizing maps in evolutionary approach for the vehicle routing problem with time windows. *IJCSNS International Journal of Computer Science and Network Security 7*(1).

Dash, M. ve Mohanty, R. (2014). Cuckoo search algorithm for speech recognition. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 3*(10).

Daskin, M.S. (1985). Logistics: an overview of the state of the art and perspectives on future research. *Transportation Research 19A*(5/6), 383-98.

De Oliveira, H. C. B. ve Vasconcelos, G. C. (2010). A hybrid search method for the vehicle routing problem with time windows. *Annals Operations Research 180*, 125–144. http://dx.doi.org/10.1007/s10479-008-0487-y

De Oliveira, H. C. B., Vasconcelos, G. C. ve Alvarenga, G. B. (2006). A multi-start simulated annealing algorithm for the vehicle routing problem with time windows. *Proceedings of the Ninth Brazilian Symposium on Neural Networks (SBRN'06).* http://dx.doi.org/10.1109/SBRN.2006.4

Demircioğlu, M. (2009). *Araç rotalama probleminin sezgisel bir yaklaşım ile çözümlenmesi üzerine bir uygulama* (Yayın Nu. 241435) [Doktora tezi, Çukurova Üniversitesi].

Dhahri, A., Zidi, K. ve Ghedira, K. (2014). Variable neighborhood search based set covering ILP model for the vehicle routing problem with time windows. *Procedia Computer Science 29*, 844–854. https://doi.org/10.1016/j.procs.2014.05.076

Di Gaspero, L. (2003). *Local search techniques for scheduling problems: Algorithms and software tools* [Unpublished doctoral dissertation]. Universit`a degli Studi di Udine Dipartimento.

Ding, D. Ve Zou, X., (2016). *The optimization of logistics distribution route based on Dijkstra's Algorithm and C-W Savings Algorithm*. 6th International Conference on Machinery, Materials, Environment, Biotechnology and Computer.

Ding, Q., Hu, X., Sun, L. ve Wong, Y. (2012). An improved ant colony optimization and its application to vehicle routing problem with time windows. *Neurocomputing 98*, 101–107. https://doi.org/10.1016/j.neucom.2011.09.040

Durmuş, Y. T. (2019). Investigation of self-efficacy sources of classroom teachers based on various variables. *International Journal of Human Sciences 16*(1), 355-369.

Dursun, P. (2009). *Zaman pencereli araç rotalama probleminin genetik algoritma ile modellenmesi* (Yayın Nu. 251628) [Yüksek Lisan tezi, İstanbul Teknik Üniversitesi]. http://hdl.handle.net/11527/5720

El Hassani, A. H., Bouhafs, L. ve Koukam, A. (2008). A hybrid ant colony system approach for the capacitated vehicle routing problem and the capacitated vehicle routing problem with time windows. In: Caric, T. ve Gold, H. (Eds.), *Vehicle Routing Problem* (pp. 142). Intech. ISBN 978-953-7619-09-1. doi: 10.5772/64.

El-Sherbeny, N. A. (2010). Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University (Science) 22*(3), 123–131. https://doi.org/10.1016/j.jksus.2010.03.002

El-Zarka, S. (2010). *Designing a competency framework for logistics executives: The case of the readymade garments manufacturers in Egypt* (Yayın Nu. U641385) [Doctoral Thesis]. ProQuest Dissertations & Theses Global https://www.proquest.com/dissertations-theses/designing-competency-framework-logistics/docview/1689623371/se-2?accountid=86200

Erol, V. (2006). *Araç rotalama problemleri için populasyon ve komşuluk tabanlı meta-sezgisel bir algoritmanın tasarımı ve uygulaması* (Yayın Nu. 180526) [Yüksek Lisans tezi, Yıldız Teknik Üniversitesi]. http://localhost:6060/xmlui/handle/1/2323

Ezzat, M., Amr, M. ve Kassem, S. S. (2019). *Logistics 4.0: Definition and historical background.* Novel Intelligent and Leading Emerging Sciences Conference, Nile University, Giza, Egypt. http://dx.doi.org/10.1109/NILES.2019.8909314

F, T. J. ve Kachitvichyanukul, V. (2009). A particle swarm optimisation for vehicle routing problem with time windows. *International Journal of Operational Research 6*(4). https://doi.org/10.1504/IJOR.2009.027156

Ferland, J. A., & Costa, D. (2001). Heuristic search methods for combinatorial programming problems. *Publication DIRO-1193, Department of Computer Science and Operations Research*.

Fister Jr., I., Fister, D. ve Fister, I. (2013). A comprehensive review of cuckoo search: variants and hybrids. *International Journal of Mathematical Modelling and Numerical Optimisation 4*(4). https://doi.org/10.1504/IJMMNO.2013.059205

Fister, I., Fister Jr., I., Yang, X. S. ve Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation 13*, 34-36. https://doi.org/10.1016/j.swevo.2013.06.001

Friedrich, H. ve Gumpp, J. (2014). Simplified modeling and solving of logistics optimization problems. *International Journal of Transportation 2*(1), 33-52. http://dx.doi.org/10.14257/ijt.2014.2.1.03

Fu, Z., Eglese, R. ve Li, L. Y. O. (2008). A unified tabu search algorithm for vehicle routing problems with soft time windows. *Journal of the Operational Research Society 59*, 663 –673. https://doi.org/10.1057/palgrave.jors.2602371

Gambardella, L. M., Taillard, E. ve Agazzi, G. (1999). A multiple ant colony system for vehicle routing problem with time windows. *Technical Report IDSIA-06-99*. IDSIA. Lugano, Switzerland.

Gao, X. Z., Govinddasamy, V., Xu, H., Wang, X. ve Zenger, K. (2015), Review article harmony search method: Theory and applications. *Computational Intelligence and Neuroscience 2015*(2), 1-10. http://dx.doi.org/10.1155/2015/258491

Ghoseiri, K. ve Ghannadpour, S. F. (2010). Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing 10*(4), 1096–1107. https://doi.org/10.1016/j.asoc.2010.04.001

Ghoumrassi, A. ve Tigu, G. (2017). *The impact of the logistics management in customer satisfaction.* Proceedings of the International Conference on Business Excellence 11(1). https://doi.org/10.1515/picbe-2017-0031

Golden, B. L. (1975). *Vehicle routing problems: Formulations and heuristic solution techniques* (Report No. ADA013639). M.I.T. Operations Research Center, Defense Technical Information Center. https://apps.dtic.mil/sti/pdfs/ADA013639.pdf

Gong, Y. J., Zhang, J., Liu, O., Huang, R. Z., Chung, H. S. H. ve Shi, Y. H. (2011). Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42*(2), 254-267.

Gothania, B., Mathur, G. ve Yadav, R. P. (2014). Accelerated artificial bee colony algorithm for parameter estimation of frequency-modulated sound waves. *International Journal of Electronics and Communication Engineering 7*(1), 63–74.

Göçken, T., Yaktubay, M. ve Kılıç, F. (2018). Zaman pencereli araç rotalama problemi çözümü için çok amaçlı genetik algoritma yaklaşımı. *Gazi University Journal of Science Part C: Design and Technology 6*(4), 774-786.

Gökçe, B. ve Taşgetiren, S. (2009). Kalite için deney tasarımı. *Makine Teknolojleri Elektronik Dergisi 61*, 71-83.

Groer, C. (2008). *Parallel and serial algorithms for vehicle routing problems* [Doctoral dissertation, University of Maryland]. http://hdl.handle.net/1903/9011

Gunawan, V. (2020). Penerapan dragonfly algorithm untuk menyelesaikan capatitated vehicle routing problem with time windows [Undergraduate Theses, Universitas Katolik Parahyangan]. http://hdl.handle.net/123456789/10845

Gupta, J. ve Diwaker, C. (2017). Evaluation of capacitated vehicle routing problem with time windows using ACO-GA. International Journals of Advanced Research in Computer Science and Software Engineering ISSN: 2277-128X (Volume-7, Issue-6). http://dx.doi.org/10.23956/ijarcsse/V7I6/0319

Gülenç, İ. F. ve Karagöz, B. (2008). E-lojistik ve Türkiye'de E-lojistik uygulamaları. *Kocaeli Üniversitesi Sosyal Bilimler Enstitüsü Dergisi 15*(1), 73-91.

Hashimoto, H., Yagiura, M. ve Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization 5*(2), 434–456. https://doi.org/10.1016/j.disopt.2007.05.004

Hertrich C., Hungerländer P. ve Truden C. (2019). Sweep algorithms for the capacitated vehicle routing problem with structured time windows. In: Fortz B., Labbé M. (eds), *Operations Research Proceedings 2018* (pp. 127-133). *Operations Research Proceedings (GOR (Gesellschaft für Operations Research e.V.))*. Springer, Cham. https://doi.org/10.1007/978-3-030-18500-8_17

Homberger, J. ve Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research 162*(1), 220–238. https://doi.org/10.1016/j.ejor.2004.01.027

Hussain, K., Salleh, M. N. M., Cheng, S., Shi, Y. ve Naseem, R. (2020). Artificial bee colony algorithm: A component-wise analysis using diversity measurement. *Journal of King Saud University – Computer and Information Sciences 32*(7), 794-808. https://doi.org/10.1016/j.jksuci.2018.09.017

Iqbal, S. (2012). *Vehicle routing problems with soft time windows* (Yayın Nu. 111279) [Master Thesis, Bangladesh University].

Jakara, M., Skrinjar, J. P. ve Brnjac, N. (2019). Vehicle routing problem: Case study on logistics company in Croatia. *International Journal for Traffic and Transport Engineering, 9*(4): 456 – 470. http://dx.doi.org/10.7708/ijtte.2019.9(4).10

Jawarneh, S. ve Abdullah, S. (2015). Sequential insertion heuristic with adaptive bee colony optimisation algorithm for vehicle routing problem with time windows. *PLoS ONE 10*(7). https://doi.org/10.1371/journal.pone.0130224

Jerabek, K., Majercak, P., Kliestik, T. ve Valaskova, K. (2016). Application of clark and Wright´s savings algorithm model to solve routing problem in supply logistics. *"Naše more" 63*(3), 115-119.

Joubert, J. W. ve Claasen, S. J. (2006). A sequential insertion heuristic for the initial solution to a constrained vehicle routing problem. *ORİON 22* (1), 105–116. http://dx.doi.org/10.5784/22-1-36

Kachitvichyanukul, V. (2012). Comparison of three evolutionary algorithms: GA, PSO, and DE. *Industrial Engineering & Management Systems 11*(3), 215-223.

Kaddouri, Z. ve Omary, F. (2017). Application of the tabu search algorithm to cryptography. (IJACSA) *International Journal of Advanced Computer Science and Applications 8*(7). http://dx.doi.org/10.14569/IJACSA.2017.080712

Kantawong, K. ve Pravesjit, S. (2020). An enhanced ABC algorithm to solve the vehicle routing problem with time windows. *Ecti Transactions on Computer and Information Technology 14*(1). https://doi.org/10.37936/ecti-cit.2020141.200016

Kao, Y., Chen, M. H. ve Huang, Y.T. (2012). A hybrid algorithm based on ACO and PSO for capacitated vehicle routing problems. *Hindawi Publishing Corporation Mathematical Problems in Engineering.* doi:10.1155/2012/726564.

Karaaslan, E. ve Zengin, K. (2016). *Ateş böceği algoritması ile haftalık ders programı hazırlama* [Konferans Sunumu]. EEB, Elektrik-Elektronik ve Bilgisayar Sempozyumu, Tokat, TÜRKİYE.

Karaboga, D. ve Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization 39*, 459–471. https://doi.org/10.1007/s10898-007-9149-x

Karaboga, D., Gorkemli, B., Ozturk, C. ve Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review 42*, 21–57. https://doi.org/10.1007/s10462-012-9328-0

Karakoyun, M. (2015). *Kurbağa sıçrama algoritmasının kümeleme problemlerine uygulanması* (Yayın Nu. 418871) [Yüksek Lisans tezi, Selçuk Üniversitesi]. http://hdl.handle.net/123456789/14318

Katiyar, S., Nasiruddin, I. ve Ansari, A. Q. (2015). Ant colony optimization: A tutorial review. *National Conference on Advances in Power and Control*. https://www.researchgate.net/publication/281432201_Ant_Colony_Optimization_A_Tutorial_Review

Kaveh, N. ve Samani, N. K. (2009). *How collaborative logistics management increases supply chain efficiency* (Yayın Nu. 2320/5405) [Master Thesis, University of Boras]. http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Ahb%3Adiva-19557

Keskintürk, T., Topuk, N. ve Özyeşil, O. (2015). Araç rotalama problemleri ile çözüm yöntemlerinin sınıflandırılması ve bir uygulama. *İşletme Bilimi Dergisi 3*(2).

Kırcı, P. (2016). An optimization algorithm for a capacitated vehicle routing problem with time windows. *Sadhana 41*(5), 519–529. http://dx.doi.org/10.1007/s12046-016-0488-5

Kiremitçi, B., Kiremitçi, S. ve Keskintürk, T. (2014). Zaman pencereli çok araçlı dağıtım toplamalı rotalama problemi için gerçek değerli genetik algoritma yaklaşımı. İstanbul Üniversitesi İşletme Fakültesi Dergisi 43(2), 391-403. https://dergipark.org.tr/tr/pub/iuisletme/issue/9253/115771

Kok, A. L., Meyer, C. M., Kopfer, H., Marco, J. ve Schutten, J. (2010). Dynamic programming algorithm for the vehivle routing problem with time windows and EC social legislation. *Transportation Science 44*(4), 442-454. http://dx.doi.org/10.1287/trsc.1100.0331

Kongkaew, W. (2017). Bat algorithm in discrete optimization: A review of recent applications. *Songklanakarin Journal of Science and Technology Songklanakarin 39* (5), 641-650.

Kukovic, D., Topolsek, D., Rosi, B. ve Jereb, B. (2014). A comparative literature analysis of definitions for logistics: Between general definition and definitions of subcategories. *Business Logistics in Modern Management, Josip Juraj Strossmayer University of Osijek, Faculty of Economics, Croatia14*,111-122.

Kumar, S. N. ve Panneerselvam, R., (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management 4*, 66-74.

Kuo, R.J., Chiu, C.Y. ve Lin, Y.J. (2004). Integration of fuzzy theory and ant algorithm for vehicle routing problem with time windows. *IEEE Annual Meeting of the Fuzzy Information*. https://doi.org/10.1109/NAFIPS.2004.1337428

Kuram, Ç. (2016). *Zaman pencereli araç rotalama problemlerinin popülasyon tabanlı sezgisel yöntemler ile optimize edilmesi* (Yayın Nu. 446437) [Yüksek Lisans tezi, İstanbul Üniversitesi].

Laporte, G., Toth, P. ve Vigo, D. (2013). Vehicle routing: historical perspective and recent contributions. *EURO Journal Transportation and Logistics 2*, 1–4. https://doi.org/10.1007/s13676-013-0020-6

Lei, L., Sun, S., Ying, C., Tan, S. ve Sun, Z. (2018). The application of cuckoo search algorithm in the path planning of logistics vehicles with time windows. *Advances in Intelligent Systems Research 161*. https://dx.doi.org/10.2991/tlicsc-18.2018.34

Li, H. ve Lim, A. (2003). Local search with annealing-like restarts to solve the VRPTW. *European Journal of Operational Research 150*(1), 115-127. http://dx.doi.org/10.1016/S0377-2217(02)00486-1

Li, X. ve Yang, G. (2016). Artificial bee colony algorithm with memory. *Applied Soft Computing 41*, 362-372. https://doi.org/10.1016/j.asoc.2015.12.046

Liberatore, F., Righini, G. ve Salani, M. (2011). A column generation algorithm for the vehicle routing problem with soft time windows. *4OR-Q J Oper Res 9*, 49–82. https://doi.org/10.1007/s10288-010-0136-6

Lin, S. W., Ying, K.C., Lee, Z. J. ve Chen, H. S. (2006). Vehicle routing problems with time windows using simulated annealing. *IEEE International Conference on Systems, Man, and Cybernetics*. https://doi.org/10.1109/ICSMC.2006.384458

Liu, C., Tao, W., Zhao, C., Li, X., Su, Y. ve Sun, Z. (2019). Research on vehicle routing problem with time windows based on the dragonfly algorithm. *IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)* https://doi.org/10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00037

Liu, X., Jiang, W. ve Xie, J. (2009). Vehicle routing problem with time windows: A hybrid particle swarm optimization approach. *Fifth International Conference on Natural Computation*. https://doi.org/10.1109/ICNC.2009.353

Luo, J. ve Chen, M. R. (2014). Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the MDVRP and the MDVRPTW. *Computers & Industrial Engineering 72*, 84-97. https://doi.org/10.1016/j.cie.2014.03.004

Luo, J. ve Chen, M. R. (2014). Improved shuffled frog leaping algorithm and its multi-phase model for multi-depot vehicle routing problem. *Expert Systems with Applications 41*, 2535–2545. http://dx.doi.org/10.1016/j.eswa.2013.10.001

Luo, J., Li, X., Chen, M. R. ve Liu, H. (2015). A novel hybrid shuffled frog leaping algorithm for vehicle routing problem with time windows. *Information Sciences 316*, 266–292. https://doi.org/10.1016/j.ins.2015.04.001

Mahmudy, W. F. (2014). Improved simulated annealing for optimization of vehicle routing problem with time windows. *Jurnal Ilmiah KURSOR 7*(3), 109-116. https://doi.org/10.21107/KURSOR.V7I3.1092

Maleki, F., Yousefikhoshbakht, M. ve Rahati, A. (2017). A hybrid self-adaptive global best harmony search algorithm for the vehicle routing problem with time windows. *BRAIN – Broad Research in Artificial Intelligence and Neuroscience 8*(4).

Mao, S., Zhao, X., Wang, Z., Zheng, M. ve Xie, W. (2016). The uncertain time dependent vehicle routing problem with soft time windows. *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. http://dx.doi.org/10.1109/FUZZ-IEEE.2016.7737665

Marinakis, Y., Marinaki, M. ve Migdalas, A. (2019). A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows. *Information Sciences 481*, 311-329. https://doi.org/10.1016/j.ins.2018.12.086

Miaoer, X. (2017). *Distribution logistics and logistics customer services of B2C E-tailing Industry in the Chinese Market.* [Degree Thesis]. (10024/15358). https://urn.fi/URN:NBN:fi:amk-2017090714893

Ming-yao, Q., Li-xin, M., Le, Z. ve Hua-yu, X. (2008). A new tabu search heuristic algorithm for the vehicle routing problem with time windows. *IEEE International Conference on Management Science and Engineering 15th Annual Conference Proceedings*. https://doi.org/10.1109/ICMSE.2008.4669126

Miranda, D. M. ve Conceiçao, S. V. (2016). The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications 64*, 104–116. https://doi.org/10.1016/j.eswa.2016.07.022

Moccia, L., Cordeau, J. F. ve Laporte, G. (2012). An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society 63*, 232–244. https://doi.org/10.1057/jors.2011.25

Mogaka, L., Murage, D. K. ve Saulo, M. J. (2016). Power optimization and prioritization in an Island supplied by a rotating machine based distributed generator using artificial bee colony algorithm. *International Journal of Energy and Power Engineering 5*(1), 15-21.

Moghaddam, R. T., Gazanfari, M., Alinaghian, M., Salamatbakhsh, A. ve Norouzi, N. (2011). A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing. *Journal of Manufacturing Systems 30*(2), 83– 92. https://doi.org/10.1016/j.jmsy.2011.04.005

Moscato, P. ve Cotta, C. (2010). A Modern Introduction to Memetic Algorithms. In: Gendreau M., Potvin JY. (eds) *Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol 146*. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-1665-5_6

Nagata, Y., Braysy, O. ve Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research 37*(4), 724—737. https://doi.org/10.1016/j.cor.2009.06.022

Najera, A. G. ve Bullinaria, J. A. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research 38*(1), 287–300. https://doi.org/10.1016/j.cor.2010.05.004

Nalepa, J. ve Blocho, M. (2016). Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows. *Soft Comput 20*, 2309–2327. https://doi.org/10.1007/s00500-015-1642-4

Nazif, H. ve Lee, L. S. (2010). Optimized crossover genetic algorithm for vehicle routing problem with time windows. *American Journal of Applied Sciences 7* (1), 95-101. http://dx.doi.org/10.3844/ajassp.2010.95.101

Neelima, S., Satyanarayana, N. ve Murthy, P. K. (2016). A comprehensive survey on variants in artificial bee colony (abc). *(IJCSIT) International Journal of Computer Science and Information Technologies 7*(4), 1684-1689.

Nesmachnow, S. (2014). An overview of metaheuristics: accurate and efficient methods for optimisation. *International Journal of Metaheuristics 3*(4). http://dx.doi.org/10.1504/IJMHEUR.2014.068914

Nikolic, M., Teodorovic, D. ve Selmic, M. (2013). Solving the vehicle routing problem with time windows by bee colony optimization metaheuristic. *1st Logistics International Conference, Belgrade, Serbia, 28 - 30 November*.

Niu, B., Wang, H., Tan, L. J., Li, L. ve Wang, J. W. (2012). Vehicle routing problem with time windows based on adaptive bacterial foraging optimization. In D. S. Huang, J. Ma, K. H. Jo and M. M. Gromiha (Eds.), *Intelligent Computing Theories and Applications* (pp. 672–679). Springer. DOI: 10.1007/978-3-642-31576-3_85

Osaba E., Carballedo R., Yang XS., Fister Jr. I., Lopez-Garcia P., Del Ser J. (2018) On Efficiently Solving the Vehicle Routing Problem with Time Windows Using the Bat Algorithm with Random Reinsertion Operators. In: Yang XS. (eds) *Nature-Inspired Algorithms and Applied Optimization. Studies in Computational Intelligence. Vol. 744*. Springer, Cham. https://doi.org/10.1007/978-3-319-67669-2_4

Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Masegosa, A. D. ve Perallos, A. (2018). Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems. *Neurocomputing 271*(3), 2-8. https://doi.org/10.1016/j.neucom.2016.11.098

Osaba, E., Yang, X. S., Diaz, F., Onieva, E., D. Masegosa, A. ve Perallos, A. (2017). A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. *Soft Computing 21*, 5295–5308. https://doi.org/10.1007/s00500-016-2114-1

Ostfeld, A. (Eds.) (2011). Ant colony optimization-methods and applications. InTech. https://doi.org/10.5772/577

Özdemir, M. (2013). *Zaman kısıtı altında takım oryantiring problemlerinin yapay arı kolonisi yaklaşımı ile çözümü* (Yayın Nu. 340428) [Yüksek Lisans tezi, İstanbul Üniversitesi].

Pan, F., Ye, C., Wang, K. ve Cao, J. (2013). Research on the vehicle routing problem with time windows using firefly algorithm. *Journal of Computers 8*(9). http://dx.doi.org/10.4304/jcp.8.9.2256-2261

Pant, M., Thangaraj, R. ve Abraham, A. (2009). Particle swarm optimization: Performance tuning and empirical analysis. *Foundations of Computational Intelligence 3*, 101–128. http://dx.doi.org/10.1007/978-3-642-01085-9_5

Pentapalli, V. V. G. ve Varma P., R. K. (2016). Cuckoo search optimization and its applications: A review. *International Journal of Advanced Research in Computer and Communication Engineering 5*(11). https://doi.org/10.17148/IJARCCE.2016.511119

Petelina, A. (2016). *Importance in the development of logistics operations in start-up e-commerce business* (Yayın Nu. 10024/287) [Bachelor's Thesis, Lahti University]. http://www.theseus.fi/handle/10024/118003

Polacek, M., Hartl, R. F. ve Doerner, K. (2004). A variable neighborhood search for the multi-depot vehicle routing problem with time windows. *Journal of Heuristics 10*, 613-627. https://doi.org/10.1007/s10732-005-5432-5

Pratiwi, A. B., Pratama, A., Sa'diyah, I. ve Suprajitno, H. (2018). Vehicle routing problem with time windows using natural inspired algorithms. *Journal of Physics: Conference Series 974*(1). http://dx.doi.org/10.1088/1742-6596/974/1/012025

Qamhan, M. A., Qamhan, A. A., Al-Harkan, İ. M. ve Alotaibi, Y. A. (2019). Mathematical modeling and discrete firefly algorithm to optimize scheduling problem with release date, sequence-dependent setup time, and periodic maintenance. *Mathematical Problems in Engineering*. https://doi.org/10.1155/2019/8028759

Qureshi, A. G., Taniguchi, E. ve Yamada, T. (2009). Column generation-based heuristics for vehicle routing problem with soft time Windows. *Proceedings of the Eastern Asia Society for Transportation Studies 7*.

Qureshi, A. G., Taniguchi, E. ve Yamada, T. (2010). Exact solution for the vehicle routing problem with semi soft time windows and its application. *Procedia Social and Behavioral Sciences 2*(3), 5931-5943. https://doi.org/10.1016/j.sbspro.2010.04.008

Rahman, C. M. ve Rashid, T. A. (2019). Dragonfly algorithm and its applications in applied science survey. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2019/9293617

Rego, C. ve Alidaee, B. (Eds.) (2005). *Metaheuristic optimization via memory and evolution: Tabu search and scatter search* (1. baskı). Springer US. https://doi.org/10.1007/b102147

Rezaeipanah, A., Ahmadi, G., Hajiani, M. ve Darzi, M. R. (2019). An improved hybrid cuckoo search algorithm for vehicle routing problem with time windows. *Journal of Quality Engineering and Production Optimization* 4(2), 189-208. DOI: 10.22070/JQEPO.2020.4978.1119.

Ropke, S. (2005). *Heuristic and exact algorithms for vehicle routing problems. Computer Science* [Ph.D. Thesis, University of Copenhagen]. https://orbit.dtu.dk/en/publications/cde05a62-8163-4693-b698-0fae72cf302c

Rushton, A., Croucher, P. ve Baker, P. (2010). *The handbook of logistics and distribution management* (3. Baskı). Kogan Page.

Said, G. A. E. N., Mahmoud, A. M. ve El-Horbaty, E. S. M. (2014). A comparative study of meta-heuristic algorithms for solving quadratic assignment problem. *(IJACSA) International Journal of Advanced Computer Science and Applications 5*(1). https://dx.doi.org/10.14569/IJACSA.2014.050101

Santana, R. M. (2016). *Heuristic algorithms and variants of the vehicle routing problem for a distribution company: A case study*. [Master of Thesis].Faculdade De Ciencias E Tecnologia Universidade Nova De Lisboa, The European Master's Program in Computational Logic.

Sarah Namany, "Capacitated Vehicle Routing Problem with Variable Fleet of Delivery Vehicles of Uniform Capacity", 2017,

http://www.aui.ma/sse-capstone-repository/pdf/spring2017/CAPACITATED%20VEHICLE%20ROUTING%20PROBLEM%20WITH%20VARIABLE%20FLEET%20OF%20DELIVERY%20VEHICLES%20OF%20UNIFORM%20CAPACITY_SARAH%20NAMANY%20(Final).pdf, (28/06/2022), s. 12.

Schulze, T. ve Fahle, T. (1999). A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Research 86*, 585-607. https://doi.org/10.1023/A:1018948011707

Seeley, T. D., Camazine, S. ve Sneyd, J. (1991). Collective decision-making in honey bees: how colonies choose among hectar sources. *Behav. Ecol. Sociobiol. 28*, 277-290.

Sharma, V., Pattnaik, S. S. ve Garg, T. (2012). A review of bacterial foraging optimization and its applications. *National Conference on Future Aspects of Artificial intelligence in Industrial Automation.*

Shi, Y. J., Meng, F. W. ve Shen, G. J. (2012). A modified artificial bee colony algorithm for vehicle routing problems with time windows. *Information Technology Journal 11* (10), 1490-1495. http://dx.doi.org/10.3923/itj.2012.1490.1495

Szeto, W. Y., Yongzhong, W. ve Sin, C. Ho. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research 215*(1), 126-135. https://doi.org/10.1016/j.ejor.2011.06.006

Şahin, Y. (2014). *Depo operasyonları ve sipariş dağıtım faaliyetlerinin sezgisel yöntemler kullanarak eş zamanlı optimizasyonu* (Yayın Nu. 369156) [Doktora tezi, Süleyman Demirel Üniversitesi]. http://tez.sdu.edu.tr/Tezler/TS01558.pdf

Şahin, Y. ve Eroğlu, A. (2014). Kapasite kısıtlı araç rotalama problemi için metasezgisel yöntemler: Bilimsel yazın taraması. *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi 19*(4), 337-355.

Taha, A., Hachimi, M. ve Moudden, A. (2017). A discrete bat algorithm for the vehicle routing problem with time windows. *International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*. http://dx.doi.org/10.1109/LOGISTIQUA.2017.7962875

Tan, K. C., Chew, Y. H. ve Lee, L. H. (2006). A hybrid multi-objective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications 34*, 115–151. http://dx.doi.org/10.1007/s10589-005-3070-3

Tan, K. C., Lee, L. H., Zhu, Q. L. ve Ou, K. (2001). Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering 15*(3), 281-295. https://doi.org/10.1016/S0954-1810(01)00005-X

Tan, L., Lin, F. ve Wang, H. (2015). Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows. *Neurocomputing 151*(3), 1208–1215. https://doi.org/10.1016/j.neucom.2014.03.082

Tan, X., Luo, X., Chen, W. N. ve Zhang, J. (2005). Ant colony system for optimizing vehicle routing problem with time windows. *International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)*. https://doi.org/10.1109/CIMCA.2005.1631470

Taner, F., Galic, A. ve Caric, T. (2012). Solving practical vehicle routing problem with time windows using metaheuristic algorithms. *Promet – Traffic&Transportation 24*(4), 343-351. https://doi.org/10.7307/ptt.v24i4.443

Taş, D., Jabali, O. ve Woensel, T. V. (2014). A vehicle routing problem with flexible time windows. *Computers &Operations Research 52*, 39–54. https://doi.org/10.1016/j.cor.2014.07.005

Tepic, J., Tanackov, I. ve Stojic, G. (2011). Ancient logistics - Historical timeline and etymology. *Tehni ki vjesnik 18*(3), 379-384.

Tezer, T. (2009). *Toplama ve dağıtım zaman pencereli araç rotalama problemi için kesin çözüm yaklaşımı ve örnek uygulamalar* (Yayın Nu. 245506) [Yüksek Lisans tezi, Balıkesir Üniversitesi].

Tom Altman. (t.y.). Computational Complexity CSC 5802. Erişim tarihi: 22.07.2022, http://cse.ucdenver.edu/~cscialtman/complexity/Report.pdf

Toro, O., Eliana, M., Escobar, Z., Antonia, H., Granada E. ve Azul, L. (2016). Literature review on the vehicle routing problem in the green transportation context. *revista.luna.azúl 42*, 362-387. https://doi.org/10.17151/luaz.2016.42.21

Tseng, Y. (2004). *The role of transportation in logistics*. [Yayınlanmamış Yüksek Lisans Tezi]. University of South Australia, School of Natural and Built Environments Transport Systems Centre.

Tseng, Y., Yue, W. L. ve Taylor, M. A. P. (2005). The role of transportation in logistics chain. *Proceedings of the Eastern Asia Society for Transportation Studies 5*, 1657 – 1672.

Tuntitippawan, N. ve Asawarungsaengkul, K. (2018). A memory integrated artificial bee colony algorithm with local search for vehicle routing problem with backhauls and time windows. *KMUTNB Int J Sci Technol 11*(2), 85-92. http://dx.doi.org/10.14416/j.ijast.2018.03.001

Verma, G. ve Verma, V. (2012). Role and applications of genetic algorithm in data mining. *International Journal of Computer Applications (0975 – 888) 48*(17).

Wang, C., Zhou, S., Gao, Y. ve Liu, C. (2018). A self-adaptive bat algorithm for the truck and trailer routing problem. *Engineering Computations 35*(2).· http://dx.doi.org/10.1108/EC-11-2016-0408

Wang, D. ve Zhou, H. (2021). A two-echelon electric vehicle routing problem with time windows and battery swapping stations. *Applied Sciences 11*(22), 10779. https://doi.org/10.3390/app112210779

Wang, D., Tan, D. ve Liu, L. (2018). Particle swarm optimization algorithm: An overview. *Soft Comput 22*, 387–408. https://doi.org/10.1007/s00500-016-2474-6

Woch, M. ve Lebkowski, P. (2009). Sequential simulated annealing for the vehicle routing problem with time windows. *Decision Making in Manufacturing and Services 3*(1), 87–100. https://doi.org/10.7494/dmms.2009.3.2.87

Worawattawechai, T. (2017). An artificial bee colony algorithm for the vehicle routing problem with backhauls and time windows. *Songklanakarin Journal of Science and Technology*.

Yadav, P. K. ve Prajapati, N. L. (2012). An overview of genetic algorithm and modeling. *International Journal of Scientific and Research Publications* 2(9).

Yang, X. S. ve He, X. (2013). Firefly algorithm: Recent advances and applications. *International Journal of Swarm Intelligence 1*(1). http://dx.doi.org/10.1504/IJSI.2013.055801

Yang, X., (2013). A review of distribution related problems in logistics and supply chain research. *International Journal of Supply Chain Management 2*(4).

Yao, B., Yan, Q., Zhang, M. ve Yang, Y. (2017). Improved artificial bee colony algorithm for vehicle routing problem with time windows. *PLOS ONE 12*(9): e0181275. https://doi.org/10.1371/journal.pone.0181275

Yassen, E. T., Ayob, M., Nazri, M. Z. A. ve Sabar, N. R. (2015). Meta-harmony search algorithm for the vehicle routing problem with time windows. *Information Sciences 325*, 140-158. https://doi.org/10.1016/j.ins.2015.07.009

Yassen, E. T., Ayob, M., Nazri, M. Z. A. ve Sabar, N. R. (2017). An adaptive hybrid algorithm for vehicle routing problems with time windows. *Computers & Industrial Engineering 113*, 382-391. https://doi.org/10.1016/j.cie.2017.09.034

Yesodha, R. ve Amudha, T. (2019, February 4-6). *An improved firefly algorithm for capacitated vehicle routing optimization* [Conference presentation]. Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates. https://doi.org/10.1109/AICAI.2019.8701269

Yeun, L. C., Ismail, W. R., Omar, K. ve Zirour, M. (2008). Vehicle routing problem: Models and solutions. *JQMA 4*(1), 205-218.

Yodwangjai, S. ve Malampong, K. (2022). An improved whale optimization algorithm for vehicle routing problem with time windows. *The Journal of Industrial Technology 18*(1). http://dx.doi.org/10.14416/j.ind.tech.2022.04.001

Yu, B. ve Yang, Z. Z. (2011). An ant colony optimization model: The period vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review 47*(2), 166–181. http://dx.doi.org/10.1016/j.tre.2010.09.010

Yu, B., Yang, Z. Z. ve Yao, B.Z. (2011). A hybrid algorithm for vehicle routing problem with time windows. *Expert Systems with Applications 38*(1), 435–441. http://dx.doi.org/10.1016/j.eswa.2010.06.082

Yu, S., Tai, C., Liu, Y. ve Gao, L. (2016). An improved artificial bee colony algorithm for vehicle routing problem with time windows: *A real case in Dalian. Advances in Mechanical Engineering 8*(8) 1–9. http://dx.doi.org/10.1177/1687814016665298

Zhikharevich, V. V., Matsiuk, N. A. ve Ostapov, S. E. (2016). Solving the routing problem by ant colony optimization algorithms. *International Journal of Computing 15*(2), 84-91.

İNTERNET KAYNAKLARI

http://www.bernabe.dorronsoro.es/vrp/index.html?/results/resultsSolom.htm

https://www.teknikturk.com.tr/TR/bilgi-merkezi/4101/karinca-ile-mucadele-nasil-yapilir

https://www.anzerbali.com/aricilikta-kafkas-ari-irki/

https://www.anzerbalirize.com/

https://www.aricilik.com.tr/bal-arisi-vucudunun-ic-yapisi/

https://media.istockphoto.com/illustrations/fly-firefly-illustration-id844831190?k=20&m=844831190&s=612x612&w=0&h=-b9adB4wlHd0GICc6psSepuyY6yILEdtuQx9SeJDvZU=