T. C.

İstanbul Üniversitesi

Sosyal Bilimler Enstitüsü

İşletme Anabilim Dalı

Sayısal Yöntemler Bilim Dalı

DOKTORA TEZİ

DİNAMİK ARAÇ ROTALAMA PROBLEMİNE PARÇACIK SÜRÜ OPTİMİZASYONU ALGORİTMASI ÇÖZÜM ÖNERİSİ

YONCA ERDEM DEMİRTAŞ

2502110410

TEZ DANIŞMANI

PROF.DR. ERHAN ÖZDEMİR

İSTANBUL – 2015

T.C.
İSTANBUL ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

DOKTORA
TEZ ONAYI

ÖĞRENCİNİN;

Adı ve Soyadı	:	YONCA ERDEM DEMİRTAŞ	**Numarası**	:	2502110410
Anabilim Dalı / Anasanat Dalı / Programı	:	SAYISAL YÖNTEMLER	**Danışmanı**	:	PROF.DR ERHAN ÖZDEMİR
Tez Savunma Tarihi	:	28.12.2015	**Saati**	:	11:00
Tez Başlığı	:	**DİNAMİK ARAÇ ROTALAMA PROBLEMİNE PARÇACIK SÜRÜ OPTİMİZASYONU ALGORİTMASI ÇÖZÜM ÖNERİSİ**			

TEZ SAVUNMA SINAVI, İÜ Lisansüstü Eğitim-Öğretim Yönetmeliği'nin **50. Maddesi** uyarınca yapılmış, sorulan sorulara alınan cevaplar sonunda adayın tezinin **KABULÜNE** OYBİRLİĞİ / ~~OYÇOKLUĞUYLA~~ karar verilmiştir.

JÜRİ ÜYESİ	İMZA	KANAATİ (KABUL / RED / DÜZELTME)
1-PROF.DR ERHAN ÖZDEMİR		Kabul
2- PROF.DR İBRAHİM DOĞAN		Kabul
3- PROF.DR MEHPARE TİMOR		Kabul
4- PROF.DR ERGÜN EROĞLU		Kabul
5- DOÇ.DR ERDAL DİNÇER		Kabul

YEDEK JÜRİ ÜYESİ	İMZA	KANAATİ (KABUL / RED / DÜZELTME)
1- PROF.DR TUNCAY CAN		
2-DOÇ.DR TİMUR KESKİNTÜRK		

DİNAMİK ARAÇ ROTALAMA PROBLEMİNE PARÇACIK SÜRÜ OPTİMİZASYONU ALGORİTMASI ÇÖZÜM ÖNERİSİ

YONCA ERDEM DEMİRTAŞ

ÖZ

Araç Rotalama Problemi (ARP) üzerinde çok uzun zamandır çalışılan bir problemdir. Her ne kadar ARP iyi bilinen statik bir problem olsa da gerçek hayatta benzeri problemler dinamik bir şekilde değişmektedir. Bu tip problemlere Dinamik ARP (DARP) denilmektedir.

Bir ARP çözümünde tüm problem girdileri önceden bilinir ve problem boyunca değişmezler. Diğer taraftan DARP'de problem girdilerinin başlangıçta tamamı veya bir kısmı bilinmez ya da planlama esnasında ortaya çıkabilir veya değişebilirler. Bu iki önemli karakteristikten dolayı DARP, ARP'ye oranla daha zor bir problem olarak bilinmektedir. Tez çalışmasında, DARP incelenmiş ve Parçacık Sürü Optimizasyonu (PSO) yöntemi probleme çözüm olarak önerilmiştir. Bilinen test problemleri önerilen yöntemle çözülmüş ve sonuçlar literatürde bilinen önceki yöntemlerle karşılaştırılmıştır.

Çalışmada elde edilen en iyi ve ortalama sonuçlar literatürde elde edilenlerle karşılaştırılmış; önerilen PSO algoritmasının sekiz problemde bilinen en iyi sonucu verdiği görülmüştür. Bu problemler genel olarak test problemleri içerisindeki büyük sayılabilecek problemler olduğu gözlenmiştir.

Gelecek çalışmalarda parametre optimizasyonu düşünülebilir. Sezgisel yöntemlerin çözüme katkıları ölçülmelidir. Ayrıca çoklu popülasyon yöntemleri de düşünülebilir.

Anahtar Kelimeler: Dinamik Araç Rotalama Problemi, Meta sezgisel Algoritmalar, Dinamik Optimizasyon, Parçacık Sürü Optimizasyonu, Yerel Arama Teknikleri.

A PARTICLE SWARM OPTIMIZATION ALGORITHM FOR DYNAMIC VEHICLE ROUTING PROBLEM

YONCA ERDEM DEMİRTAŞ

ABSTRACT

Vehicle Routing Problem (VRP) has been studied for many years. VRP is a well-known optimization problem. It is basically a static problem, although many VRP problems change over time in the real world. This kind of problems are called as Dynamic VRP (DVRP) in the literature.

In the VRP, all information about the problem is known at the beginning of the solution procedure. On the other hand, in most cases the real world information can change or appear after the solution process begins. These characteristics make the DVRP harder to solve than static VRP.

The DVRP is examined and a Particle Swarm Optimization algorithm is proposed. The most known benchmarks are solved with the proposed algorithm and the results are compared with the previously employed methods in the literature.

In this study, the obtained best and average results are compared with those found by the algorithms previously proposed in the literature. The PSO algorithm finds the best known results for 8 problem instances. These problems are mainly big problems in benchmarks.

For future research, parameter optimization can be considered and contributions of heuristic algorithms can be examined. Multi-population approaches can be addressed as well.

Key Words: Dynamic Vehicle Routing Problem, Metaheuristics, Dynamic Optimization, Particle Swarm Optimization, Local Search Techniques.

ÖNSÖZ

Gelişen teknoloji ve artan rekabet ortamında dağıtım ve taşımacılık sektöründe hızlı olmak önem kazanmaktadır. Tedarik ve dağıtım şirketlerini sektörlerinde bir adım öne çıkaracak unsur müşterilerinin taleplerine olabildiğince erken geri dönüşlerde bulunmalarıdır. Bu nedenle rotalama yaparken zaman faktörünün dikkate alınması gerekmektedir. Yaklaşık elli yıldır araştırmacıların ilgisini çeken geleneksel Araç Rotalama Problemine zaman faktörü de eklenerek geliştirilen Dinamik Araç Rotalama Problemi bu noktada gerçek hayatı daha iyi temsil etmektedir. Çalışmada Dinamik Araç Rotalama Problemi ele alınarak çözüm önerisinde bulunulmuştur.

Tez çalışması beş bölümden oluşmaktadır. Birinci bölümde, tezin kapsamı ve literatüre katkısından bahsedilmiştir. İkinci bölümde, Dinamik Araç Rotalama Problemlerinin tanımı verilerek hangi özelliklerine göre sınıflandırıldıklarına değinilerek literatürde yer alan çalışmalar özetlenmiştir. Üçüncü bölümde, Dinamik Araç Rotalama Problemleri için daha önceden önerilen çözüm yöntemleri detaylı bir şekilde verildikten sonra çözüm yöntemlerini değerlendirmek adına önerilen performans ölçülerinden bahsedilmiştir. Dördüncü bölümde, Dinamik Araç Rotalama Probleminin çözümü için Parçacık Sürü Optimizasyonu algoritması anlatılarak önerilen çözüm tasarımı detaylı bir şekilde anlatılmıştır. Beşinci ve son bölümde, önerilen çözüm yöntemi literatürde çalışılmış Dinamik Araç Rotalama test problemlerinin çözümünde kullanılmıştır. Elde edilen sonuçlar aynı test problemleri ile çalışan diğer akademik çalışmaların sonuçları ile karşılaştırılmıştır. Son olarak geliştirilen algoritma bir gerçek hayat problemine uygulanarak var olan rotalara nazaran maliyetler açısından daha uygun olan rotalar işletmeye sunulmuştur. Algoritma Java programlama dilinde kodlanmıştır.

Tez çalışması boyunca değerli yardım ve katkılarını esirgemeyerek bana her zaman yol gösterici olan tez danışmanım Sayın Prof. Dr. Erhan Özdemir Hocama çok teşekkür ederim. Çalışmanın gerçek hayat verisi elde etme süresince benden desteğini esirgemeyerek bu süreçte somut katkıları olan Sayın Prof. Dr. Eyüp Çetin Hocama çok teşekkür ederim. Her zaman benim yanımda olan ve birçok konuda fikirlerine başvurduğum sevgili çalışma arkadaşların Dr. Neslihan ve Dr. Sinem'e teşekkürü bir borç bilirim.

Tezimin ortaya çıkma süresince çok büyük destek ve yardımları olmasının dışında her konuda yol arkadaşım olan sevgili eşim Umut Demirtaş'a ne kadar teşekkür etsem azdır. Tüm hayatım boyunca her türlü desteğini esirgemeyerek benim arkamda olan canım annem Gülten ve babam Zafer Erdem başta olmak üzere tüm aileme sonsuz teşekkürlerimi sunarım.

Ayrıca çalışma kapsamında sağladığı maddi destek ile bilgisayar ve kitap desteği sebebiyle İstanbul Üniversitesi Bilimsel Araştırma Projeleri Birimine Teşekkürü borç bilirim.

Yonca ERDEM DEMİRTAŞ

Aralık,2015

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

KISALTMALAR LİSTESİ

ARP	Araç Rotalama Problemi
DARP	Dinamik Araç Rotalama Problemi
PSO	Parçacık Sürü Optimizasyonu
GA	Genetik Algoritma
KKA	Karınca Kolonisi Algoritması
GSP	Gezgin Satıcı Problemi
DTRP	Dinamik Tamirci Rotalama Problemi

GİRİŞ

Günümüz gerçek hayat problemlerinin birçoğu zaman faktörünü göze aldığımız müddetçe geçerlidir. Örneğin; çizelgeleme, araç rotalama, portföy optimizasyonu problemleri genellikle dinamiktir, yani optimum çözüm veya çözüm uzayı zamana bağlı değişebilmektedir.

Literatürde yer almış çalışmalar genellikle statik optimizasyon problemleri üzerine yoğunlaşmıştır. Statik versiyonları dahi literatürde NP-Zor sınıfına dahil olan bu problemlerin dinamik halleri günümüz araştırmacıları için yeni ve ilgi çekici bir alandır. Bununla birlikte dinamik problemin statik halini de kapsadığı ve statik haline göre daha zor olduğu ispatlanmıştır.

NP-Zor sınıfında bulunan problemlere, analitik yöntemlerle kabul edilebilir bir zaman içerisinde kaliteli sonuçların üretilmesi pek mümkün gözükmemektedir. Bu gerçek meta sezgisel yöntemleri ön plana çıkarmaktadır. Araç Rotalama Problemi (ARP) NP-Zor sınıfına dahil bir problem olduğundan meta sezgisel çözüm önerileri oldukça kullanılmaktadır. Dolayısıyla Dinamik ARP (DARP) için de analitik yöntemler etkisiz kalmakta ve meta sezgisel yöntemler önerilmektedir.

Parçacık Sürü Optimizasyonu (PSO) NP-Zor sınıfındaki problemlere çözüm önermek için kullanılan meta sezgisellerden biridir. Özellikle ARP gibi lojistik problemlerine yapısı gereği uyum sağladığı çeşitli yayınlarda vurgulanmış ve etkinliği ortaya konulan sonuçlarla kanıtlanmıştır.

BİRİNCİ BÖLÜM

1 TEZİN KAPSAMI VE LİTERATÜRE KATKISI

Bu tez çalışmasında dinamik optimizasyon problemlerinden "Dinamik Araç Rotalama Problemleri" (DARP) ile ilgilenilmiştir. Araç rotalama problemi, merkezi depodan yola çıkan bir veya birden fazla aracın dağıtım ve /veya toplama yapacağı konumlara maliyeti en küçükleyecek şekilde yönlendirilmesi şeklinde özetlenebilir. DARP'de yer alan "dinamik"lik yaklaşımı karar vericiler için, araç rotaları ve çizelgelenmesinde ihtiyaç duyulan bilgilerin dinamik olarak açığa çıkmasını göstermektedir. Yolların durumu, müşterilerin özellikleri ve dinamik ortamın çözüm düzenlenerek araç rotalama problemleri çeşitlendirilmektedir.

Araç Rotalama Problemleri sınıflandırma açısından literatürde kombinatoryal optimizasyon problemleri içerisinde gösterilmektedirler. Kombinatoryal optimizasyon problemlerinin karmaşıklığı ve problem boyutu genişlediğinde kesin çözüm yöntemlerinin yetersiz kalması sebebiyle, sezgisel optimizasyon teknikleri çözüm için kullanılan etkin yöntemlerdir. Sezgisel yöntemler optimum çözümü bulmayı garanti etmezler ancak kısıtları olabildiğince sağlayan iyi bir çözüm önerisinde bulunurlar. Öte yandan tüm çözüm uzayını taramak esasıyla çözüm aramaya başvurulması ile geçerli bir çözüm bulmak neredeyse imkansızdır.

Kombinatoryal optimizasyon problemleri ve sezgisel çözüm tekniklerine yönelik literatür oldukça hızlı gelişmektedir. Yapılan literatür araştırmasına göre, dinamik optimizasyon problemlerinin çözümünde kullanılan yöntemlerden bazıları aşağıdaki gibi sıralanabilir:

- Karınca Kolonisi Algoritması
- Evrimsel Algoritmalar
- Evrimsel Programlama
- Genetik Algoritmalar
- Memetik Algoritmalar
- Bağışıklık Sistemi Tabanlı Algoritmalar
- Yapay Sinir Ağları

- Parçacık Sürü Optimizasyonu

Dinamik optimizasyon problemlerine çözüm ararken, uzun süre yerel minimum veya yerel maksimuma takılmamak önemlidir. Bunun yanında çözüm uzayında gezinirken zaman ve çözüm kalitesi bakımından mümkün olan en iyi yolda ilerlemek esastır. Bu ilerlemeyi takip edebilmek için genelde bir grup çözüm ailesi ile uzayı taramak daha iyi sonuç vermektedir. Literatürde bu amaca yönelik olarak popülasyon tabanlı algoritmalar kullanılmaktadır ve iyi sonuçlar elde edilmektedir. Bu amaca yönelik olarak evrimsel algoritmalar, parçacık sürü optimizasyonu, karınca kolonisi algoritmaları, genetik algoritmalar iyi birer çözüm yöntemi olabilmektedirler.

Bu tez çalışmasının amacı literatürde çokça çalışılmış olan Araç Rotalama Problemlerinin gerçek hayat problemine daha yakın ve daha zor bir versiyonu olan Dinamik Araç Rotalama Probleminin çözüm yöntemlerini incelemek ve bu yöntemlere alternatif olarak literatürdeki diğer meta sezgisel yöntemlerden uygun olan bir tanesini önermektir.

Bu çalışmanın kavramsal içeriğini, gelişmekte olan ulaştırma ve lojistik sektörünün önemli problemlerinden biri olan dinamik araç rotalama probleminin meta sezgisel optimizasyon teknikleri ile çözümü teşkil etmektedir.

Bu tez çalışmasında ilk aşamada Dinamik Araç Rotalama Problemi ve türlerinin tanımı yapılmakta, sezgisel ve meta sezgisel yöntemler ele alınmakta ve konuyla ilgili yerli ve yabancı makaleler, kitaplar, tez çalışmaları, raporlar, incelenmektedir. Önerilen Parçacık Sürü Optimizasyonu algoritması Java programlama dili ile kodlanarak literatürde bilinen bazı dinamik araç rotalama test problemlerine uygulanmaktadır. Son olarak ise seçilen bir bölgeye ait gerçek hayat verileri kullanılarak meta sezgisel yöntemlerle çözüm önerisinde bulunulmaktadır.

İKİNCİ BÖLÜM

2 DİNAMİK ARAÇ ROTALAMA PROBLEMLERİ

2.1 Giriş

Günümüzde teknoloji birçok alanda hayatımızı kolaylaştıracak seviyeye gelmiştir. Bilgi transferi ve iletişim teknolojilerindeki gelişmeler sayesinde anlık haberleşme ve eş anlı operasyonlar yapabilmek mümkün olmaktadır. Bu açıdan bakıldığında; lojistik sektöründe kullanılan araçlarda coğrafi bilgi sistemleri (GIS) ve küresel konumlama sistemleri (GPS) kullanıldığında araçları gerçek zamanlı yönlendirmek ve yönetmek kolaylaşmaktadır.

Taşımacılık ve dağıtım sektöründe araç rotalamalar halen yaygın olarak statik şekilde yapılmaktadır. Yani dağıtım ve/veya toplama yapılacak hizmet/müşteri/ürün bileşenleri planlama periyodundan önce bilinmekte ve araçların rotaları önceden hesaplanmaktadır. Ancak araç bozulması, şoför rahatsızlığı ve/veya yol çalışması gibi özel bir durumla karşılaşıldığında planlanan rotaların değiştirilmesi veya yeni rota eklenmesi söz konusu olabilmektedir.

Statik bir şekilde yapılan planlamalarda istenildiği kadar esnek olunamamaktadır ve yapılacak her hangi bir değişikliğin sisteme adaptasyonu güçtür. Günümüzde piyasalardaki rekabet ortamı göz önüne alındığında esnek ve değişikliğe çabuk adapte olabilen sistemler bir adım öne geçmektedir.

Araç rotalama problemi gerçek hayat uygulaması çok yaygın olan bir kombinatoryal optimizasyon problemidir. Literatüre kazandırıldığı 1959 (Dantzig ve Ramser 1959) tarihinden itibaren geniş bir kitle tarafından üzerinde çalışılan bir problemdir. Problemin zaman kavramını dâhil eden ve daha karmaşık bir yapıya sahip olan dinamik versiyonu Dinamik Araç Rotalama problemidir. Bahsedilen teknolojik ilerlemeler yardımıyla hem akademik alanda hem de reel sektör alanında çokça uygulama alanı bulunmaktadır.

Bu bölümde dinamik araç rotalama probleminin tanımı ve çeşitlerinden bahsedilecektir.

2.2 Problem Tanımı

Kombinatoryal optimizasyon problemleri içerisinde yer alan Gezgin Satıcı Problemi 1800 lü yıllardan beri üzerinde çalışılan önemli problemlerden biridir. Gezgin Satıcı Problemi (GSP) aralarındaki uzaklıkları bilinen *n* tane şehrin her birinden yalnız bir kez geçen ve başlanılan noktada tamamlanan, en kısa (en az maliyetli) turun bulunmasını hedefleyen bir rotalama problemidir.

Araç rotalama problemi ise Gezgin satıcı probleminin daha genel hali olarak düşünülebilir. Burada *m* tane aracın rotalarının belirlenmesi söz konusudur, her bir aracın rotası depodan başlayan ve müşterilerin bulunduğu şehirlere ait bir alt kümeyi gezerek depoda son bulan bir turdan oluşmaktadır. Her bir müşteri yalnızca bir kez ziyaret edilmelidir ve bir rotaya ait müşteri taleplerinin toplamı araç kapasitesini aşmamalıdır. ARP de amaç tur maliyetini en küçüklemektir. Hesaplama karmaşıklığı açısından ARP NP-zor sınıfına ait bir problemdir (Cormen, Leiserson, Rivest, ve Stein, 2009: 1086).

Dinamik araç rotalama problemi (DARP)'de yer alan "dinamik"lik yaklaşımı karar vericiler için, araç rotaları ve çizelgelenmesinde ihtiyaç duyulan bilgilerin zamana bağlı olarak açığa çıkmasını göstermektedir. Yolların durumu, müşterilerin özellikleri, dağıtım ve toplama işlemlerinin aynı anda bulunup bulunmaması gibi unsurlar dikkate alınarak problem türleri çeşitlendirilmektedir.

Geleneksel ARP ile karşılaştırıldığında DARP daha genelleştirilmiş bir problem sınıfı olarak karşımıza çıkmaktadır, ARP'nin ait olduğu küme DARP'nin bir alt kümesi olarak ele alınır. ARP'de probleme ait bilgiler planlama periyoduna başlamadan önce tamamıyla bilinmektedir ve planlama sonuna kadar değişmemektedir. Öte yandan DARP'de problem ile ilgili bilgilerin tamamı bilinmemekle beraber, planlama periyodu boyunca değişebilmesi de söz konusudur (A. Larsen 2000). Problemle ilgili bilgiler müşterilerin coğrafi konumları, müşteri sayısı, talep veya arz miktarları ve müşterilerin servis edilme süreleri olarak ele alınmaktadır.

Psaraftis (1988) dinamik araç rotalama problemi ile statik araç rotalama problemini karşılaştırmış ve aralarındaki farkları detaylı bir şekilde açıklamıştır. On iki maddede özetlenmiş olan bu farklar aşağıdaki şekilde sıralanabilir:

1. Zaman Boyutunun Gerekliliği: Geleneksel statik araç rotalama problemlerinde zaman faktörü hemen hemen hiç önemli değildir. Genellikle zaman kavramı ancak iki konum arasında gidilen mesafeyi ölçmek için kullanılır ve problem çözülürken dikkate alınmaz. Ancak çözümün yorumlanmasında ve diğer çözümlerle karşılaştırılması adına maliyet olarak turun ne kadar sürede tamamlandığını göstermek için kullanılır. Öte yandan dinamik araç rotalama problemlerinde, tamamlanma süresi ilgili bir kısıt olmasa dahi, en azından yeni bir müşteri ortaya çıktığı an araçların anlık konum bilgilerine ihtiyaç duyulmaktadır.

2. Açık Uçlu Problem Olma: Statik araç rotalama problemlerinde rotanın tamamlanma süresi ile ilgili alttan ve üstten sınırlamalar getirilebilmektedir, ancak dinamik araç rotalamada böyle bir durum söz konusu değildir. DARP de yeni bir müşteri ortaya çıkma ihtimaline karşın araçlar son konumlarında bekleyebilmektedir. Depoya hemen dönemedikleri için tamamlanan yollar bulunur, ancak tüm süreç tamamlandığında araçlar depoya dönerek tur oluştururlar.

3. Gelecek ile İlgili Bilginin Kesin Olmaması veya Bilinememesi: Statik araç rotalama problemlerinde planlama periyodundan önce probleme ait tüm girdiler bilinmektedir. Ancak gerçek hayatta her zaman geleceğe dair bilgiler önceden bilinememektedir. Bu yüzden gerçek hayatın temsili olacak şekilde dinamik araç rotalama problemlerinde, bir müşterinin hangi konumda ve ne zaman ortaya çıkacağı kesin olarak bilinememektedir. Bazı planlama durumlarında geçmiş veriler göz önünde tutularak olasılık dağılımlarına uygunluğu doğrultusunda tahmin edilebilirler. Yine de gerçekleşecek durumlar ile ilgili kesin bilgi yoktur.

4. Yakın Dönemdeki Olayların Önemi: Dinamik araç rotalamada yakın zamanda gerçekleşecek olaylar uzun vadede gerçekleşecek olanlara göre daha önemlidir. Çünkü zaman faktörü göz önüne alındığında uzun vadede gerçekleşecek olayların hemen dikkate alınmasına gerek yoktur, bir müddet bekletilebilirler; ancak yakın zamandaki olayların planlama sürecine dahil edilmelerinde öncelikleri

bulunmaktadır. Statik surumda ise böyle bir ayrım yoktur, tüm müşterilerin eşit ağırlıkta önemleri bulunmaktadır.

5. Bilgi Güncelleme Mekanizmasının Gerekliliği: Dinamik olarak yapılacak planlamalar esnasında beklenmedik durumlarla karşılaşıldığında ve/veya planlanan rotalarda yapılacak bir değişiklik olduğunda, bilgilerin güncellenmesi gerekmektedir. Örneğin bir aracın bozulması, bir müşterinin talebinden vaz geçmesi veya beklenmeyen hava koşulları neticesinde bir noktaya ulaşımın engellenmesi gibi durumlarda çözüm algoritması ve uygulanması süreci arasında bilgi transferi mümkün olmalıdır.

6. Yeniden Sıralama Ve Yeniden Atama Kararlarına İzin Verilmesi: Bilgi transferinin mümkün olduğu düşünüldükten sonra, elde edilen bilgi doğrultusunda önceden planlanmış rotalar üzerinde değişiklik yapılması gerekebilir. Örneğin yeni bir müşterinin ortaya çıkması halinde önceden optimal olarak planlanan rotanın, gidilen yolları en küçükleyecek şekilde tekrar hesap edilmesi gerekebilir ve araçlara yapılan atamanın güncellenmesi gerekebilmektedir. Bu sebeple henüz hizmet verilmeyen şehirlerin yeniden sıralanmasına ve yeniden atanmasına izin verecek şekilde çalışılmalıdır.

7. Hesaplamaların Hızlı Yapılabilmesi: Rotaları yeniden planlamak ve/veya araçlara atamaların gerçek zamanlı bir şekilde yapılabilmesi için hesaplamaların hızlı olması önemlidir. Statik bir şekilde yapılan planlamalarda, hesaplamalar çok önceden başlatılabileceğinden hızlı bir şekilde sonuçlandırma gerekliliği yoktur. Ancak dinamik ortamların hızlı bir şekilde değişebileceği göz önünde bulundurulduğundan sonuçların hızlı bir şekilde elde edilmesi önemlidir.

8. Erteleme Mekanizmalarının Gerekliliği: Talepler arasında öncelik sıralamalarının olmadığı durumlarda, konumu nedeniyle hizmet verilmesi maliyeti artıracak olan talepler belirsiz bir zamana ertelenebilirler. Eğer rotalama tek bir araç ile yapılıyorsa bu şekildeki noktalara en son hizmet vermek tercih edilebilir, birden fazla araç var ise kimi zaman sadece tek bir araç bu noktaya hizmet verebilir. Hem toplam gidilen yolu en küçüklemek, hem de hiçbir müşteriyi göz ardı etmemek gerekmektedir. Bunu sağlayabilmek için probleme belirli kısıtlar getirilmektedir,

örneğin zaman penceresi kısıtı getirilerek, belirli müşterilerin belirli bir zaman diliminde hizmet verilmiş olması sağlanır. Aynı zamanda müşterilerin ertelenmelerine ait maliyetler de amaç fonksiyonlarına eklenerek denge sağlanmış olur.

9. Amaç Fonksiyonundaki Farklılıklar: Tüm girdilerin önceden bilindiği statik araç rotalamada, amaç fonksiyonu genellikle toplam gidilen yolun en küçüklenmesi veya toplam hizmet süresinin en küçüklenmesi olarak belirlenmektedir. Ancak dinamik olarak yapılacak planlamalarda toplam tur süresini optimize etmek, problem açık uçlu olduğu durumlarda anlamsızlaşır. Yeni bir müşteri ortaya çıkması ihtimali düşünülerek, kapasitesinin tamamını kullanmamış araçlar en son konumlarında bekletilebilirler, araçlara depoya dönme talimatı genellikle iki koşuldan biri sağlandığında verilir. Bu koşullardan biri ya tüm müşterilere hizmet verilmesi ya da aracın kapasitesini doldurmuş olmasıdır. Ayrıca müşterileri en az süre ertelemek, bir aracın kapasitesini en verimli şekilde kullanmak gibi önceliklerin bulunduğu dinamik süreçlerde doğrusal olmayan amaç fonksiyonları da kullanılabilmektedir.

10. Zaman Kısıtlarındaki Farklılıklar: Dinamik araç rotalama probleminde erken hizmet verme zamanı veya geç hizmet verme zamanı statik araç rotalamaya göre daha esnektir. Tabii ki esnek olması bu koşulun göz ardı edilebileceği anlamına gelmemektedir. Planlayıcının bakış açısına göre; statik olarak düşünüldüğünde probleme dair yeni bir bilgi ortaya çıkmaması halinde, eldeki verilere göre rotalama yapılacaktır ve her hangi bir esneme payına gerek yoktur. Ancak dinamik olduğunda, o an için belirli bir müşteriye hizmet vermek çok karlı olmayabilir, bu sebeple müşterinin belirli bir süre ertelenmesi gerekeceğinden hizmet verme zamanları ile ilgili kısıtların da esnek olarak modellenmesi gerekmektedir.

11. Değişken Araç Filo Boyutuna Esneklik Düşüktür: Teorik olarak düşünüldüğünde, hizmet verilecek müşterilere planlama periyodu içerisinde yetersiz kalınacağı öngörüldüyse ya araç sayısında ya da araç kapasitesinde artışa gidilmektedir. Statik olarak yapılan rotalamalarda, çözüm süreci ve uygulama zamanları arasında mutlaka bir zaman dilimi bulunduğundan gerekli güncellemeler yapılabilmektedir. Ancak dinamik olarak yapılan planlamalarda böyle bir esneklik söz konusu değildir, anlık olarak araç bulabilmek veya değiştirebilmek mümkün

olamayacaktır. Her hangi bir sebeple hizmet verilemeyecek olan müşteriler ya ertelenebilir ya da eksik hizmet görebilirler.

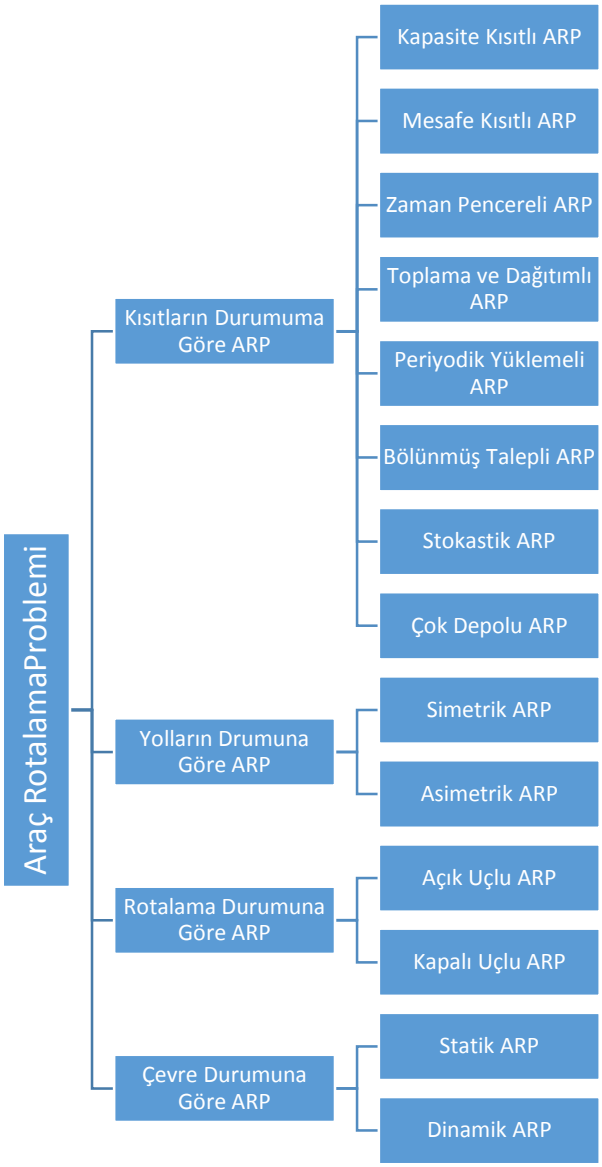
12. Kuyrukları Dikkate Almak Önemlidir: Açığa çıkan müşterilerin talepleri eldeki imkânlarla karşılanamadığı durumlarda sistem aşırı yüklü olmaktadır. Bu durumda çözüm algoritmaları geçersiz sonuçlar üretebilmektedir. Bu sebeple yeni müşterilerin sisteme aşırı bir yük getirmemesi için statik sistemlerdeki gibi atama yapmak yerine, bekleme hattı ve kuyruk teorilerinin dayandığı temellere göre hareket edilmesi gerekmektedir.

2.2.1 ARP Sınıflandırmaları

Araç rotalama probleminin matematiksel modelinden bahsetmeden önce problemin sınıflandırmalarından bahsetmek gerekmektedir. Daha önce de bahsedildiği gibi ARP 'de amaç şehirlere/müşterilere/konumlara ait alt rotaların verilen kısıtlar doğrultusunda ve maliyeti en küçükleyecek şekilde ziyaret edilmesi şeklinde özetlenebilir.

Literatürde farklı açılardan ele alınmış ARP sınıflandırmaları mevcuttur. Probleme ait ortamın durumu ve girdilerin türü açısından sınıflandırma yapan (H. N. Psaraftis 1988)' e göre; ortam dinamik veya statik olabilir ve problem girdileri deterministik veya stokastik olabilir. Örneğin probleme dair tüm bilgiler önceden biliniyorsa problem statik ve eğer herhangi bir olasılık dağılımı veya bir simülasyona bağlı olarak veriler elde ediliyorsa stokastik olarak sınıflandırılmaktadır.

Bu tez kapsamında dinamik araç rotalama problemleri ile ilgilenildiğinden araç rotalama problemi türlerinin kısaca tanımlarına ve matematiksel modellerine yer verilecektir. Bir sonraki bölümde ise dinamik araç rotalama problem türleri ele alınırken literatürde yer alan çalışmalar probleme getirilen çözüm önerileri dikkate alınarak detaylı bir şekilde incelenecektir. Bu bağlamda literatürde yer alan çalışmalar incelendiğinde ARP türleri Şekil 2-1'de gösterildiği gibi sınıflandırılmaktadır (Yeun, ve diğerleri 2008) (Ercan ve Gencer 2013):



Şekil 2-1 ARP Sınıflandırması

1. Kısıtların Durumuna Göre ARP

a. Kapasite kısıtlı ARP: Kapasite kısıtlı ARP en bilinen ve üzerinde en çok çalışılan ARP türlerinden biridir. Müşterilerin konumları, talep miktarları, iki müşteri arasındaki mesafenin ne kadar zamanda kat edileceği ve her bir müşterinin ne kadar süre hizmet göreceği önceden bilinmektedir. Bu bilgiler doğrultusunda tüm müşterilere hizmet vermek ve başlangıç noktası olan depoda son bulacak rotayı en küçük maliyetle bulmak amaçlanmaktadır.

Kapasite kısıtı bulunan ARP'lerde araçların belirli bir kapasiteleri bulunmaktadır. Her bir aracın ziyaret ettiği müşterilerin toplam talep miktarı araç kapasitesini aşmamalıdır. Kapasitelerin özelliğine göre de problemi iki alt sınıfa ayırmak mümkündür; araçların kapasitelerinin homojen olduğu problem türlerinin yanı sıra heterojen bir filo da söz konusu olabilir.

b. Mesafe Kısıtlı ARP: ARP'nin mesafe kısıtı eklenmiş türü olan mesafe kısıtlı ARP'de her bir aracın gidebileceği toplam mesafe için bir üst sınır getirilmektedir. Bir aracın rotasının uzunluğu için getirilen bu kısıt her bir araç için eşit olabileceği gibi, araçlar için farklı limitler de belirlenebilir. Karar verici bu konuda sınırlamaları kendi belirlemektedir. Öte yandan mesafe kısıtı bir aracın yolculuk süresi olarak da yorumlanabilmektedir. Bozulabilecek ürün dağıtımı yapan bir aracın yolda geçireceği zaman için bir üst sınır getirilebilir. Bu durumda müşterilere verilen servis süreleri (s_i) de dikkate alınmaktadır.

c. Zaman pencereli ARP: Zaman pencereli araç rotalama probleminde, her bir müşteriye belirli zaman dilimi içerisinde hizmet vermek şeklinde bir kısıt bulunmaktadır. *i.* müşteri için $[a_i, b_i]$ şeklinde belirlenen zaman penceresine göre problem alt iki sınıfa ayrılmaktadır:

Sıkı zaman pencereli ARP: Bu tür zaman pencereli ARP de bir araç ilgili müşteriye belirlenmiş zaman aralığından önce varmış ise, hizmet vermek için beklemektedir. Eğer zaman aralığından geç gelmiş ise hizmet verememektedir, müşteri başka zaman dilimine ertelenememektedir.

Esnek Zaman Pencereli ARP: Esnek zaman pencereli türde ise müşteriye zaman aralığı dışında hizmet verilebilmektedir. Ancak böyle bir durumla karşılaşıldığında bir penaltı fonksiyonu kullanılarak bu durum cezalandırılmaktadır. Penaltı fonksiyonu genellikle maliyet türünden belirlenmektedir.

d. Toplama ve dağıtımlı ARP: Bu tür araç rotalama problemlerinde hem dağıtım hem de geri toplama söz konusudur. Kimi zaman aynı müşteri için hem dağıtım hem de toplama olduğu gibi, bir müşteri sadece dağıtım başka bir müşteri de sadece toplama hizmeti görebilmektedir. Toplama ve dağıtmalı ARP türlerinde iki alt

kısım bulunmaktadır. Tüm müşteriler için toplama ve dağıtım işlemi tek bir merkez üzerinden yapılabileceği gibi, bir müşteriden toplanan ürün bir başka müşteriye teslim edilebilir. Örneğin kan bankası tek bir depo üzerinden işlem gören bir toplama dağıtmalı arp örneği iken, kurye servisleri bir müşteriden başka bir müşteriye teslimat götüren problem türüne örnektir (Cordeau, Laporte ve Ropke 2008, 327-357).

Çözüm stratejileri olarak ele alındığında, önce dağıtım işlemleri yapılıp sonra toplama işlemleri yapılabildiği gibi eş zamanlı olarak da toplama ve dağıtım işlemlerinin yapıldığı çalışmalar mevcuttur.

e. Periyodik Yüklemeli ARP: Genellikle planlamalar belirli bir zaman dilimi örneğin 1 gün veya 1 hafta için yapılabilir. Daha uzun planlamalar aylık veya yıllık da olabilir ancak rotalama problemleri için bu kadar uzun süreler hesaplamaları karmaşık hale getirebilmektedir.

Bir planlama süresinin gün üzerinden hesaplandığını düşünelim; böyle bir yapıda eğer ardışık olarak birden fazla, *t* gün için planlama yapılıyor ise bu durum periyodik araç rotalama problemi olarak isimlendirilmektedir. Toplam *t* gün içerisinde her bir müşteri için servis edilme sıklığı (e_i) önceden belirli bir parametredir. Bunun yanında hizmet verilecek günlerin olası kombinasyonlarını içeren küme C_i yine karar verici tarafından önceden belirlenmelidir. Örneğin *i.* müşteri için $e_i = 2$ ve $C_i = \{(1,3), (2,4), (3,5)\}$ olsun; *i.* müşteri *t* periyottan oluşan planlama sürecinde toplam 2 ayrı günde hizmet görecektir ve bu günler 1 ve 3 veya 2 ve 4 ya da 3 ve 5 olabilecektir (Cordeau, Gendreau ve Laporte 1997).

f. Bölünmüş Talepli ARP: Klasik ARP de her bir müşterinin yalnız bir araç tarafından bir kere ziyaret edileceği kısıtı bulunmaktadır. Bölünmüş talepli ARP de ise bu kısıt kaldırılmıştır. Bir müşteriye birden fazla araç hizmet verebilmektedir. Yine her bir aracın belirli bir kapasitesi bulunmaktadır. Müşterilerin talep miktarlarının araç kapasitelerini aştığı durumlarda geçerli olabilen bu durum problemi daha zor bir hale getirmektedir.

Problemin en genel halinde araç sayısının limitsiz olduğu varsayımı bulunmaktadır. Ancak bu durum gerçekçi bulunmadığından Archetti ve Speranza tarafından (Archetti ve Speranza 2008) tüm müşterilere hizmet verecek araç sayısı için

bir üst sınır (m) getirilmesi önerilmiştir. $m = \sum_{i=1}^{n} \left\lceil \frac{d_i}{Q} \right\rceil$ olarak hesaplanabilmektedir, burada Q özdeş araçların her birinin kapasitesini, d_i ise müşterilerin talep miktarlarını göstermektedir. Örneğin klasik ARP için, 3 müşteriden oluşan bir sistem düşünelim, kapasite miktarı 4 olan araçlardan en az 3 tane gerekmektedir ki müşterilere hizmet verebilsin. Ancak talepler bölünebilir olarak düşünüldüğünde aynı kapasitede 3 araç ile hizmet tamamlanabilir.

g. Stokastik ARP: Geleneksel ARP problemi deterministiktir. Yani probleme dair tüm girdiler kesin bir şekilde bilinebilmektedir. Stokastik ARP de ise probleme dair girdilerinden bir veya birden fazlası rassal olarak ve/veya belirli bir olasılık dağılımına uygun olarak hesaplanmaktadır.

Stokastik ARP, girdilerinin hangilerinin stokastik olduğuna göre alt gruplara ayrılabilmektedir (Gendreau, Laporte ve Séguin 1996):

Stokastik Müşteriler: Hizmet verilecek müşterilerin kümesinin tam olarak belirli olmadığı problem türüdür. *i.* müşteri p_i olasılığı ile mevcut (1- p_i) olasılığı ile mevcut değildir.

Stokastik Talepler: Her bir müşterinin talepleri rassal olduğu problem türüdür.

Stokastik Maliyet Matrisi: İki müşteri arası mesafenin maliyeti (c_{ij}) hesaplanırken ya mesafe uzunluğu (d_{ij}) ya da ulaşım süresi (t_{ij}) kullanılmaktadır. Maliyet matrisin elemanları rassal ve/veya bir olasılık dağılımına uygun olarak hesaplandığı problem türüdür.

Stokastik ARP'de rassal veriler bulunduğundan tüm kısıtların gerçekleşecek şekilde optimal bir çözümü bulmak neredeyse imkansızdır. Bu durumda karar vericinin bazı kısıtların belirli olasılıklarla karşılanması koşulu koyması veya sağlanmayan bir kısıta penaltı fonksiyonu belirlemesi gerekebilir. Hangi problem girdisinin stokastik olarak belirlendiği burada çözüm stratejisi geliştirmede ayırt edici rol oynamaktadır.

h. Çok Depolu ARP: Bir şirketin müşterilerine hizmet verebileceği birden fazla deposu bulunabilir. Bu durumda geçerli olan çok depolu arp için müşterilere

hizmet verecek araçların konum olarak müşterilere yakın depolardan rotalarına başlayıp bitirmeleri önemlidir. Genellikle önce kümeleme algoritmaları kullanarak depoların yakınlarındaki müşteriler belirlenir, daha sonra her bir depo için klasik tek depolu araç rotalama problemi çözülür. Bu yaklaşımda kümeleme algoritmasının performansı rotalama probleminin etkinliğini büyük ölçüde etkilemektedir (Nagy ve Salhi 2005).

2. Yolların Durumuna Göre ARP: Bir konumdan diğerine giden yolların durumu mesafe ve trafik açısından değerlendirilebilir. Gerçek hayat problemleri göz önüne alındığında yolların durumuna göre araç rotalama problemleri iki alt sınıfa ayrılmaktadır:

a. Simetrik ARP: İki şehir arasındaki gidiş ve geliş maliyeti ve/veya uzaklığının eşit olduğu rotalama problem türüdür. Yani *i* şehrinden *j* şehrine gidiş maliyeti c_{ij} dönüş maliyeti c_{ji} 'ye eşit olduğu durumdur. Böyle bir problemde maliyetlerin oluşturduğu matris simetrik matris olarak kullanılabileceği gibi üst veya alt üçgensel matrisler de gösterim açısından tercih edilebilir.

b. Asimetrik ARP: İki şehir arasında yolların durumundan ötürü gidiş geliş mesafesi eşit olmadığı durumlardır. Gerçek hayat problemlerinde yolların tek yönlü olduğu göz önünde bulundurulduğunda asimetrik ARP gerçeği daha çok yansıtmaktadır. Böyle bir problemde maliyetlerin oluşturduğu matris simetrik değildir.

3. Rotalama Durumuna Göre ARP: Rotaların durumuna göre ARP iki alt gruba ayrılmaktadır:

a. Açık Uçlu ARP: Depodan başlayan rotaların depoda son bulma şartı olmayan ARP türlerine Açık Uçlu ARP ismi verilmektedir. Araçlar tüm müşterilerine servis verdikten sonra en son servis verdiği müşteri konumuna bekletileceği gibi başka bir konuma da yönlendirilebilirler. Son yönlendirildikleri konuma gidiş maliyeti problem sınırları dahilinde göz önüne alınmamaktadır (Li, Golden ve Wasil 2007).

Her bir aracın rotası müşteri kümesinin bir alt kümesi üzerinde Hamilton yolu[1] oluşturmaktadır. ARP'de Hamilton yolu bulmak problemine ek olarak kullanılan araç sayısının da en küçüklenmesi kısıtı eklenmiştir.

Açık uçlu ARP literatüre yaklaşık 25 yıl önce kazandırılmasına rağmen son yıllarda üzerinde yapılan çalışmaların sayısı artmaya başlamıştır. Evlere paket veya gazete dağıtımı yapan şirketlerin rotalama problemleri açık uçlu ARP 'lere örnek olarak verilebilir.

b. Kapalı Uçlu ARP: Geleneksel araç rotalama problemi türüdür. Tüm araçların rotalarına depodan başlayıp rotalarını depoda sonlandırma kısıtı bulunmaktadır. Her bir aracın rotası müşteri kümesinin bir alt kümesi üzerinde Hamilton turu[2] oluşturmaktadır. Genellikle ARP çalışmaları kapalı uçlu ARP 'ler üzerinde yoğunlaşmaktadır.

4. Çevre Durumuna Göre ARP: Planlama süreci boyunca çevre durumuna göre araç rotalama problemleri ikiye ayrılmaktadır:

a. Statik ARP: Statik ortam problemlerinde planlama ile ilgili tüm bilgiler rotalama algoritmasına başlamadan önce karar verici tarafından bilinmektedir. Rotaların belirlenmesi ve araçların rotalara atanması süreci sona erene kadar da bilgilerin değişmeyeceği varsayılmaktadır (A. Larsen 2000).

Araç rotalama problemleri literatüre kazandırıldığı zamandan günümüze kadar araştırmacıların ilgisini çeken Statik ARP NP-Zor problem sınıfına girmektedir. Bu nedenle problem boyutu büyüdükçe çözüm uzayı üstel olarak genişlemektedir ve kesin çözüm yöntemleriyle optimal çözümü bulmak neredeyse imkansızdır. Sezgisel ve meta sezgisel algoritmalar kullanılarak kabul edilebilir çözümler üretilmektedir (Cormen, ve diğerleri 2009).

b. Dinamik ARP: Rotalama problemine ait bilgilerin planlama sürecine başlamadan önce tamamının bilinmediği problem türüdür. Bilinen bilgilerin planlama esnasında değişebileceği gibi yeni bilgilerin de eklenmesi söz konusudur. Zamana

[1] Hamilton Yolu (Hamiltonian Path): Bir graftaki her düğümden (node) yalnızca 1 kere geçilmelidir.
[2] Hamilton turu (Hamiltonian cycle): Bir graftaki her düğümden (node) yalnızca 1 kere geçilmelidir ve tur başlangıç düğümünde son bulmalıdır.

bağlı olarak problem girdilerinin değişmesi ARP'ne dinamiklik özelliğini getirmektedir. Bu nedenle statik ARP ile karşılaştırıldığında bilinmeyen durumların etkisi ve dikkate alınması özelliği açısından daha karmaşık bir yapıya sahiptir.

Gerçek hayat problemleri açısından ele alındığında aslında problemlerin gerçek zamanlı çözülebilmeleri, günümüz teknolojik ilerlemeleri ışığında gerekli olmaktadır. Bu açıdan ele alındığında dinamik araç rotalama problemlerinin gerçek hayat uygulama alanlarının çokluğu son yıllarda yapılan akademik çalışmaların ilgi odağı olmuştur.

DARP'nin gerçek hayat uygulama alanlarını aşağıdaki şekilde örneklendirmek mümkündür (A. Larsen 2000), :

- Tedarik ve dağıtım şirketleri
- Kurye servisleri
- Gezgin tamirci problemi
- Kurtarma servisleri
- Taksi servisleri

şeklinde örneklendirilebilir.

2.2.2 Matematiksel Model

Bu tez kapsamında incelenen dinamik araç rotalama problemi, kapasite kısıtlı dinamik araç rotalama problemidir. Bu sebeple ARP türlerinden kapasite kısıtlı statik araç rotalama probleminin matematiksel modelinden bahsetmek ve daha sonra problemin dinamik versiyonuna geçiş yapılarak dinamizm değerlendirmesi yapılacaktır.

(Toth ve Vigo 2001) 'de önerilen matematiksel model araç sayısı da göz önüne alınarak düzenlenmiştir ve aşağıdaki şekilde ifade edilebilir: Araç rotalama problemi $G = (V, A)$ çizgesi üzerinde tanımlı olsun. Burada $V = \{0,1, \dots, n\}$; 0. düğüm depoyu göstermek üzere şehirlere/müşterilere ait düğümleri, $A = \{(i, j): i, j \in V, i \neq j\}$ ve düğümleri birbirine bağlayan doğru parçaları kümesini göstermektedir. Her bir düğüm tek bir araç tarafından ziyaret edilecektir. Problemde özdeş "Q" kapasiteye ait "m" adet aracın hizmet verdiği varsayımı bulunmaktadır.

Probleme ait çizge eğer simetrik ise $G = (V, E)$ şeklinde gösterilmektedir, burada iki şehrin arasındaki yolların temsil edildiği E kümesi, yönlendirilmiş doğru parçalarından (arklardan) oluşmaktadır. Bu durumda, *i*' den *j*'ye gitme maliyeti ile *j*'den *i*'ye gitme maliyeti eşittir dolayısıyla problem simetrik bir yapıya sahiptir.

Parametreler:

$V = \{0,1, \dots, n\}$: Düğüm kümesi,

$K = \{1,2, \dots, m\}$: Araç kümesi,

Q: Araç kapasitesi,

c_{ij}: i düğümünden j düğümüne gitme maliyeti,

d_i: i düğümünde yer alan müşterinin talep miktarı,

Karar değişkeni:

$$x_{ij}^k = \begin{cases} 1, & k\ aracı\ i\ düğümünden\ j\ düğümüne\ giderse \\ 0, & aksi\ taktirde \end{cases}$$

Amaç fonksiyonu:

$$Minimize \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} c_{ij} x_{ij}^k \qquad (2\text{-}1)$$

Kısıtlar:

$$\sum_{i=0}^{n} \sum_{k=1}^{m} x_{ij}^k = 1, \qquad j \in V \setminus \{0\}, \qquad (2\text{-}2)$$

$$\sum_{j=0}^{n} \sum_{k=1}^{m} x_{ij}^k = 1, \qquad i \in V \setminus \{0\}, \qquad (2\text{-}3)$$

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{i0}^k = m, \qquad (2\text{-}4)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x_{0j}^k = m, \qquad (2\text{-}5)$$

$$\sum_{i=0}^{n} x_{ih}^{k} = \sum_{j=0}^{n} x_{hj}^{k}, \qquad \forall h \in V, k \in K \tag{2-6}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} d_i x_{ij}^{k} \leq Q, \qquad k \in K \tag{2-7}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k} \leq |S| - 1, \qquad \forall S \subseteq V - \{1\}, S \neq \emptyset, k \in K \tag{2-8}$$

Amaç fonksiyonu toplam tur maliyetini minimize edecek şekilde modellenmiştir. (2-2) ve (2-3) numaralı kısıtlar depo dışındaki her bir müşterinin yalnızca bir kez ziyaret edilmesi gerektiğini göstermektedir. (2-4) ve (2-5) numaralı kısıtlar depoya araç sayısı kadar giriş ve çıkış olmasını kontrol etmektedir. Yani *m* adet alt rota oluşturulmaktadır. Kısıt (2-6) ise her bir düğüme giriş ve çıkış aynı araç tarafından yapılması gerektiğini söylemektedir. Kısıt (2-7) bir araca ait müşterilerin toplam talebinin araç kapasitesini aşmamasını kontrol etmektedir. Son olarak (2-8) numaralı kısıt bir araç için alt tur oluşmasını engellemektedir.

Tez çalışmamızda ilgilenilen araç rotalama problemi kapasite kısıtlı dinamik araç rotalama problemidir ve matematiksel model yapısı statik problemle aynı özelliklerdedir. Tabii ki dinamik olarak müşteri bilgileri değişmektedir, ancak her bir değişim kontrolünde eldeki bilgiler ışığında anlık statik problemler çözülmektedir.

2.2.3 Dinamizmin Derecesi

Dinamik olarak yapılan rotalamalarda, probleme dair bilgilerin tamamı sistem işleyişine başladıktan sonra ortaya çıkabilir. Ancak, bazı yapılarda problem girdilerinin bir kısmı sistem işleyişine başlamadan önce biliniyor olabilir. Örneğin kargo dağıtımı yapılan bir şirketin, bir önceki günden kalan hizmet verilmemiş müşterileri bulunabilir ve bunlara ek olarak yeni iş gününde ortaya çıkacak müşterilerini de dikkate alarak planlamalarının yapılması gerekebilir.

Rotalama probleminin dinamizminin ölçülmesi ve geliştirilecek algoritmaların bu yönde tasarlanması önemlidir. Literatürde problemin dinamikliğini ölçmek için bir takım metrikler önerilmiştir. Bunlar içerisinde ilk olarak (Lund, Madsen ve Rygaard.

1996) tarafından önerilen metrik en yalın olanıdır ve eşitlik (2-9) ile ifade edilmektedir.

$$dod = \frac{n_d}{n_s + n_d} \qquad (2\text{-}9)$$

Burada "*dod"* ile gösterilen metrik dinamizmin derecesini veya yüzdesini ölçmektedir ve İngilizce "*degree of dynamism*" cümlesinin baş harflerinden oluşmaktadır. n_d ve n_s sırasıyla, dinamik müşterilerin sayısını ve statik müşterilerin sayısını göstermektedir. Belirli bir planlama periyodu, [0,T], için dinamik müşterilerin sayısının toplam müşterilerin sayısına oranlanmasıyla bulunan dinamizmin derecesi [0,1] aralığında bir değer almaktadır. Dinamizmin derecesi 1'e ne kadar yakınsa sistem o derece dinamik ve karmaşık bir yapıya sahiptir. Örneğin 10 müşteriden 2 sinin dinamik olarak ortaya çıktığı bir durumda dinamizmin derecesi 0,2 veya %20 olarak hesaplanmaktadır.

(A. Larsen 2000) çalışmasında dinamizmin derecesini ölçmek için zaman kavramını da hesaplamalara dahil edilmesi gerektiğini önermektedir. Buna göre zamana bağlı olarak ortaya çıkan müşterilerin ortaya çıkma zamanlarını da formülasyonuna eklemiştir. Çünkü iki sistemin dinamizmini karşılaştırmak adına sadece dinamik ve statik müşterilerin sayısı yeterli olamamaktadır. Dinamik olarak ortaya çıkan müşterilerin planlama periyodunun hangi zaman dilimlerinde ortaya çıktığı da karar verme sürecini etkileyen bir unsur olmaktadır.

Zaman kavramını dikkate alan metriğin hesaplanmasına geçmeden önce pratikte nasıl değerlendirildiğinden bahsetmek gerekmektedir. Dinamik ve statik müşteri sayıları eşit olan iki senaryo ele alalım; birincisinde dinamik olan müşteriler planlama periyodunun hemen ilk dönemlerinde ortaya çıkmış olsun, ikincisinde ise planlama periyoduna uzun vadede dağılmış olarak ortaya çıksınlar. İkinci sistem karar verici açısından daha esnektir. Çünkü karar vericinin değişiklik ortaya çıktığında bir sonraki değişikliğe kadar geçen süresi daha uzundur. Ortaya çıkma zamanlarını dikkate alarak sistem dinamizmini ölçen, "*effective degree of dynamisim*" cümlesinin kısaltması olarak "*edod*" dinamizmin efektif derecesi olarak çevrilebilen metrik eşitlik (2-10) ile gösterildiği şekilde hesaplanmaktadır.

$$edod = \frac{\sum_{i=1}^{n_s+n_d}(\frac{t_i}{T})}{n_s + n_d} \qquad (2\text{-}10)$$

Burada $t_i \in [0, T]$, i. müşterinin ortaya çıkma zamanını göstermektedir. Statik müşteriler için $t_i = 0$ dır. Kolayca görülebileceği gibi $edod$ da [0,1] aralığında bir değer almaktadır. Tüm müşteriler planlama periyoduna başlamadan önce biliniyorsa $edod = 0$, eğer tüm müşteriler T zamanında (yani planlama periyodunun en sonunda) ortaya çıkıyor ise $edod = 1$ olmaktadır.

Araç rotalama probleminin zaman pencereli türü de bulunduğunu daha önce söylemiştik. Zaman penceresi de dikkate alınarak dinamizmin derecesini ölçmeyi öneren (A. Larsen 2000) aşağıdaki formülasyonu geliştirmiştir. Zaman pencereli araç rotalama probleminde her bir müşterinin hizmet görebileceği bir zaman dilimi mevcuttur, i müşterisine ait hizmet görebileceği zaman aralığı (penceresi) $[a_i, b_i]$ olsun, buna göre dinamizmin derecesi eşitlik (2-11) şeklinde hesaplanmaktadır.

$$edod_{tw} = \frac{1}{n_s + n_d} \sum_{i=1}^{n_s+n_d} [T - (b_i - t_i)]/T \qquad (2\text{-}11)$$

Yine kolayca görüleceği gibi $edod_{tw}$ [0,1] aralığında bir değer almaktadır. Eğer zaman penceresi yok ise, yani $a_i = t_i$ve $b_i = T$ ise $edod = edod_{tw}$ olmaktadır.

Sonuç olarak dinamizmin derecesinin ölçülebilmesi problemin ne derece karmaşık bir yapıya sahip olduğunun göstergesi olmaktadır. Bu nedenle rotalama problemlerini karşılaştırmak adına önemlidir.

Dinamizmin ölçülebilir olduğundan bahsettikten sonra, ölçülen değerlerin problemlerin kategorize edilmesinde kullanıldığından bahsetmek gerekmektedir. Dinamik araç rotalama problemleri, dinamizm metriğine göre üç farklı kategoride sınıflandırılmaktadır; bunlar zayıf dinamik, orta dinamik ve güçlü dinamik şeklindedir. Örneğin tedarik ve dağıtım şirketlerinde dağıtım yapılacak müşterilerin %80 ve daha fazlası planlamaya başlamadan önce bilindiği için bu tip şirketlerin dağıtım problemleri zayıf dinamik sınıfına girmektedir. Kargo ve posta servisleri orta dinamik olarak, acil servis ve taksi gibi birimlere ait rotalama problemleri ise güçlü dinamik kategorisinde yer almaktadır.

2.2.4 Amaç fonksiyonları

Statik ARP'de amaç fonksiyonu taşımacılık maliyetini en küçüklemektir. Ek olarak kat edilen yola bağlı olarak değişken maliyetler de dikkate alınmaktadır. Araç sayısı önceden belirlenmemiş ise kullanılan araç sayısının da maliyeti etkilediği göz önünde bulundurularak, araç sayısının da minimize edilmesi gerekmektedir.

Dinamik ARP'lerde dinamizme sebep olan parametrenin durumuna göre farklı öncelikler bulunmaktadır. Müşterilerin beklentileri veya firma yetkililerinin tercihlerine göre amaç fonksiyonları bir problemden diğerine değişebilmektedir. Dinamik ARP'de amaç fonksiyonları belirlenirken aşağıdaki maliyetler dikkate alınmaktadır:

Taşımacılık Maliyetleri: Taşımacılık sektöründe bir rotalamanın başarılı olarak değerlendirilmesi için öncelikle müşteriler arası en kısa yollardan gidilmesi gerekmektedir. Gidilen yol maliyet açısından en büyük ağırlığa sahip parametre olmaktadır. Bu nedenle DARP için amaç fonksiyonları belirlenirken kat edilen yolun minimizasyonu mutlaka dikkate alınmaktadır.

Hizmet Zamanları: Dinamik olarak yapılan rotalamalarda bir müşteriye ne zaman hizmet verildiği önemlidir. Çünkü müşterinin ortaya çıkma zamanı ile hizmet edildiği zaman arasındaki farkın fazla olması, müşteriyi gereğinden fazla bekletmek anlamına gelmektedir. Gerçek hayat problemlerinde, özellikle acil bir durum söz konusu ise bekleme süreleri büyük önem taşımaktadır. Bir müşterinin hizmet görmeden önceki bekleme süresinin minimize edilmesi amaç fonksiyonuna eklenmesi bu nedenle önem taşımaktadır.

2.3 Dinamik Araç Rotalama Problem Çeşitleri Literatür Araştırması

Araç rotalama problemi geniş bir uygulama alanına sahip olduğu için, problem kısıtlarına ve özelliklerine göre farklı isimlendirilmeleri mevcuttur. Bu bölümde dinamik araç rotalama problemine ait son 20 yılda yapılan çalışmalardan bahsedilecektir. Dinamik araç rotalama problemi için (Psaraftis, Wen ve Kontovas

2015), (Pillac, ve diğerleri 2013), (Bianchi 2000), (Cruz, González ve Pelta 2011) (Ghiani, ve diğerleri 2003) tarafından çalışılan geniş kapsamlı araştırma makaleleri incelenmiştir. Literatür araştırmaları sonucunda DARP alanında çeşitli dergi ve konferans kitaplarında basılan bilimsel yayınlar bu bölümde özetlenecektir.

Yapılan çalışmalar incelendiğinde, dinamik araç rotalama problemlerinin iki ana alt başlık altında toplanabileceği görülmüştür (Pillac, ve diğerleri 2013). Probleme dair girdilerin deterministik yani kesin olarak bilinen veya stokastik yani bir olasılık dağılımı veya bir tahmin yöntemi ile belirlendiği çalışmalar bulunmaktadır. Bu doğrultuda geçmiş çalışmalar bu bölümde sunulacaktır.

Deterministik:

(Tian, Song, ve diğerleri, Dynamic vehicle routing problem using hybrid ant system 2003): Tek araçlı dinamik araç rotalama problemi ele alınmıştır. Problem bu haliyle gezgin satıcı problemi olarak isimlendirilmektedir. Hibrit Karınca Kolonisi algoritması (KKA) önerilmektedir. Feromen matrisinin güncellenmesi ile dinamizmin takip edilmesi hedeflenmektedir. Yerel arama metotları kullanılarak KKA bulduğu sonuçlar iyileştirilmektedir.

(Taniguchi ve Shimamoto 2004): Dinamik araç rotalama probleminde, trafiğin dinamik simüle edildiği bir ortamdan trafik bilgileri gerçek zamanlı kullanılmıştır. Böylelikle seyahat süreleri dinamik olarak güncellenmektedir. Çalışmada, seyahat süreleri iki türlü dikkate alınarak iki farklı model kurulmuştur. İlk olarak planlama yapılacak güne dair seyahat süreleri geçmiş veri kullanılarak tahmin edilerek zaman pencereli araç rotalama problemi modellenmiştir. Tahmin verileri kullanılarak araç rotaları Genetik Algoritma ile belirlenmiştir. Daha sonra, gerçek zamanlı seyahat süreleri ile zaman pencereli araç rotalama problemi modellenmiş ve araç rotaları belirlenmiştir. Her iki model sonucunda toplam tur maliyetleri hesaplanarak karşılaştırılmıştır. Maliyetler açısından karşılaştırıldığında gerçek zamanlı veriler kullanılan dinamik araç rotalama probleminin maliyetinin daha düşük olduğu raporlanmıştır.

(Chitty ve Hernandez 2004): Dinamik kapasite kısıtlı araç rotalama problemine dinamik programlama ve karınca koloni algoritması ile melez bir çözüm tekniği önermişlerdir.

(Attanasio, ve diğerleri 2004): Zaman pencere kısıtı bulunan dinamik araç rotalama problemine Paralel Tabu Arama meta sezgiseli ile çözüm önerisi getirilmiştir. Ele alınan problemde bir konumdan başka bir konuma yolcu götürmek söz konusudur. Birden fazla kullanıcı, bir araç ile taşınmaktadır. Kullanıcılar bulundukları konumdan alınmak ve varmak istedikleri konuma varış zaman aralıklarını bildirmektedirler. Aralıklar dâhilinde kullanıcılar uygun araçlar ile taşınmaktadırlar.

(Fleischmann, Gnutzmann ve Sandvoß 2004): Bu çalışmada zaman pencereli dinamik araç rotalama problemi ile ilgilenilmiştir. Gerçek hayat problemi çözülmüştür ve ele alınan problemde trafik bilgisi de göz önünde bulundurulduğundan, iki konum arası seyahat süresi zamana bağlı olarak değişmektedir.

(Haghani ve Jung 2005): Çalışmada iki konum arası seyahat süresinin zamana göre değiştiği dinamik araç rotalama problemi ele alınmıştır. İki konum arasındaki seyahat zamanının değişmesindeki en büyük etken trafik yoğunluğu olarak düşünülebilir. Problemin matematiksel modeli önerilmiş ve çözüm için genetik algoritma yaklaşımı kullanılmıştır. Çalışmada rasgele üretilen test problemleri çözülmüştür. Önerilen algoritmanın performansını ölçmek için elde edilen sonuçlar kesin çözüm yöntemleri ile karşılaştırılmıştır.

(Pankratz 2005): Zaman pencereli dinamik toplama ve dağıtımlı araç rotalama problemine genetik algoritma ile çözüm önerisinde bulunulmuştur. İlk olarak bilinen problem verileri ile başlangıç popülasyonu üretilmiştir. Önerilen gruplama temelli genetik algoritma başlangıç popülasyonunu iyileştirmek üzere uygulanmıştır. Yeni bir talep ortaya çıkana kadar algoritma çalıştırılmıştır. Problem verileri değiştikçe popülasyon güncellenmektedir. Popülasyonun güncellenmesi ardından tekrar önerilen gruplama temelli genetik algoritma ile çözüm üretilmektedir. Önerilen algoritma 5600 problem çözülerek test edilmiştir. Çözülen test problemleri literatürde çalışılan statik problemlerin dinamik hale getirilmesi ile üretilmiştir.

(Du, Li ve Chou 2005): Elektronik ticaret uygulaması olarak üreticiden direkt tüketiciye teslimat şeklinde dinamik araç rotalama problemi ele alınmıştır. Üç aşamalı olarak çözüme gidilmektedir; ilk olarak başlangıç rotaları oluşturulur. Başlangıç rotaları oluştururken kümeleme yaklaşımı kullanılmıştır, bilinen müşteriler konumları gözetilerek alt kümelere ayrılmıştır ve her bir kümeye bir araç hizmet verecektir. Yeni ortaya çıkan müşteriler ise dört farklı çok bilinen sezgisele göre rotalara eklenmektedir. Müşteriler açığa çıkma sürelerine göre önceliklendirilir; yani ilk gelen ilk servis edilir (*First-In-First-Serve*). Rotalara eklenirken ise, ilk uygun rotaya son müşteri olarak eklenebilirler (*First-Fit-Nearest*), ya da tüm rotalar incelendikten sonra en uygununa eklenebilirler (*Best-Fit-Nearest*). Rotaya eklemek için son olarak kullanılan sezgisel yöntem ise yer değiştirme (*swap*) operatörüdür. Daha sonra rota içi ve rotalar arası rota iyileştirme aşamaları sırası ile uygulanmaktadır. Bu aşamalarda da yine bilinen sezgisel yöntemler kullanılmıştır. Elektronik ticaret ortamı simüle edilerek yaklaşım değerlendirilmiştir.

(Montemanni, ve diğerleri 2005): Dinamik araç rotalama problemine karınca kolonisi algoritması ile çözüm önerisi getirilen çalışma, problemi ele alış biçimi açısından ileriki yıllarda yapılan birçok çalışmaya referans olmaktadır. Tez çalışmamızda da uygulamış olduğumuz bu yaklaşımda, planlama periyodu eşit zaman dilimlerine ayrılarak her bir zaman diliminde bilinen verilerle üretilmiş statik problem çözülmektedir. Veri seti olarak sentetik veri kullanılan çalışmadaki sonuçlar, aynı veri seti kullanılarak çalışılan diğer yayınlarda karşılaştırmalı olarak incelenmiştir.

(Alvarenga, Silva ve Mateus 2005): Zaman pencereli dinamik araç rotalama problemi için melez bir çözüm önerisi sunulmuştur. Öncelikle genetik algoritma kullanarak bilinen problem verileri ile rotalar üretilmektedir. Daha sonraki aşamada ise önerilen sezgisel yöntem ile dinamik olarak değişen bilgiler dikkate alınarak rotalar güncellenmektedir. Sentetik problemler önerilen çözüm yöntemi ile çözülerek sonuçları raporlanmıştır.

(Chen, Hsueh ve Chang 2006): Zaman pencereli dinamik araç rotalama problemi karma tam sayılı programlama problemi olarak modellenmiştir. Zaman

penceresi kısıtları da dikkate alınmıştır. Probleme özgü geliştirilen sezgisel yöntemler ile literatürde bilinen test problemleri çözülmüştür.

(Gendreau, Guertin, ve diğerleri 2006): Toplama ve dağıtım unsurlarının birlikte yer aldığı dinamik araç rotalama problemine komşuluk arama temelli sezgisel yöntem önerilmiştir. Planlanmış rotaların, zaman içerisinde yeni bir toplama veya dağıtım talebi ortaya çıktıkça sezgisel yöntem kullanılarak iyileştirilmesine dayanmaktadır. Toplam tur maliyetini en küçükleyecek şekilde, bir müşteri rotalar arasında en uygun yere yerleştirilmektedir. Aynı zamanda elde edilen sonuçların iyileştirilmesi için Tabu Arama meta sezgisel yöntemi de kullanılmıştır. Simülasyon ile üretilen veriler algoritmayı test etmek amaçlı çözülmüştür.

(Fabri ve Recht 2006): Zaman pencereli dinamik toplama ve dağıtımlı araç rotalama problemi ele alınan çalışmada araçların bekleme süreleri de dikkate alınmıştır. Planlama periyodunun her hangi bir anında ortaya çıkan müşteriler online olarak değerlendirilmektedir, kabul edilir veya reddedilir. Kabul edilen müşteriler uygun olan araçların rotalarına eklenir. Çözüm algoritması olarak sezgisel bir yöntem önerilmiştir. Önerilen sezgisel yöntem tek araç rotalama ve çoklu araç rotalama problemleri için ayrı ayrı detaylandırılmıştır. Yerel arama yöntemleri ile sonuçlar iyileştirilmiştir.

(Housroum, ve diğerleri 2006): Çalışmada zaman pencereli dinamik araç rotalama problemine genetik algoritma kullanılarak çözüm önerisi getirilmiştir. Algoritma parametrelerini tasarlama amacıyla deney tasarımı yaklaşımı kullanılmıştır. Literatürde çalışılmış bilinen test problemleri ile algoritmanın işleyişi test edilmiştir ve karşılaştırmalı sonuçlar raporlanmıştır.

(Chen ve Xu 2006): Zaman pencereli dinamik araç rotalama problemini ele almışlardır. Karar değişkeni sayısının çok fazla olduğu doğrusal programlama problemlerinin çözümü için önerilen kolon üretme yöntemi temelli bir çözüm önerisinde bulunulmuştur. Kolon üretme yöntemi, karar değişkenlerinin tamamını bir anda ele almak yerine aşamalı olarak karar değişkenlerini modele dahil ederek optimal çözüme ulaşan bir tekniktir. Çalışmada kolon üretme tekniğini dinamik probleme uyarlamak adına periyodik olarak optimizasyon işlemi tekrarlanmaktadır. Statik test

problemleri her bir karar verme periyodunda ardışık olarak önerilen yöntem ile çözülerek sonuçları raporlanmıştır.

(Hanshar ve Ombuki-Berman M. 2007): Çalışmada kapasite kısıtlı dinamik araç rotalama problemi ele alınmıştır. Genetik algoritma ile probleme çözüm önerisi getirilmiştir. Amaç fonksiyonu seyahat maliyeti olan, araçların yol almış olduğu mesafeyi minimize etmektir. Aynı zamanda Tabu Arama meta sezgiseli kullanarak da çözüm önerisi sunulmuştur. Montemanni ve diğerlerinin önermiş olduğu çözüm stratejisi kullanılarak aynı test problemlerine çözüm getirilmiştir. Karşılaştırmalı sonuçlar değerlendirilmiştir.

(Wen, ve diğerleri 2010): Birden fazla planlama periyodu (veya günü) olan dinamik araç rotalama problemi ele alınmıştır. Planlamalar tek bir gün veya dönem için değil de birden fazla ardışık gün (dönem) için yapılmaktadır. Müşterilerin talepleri ve hangi periyotta hizmet görmesi gerektikleri zamana bağlı olarak dinamik bir şekilde ortaya çıkmaktadır. Problem, karma tam sayılı doğrusal programlama problemi olarak modellenmiştir. Amaç fonksiyonu olarak, toplam tur maliyeti ve müşterilerin bekleme zamanının minimizasyonu seçilmiştir. İsviçre'de dağıtım yapan bir şirketin verileri kullanılarak, önerilmiş olan üç aşamalı sezgisel yöntemin performansı test edilmiştir.

(Lorini, Potvin ve Zufferey 2011): Zaman pencereli dinamik araç rotalama problemi ele alınmıştır. Müşterilerden araçlar ile toplanmış ürünler, tek bir bölgede bulunan depoya getirilmektedir. Zaman penceresi kısıtı bulunduğundan, müşteriye belirlenmiş zaman aralığından daha geç bir zamanda varıldığında bu durum artı maliyet olarak kaydedilmektedir. Amaç fonksiyonu araçlarım toplam seyahat süresi ve müşterilere gecikme maliyetinin minimize edilmesidir. Gerçek hayat problemini daha iyi temsil etmek adına zaman bağımlı seyahat süreleri kullanılmaktadır. Çalışma günü üç farklı zaman periyodunda değerlendirilmiştir; sabah, öğle yemeği vakti ve öğleden sonra şeklinde. Her bir zaman periyodu için belirlenmiş katsayılar (1,25; 0,5; 1,25) ile iki şehir arasındaki seyahat süresi çarpılmaktadır. Örneğin şehir *i* den *j* ye ulaşım süresi t_{ij} öğle saatinde gidilmesi gerektiği durumda 0,5 ile çarpılmaktadır. Bu durum trafiğin yolculuk süresine etkisinin daha iyi temsil edilmesine imkân vermektedir.

(Liao ve Hu 2011): Gerçek zamanlı trafik bilgilerinin sisteme dahil edildiği dinamik araç rotalama problemi türü ele alınmıştır. İki aşamadan oluşan bir çözüm stratejisi önerilmiştir. Sezgisel bir algoritma ile öncelikle taslak rotalar belirlenmektedir. Daha sonra gerçek zamanlı bilgiler gelince, taslak rotalar Tabu Arama meta sezgiseli kullanılarak güncellenmektedir. Gerçek zamanlı trafik bilgilerinin oluşturulmasında ise DynaTAIWAN ismi verilen bir simülasyon modeli kullanılmıştır.

(Khouadjia, ve diğerleri 2012): Dinamik kapasite kısıtlı araç rotalama problemine çözüm önerisi getirilmiştir. Montemanni ve arkadaşlarının (Montemanni, ve diğerleri 2005) önermiş olduğu, planlama periyodunu alt zaman dilimlerine bölmek ile elde edilecek statik problemlerin çözümünden oluşan yaklaşım kullanılmıştır. Her bir statik problem popülasyon tabanlı PSO ve tek çözüm tabanlı Değişken komşuluk arama algoritmaları ile çözülmüştür. Dinamik problemlerde PSO uygulamasını DAPSO olarak isimlendirilmiştir, DAPSO ile elde edilen çözümleri iyileştirmek adına yerel arama algoritmalarından 2-opt kullanılmıştır. Algoritma ile elde edilen sonuçlar karşılaştırmalı olarak tablolar ile sunulmuştur.

(Mavrovouniotis ve Yang 2012): Trafik durumunun dinamik olarak belirlendiği araç rotalama problemi ele alınmıştır. Trafik değişikliği bir döngü oluşturmaktadır, yani bir önceki trafik ortamı gelecekte de ortaya çıkmaktadır. Hafızaya dayalı Karınca Kolonisi algoritması ile o ana kadar elde edilmiş en iyi çözüm bilgisi saklanmaktadır. Bu bilgi çözüm uzayını tarama esnasındaki çeşitliliği sağlamak ve dinamik ortam değişikliği olduğunda çözümleri transfer etmek için kullanılmaktadır. Karınca kolonisi algoritmasını test etmek için trafik faktörünü de dikkate alarak test problemleri üretilmiştir. İki şehir arasındaki uzaklık (d_{ij}) aynı iki şehir arası trafik faktörü (t_{ij}) ile çarpılarak bir şehirden diğerine gidiş maliyeti hesaplanmaktadır. Beklenmedik bir trafik sıkışıklığını temsil etmek adına belirli bir aralıkta rassal olarak üretilmiş bir değer trafik faktörüne eklenmektedir. Karınca kolonisi algoritmasının hafıza bileşeninin farklı parametrelerle test problemlerine uygulanmış halleri karşılaştırılmıştır.

(Xu, Wang ve Yang 2013): Zaman pencereli dinamik araç rotalama problemi ele alınmıştır. Değişken komşuluk arama meta sezgisel yöntemi kullanılarak probleme çözüm önerisi getirilmiştir. Sentetik bir problem üretilmiştir ve önerilen algoritma bu problem üzerinde test edilmiştir. Problemde statik olarak 30 adet müşteri bulunmaktadır bu müşterilerin pozisyon ve talep miktarları bilinmektedir. 10 adet müşteri ise dinamik olarak açığa çıkmaktadır, dinamik müşterilerin ortaya çıkma zamanı 2 periyot olarak ele alınmıştır. Müşterilerin bir kısmı periyot 1 'de bir kısmı periyot 2'de ortaya çıkmaktadır. Bu nedenle problem 2 zaman diliminde ele alınarak çözümlenmektedir. Değişken komşuluk arama algoritması içerisinde, başlangıç çözümü üretilirken Clarke ve Wright tasarruf algoritması kullanılmıştır, shaking aşamasında yer değiştirme ve ekleme sezgisel yöntemleri ve son olarak yerel arama safhasında is e 2-opt ve 3-opt sezgisel yöntemleri kullanılmıştır.

(van Veen, ve diğerleri 2013): Zaman pencereli dinamik araç rotalama problemi için Karınca Kolonisi meta sezgisel algoritması kullanılarak çözüm önerisi sunulmuştur. Müşterilere dair bilgiler ve zaman pencereleri çalışma günü ilerledikçe sisteme dahil edilmektedir ve alt problemlerin çözümleri oluşturulurken bu bilgiler kullanılmaktadır. Karınca kolonisi algoritmasının işleyişini dinamikleştirmek adına bir kontrol mekanizması eklenmiştir. Kontrol mekanizması probleme dair bilgileri okuyarak veri yapılarını düzenleyip başlangıç çözümlerini oluşturmaktadır. Başlangıç çözümleri oluşturulurken, en yakın komşu arama sezgisel yöntemi kullanılmıştır, zaman penceresi ve kapasite kısıtı dikkate alınarak başlangıç çözümleri düzenlenmektedir. Başlangıç çözümleri tamamlandıktan sonra, zaman ilerlemeye başlatılır ve her bir zaman aralığında (kullanıcı tarafından başlangıçta zaman aralıkları uzunluğu belirlenmiştir) yeni bir müşteri bilgisi olup olmadığı kontrol edilir, eğer var ise sezgisel bir yöntem (*InsertMissingNodes*) kullanılarak uygun rotaya eklenir. Tüm gerekli eklemeler yapıldıktan sonra iki farklı Karınca kolonisi algoritması devreye girmektedir ve yeni zaman dilimi başlayana kadar çözümleri iyileştirmektedir. Algoritmanın çalışma yapısı Montemanni ve diğerlerinin önermiş olduğu yapıya

benzemektedir. Önerilen çözüm algoritması Solomon test problemleri[3] kullanılarak test edilmiştir.

Stokastik:

(Bent ve Van Hentenryck 2003): Müşterilerin konumlarının ve servis sürelerinin rassal olarak hesaplandığı dinamik araç rotalama problemi ele alınmıştır. Amaç fonksiyonu olarak, hizmet edilen müşterilerin sayısını maksimize etmek ve tur maliyetini minimize etmek belirlenmiştir. Gerçek veriler planlama süresi boyunca dinamik olarak netleşmektedir. Rotalar sürekli bir şekilde, veriler ortaya çıktığında tekrar gözden geçirilerek gerekli değişiklikler yapılmaktadır.

(Hvattum, Løkketangen ve Laporte 2006): Müşteri taleplerinde iptal durumunun söz konusu olduğu araç rotalama problemi ele alınmıştır. Problem, bu özelliğinden dolayı dinamik bir yapıya sahiptir. Ek olarak müşterilerin konumları ve talep miktarları planlamaya başlandığı anda bilinmemektedir. Geçmiş veriye bağlı olarak tahmin edilmektedir. Ancak bir müşteri çağrısı kabul edildiği vakit, konumu ve talebin gerçek değeri öğrenilmektedir. Bu nedenle de problem stokastik özelliktedir. İki aşamalı bir stokastik model önerilmiştir. Bir gerçek hayat problemi ele alınmıştır. Örnek senaryolar oluşturulup sezgisel teknikle çözüm önerisi getirilmiştir.

(Pavone, ve diğerleri 2009): Çalışmada, stokastik ve zaman pencereli dinamik araç rotalama problemi ele alınmıştır. Müşteri taleplerinin Poisson dağılımına uygun olduğu varsayılmıştır. Müşterilerin konumlarının dağıldığı olarak depo merkezli Q birim çaplı çember servis bölgesi olarak belirlenmiştir. Her bir müşterinin belirli bir zaman aralığında servis edilmesi gerekmektedir ki bu da zaman penceresi özelliğinden gelmektedir. Servis alanı araç sayısı kadar alt bölgeye ayrılarak her bir bölge için tek bir araç rotası belirlenmektedir. Yani her bir alt bölge için gezgin satıcı problemi çözülmektedir.

(Branchini, Armentano ve Løkketangen 2009): Dinamik kapasite kısıtlı araç rotalama problemi için uyarlanabilir tanecikli yerel arama yöntemi adlı bir sezgisel önermişlerdir. Önerilen çözüm stratejisine göre, araçlar müşterilerin muhtemel ortaya

[3] Solomon Benchmark adı ile zaman pencereli ARP için üretilen test problemleri: https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/

çıkacakları konumlara yakın bir şekilde dağıtılmaktadır. Araçların bir sonraki konumları uyarlanabilir tanecikli yerel arama yöntemi ile belirlenmektedir. Uyarlanabilir tanecikli yerel arama yöntemi, bir taneciğin komşuluk sayısında belirli bir kurala göre azaltmaya giden bir yerel arama yöntemidir. Böylelikle çözüm uzayında potansiyel çözüm noktalarında azalmaya gidilmektedir. Gerçek hayat problemi simüle edilerek önerilen algoritmanın performansı test edilmiştir. Planlama periyoduna başlamadan öncesinden bilinen verilerle ile sistem araçlara atama yapmaktadır, daha sonra ortaya çıkacak müşteriler simülasyon ile stokastik olarak tahmin edilmektedir. Gerçek zamanlı ortaya çıkan müşteriler simülasyona bilgi olarak girilmektedir.

(Novoa ve Storer 2009): Taleplerin stokastik olarak belirlendiği dinamik araç rotalama problemi ele alınan çalışmada dinamik programlama temelli bir sezgisel yöntem kullanılmıştır. Müşterilere ait talepler birbirinden bağımsız olmakla beraber belirli bir olasılık dağılımına uygun olduğu varsayılmaktadır. Çalışmada kullanılan dağılım genellikle düzgün dağılımdır, farklı aralıklarda düzgün dağılıma uygun müşteri talepleri kümeleri üretilmiştir. Tahmin edilen müşteri taleplerine göre rotalama yapılmaktadır. Araçlar yola çıkarıldıktan sonra müşterilere ulaştıklarında gerçek talepler öğrenilmektedir. Tahmin edilen talepler gerçek taleplerden yüksek ise genellikle araç kapasitelerinde sorun olmamakta, ancak düşük tahmin edilmiş ise rotaları gerçekleştirmek için farklı bir çözüm önerilmektedir. Ya eksik kapasite ile de olsa müşteriye hizmet verilir ve eksik kalan ürün en kısa zamanda müşteriye temin edilir, ya da araç kapasitesini artırmak için depoya geri gönderilir. Tur maliyetini en küçükleyecek şekilde rotalar güncellenmektedir.

(Pavone, Frazzoli ve Bullo 2011): Müşterilerin ortaya çıkış zamanlarının, konum bilgilerinin ve servis sürelerinin stokastik olarak ortaya çıktığı rotalama problemi ele alınmıştır. Amaç fonksiyonu sistem için gerekli olan ortalama servis süresinin minimizasyonu olarak belirlenmiştir. Literatürde *m*- araçlı Dinamik tamirci problemi olarak da bilinmektedir.

(Azi, Gendreau ve Potvin 2012): Dinamik araç rotalama problemi için Uyarlanabilir Geniş Komşuluk Arama (*Adaptive large neighborhood search)* sezgiseli

ele alınmıştır. Ele alınan sezgisel yöntem problemin statik versiyonu için daha önceki çalışmalarda kullanılmıştır, çalışmada dinamik problemler için tekrar çalışılmıştır. Başlangıç rotaları sezgisel olarak, maliyeti en küçükleyecek şekilde müşterilerin sıralanması ile belirlendikten sonra Uyarlanabilir Geniş Komşuluk Arama ile sonuçlar iyileştirilmeye çalışılmıştır. Ele alınan problemde stokastik veri kullanılmıştır. Müşteri taleplerinin Poisson dağılımına uygun olduğu varsayılmıştır ve veriler üretilirken Poisson dağılımı kullanılmıştır. Önerilen yöntemin test edilmesinde simüle edilen veriler kullanılmıştır.

Stokastik ve dinamik özellikteki araç rotalama problemleri; otonom sistemlerde, robotik ve karmaşık sistemlerde çalışan araştırmacıların ilgi alanına girmektedir. Bu bölümde belli başlı birkaç çalışmadan bahsetmek yeterli görülmektedir.

ÜÇÜNCÜ BÖLÜM

3 DİNAMİK ARAÇ ROTALAMA PROBLEMLERİ İÇİN ÇÖZÜM YÖNTEMLERİ

Araç rotalama problemleri akademik literatürde uzun zamandır çalışılan bir problem türüdür. Bu nedenle problemin dinamik versiyonu için önerilen çözüm yöntemlerinin temelleri statik ARP için önerilen yöntemlere dayanmaktadır. Geleneksel statik ARP için önerilen çözüm yöntemleri ile ilgili çeşitli araştırma makaleleri bulunmaktadır (Prins 2004), (Laporte, ve diğerleri 2000), (Taillard 1993).

Dinamik araç rotalama problemleri için önerilen çözüm yöntemleri üç temel başlıkta incelenebilir; strateji temelli algoritmalar, sezgisel yöntemler ve meta sezgisel yöntemler. Meta sezgisel yöntemler ise kendi aralarında tek çözüm temelli ve popülasyon temelli olarak ikiye ayrılmaktadır.

Bu bölümde DARP için önerilen çözüm yöntemlerine değinilecektir.

3.1 Strateji Temelli Algoritmalar

DARP'nin en basit hali kapasite kısıtı olmayan tek aracın servis için kullanıldığı, literatürde dinamik tamirci rotalama problemi (DTRP) olarak bilinen problemdir. Daha karmaşık hali ise kapasite kısıtı bulunan *m* farklı aracın rotalanması problemidir.

Dinamik tamirci rotalama problemi (Bertsimas ve Van Ryzin, A stochastic and dynamic vehicle routing problem in the Euclidean plane 1991) tarafından kuyruk teorisine dayandırılarak modellenmiştir. Taleplerin konumların konveks ve sınırlı bir *A* bölgesinde düzgün olarak dağıldığı varsayılmaktadır. Taleplerin Poisson dağılımına uygun olarak λ geliş oranı ile ortaya çıkmaktadır. *A* bölgesindeki herhangi bir müşterinin depo ile olan uzaklığın beklenen değeri *r* olduğu kabul edilmektedir. Her bir aracın sabit *v* hızıyla seyahat ettiği ve müşterilere hizmet verme sürelerinin ortalamasının *s* olduğu düşünülmektedir. Bu durumda bir aracın sistem kullanım oranı ρ olmaktadır ve durağan sistemlerde $\rho = \lambda s$ olarak hesaplanmaktadır.[4]

[4] Kuyruk teorisi ile ilgili detaylı bilgi için: Zukerman, Moshe. "Introduction to queueing theory and stochastic teletraffic models"

Amaç fonksiyonu, beklenen sistem zamanını (her bir alt problemin optimal bir şekilde çözüldüğü varsayımı altında) minimize edecek rotalama stratejisinin bulunması olarak belirlenmiştir.

DTRP için strateji temelli algoritmaların geliştirildiği çalışmalarda, probleme ait belli başlı özellikler tanımlanmıştır. Örneğin durağanlık durumu, trafiğin yoğun veya hafif olmasının modellenmesine göre optimal sistem zamanın beklenen değeri T^* ın değişimi gibi. Bu tanımlamalar aşağıdaki gibi ifade edilmektedir:

- **Yoğun olmayan trafik durumu:** $\lambda \to 0$ için optimal sistem zamanının beklenen değeri T^* basit bir formülasyona sahiptir. $T^* \to \frac{r}{v} + s$ şeklinde hesaplanmaktadır, yani gidilen yol hıza oranlanıp servis süresi eklenmektedir.
- **Durağanlık koşulu:** Durağanlık koşulu sistem zamanının sonlu olmasını garanti etmektedir. Bertsimas ve van Ryzin (Bertsimas ve Van Ryzin 1993) tarafından durağanlık koşulunun $\rho + \frac{2\lambda r}{mvq} < 1$ olacağı gösterilmiştir. Burada q, her bir aracın kapasite kısıtını göstermektedir ve tamirci rotalama problemi olduğu için bir tamircinin gidebileceği maksimum müşteri sayısını temsil etmektedir. Eğer araçlar için her hangi bir kapasite kısıtı yoksa yani $q = \infty$ ise, durağanlık koşulu $\rho = \lambda s < 1$ şeklinde olmaktadır.
- **Yoğun trafik durumu:** Optimal sistem zamanının beklenen değeri yaklaşık olarak: $T^* \sim \frac{\gamma^2 \lambda A (1-\frac{1}{q})^2}{m^2 v^2 (1-\rho-\frac{2\lambda r}{mqv})^2}$, $\rho + \frac{2\lambda r}{mvq} \to 1$ iken şeklinde öngörülmektedir. Burada γ kullanılan stratejiye bağlı bir sabit değerdir, m ise araç sayısıdır. Eğer kapasite ile ilgili bir limit bulunmuyor ise $T^* \sim \frac{\gamma^2 \lambda A}{v^2 (1-\rho)^2}$, $\rho \to 1$ şeklindedir.

Dinamik tamirci rotalama problemi ile ilgili yukarıda bahsedilen problem dinamikleri için detaylı açıklamalar (Bertsimas ve Van Ryzin 1991), (Bertsimas ve Van Ryzin 1993), (Papastavrou 1996) çalışmalarında bulunmaktadır. Bu bölümün devamında strateji temelli algoritmaların neler olduğuna yer verilecektir.

DTRP için belli başlı kullanılan strateji temelli algoritmalar aşağıdaki gibi sıralanmaktadır:

i. **İlk Gelen İlk Hizmet Görür (First Come First Served-FCFS):** Müşteriler ortaya çıkma zamanlarına göre sıralanırlar ve ilk önce hangi müşteriden talep gelmiş ise o müşteri hizmet görmektedir.

ii. **Stokastik Kuyruk Medyan (Stochastic Queue Median-SQM):** Bu strateji FCFS stratejisinin biraz daha geliştirilerek uygulanmış halidir. Araçlar, müşterilerin yer aldığı servis bölgesinin orta kısmında konumlandırılmaktadırlar. İlk ortaya çıkan müşteri kuralına göre müşterilere hizmet verilir ve hizmetini tamamlayan araç tekrar servis alanının orta konumuna dönmektedir.

iii. **En Yakın Komşuluk Kuralı (Nearest Neighbor):** En yakın komşuluk kuralına göre; en son hizmet verilen müşterinin işlemi tamamlandıktan sonra, servis bekleyen müşteriler içerisinde konum olarak en yakın müşteriye hizmet verilir.

iv. **Gezgin Satıcı Kuralı (Traveling Salesman-TSP):** Önceden belirlenmiş bir sayı kadar müşteri ortaya çıkması beklenir ve o müşteriler Gezgin Satıcı problemi gibi düşünülerek çözülür. Gezgin satıcı problemini çözen bir algoritma kullanılarak çözüm elde edilir (Bertsimas ve Van Ryzin 1993).

v. **Nesil Üretim (Generation) (GEN) Kuralı:** Papastavrou (Papastavrou 1996) tarafından önerilen GEN stratejisi SQM ve TSP stratejilerinin birleşimi şeklindedir. Tek bir araç ile hizmet verilen dinamik tamirci rotalama problemi için uygulanmıştır. Yöntemin işleyişi şu şekildedir, ilk olarak araç servis bölgesinin orta noktasına konumlandırılmaktadır. İlk müşteri ortaya çıktığı anda araç müşteriye hizmet vermek için yola çıkmaktadır. Eğer müşterinin hizmeti tamamlanana kadar başka bir müşteri ortaya çıkmaz ise araç başlangıç konumuna geri dönmektedir. Fakat hizmet tamamlanana kadar başka müşteriler ortaya çıkmış ise TSP prosedürü uygulanır ve gezgin satıcı problemi oluşturulup en iyi şekilde çözüm elde edilir.

vi. **Kısımlara Ayırma Kuralı (Partitioning-PART):** Tek bir araçla, kare olduğu düşünülen hizmet A bölgesi için uygulanan bir stratejidir. İlk olarak servis bölgesi A, alt karelere ayrılır. Alt bölgeler belirli bir sırada araç tarafından gezilir. Aracın bu gezintisi esnasında alt bölgelerde bulunan müşteriler, kendi bulundukları bölge içerisinde FCFS kuralına göre ziyaret edilirler. A bölgesinin alt bölgelerini gezme döngüsü tüm müşteriler hizmet verilinceye kadar devam ettirilir (Bertsimas ve Van Ryzin 1991).

vii. **Bekleme Stratejileri (Waiting Strategies-WAIT):** Rotalama problemlerinde dinamik olarak ortaya çıkan müşterilerin hizmet görmek için beklemiş olacakları sürenin minimum düzeyde olması önemlidir. Bu süreler toplam tur süresinin de uzunluğunu etkilediğinden amaç fonksiyonları belirlenirken dikkate alınmaları gerekmektedir. Larsen ve diğerleri (Larsen, Madsen ve Solomon 2004), zaman pencereli dinamik gezgin satıcı problemi için gecikmeleri minimize edecek şekilde stratejiler önermişlerdir. Önerilen bir stratejiye göre, bir araç en son hizmet verdiği müşteride beklemektedir. Bu bekleme süresi, bir sonraki hizmet vereceği müşterinin zaman penceresine göre, erken hizmet verme süresi gelene kadar olmaktadır. Bir diğer stratejiye göre, araç en son hizmet verdiği müşteride değil de müşterilerin öncelik sıraları gözetilerek, önceliği olan ilk müşteride beklemektedir. Bent ve Van Hentenryck (Bent ve Van Hentenryck 2007) tarafından yeniden konumlandırma stratejileri tekrar ele alınmıştır ve geliştirilmiştir. Deneysel sonuçlar incelendiğinde bu iki bekleme stratejisi dinamizm derecesi yüksek olan problem türlerinde, hizmet edilecek müşteri sayısını maksimize etmek konusunda başarılı olmaktadır.

3.2 Sezgisel Yöntemler

Sezgisel algoritmalar problem bağımlı algoritmalardır. Yani işleyiş kuralı probleme özgü olarak belirlenmektedir. Yapıcı ve geliştirici olarak iki alt sınıfa ayrılabilirler. Yapıcı sezgisel algoritmalar araç rotalama problemi için başlangıç rotalarını oluşturabilecek türde algoritmalardır. Geliştirici algoritmalar ise, var olan bir tur veya çözümün iyileştirilmesine yönelik algoritmalardır. Planlama periyodu boyunca yeni bir talep/müşteri ortaya çıktığında sezgisel algoritmalar tekrar tekrar çalıştırılmaktadırlar.

Sezgisel yöntemlerin en çok kullanılanlarından olan ekleme (*insertion*) sezgiselleri, ilk olarak Psaraftis (H. N. Psaraftis 1988) tarafından çalışılmıştır. Müşteriler, maliyeti minimum tutacak şekilde rotalardaki en uygun konumlara eklenmektedirler.

Mitrović –Minić ve diğerleri (Mitrović-Minić, Krishnamurti ve Laporte 2004), ekleme sezgiselini geliştirerek zaman pencereli toplama ve dağıtımlı dinamik araç

problemine uygulamışlardır. Zaman periyodu, uzun zaman ve kısa zaman dönemi olarak iki alt periyoda ayrılmıştır. Böylelikle planlama süreci için kısa vadeli amaç fonksiyonu ve uzun vadeli amaç fonksiyonu tanımlamaları yapılmıştır. Rotalama için iki tane sezgisel yöntem önermişlerdir, bunlardan ilki maliyeti en az olan konuma ekleme (*cheapest insertion*) sezgiseli başlangıç rotasını oluşturmaktadır. Diğer sezgisel yöntem ise, yine maliyeti göz önünde bulundurarak rotayı iyileştirmektedir. Rotayı maliyet açısından geliştiren sezgisel yöntem Tabu Arama temelli bir algoritmaya dayanmaktadır. Yeni talepler ortaya çıktığı müddetçe sezgisel algoritmalar çalıştırılmaktadır.

Kilby ve diğerleri tarafından yapılan çalışmada (Kilby, Prosser ve Shaw 1998), müşterileri rotalardaki en uygun yere ekleyen ekleme sezgiselleri kullanılmıştır. Daha sonra oluşturulan rotalar 2-Opt (Lin 1965), Or-Opt, yer değiştirme ve çapraz yer değiştirme (Van Breedam 1994) sezgisel yöntemleri kullanılarak iyileştirilmiştir.

Branchini ve diğerleri (Branchini, Armentano ve Løkketangen 2009) tarafından uyarlanabilir tanecikli yerel arama yöntemi önerilmiştir. Uyarlanabilir tanecikli (*Adaptive Granular*) yerel arama yöntemi, bir taneciğin komşuluk sayısında belirli bir kurala göre azaltmaya giden bir yerel arama yöntemidir. Böylelikle çözüm uzayında potansiyel çözüm noktalarında azalmaya gidilmektedir. Gerçek hayat problemi simüle edilerek önerilen algoritmanın performansı test edilmiştir. Planlama periyoduna başlamadan önce bilinen veriler ile sistem araçlara atama yapmaktadır, daha sonra ortaya çıkacak müşteriler simülasyon ile stokastik olarak tahmin edilmektedir. Gerçek zamanlı ortaya çıkan müşteriler simülasyona bilgi olarak girilmektedir.

3.3 Meta Sezgisel Yöntemler

Sezgisel algoritmalar belirli çerçevelerde çözüm bulmak için kullanılsalar da, çözüm kalitesini belirli bir seviyenin üzerine çıkarmada etkin olamamaktadırlar. Basit kurallara dayanmaları ve probleme özgü tasarlanmış olmaları sebebiyle genellikle yerel arama prosedürleri olarak kullanılmaktadırlar. Problemden bağımsız ve çözüm uzayını taramakta daha etkin olan meta sezgisel algoritmalar çözüm kalitesi açısından sezgisel algoritmaların bir adım ötesine geçmektedirler.

Probleme dair herhangi bir varsayıma gerek duymayan meta sezgisel algoritmalar büyük boyutlardaki karmaşık optimizasyon problemlerin çözümünde etkin sonuçlar elde edilmesine imkan vermektedirler. İteratif olarak çözüm uzayını tarayan ve kabul edilebilir zaman dilimlerinde çözüm elde eden meta sezgisel algoritmaların temel prensibi çeşitlendirme (*diversification*) ve yoğunlaştırma (*intensification*) prensibine dayanmaktadır. Yani önce çözüm uzayında bir pozisyondan başka bir pozisyona geçme ve yeni pozisyonun komşuluklarını detaylıca tarama temeline dayanmaktadır. Bu prensiplere dayalı iyi tasarlanmış ve modellenmiş meta sezgisel bir algoritmanın yerel optimum çözümlere takılmadan çözüm uzayını gezmesi beklenmektedir.

Genellikle doğadan esinlenerek tasarlanan evrimsel algoritmalar sezgisel algoritmalar gibi optimal çözümü bulmayı garanti etmezler, büyük boyutlu problemler için kabul edilebilir iyi bir çözüm bulurlar. Meta sezgisel algoritmalar tek çözüm tabanlı ve popülasyon tabanlı olmak üzere iki alt gruba ayrılmaktadırlar (Talbi 2009).

Tek çözüm tabanlı algoritmalarda bir başlangıç çözümü ile aramaya başlanır ve o çözüm algoritmanın kuralına göre iyileştirilir. Ardışık olarak çözüm uzayında bir noktadan daha iyi bir noktaya ilerleyerek çözüm aranmaktadır. Tek çözüm temelli algoritmalar; Tabu Arama, Tavlama Benzetim, Değişken Komşu Arama, Ardışık Yerel Arama, GRASP şeklinde örneklendirilebilir. Bu örnekler çoğaltılabilir.

Popülasyon temelli algoritmalarda ise çözüm uzayı bir topluluk tarafından gezilmektedir. Bu topluluğa sürü veya popülasyon adı verilmektedir. Başlangıç popülasyonunda, her bir popülasyon elemanı bir başlangıç çözümünü temsil etmektedir. Her bir iterasyonda, popülasyon elemanları bulundukları noktadan daha iyi bir noktaya gitmeye çalışmaktadırlar. Böylelikle çözüm uzayı popülasyon büyüklüğü kadar eleman tarafından farklı açılardan gezilmektedir. Tek çözüm temelli algoritmalara göre çözüme ulaşmakta daha güçlü bir yapıya sahiptirler. Genetik Algoritmalar, Evrimsel Algoritmalar, Karınca Kolonisi Algoritmaları, Parçacık Sürü Optimizasyonu Algoritmaları, Arı Kolonisi Algoritmaları, Yapay Bağışıklık Sistemi Algoritmaları bu alanda yaygın olarak başarılı bir şekilde uygulanan algoritmalardır.

Bu bölümde dinamik araç rotalama problemlerinin çözümünde kullanılan meta sezgisel algoritmalardan tek çözüm temelli ve popülasyon temelli olarak ayrı ayrı bahsedilecektir.

3.3.1 Tek Çözüm Temelli Meta Sezgiseller

Tek çözüm temelli algoritmalarda tek bir çözüm ardışık olarak iyileştirilir. Tek bir çözümden, yerel arama prosedürleri kullanılarak bir sonraki adımda çözüm olabilecek aday noktalar üretilir. Bu çözüm adayı noktaların oluşturduğu kümenin en iyi noktası bir sonraki adımdaki çözüm noktası olur. Önceden belirlenmiş bir durma kriteri sağlanana kadar bu döngü tekrar edilir.

Çözüm adayları kümesi belirlenirken, o anki çözüm noktasının komşuluğu gezilmektedir. Yani yerel arama yapılmaktadır. Bu noktada komşuluk tanımlarının önceden algoritma için belirlenmiş olması gerekmektedir.

Tabu Arama (Tabu Search-TS): Tabu arama algoritması Glover (Glover 1986) tarafından literatüre kazandırılmıştır. Yerel en iyi çözüme takılmadan global en iyiye yakınsaması açısından kuvvetli bir algoritmadır. Tabu yani yasaklanmış noktaların bir listesi oluşturulur, böylelikle mevcut çözümden hangi çözüm noktalarına gidileceği ve gidilemeyeceği belirlenir. Tabu listesi her adımda güncellenir. Kısa-dönem strateji ve uzun-dönem strateji olmak üzere iki stratejiye sahip algoritmada, kısa-dönem stratejisi kullanılarak hangi noktaların tabu listesine gireceği ve listeden çıkacağı belirlenir.

Gendreau ve diğerleri (Gendreau, Guertin, ve diğerleri 1999) paralel tabu arama algoritması kullanarak gerçek zamanlı araç rotalama problemini çözmüşlerdir. Problemin statik versiyonu için başlangıç çözümü oluşturulmuştur ve çözüm uzayı dinamik olarak değiştikçe geçmiş iyi değerleri hafızada tutan algoritma ile dinamik problemi çözmüşlerdir. Hafızadaki bilgiler kullanılarak tabu listesi güncellenmektedir. Eş zamanlı olarak, farklı tabu listeleri kullanılmıştır ve bu tabu listelerinin her biri farklı bir başlangıç çözümünü geliştirmek için kullanılmıştır. Böylelikle paralel olarak geliştirilen algoritmanın performansı artırılmıştır.

2006 yılında Gendreau ve diğerleri (Gendreau, Guertin, ve diğerleri 2006) toplama ve dağıtımlı dinamik araç rotalama problemi için tabu arama algoritması

kullanarak çözüm önerisinde bulunmuşlardır. Komşuluk tanımı ekleme sezgiseli kullanılarak yapılmıştır. Var olan bir çözümün komşu çözümleri aranırken, bir rotadaki müşteri diğer rotalardaki her iki şehir arasına sırası ile eklenir ve maliyeti en küçük kılan konumda tutulur. Tabu listesi göz önünde tutularak, yani gidilmemesi gereken konumlar dikkate alınarak, çözümlerin komşu çözümleri oluşturulur ve en iyi komşu çözümün değeri hafızada tutulur. Böylelikle en iyiye ulaşılmaya çalışılmaktadır.

Mitrović-Minić ve diğerleri (Mitrović-Minić, Krishnamurti ve Laporte 2004) ekleme sezgiseli ile başlangıç çözümlerini oluşturduktan sonra çözümü iyileştirmek için tabu arama meta sezgiselini kullanmışlardır.

2007 yılında yapılan çalışmada (Hanshar ve Ombuki-Berman M. 2007), kapasite kısıtlı dinamik araç rotalama problemi tabu arama algoritması ile de çözülerek diğer algoritmalar (Genetik algoritma ve karınca kolonisi algoritması) ile karşılaştırılmıştır. Klasik tabu arama metodolojisi kullanılmıştır, komşuluk tanımı olarak λ-değiştirme (Osman 1993) ve bir rotayı tersine çevirme sezgisel yöntemleri kullanılmıştır. Bu iki yerel arama sezgisellerinden ilki %40 diğeri %60 olasılıkla uygulanmaktadır.

Kergosien ve diğerleri (Kergosien, ve diğerleri 2011) Fransa'da bulunan bir hastaneye ait hastaların taşınması problemini ele almışlardır. Taleplerin bir kısmı önceden bilinmekle beraber bir kısmı dinamik olarak ortaya çıkmaktadır. Her bir talep belirli bir zamanda ortaya çıkmaktadır ve her bir araç tek seferde bir hasta taşıyabilmektedir. Problemin çözümü içn tabu arama algoritması öneren yazarlar, bir gerçek hayat problemini çözmüşlerdir. Ek olarak rasgele üretilen problemler üzerinde de algoritmayı test etmişlerdir.

Değişken Komşuluk Arama (Variable Neighborhood Search-VNS): 1997 yılında (Mladenović ve Hansen 1997) literatüre kazandırılan değişken komşuluk arama algoritmasının temeli komşuluk kavramlarına dayanmaktadır. Bir başlangıç çözümünün, önceden tanımlanmış komşuluk kuralına göre komşuları belirlenir ve en iyi çözüm değerini veren komşu noktaya doğru ilerlenir.

Öncelikle en az 3 adet komşuluk tanımı yapılır. Başlangıç noktasının, ilk olarak en yakın komşuluğu içerisinde daha iyi bir nokta olup olmadığı araştırılır. Eğer var ise o noktaya geçilir ve onun komşulukları belirlenir. Eğer en yakın komşuluk içerisinde başlangıç çözümünden daha iyi bir nokta yoksa sırası ile diğer komşuluklar aranır. Hiçbir komşuluk içerisinde başlangıç noktasından iyi bir nokta bulunmadığı takdirde, rassal olarak yeni komşuluklar üretilir.

Değişken komşuluk arama algoritmasın kullanıldığı çalışmada (Azi, Gendreau ve Potvin 2012) kapasite kısıtlı dinamik araç rotlama problemi ele alınmıştır. Problemin statik versiyonu için daha önceden önerilmiş uyarlanabilir geniş komşuluk arama sezgisel algoritmasından esinlenilmiştir.

2012 yılında yapılan bir diğer çalışmada (Khouadjia, ve diğerleri 2012) değişken komşuluk arama algoritması kapasite kısıtlı dinamik araç rotalama problemi için ele alınmıştır. Literatürde çalışılan test problemleri çözülerek karşılaştırmalı sonuçlara yer verilmiştir.

Schilde ve diğerleri (Schilde, Doerner ve Hartl 2014) dinamik araç rotalama problemi için geçmiş verileri kullanarak bir sonraki dönemlerdeki talepleri tahmin etmişlerdir. Bu nedenle ele alınan problem stokastik bir yapıya sahiptir. İkili bir yapıya sahip meta sezgisel bir yöntem önermişlerdir. Deterministik olarak ulaşım sürelerinin ortalamalarını kullanmışlardır, planlama esnasında da stokastik olarak tahmin verilerini göz önünde bulundurmuşlardır. 762 farklı noktadan gelen talepler içeren bir gerçek hayat probleminin çözümünde önerilen yöntemi test etmişlerdir.

Hırslı Rassal Uyarlamalı Arama Prosedürü (Greedy Randomized Adaptive Search Procedure-GRASP): GRASP algoritması 1989 yılında (Feo ve Bard 1989) tarafından literatüre kazandırılmıştır. Araç rotalama problemi üzerinden örnek vermek gerekirse, rassal çalışan bir algoritmada bir sonraki gidilecek şehir rassal olarak belirlenmektedir. Böyle bir süreçte çeşitliliği sağlamak kolaydır ancak çözüm kalitesini artırmak kolay değildir. Öte yandan Greedy (Hırslı) bir algoritmada bir sonraki gidilecek şehir en son gidilmiş şehre en yakını olarak veya getirisi en yüksek olan şehir olarak belirlenir. Bu noktada da çözüm kalitesi yüksek olmakla beraber, ileri görüşlülük açısından çözüm çeşitliliği daha düşük olur. Anlık olarak en

yakına gidilmesi kimi zaman toplam tur için optimal olmayabilir. GRASP algoritması Hırslı ve Rassal stratejilerinin ikisinin de birlikte kullanıldığı bir algoritmadır.

Hırslı bir algoritma kullanılarak bir küme oluşturulur ve bu kümeden rassal olarak seçilen bir noktanın etrafı yerel arama teknikleri ile araştırılır. Oluşturulacak kümenin eleman sayısı genellikle 3 ile 5 arasında olmaktadır. Eğer kümenin eleman sayısı yeteri kadar büyük seçilirse prosedür Rassal bir algoritmaya dönüşür, 1 olarak seçilirse yalnızca Hırslı bir algoritma gibi davranır.

2005 yılında yapılan çalışmada (Montemanni, ve diğerleri 2005) dinamik araç rotalama problemi için GRASP algoritması ile çözüm üretilmiştir. Aynı çalışmada karınca kolonisi algoritması ile de aynı test problemleri çözülerek sonuçlar karşılaştırılmıştır.

3.3.2 Popülasyon Temelli Meta Sezgiseller

Tek bir çözüm ile çözüm uzayını gezmek yerine çözümlerin topluluğu ile çözüm uzayının gezilmesi daha etkin bir aramaya imkan vermektedir. Aynı zamanda yerel en iyiye takılma probleminde daha başarılı olunmaktadır. Dinamik araç rotalama problemi için geliştirilen popülasyon temelli meta sezgisel algoritmalara bu bölümde değinilecektir.

Karınca Kolonisi Optimizasyonu (Ant Colony Optimization-ACO): Doğal yaşamdan esinlenerek, doğadaki olayların modellenmesine bir örnek olan karınca kolonisi optimizasyonu gerçek karıncaların davranışlarını örnek almaktadır. Doğada, gerçek karıncalar besinlere ulaşmak için en kısa yolu kullanmakta başarılı olmaktadırlar. Ek olarak çevresel faktörlerden dolayı en kısa yol değiştiği takdirde bu duruma da adaptasyon söz konusudur.

Kimyasal bir haberleşme ile en kısa yolu bulan karıncalar, geçtikleri yerlerde *feromen* adı verilen kimyasal bir madde bırakmaktadırlar. Böylece feromen yoğunluğu hangi yolda daha fazla ise diğer karıncalar o yolu tercih etmektedirler. Eğer tüm alternatiflerin feromen miktarı eşit ise, tüm yollar eşit sıklıkta kullanılmış anlamına gelmektedir.

Karınca kolonisi algoritması ise bu doğal sürecin optimizasyonda kullanılması fikrine dayanmaktadır. İlk olarak (Colorni, Dorigo ve Maniezzo 1991) tarafından literatüre kazandırılmıştır.

Daha sonra çeşitli optimizasyon problemlerinin çözümünde kullanılan ACO dinamik araç rotalama problemleri için de iyi sonuçlar üretilmesinde kullanılmıştır.

DARP için ilk çalışmalardan biri Tian ve diğerleri (2003) tarafından yapılmıştır. Feromen matrisini dinamik olarak güncelleyerek dinamizmin takip edildiği hibrit karınca kolonisi algoritması önerilmiştir. Bir önceki iterasyondaki problem bilgileri kullanılarak matris güncellenmektedir. Başlangıç feromen matrisinin oluşturulmasında yeni bir yaklaşım önermişlerdir. Ayrıca her bir zaman diliminde ortaya çıkan yeni talepleri gruplayarak sisteme tanıtmışlar ve elde edilen rotaların iyileştirilmesi için 2-opt sezgisel yöntem kullanmışlardır.

Montemanni ve diğerleri (2005) tarafından strateji temelli ACO algoritması dinamik araç rotalama probleminin çözümü için önerilmiştir. Feromen koruma mantığına dayanan önerilen algoritmada, iyi çözümlerin bilgileri saklanmaktadır. İyi çözümlere feromen matrisinde yüksek değerler atanmaktadır. Çalışmanın en önemli özelliği dinamik bir problemin, zaman dilimleri bazında değerlendirilebileceğini önermiş olmasıdır. Her bir zaman diliminde ortaya çıkan talepler statik olarak ACO algoritması ile çözülmektedir. Bu yaklaşım daha sonraki çalışmalarda referans olarak kullanılmaktadır.

Montemanni ve diğerleri (2005) tarafından önerilen algoritma Rizzoli ve diğerleri (2007) tarafından kullanılarak bir gerçek hayat araç rotalama problemine uygulanmıştır. İsviçre'de yakıt dağıtımı yapan araçların çevrimiçi olarak rotalanması problemi bir zaman pencereli dinamik araç rotalama problemi uygulamasıdır.

(Jun, Wang ve Zheng 2008), zaman pencereli dinamik araç rotalama probleminin çözümü için çok amaçlı hibrit karınca kolonisi algoritması önermişlerdir. Araç sayısını ve zaman maliyetini birbirinden bağımsız iki amaç olarak düşünüp minimize etmişlerdir. Önerilen ACO algoritmasının feromen matrisinin güncellenmesinde evrimsel algoritma kullanılmıştır. Böylelikle daha hızlı en iyiye yakınsayan bir algoritma elde edilmiştir. Bu nedenle hibrit olarak değerlendirilmiştir.

(Dan, ve diğerleri 2013), çalışmada acil bir durumda ihtiyaç malzemelerinin taşımacılığı problemine değinilmiştir. Örneğin bir doğal afet, yangın gibi durumlarda ihtiyaç malzemelerinin dağıtımı problemi dinamik araç rotalama problemi uygulaması olarak düşünülmüştür. Problemi, zaman ilerledikçe ortaya çıkan statik problemlerin serisi olarak modellemişlerdir. Çok amaçlı bir optimizasyon problemi olarak düşünüp ACO algoritması ile çözüm önerisinde bulunmuşlardır. Önerilen matematiksel modelin ve çözüm algoritmasının geçerliliğini ve uygunluğunu sayısal bir örnekle test etmişlerdir.

Elhassania ve diğerleri (2013), ACO ile Geniş Komşuluk Arama sezgisel yöntemini hibrit olarak ele almışlar ve DARP için çözüm önerisinde bulunmuşlardır. 22 farklı test problemi üzerinde çalışılmıştır. 385 adet müşteri sayısına sahip büyük boyutlu test problemleri çözülmüştür. Önerilen algoritmanın başarılı sonuçlar elde ettiği, daha önce aynı test problemlerini çözen algoritmaların sonuçları ile karşılaştırmalı olarak gösterilmiştir.

Genetik Algoritma (Genetic Algorithm-GA): Genetik algoritmalar, doğada gözlemlenen doğal ayıklanma ve en iyinin hayatta kalma prensibini model almaktadır. Bir diğer deyişle türlerin bozulmadan sabit kalmalarını sağlayan bir mekanizma olarak tanımlanabilir. İlk olarak 1975 yılında (Holland 1975) yayımlanan kitapta temel prensipleri tanıtılmıştır. Genetik algoritmalar popülasyon temelli bir arama algoritmasıdır.

Problem için olası pek çok çözümü temsil eden popülasyonda bulunan her bir vektör, kromozom veya birey adı verilen sayı dizilerinden oluşur. Birey içindeki her bir elemana gen adı verilir. Popülasyondaki bireyler evrimsel süreç içinde genetik algoritma işlemcileri tarafından belirlenirler.

Genetik algoritmalardaki en önemli iki operatör, mutasyon ve çaprazlama operatörleridir. Bu operatörlerin uygulanmasıyla yeni bir popülasyon üretilir ve popülasyondaki bireylerin uygunluk değerleri hesaplanır. Uygunluk değerleri, uygunluk fonksiyonu yardımı ile belirlenir. Bu fonksiyon bir çözümün kalitesini ölçmek için kullanılır. Eğer üretilen yeni popülasyon bu anlamda kabul edilir ise, bir önceki popülasyon ile yer değiştirilir. Çaprazlama ve mutasyon operatörleri, biyolojide

genlerin çeşitliliğinden esinlenerek modellenmektedir. Bu nedenle yeni nesillerin üremesinde (yeni çözümlerin üretilmesinde) çeşitliliği sağlamaktadırlar.

Genetik algoritma önerildiği ilk yıllardan günümüze kadar çok çeşitli optimizasyon problemlerinde başarılı sonuçlar elde edilmesine imkan vermektedir. Hatta meta sezgisel algoritmalar içerisinde başarılı sonuçlar elde edilmesinde ilk sıralarda yer almaktadır, bir çok algoritmaya göre çalışma prensibi çözüm arama anlamında üstün olmaktadır.

(Jih ve Hsu 1999), tek araçlı zaman pencereli toplama dağıtımlı dinamik araç rotalama problemi için hibrit bir genetik algoritma önermişleridir. Dinamik programlama ve genetik algoritmanın beraber kullanıldığı algoritmada, başlangıç popülasyonu dinamik programlama ile üretilmiştir. Genetik algoritma daha sonra çözümleri geliştirerek son haline getirmek için kullanılmaktadır. Genetik algoritmanın çaprazlama operatörü için 4 farklı yöntem karşılaştırılmıştır. Ek olarak mutasyon operatörü için ise 3 farklı prosedür karşılatırılmıştır.

(Taniguchi ve Shimamoto 2004), çalışmalarında ulaşım sürelerinin gerçek zamanlı bildirildiği dinamik araç rotalama problemine genetik algoritma ile çözüm önerisinde bulunulmuştur. Dinamik olarak trafik simülasyonu kullanılmıştır ve böylelikle ulaşım süreleri güncellenmiştir.

(Hanshar ve Ombuki-Berman M. 2007), statik olarak alt dönemlerde ele alınan dinamik araç rotalama problemlerinin çözümünde genetik algoritma kullanılmıştır. Popülasyondaki her bir bireyin uygunluk değeri temsil ettiği çözümdeki rota uzunluğunu yani maliyeti olarak belirlenmiştir. Mutasyon operatörü olarak, bir rotanın içerisindeki bir kısmı tersine çevirme işlemi (*inversion operatör*), çaprazlama operatörü olarak ise en düşük maliyeti hesaplayan *"Best-Cost Route"* çaprazlama operatörü kullanılmıştır. Bu operatöre göre, popülasyona ait iki birey seçiliyor. Her bir birey bir araç rotasını temsil etmektedir. İki bireyden rasgele şehirler seçiliyor, daha sonra bireylerden birinden seçilen şehirler diğerinin rotasından çıkarılıyor. Ve çıkarılan rotada maliyet göz önünde bulundurularak daha uygun bir konuma ekleniyor. Bu çaprazlama ile kapasite kısıtı hiçbir şekilde göz ardı edilmemiş olmaktadır.

3.4 Performans Ölçümü

Bir problemin çözümünde kullanılacak farklı algoritmaların başarılarının karşılaştırılması önemli bir araştırma konusudur. Problemin çözümü için elde edilen değerleri karşılaştırmak statik bir problem için yeterli olsa da, dinamik optimizasyonda yeterli olmamaktadır. Dinamik optimizasyonda, algoritmanın çözüm uzayının değişimlerine hızlı bir şekilde adapte olması önemlidir. Bu bölümde, dinamik optimizasyon problemlerinin çözümünde kullanılan algoritmaların performansını ölçmek için yaygın olarak kullanılan metriklere değinilecektir.

Çevrimdışı (Offline) Performans Ölçüsü: Çevrimdışı performans ölçüsü (Branke 2012) tarafından tanıtılmıştır. Bir dinamik problemde çözüm uzayı belirli dönemlerde, dinamik bir bilgi değişikliği ile değişmektedir. Eğer problemin yapısı *T* defa değişiyor ise, *T* farklı periyotta algoritma birden fazla kez koşturulmaktadır. Buna göre, bu periyotların kapsadığı her bir zaman aralığında bulunan en iyi değer f_i olsun. Çevrimdışı performans ölçüsü, periyotlarda bulunan en iyi değerlerin ortalaması olarak eşitlik (3-1) ile verilmektedir.

$$x^* = \frac{1}{T}\sum_{i=1}^{T} f_i \qquad (3\text{-}1)$$

Çevrimdışı performans ölçüsünün hesaplanabilmesi için, problemde gerçekleşecek dinamik değişikliğin ne zaman olacağının önceden bilinmesi ve periyotların hesaplanabilmesi gerekmektedir. Ayrıca normalize edilmemiş bir değer olduğundan yanlılık içermektedir.

En İyiyi Türetme (Best-Of-Generation) Performans Ölçüsü: Bir optimizasyon algoritması bir problem için bir den fazla kez koşturulabilir (çalıştırılabilir). Her bir çalıştırma için de iterasyon (nesil veya *generation*) sayısı belirlenmektedir. Eğer *N* defa koşturulan bir algoritmanın her bir koşumunda *G* defa algoritma çalıştırılıyorsa *Best-of-Generation* metriği (Morrison 2003)' de önerildiği haliyle (3-2) eşitliği ile verilmektedir.

$$\bar{F}_{BOG} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{G}\sum_{j=1}^{G} F_{BOG_{ij}}\right) \qquad (3\text{-}2)$$

Burada $F_{BOG_{ij}}$değeri algoritmanın her bir çalıştırılmasındaki iterasyonlar sonucunda bulunan en iyi değerlerdir. Yani her bir iterasyonda çözüm değeri olarak kabul edilen değerlerdir.

Kesinlik (Accuracy), Kararlılık (Stability), Tepkisellik (Reactivity): Birbirleri ile ilişkili olarak değerlendirilen kesinlik, kararlılık ve tepkisellik metrikleri Weicker (2002) tarafından tanıtılmıştır.

Kesinlik (*Accuracy*), belirli bir anda bulunan en iyi amaç fonksiyonu değerinin bilinen en iyi (optimal) değere ne kadar yakın olduğunun ölçüsüdür. Bir maksimizasyon problemi için, *t* anında, *A* algoritmasının kesinlik değeri *F* amaç fonksiyonu olmak üzere eşitlik (3-3) ile verilmektedir.

$$accuracy_{F,A}^{(t)} = \frac{F\left(best_A^{(t)}\right) - Min_F^{(t)}}{Max_F^{(t)} - Min_F^{(t)}} \qquad (3\text{-}3)$$

Burada $best_A^{(t)}$, *t* anında algoritma tarafından bulunan en iyi nokta, $F\left(best_A^{(t)}\right)$ ise amaç fonksiyonu değeridir. Çözüm uzayında yer alan maksimum ve minimum amaç fonksiyonu değerleri ise sırasıyla $Max_F^{(t)}$ ve $Min_F^{(t)}$ile gösterilmektedir. Bir algoritmanın kesinlik değeri [0,1] aralığında değer almaktadır. Kesinlik değeri 1'e yaklaştıkça algoritmanın başarısı yüksek anlamına gelmektedir.

Kesinlik değerine bağlı olarak hesaplanan kararlılık (*stability*) metriği, dinamik bir ortamda gerçekleşen değişikliğin algoritmanın kesinlik değerini ne derece etkilediğini ölçmektedir. Başka bir ifadeyle, çözüm uzayında en iyi nokta yer değiştirdikçe optimizasyon algoritmasının bu duruma kendini ne derece adapte edebildiğinin ölçüsüdür. Eşitlik (3-4) ile hesaplanmaktadır.

$$stability_{F,A}^{(t)} = \max\{0, accuracy_{F,A}^{(t)} - accuracy_{F,A}^{(t-1)} \qquad (3\text{-}4)$$

[0,1] aralığında değer alan kararlılık ölçüsünün 0'a yakın olması algoritmanın kararlı olduğunu göstermektedir.

Son olarak tepkisellik (*reactivity*) ölçüsü, çözüm uzayında gerçekleşen bir değişikliğe algoritmanın tepkisini ölçmektedir. Eşitlik (3-5) ile ifade edilmektedir.

$$react_{F,A,\varepsilon}^{(t)} = \min\{t' - t \mid t < t' \leq maxgen, t \in \mathbb{N}, \frac{accuracy_{F,A}^{(t')}}{accuracy_{F,A}^{(t)}} \geq (1-\varepsilon)\} \cup \{(maxgen - t)\} \quad (3\text{-}5)$$

Burada *maxgen*, toplamda yapılan iterasyon sayısını göstermektedir. Tepkisellik ölçüsünün değeri ne kadar küçük çıkarsa algoritmanın performansı o derece iyi olarak değerlendirilir.

DÖRDÜNCÜ BÖLÜM

4 PARÇACIK SÜRÜ OPTİMİZASYONU

4.1 Giriş

Bu bölümde NP-zor sınıfına ait optimizasyon problemlerine çözüm önerisi getirmek amacıyla, literatürde sıkça kullanılan meta sezgisellerden olan Parçacık sürü optimizasyonu algoritmasına değinilecektir. PSO algoritmasının birkaç farklı formülasyonu bulunmaktadır. Bu farklı formülasyonlardan kısaca bahsedildikten sonra çalışmamızda kullanılan formülasyon detayları ile anlatılacaktır. Daha sonra algoritmamızın son basamağı olan, çözümü iyileştirmek adına başvurduğumuz yerel arama metotları detaylı bir şekilde verilecektir.

4.2 PSO Algoritması

PSO, 1995 yılında Kennedy ve Eberhart (Kennedy ve Eberhart 1995) tarafından önerilen popülasyon temelli meta sezgisel bir optimizasyon tekniğidir. Temel prensip olarak kuş ve/veya balık sürülerinin sosyal davranışlarına dayanmaktadır. Kuş, balık ve hayvan sürülerinin bir "bilgi paylaşma" yaklaşımı uygulayarak çevrelerine adapte olabilme, zengin yiyecek kaynağı bulabilme ve avcılardan kaçabilme yeteneklerinden esinlenmiştir. Bu davranış biçimine sürü zekası (swarm intelligence) ismi verilmektedir.

Sürü zekası belirli bir algoritma veya bir sistem değildir. Sürü zekası doğal veya yapay dağıtılmış, kendi kendine organize sistemlerin, kolektif bir davranış biçimidir. Sürü zekasını baz alarak işleyen algoritmalara verilebilecek en bilinen örnekler karınca kolonisi algoritması ve parçacık sürü optimizasyonu algoritmasıdır. Diğer evrimsel algoritmalar ile karşılaştırıldığında bu algoritmalarda, birbirlerinin davranışlarından etkilenen sürü elemanlarının, bireysel hareket edenlere nazaran çözüm uzayına daha uygun bir şekilde yayıldığı görülmüştür. Bu durum dinamik olarak değişen çözüm uzaylarındaki değişimin daha rahat takip edilebilmesine ve adaptasyonun daha hızlı olmasına kolaylık sağlamaktadır. (Yu & Gen, 2010: 328).

PSO algoritmasında, genetik algoritmaya benzer şekilde çözüm uzayındaki aramalar popülasyon temelli yapılmaktadır, popülasyona sürü, sürü elemanlarının her

birine de parçacık ismi verilmektedir. Her bir parçacık aslında probleme ait bir çözümü (çözüm önerisini) temsil etmektedir. Başlangıç popülasyonu rasgele seçilen parçacıklardan oluşmaktadır ve genetik algoritmalardan farklı olarak, her bir parçacığın kendine ait hız ve yön bilgileri bulunmaktadır. Yine bu hız ve yön bilgileri başlangıç popülasyonu üretilirken her bir parçacık için rasgele üretilmektedir.

Her bir parçacık kendi geçmiş pozisyonlarından en iyi olanına dair bilgileri hafızasında tutar. Bu pozisyondaki amaç fonksiyonu (uygunluk değeri) değerine kişisel en iyi ($pbest_i$) denir. Sürü elemanlarının en iyi pozisyonuna ise global en iyi ($gbest_i$) adı verilmektedir. Sürüye ait belirli bir alt küme için de en iyi pozisyon bilgisi saklanabilir buna da yerel en iyi ($lbest_i$) adı verilir. Çözüm uzayında gezinirken her bir ilerleme aşamasında parçacıklar kişisel en iyi ve global en iyi pozisyonları gözeterek hızlarını ve yönlerini tayin ederler. Bu bağlamda her bir parçacığın komşuluğunun tanımlanması gerekmektedir. Çünkü bir sonraki pozisyona yönelirken parçacığın kendi pozisyonu ve komşusunun en iyi pozisyonu arasındaki fark dikkate alınacaktır. Çözüm uzayındaki hızının ve yönünün her seferinde nasıl değişeceği, komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının bir birleşimi olacaktır. Literatürde sıkça kullanılan iki komşuluk kavramı bulunmaktadır:

- *gbest* yöntemi: Sürü elemanlarının tümünü komşu olarak düşünmek, ve parçacığın bir sonraki hızını hesaplarken global en iyi pozisyonu dikkate almak.
- *lbest* yöntemi: Her bir parçacık için sürünün bir alt kümesini komşu varsaymak ve parçacığın bir sonraki hızını hesaplarken bu alt kümenin en iyi pozisyonu olan yerel en iyiyi dikkate almak.

Karar verici tarafından önceden belirlenmesi gereken bazı parametreler bulunmaktadır. Bunlar sürü yani popülasyonun kaç parçacıktan oluşacağı, her bir parçacığın hızının maksimum ne kadar olabileceği şeklindedir. Algoritma tasarlanırken bu bilgiler dikkate alınmaktadır.

Örneğin, D tane bilinmeyen içeren bir optimizasyon problemini ele alalım. Bu durumda her bir parçacığın bir çözümü temsil edebilmesi için D boyutlu bir vektör ile gösterilmesi gerekmektedir. Popülasyonda n adet parçacık olduğunu varsayarsak her

bir parçacığın pozisyon vektörlerinden oluşan pozisyon matrisi aşağıdaki gibi olacaktır.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \dots & \dots & & \dots \\ x_{i1} & x_{i2} & \dots & x_{iD} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & & x_{nD} \end{bmatrix}$$

Pozisyon matrisindeki *i.* satır *i.* parçacığın pozisyonunu temsil eder ve $X_i = [x_{i1}, x_{i2}, \dots, x_{iD}]$ olarak ifade edilir. Aynı parçacığın o anki en iyi uygunluk değerini elde ettiği kişisel en iyi pozisyonu $pbest_i = [p_{i1}, p_{i2}, \dots, p_{iD}]$ şeklinde gösterilir. Popülasyonun, yani sürünün en iyi pozisyonu olan global en iyi $gbest = [g_1, g_2, \dots, g_D]$ ile temsil edilir. Eğer bir alt küme üzerinde komşuluk tanımlanacaksa, formülasyonlardan $gbest$ yerine yerel en iyiyi temsil eden $lbest = [l_1, l_2, \dots, l_D]$ vektörü kullanılacaktır. Her bir iterasyonda tüm parçacıkların pozisyonu hız vektörüne göre güncellenir, *i.* parçacığın pozisyonundaki değişim miktarını veren hız vektörü ise $V_i = [v_{i1}, v_{i2}, \dots, v_{iD}]$ olarak ifade edilir.

Daha önce de bahsedildiği gibi ilk iterasyon için tüm vektörler rassal olarak üretilir, daha sonra her bir iterasyonda belirli formüllere göre güncellemeler yapılmaktadır. Kennedy ve Eberhart (Kennedy ve Eberhart 1995) ın önermiş olduğu ilk formülasyona göre *i.* parçacığın *k.* iterasyonda hız vektörünün *j.* elemanı denklem (4-1) şeklinde güncellenmektedir:

$$v_{ij}^{k+1} = v_{ij}^{k} + c_1 rand_1\left(p_{ij}^{k} - x_{ij}^{k}\right) + c_2 rand_2(g_j^{k} - x_{ij}^{k}) \qquad (4\text{-}1)$$

Burada, $rand_1$ ve $rand_2$ [0,1] aralığında düzgün dağılıma ait rasgele sayılardır ve c_1 ile c_2 hızlandırma sabiti olarak adlandırılan pozitif iki sabit sayıdır[5]. c_1, parçacığın kendi tecrübelerine göre hareket etmesini, c_2 ise sürüdeki diğer parçacıkların tecrübelerine göre hareket etmesini sağlar.

Hız vektörünün güncellenmesinden sonra *i.* parçacığın pozisyon vektörünün elamanları denklem (4-2) 'ye göre güncellenir:

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \qquad (4\text{-}2)$$

[5] Kennedy ve Eberhart $c_1=c_2=2$ olarak seçmeyi önermişlerdir.

1999 yılında Shi ve Eberhart parçacığın hızının güncellenmesinde kullanılan formüle bir bilgi daha eklemişlerdir ve yeni formülasyon denklem (4-3) ile gösterilen şekilde olmuştur (Shi ve Eberhart 1999):

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 rand_1\left(p_{ij}^{k} - x_{ij}^{k}\right) + c_2 rand_2\left(g_j^{k} - x_{ij}^{k}\right) \qquad (4\text{-}3)$$

Burada $0 < w < 1$ olmak koşulu ile atalet ağırlığıdır.[6] Düşük atalet ağırlığı değerleri yerel aramayı, yüksek atalet ağırlığı değerleri ise global aramayı kolaylaştırmaktadır. Parçacığın hızının belirlenmesindeki ilk terim mevcut hareketinin etkisidir, ikinci terim kendi belleğinin etkisi, son terim ise sürünün belleğinin etkisidir.

Özetlemek gerekirse; her bir parçacık rassal olarak çözüm uzayına dağıtılmaktadır ve bulundukları pozisyonların uygunluk değerleri amaç fonksiyonu gözetilerek hesaplanmaktadır. Tüm parçacıkların bir sonraki pozisyonlarını belirlemek adına hız vektörleri hesaplanır. Hız vektörleri hesaplanırken; kişisel en iyi pozisyonları ile o anki pozisyonları arasındaki farkın ve global en iyi ile o anki pozisyonları arasındaki farkın (veya yerel en iyi ile o anki pozisyonları arasındaki farkın) bir önceki hız vektörü ile doğrusal kombinasyonu alınır. Anlık pozisyon vektörü ile hız vektörü toplanarak yeni pozisyonlar üretilir ve algoritma bitirme koşulları sağlanana kadar bu döngü devam ettirilir.

Sürü elemanları birbirine yaklaşmaya başladığında yani $pbest_i$ ve *gbest* ikisi birden parçacıkların o anki pozisyonlarına yaklaştığında, hız vektörü küçülmeye başlar. Yani bir sonraki pozisyona geçiş adımı küçülür. Sonuç olarak PSO nun kendi kendine bir kontrol mekanizması olduğunu söyleyebiliriz (Yu & Gen, 2010:341).

Hız vektörünün boyunun çok büyük olması ise parçacıkların çok büyük adımlarla ilerlemesine sebep olmaktadır ve sürü çok yayılabilir. Bu yüzden hız vektörü bileşenleri için bir alt ve üst sınır oluşturmak gerekir. Genellikle hız vektörü bileşenleri $[-v_{max}, v_{max}]$ şeklinde gösterilen bir aralık içerisinde tutulur (Yu & Gen, 2010:342).

[6] Shi ve Eberhart atalet ağırlığının algoritmanın işleyişi süresince 0,9 dan başlayıp 0,4 'e kadar doğrusal olarak azalmasını önermektedir.

PSO algoritmasının en önemli özelliklerinden biri hızlı yakınsaması ve kolay uygulanabilir olmasıdır. PSO algoritmasının özet bir şekilde işleyiş adımları Algoritma 1 ile verilmektedir.

Algoritma 1: PSO algoritmasının genel işleyişi

```
PSO parametrelerinin belirlenmesi;
For  Her Bir Parçacık İçin{
        Pozisyonlarını ve hızlarını belirle
        }
        End For
Do   {
        For Her Bir Parçacık İçin{
                Uygunluk değerini hesapla;
                Kişisel en iyi ve global en iyi değerlerini güncelle;
        }
        End For
        For Her Bir Parçacık İçin{
                Parçacığın hızını hesapla
                Parçacığın pozisyonunu güncelle
        }
        End For
}
While {
          Maksimum iterasyon sayısına veya minimum hata koşulu sağlanana kadar
          devam et
}
```

4.2.1 PSO Parametreleri

Bu bölümde PSO algoritması işleyişinde kullanılan parametrelere değinilecektir.

- **Sürü Büyüklüğü: S**ürü büyüklüğü genellikle ilgilenilen problem türüne göre seçilebilir. Genellikle bir sürüdeki parçacıkların sayısı 20 ile 50 arasında olabilmektedir. Bu konu ile ilgili geçmiş çalışmalar incelendiğinde bu sınırlar arasındaki değerlerin kullanımı daha yaygındır. Sürü elemanları ne kadar fazla ise çözüm uzayı o kadar farklı koldan taranmaktadır. Ancak bu durum, her bir parçacık için hesaplamalar yapıldığından algoritmanın işleyiş süresini uzatmaktadır. Bu sebeple sürü büyüklüğünü çok tutmak yerine optimal bir değer seçmek yerinde bir karardır.
- **Hız Sınırları:** Her bir parçacığın bir sonraki konumunun belirlenmesinde kullanılacak olan hız vektörü olduğundan daha önce bahsetmiştik. Hız vektörünün her

bir bileşeni için $[-v_{max}, v_{max}]$ şeklinde bir sınırlama getirilmesi gerekmektedir. Eğer hız vektörünün güncellenmesi esnasında, vektörün bileşenlerinden her hangi bir tanesi üst sınır değerini aşıyor ise o bileşen v_{max} olarak atanır veya alt sınırın altında kalıyor ise $-v_{max}$ olarak atanır.

v_{max}, algoritmanın işleyişinde önemli bir rol oynamaktadır ve belirlenmesi önemli bir sorumluluk gerektirmektedir. Belirlenen bu sınırlar, her hangi bir iterasyonda bir parçacığın hızının hangi sınırlar dahilinde olabileceğinin kontrol edilmesini sağlamaktadır. Bu da aslında çözüm uzayının parçacıklar tarafından gezilmesi esnasında, elde edilecek sonuçların birbirleri ile göreli olarak mesafelerinin bir parametre ile kontrol edilmesidir.

Hız vektörü bir sonraki konumun belirlenmesinde kullanılır. Çok büyük hız vektörü adımları, parçacığın iki konumu arasındaki mesafeyi artırmaktadır bu da arada daha iyi bir çözüm eğer varsa onun gözden kaçırılmasına sebep olur. Öte yandan gereğinden daha yavaş ilerleyen bir parçacık yerel bir noktanın etrafından çok uzun zamanda uzaklaşamaz. Dolayısıyla iterasyon adımları süresince hep birbirine yakın çözüm değerleri elde etmiş olur ve algoritmanın etkinliğini azaltır. İki durumu da dengeleyici bir şekilde v_{max} değerinin belirlenmesi bu sebeple önemlidir (Eberhart ve Shi 2001).

$V_i = [v_{i1}, v_{i2}, \dots, v_{iD}]$, i. parçacığın hız vektörünü temsil etmektedir, her hangi bir iterasyonda vektör elemanı değerleri hesaplanır ve,

eğer $v_{ij} > v_{max}$ ise $v_{ij} = v_{max}$

eğer $v_{ij} < -v_{max}$ ise $v_{ij} = -v_{max}$

kuralına göre atanır.

- **Hızlandırma (Öğrenme) Sabitleri:** Parçacığın hız vektörünün güncellenmesinde kullanılan, c_1 ve c_2 ile gösterilen katsayılara hızlandırma veya öğrenme sabitleri adı verilmektedir. Parçacığın kişisel en iyi pozisyonu (p_{best})ve global en iyi pozisyonların (g_{best}) hız vektörüne etkisini gösterdiğinden öğrenme sabitleri adı ile kullanmayı uygun görmekteyiz.

PSO algoritmasının davranışı öğrenme faktörlerinin değerlerinden hızlı bir şekilde etkilenmektedir (Poli, Kennedy ve Blackwell 2007). Hız vektörünün güncellendiği eşitliği vektörel olarak yazacak olursak;

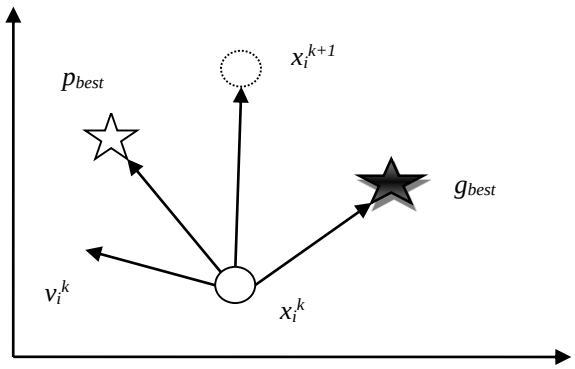
$$\mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + c_1 rand_1\left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2 rand_2\left(\mathbf{g}^k - \mathbf{x}_i^k\right) \quad (4\text{-}4)$$

şeklindedir.

Vektörel formülasyondan açıkça görülmektedir ki, hız vektörü üç farklı vektörün vektörel toplamına eşittir. Öğrenme faktörleri seçilirken, parçacığın en iyi pozisyonuna uzaklığı ile kendi en iyi pozisyonuna uzaklığına ne kadar ağırlık verileceği belirlenir. Bu noktada parçacığın sürü elemanlarının ve kendi geçmiş bilgilerini de kullanarak hareket ettiği düşünülmektedir. Öğrenme sabitlerinden c_1, parçacığın kendi geçmişinden öğrendiği bilginin ağırlığı, c_2 ise komşularının geçmişinden öğrendiği bilginin ağırlığını göstermektedir. Genellikle $c_1 = c_2 = 2$ olarak seçilmektedir ve algoritma sonuna kadar sabit tutulmaktadır. (Eberhart ve Shi 2001). Farklı değerlerin seçildiği çalışmalar da mevcuttur.

Düşük öğrenme faktörleri değerleri parçacığın öğrenmesini kısıtlamaktadır. Yani kendi en iyi veya sürünün en iyi konumuna yakınsaması yavaş olacaktır. Yüksek öğrenme faktörleri ise bu yakınsamayı hızlandıracağından, yerel en iyiye takılmaya sebep olacaktır. Çünkü bir iterasyon sonra global iyi güncellenebilir ve parçacığın bir önceki hareketi onu yanıltmış olabilir.

Bir parçacığın hareketinin görsel olarak ifade edilmesi bu noktada daha açıklayıcı olabilir. Bu sebeple Şekil 4-1 açıklayıcı olmaktadır. Görüldüğü gibi, parçacığın bir sonraki hız vektörü üç vektörün doğrusal kombinezonu olarak hesaplanmaktadır. Bunlardan ilki parçacığın anlık hız vektörü V_i^k'dir. Diğer ikisi ise, parçacığın geçmişte bulunduğu en iyi pozisyonu ile sürünün en iyisine yönlenmiş vektörlerdir.



Şekil 4-1 Parçacığın hareketinin görselleştirilmesi

- **Atalet Ağırlığı:** Bir parçacığın hız vektörünün elemanlarının v_{max} üst sınırı ile sınırlandırıldığından bahsedilmişti. Bu üst sınır global olarak en iyi noktanın aranmasında bir sınırlama getirmektedir. Böylelikle, global olarak çok fazla yayılmayı önlerken yerel olarak da bir nokta etrafında toplanmayı engellemektedir. Atalet ağırlığı ise, global olarak yayılmanın ve yerel olarak bir noktada toplanmanın kontrolünün daha iyi yapılabileceği mantığına dayanarak Shi ve Eberhart tarafından (Shi ve Eberhart 1998a) (Shi ve Eberhart 1998b)1998 yılında literatüre kazandırılmıştır.

Hız vektörünün güncellenmesinde kullanılan eşitlikte bir iterasyon önceki hız vektörünün katsayısı w, atalet ağırlığıdır. Atalet ağırlığının kullanılması çözüm arama performansını iyi yönde etkilemektedir. Önerildiği ilk çalışmalarda, 0,9'dan başlayarak algoritmanın ileriki iterasyonları ile 0,4'e doğrusal olarak azalan değerler almaktadır. Atalet ağırlığının uygun seçilmiş değerleri yerel ve global aramaların arasında denge oluşturmaktadır ve optimale ulaşmak için yapılacak iterasyonlarda azalmaya sebep olmaktadır.

(4-4) ile verilen eşitliği göz önüne alalım, bir parçacığın hız vektöründeki değişim, $\Delta\mathbf{v_i}$ ile gösterilecek olursa;

$$\begin{aligned}\Delta\mathbf{v_i} &= \mathbf{v}_i^{k+1} - \mathbf{v}_i^k \\ &= w\mathbf{v}_i^k + c_1\mathbf{rand}_1^k .* \left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2\mathbf{rand}_2^k .* \left(\mathbf{g}^k - \mathbf{x}_i^k\right) - \mathbf{v}_i^k \\ &= w\mathbf{v}_i^k + \mathbf{f_i} - \mathbf{v}_i^k \\ &= \mathbf{f_i} - (1-w)\mathbf{v}_i^k\end{aligned}$$

burada $\mathbf{f_i}$ vektörü ile $c_1 rand_1\left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2 rand_2\left(\mathbf{g}^k - \mathbf{x}_i^k\right)$ toplamı gösterilmektedir.

Görüldüğü gibi, w değerlerinin yüksek seçilmesi parçacığın hız vektöründeki değişim miktarını yükseltmektedir. Bu durumda bir sonraki konumu önceki konumundan daha uzakta olacaktır. Böylelikle çözüm uzayının global aranması sağlanmaktadır. Düşük atalet ağırlığı değerleri ise hız vektöründeki değişim miktarını düşük tutar, bu da yerel aramaları desteklemektedir.

Algoritmanın ilk işleyişi sırasında çözüm uzayının büyük adımlarla gezilmesi, sonlara doğru ise detaylı aramaların yapılması önemlidir. Bu sebeple atalet ağırlığının bu şekilde seçilmesinin önemi desteklenmektedir. Öte yandan $w > 1$ olarak seçmek de mümkündür. Ancak bu sürüyü kararsız bir duruma götürebilmektedir (Poli, Kennedy ve Blackwell 2007).

İlk önerildiği çalışmalardan itibaren atalet ağırlığının belirlenmesi ile ilgili farklı stratejiler geliştirilmiştir. Bu konuda Bansal ve diğerleri tarafından yapılan çalışmada (Bansal, ve diğerleri 2011) kronolojik olarak geliştirilen stratejilerden bahsedilmiştir. Aşağıda atalet ağırlığının belirlenmesi ile ilgili literatürde önerilen yöntemlerden bahsedeceğiz..

Eberhart ve Shi (Eberhart ve Shi 2001), atalet ağırlığın rassal olarak belirlenmesini önermişlerdir. Yapılan deneysel çalışmalarda bu durumun, PSO algoritmasının ilk iterasyonlarıyla yakınsama hızını artırdığını gözlemlemişlerdir. Xin ve diğerleri (Xin, Guimin ve Hai 2009) 0,9 ile 0,4 arasında doğrusal olarak azalan atalet ağırlığı değerlerinin daha etkin sonuçlar ürettiğini farklı optimizasyon problemleri üzerinde test ederek doğrulamışlardır.

Arumugam ve Rao (Arumugam ve Rao 2006), atalet ağırlığının parçacıkların her iterasyonundaki yerel ve global en iyi pozisyonlarının bir fonksiyonunu kullanarak belirlemeyi önermişlerdir. Önerdikleri bu yönteme "global-yerel en iyi atalet ağırlığı" ismini vermişlerdir. Bu durumda atalet ağırlığı sabit bir değer değildir, doğrusal olarak azalan bir değer de almamaktadır. Yerel minimuma erken bir şekilde yakınsama probleminin üstesinden gelebilmek için uyarlanabilir atalet ağırlığı stratejisi önerilmiştir (Nikabadi ve Ebadzadeh 2008). Böylelikle çözüm uzayında daha verimli

bir arama elde edilmiştir. Uyarlanabilir özelliği ile sürünün çeşitliliği kontrol edilebilmektedir.

Chen ve diğerleri, (Chen, ve diğerleri 2006) atalet ağırlığı belirlenmesinde iki adet yöntem önermişlerdir. Üstel fonksiyonlar içeren iki eşitlikleri $e_1 - PSO$ ve $e_2 - PSO$ olarak isimlendirmişlerdir. İki yöntemin de temeli azalan atalet ağırlığı belirlemeye dayanmaktadır. Yapılan deneysel çalışmalar sonucunda, doğrusal olan atalet ağırlığı fonksiyonlarına göre ilk iterasyonlarda üstel fonksiyonların daha hızlı yakınsadığı görülmüştür. Bu durum sürekli optimizasyon problemlerinin çözümünde avantaj olarak görülmüştür.

Kaotik atalet ağırlığı önerisi Feng ve diğerleri tarafından yapılmıştır (Feng, ve diğerleri 2007). Kaosun doğrusal olmayan dinamiklerin hesabında, rassallık, ergodiklik ve düzenlilik bileşenlerinden faydalanılmıştır. Rassal olarak hesaplanan atalet ağırlığı $w = 0.5 + 0.5rand()$ formülüne göre hesaplanmaktadır ve çalışmada kaotik parametre eklenilerek kaotik rassal olarak hesaplanan atalet ağırlığı ile sonuçlar karşılaştırılmıştır. Ele alınan test problemleri için kaotik rassal atalet ağırlığı rassal atalet ağırlığına göre daha iyi sonuçlar vermiştir.

Malik ve diğerleri (2007) Sigmoid fonksiyonu kullanarak atalet ağırlığı belirlemeyi önermişlerdir. Çalışmada Sigmoid azalan ve Sigmoid artan fonksiyonları ayrı ayrı kullanılarak iki farklı eşitlik ile atalet ağırlığı belirleme yöntemi kullanılmıştır. Bilindiği gibi Sigmoid fonksiyonu genel olarak $S(x) = \frac{1}{1+e^{-x}}$ formundadır. Bu form atalet ağırlığı hesabı için güncellenmiştir. Detaylı eşitlik Tablo 4-1 de verilmektedir. Çalışmada görülmüştür ki sigmoid fonksiyonu amaç fonksiyonu değerlerinin minimize edilmesinde büyük rol oynamakta ve doğrusal olarak artan atalet ağırlığı ise hızlı yakınsamayı kolaylaştırmakta. Çalışmanın sonuç bölümünde bu iki fonksiyonun birleşiminin daha iyi sonuçlar getireceğinden bahsederek ileriki çalışmalar için yol göstermişlerdir.

Tablo 4-1 Atalet ağırlığı belirleme stratejileri

Strateji ismi	Matematiksel formül
Sabit atalet ağırlığı	$w = c\ ; c = 0{,}7$
Rassal atalet ağırlığı	$w = 0{,}5 + \frac{Rand()}{2}$
Doğrusal olarak azalan	$w_k = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} * k$
Global-yerel en iyi poz. kullanmak	$w_i = (1{,}1 - \frac{gbest_i}{pbest_i})$
Uyarlanabilir atalet ağırlığı	$w_i(t+1) = w(0) + (w(n_t) - w(0))\frac{e^{m_i(t)} - 1}{e^{m_i(t)} + 1}$
Üstel atalet ağırlığı-1 $(e_1 - PSO)$	$w(t) = w_{min} + (w_{max} - w_{min}).e^{-t/(\frac{max_{iter}}{10})}$
Üstel atalet ağırlığı-2 $(e_2 - PSO)$	$w(t) = w_{min} + (w_{max} - w_{min}).e^{-[t/(\frac{max_{iter}}{4})]^2}$
Kaotik rassal atalet ağırlığı	$z = 4z(1-z)$ $w = 0{,}5rand() + 0.5z$
Sigmoid azalan atalet ağırlığı	$w_k = \frac{(w_{start} - w_{end})}{(1 + e^{-u*(k - n*gen)})} + w_{end}$ $u = 10^{(\log(gen) - 2)}$
Sigmoid artan atalet ağırlığı	$w_k = \frac{(w_{start} - w_{end})}{(1 + e^{u*(k - n*gen)})} + w_{end}$ $u = 10^{(\log(gen) - 2)}$

- **Sonlandırma Kriteri:** NP-zor problem sınıfına giren problemlerin boyutları büyük ise optimal değerinin bulunabilmesinin neredeyse imkansız olduğundan daha önce bahsedilmişti. Bu tür problemleri çözmek için tasarlanan algoritmalar, eğer problem boyutu yeterince küçük ise optimal değeri bulabilecek şekilde tasarlanabilirler. Çözüm uzayını gezerken iteratif bir şekilde bir noktadan daha iyi bir noktaya ulaşmaya çalışan çözüm algoritmasının optimal değeri bulduğunda sonlanması beklenir. Bu durum aşikardır ve optimal değerin bulunması bir durdurma kriteri olarak belirlenir.

Ancak çok küçük boyutlu problemler yeterince ilgi çekici olamamaktadır. Gerçekte problemler daha karmaşık ve boyutları daha büyüktür. Bir çok problem için de optimal değer bilinememektedir. Bu sebeple algoritmalar optimal değeri bulabilmek için belki günlerce, yıllarca çalıştırılmak zorundadır. Pratik olamayan bu durum yerine, algoritmaların belirli durumlarda durdurulma kriterleri belirlenmelidir.

Kullanıcı tarafından belirlenen bu kriterlerden biri algoritmanın yapacağı iterasyon sayısı ile ilgilidir. İterasyon sayısına bir üst sınır getirilir ve maksimum iterasyon sayısı denir, $max_{iter}, iter_{max}$ gibi ifadeler ile temsil edilir. Farklı algoritmaların performansları karşılaştırılırken daha az iterasyon ile daha iyi sonuçlar elde eden algoritma başarılı kabul edilir. Bir diğer durma kriteri ise, algoritmanın bulacağı sonuçlar ile ilgilidir. Kullanıcı tarafından kabul edilebilir bir sonuç var olabilir ve buna belli uzaklıktaki değerler de kabul edilebilir. Örneğin amaç fonksiyonu değeri belirli bir sabit değerden küçük oldukça dur şeklinde bir sonlandırma kriteri belirlenebilir.

4.3 Önerilen Çözüm Tasarımı

DARP için çözüm önerisi geliştirilen tez çalışmamızda çözüm algoritması PSO kullanılmıştır. PSO'nun dinamik problemlere uygulanmasına literatürde yakın zamanda yeni yeni karşılaşılmaktadır. Çalışmada çözüm yaklaşımı olarak Montemanni ve arkadaşları tarafından (Montemanni, ve diğerleri 2005) önerilen yaklaşım uygulanmıştır.

DARP için önerilen çözüm yaklaşımında iki ana unsur söz konusudur, ilki çözüm aşamalarının işleyişini kontrol eden "Olay Yöneticisi" ("*Event Manager*") kısmı, diğeri ise problemin çözümlerinin üretildiği çözüm algoritmasıdır. "Olay Yöneticisi", çözüm algoritmasının ürettiği çözümlerin uygunluğunu denetleyerek optimizasyon prosedürü ile müşteriler arasında köprü görevi üstlenmektedir.

DARP için önerilen çözüm stratejisi, planlama periyodunun eşit uzunluktaki zaman dilimlerine bölünmesine dayanmaktadır. Tüm planlama periyodunun uzunluğu T ile gösterilecek olursa ve n_{ts} alt dönem sayısı olmak üzere; her bir alt dönemin uzunluğu T/n_{ts} olacaktır. Problem alt zaman aralıklarına ayrıldıktan sonra her bir zaman diliminde o anki problem girdileri göz önünde bulundurularak statik araç rotalama problemi çözülür. Bir sonraki zaman diliminde önceki statik problemin bilgileri hafızada tutulur ve aradaki zamanda eğer bilgi değişikliği olmuş ise problem tekrar güncellenerek yeni statik ARP üretilir ve çözülür.

Zaman periyotları içerisinde, üretilen statik ARP 'ler ardışık olarak çözülerek dinamizmin takip edilmesine çalışılmaktadır. Yapıda karar verici tarafından önceden

belirlenmesi gereken bir takım parametreler bulunmaktadır. Bunlardan ilki planlama periyodunun kaç alt döneme ayrılacağıdır, yani n_{ts} belirlenmelidir.

Bir planlama periyodu içerisinde, örneğin 1 gün içerisinde, belirli bir zamandan sonra gelen taleplerin ertesi güne ertelenmesi gerekmektedir. Örneğin günün ilk yarısında gelen talepler aynı gün değerlendirilir, ikinci yarısında gelen talepler ertesi güne aktarılır. Ertesi günün ilk alt zaman diliminde bir önceki günden bilinen taleplerin tamamı değerlendirilir, diğer zaman dilimlerinde yine dinamik olarak açığa çıkan talepler dikkate alınır. Böylelikle ardışık günler boyunca sistem kesintisiz bir şekilde işleyişine devam eder.

Belirlenmesi gereken ikinci parametre bu noktada ortaya çıkmaktadır. Günün veya planlama periyodunun hangi zamanından sonra gelen taleplerin ertesi güne erteleneceğinin kararı T_{co} parametresi ile belirlenir. T_{co} zamanından sonra gelen talepler ertesi güne ertelenir.

Bir aracın planında aksaklık olmaması ve aralarda boşa harcanan zaman olmaması planlayıcı açısından önemli bir kısıttır. Bu nedenle araçlara bir sonraki gidecekleri müşteriler belirli bir esneklik payı bırakılarak bildirilmelidir. Böylelikle hem müşterilerin gereksiz yere beklemeleri engellenmiş olur hem de araçlar daha verimli bir şekilde kullanılır. Çözüm stratejisi içerisinde bu kısıt T_{ac} parametresi ile değerlendirilmektedir. T_{ac} değeri belirlenirken, her bir aracın son konumlarından tahmini ayrılma süreleri dikkate alınır. Araçların son konumlarından tahmini ayrılma zamanlarından en az T_{ac} zaman öncesinden bir sonraki gidecekleri adres araçlara bildirilir. Böylelikle araç kullanıcıları daha uygun davranabilirler.

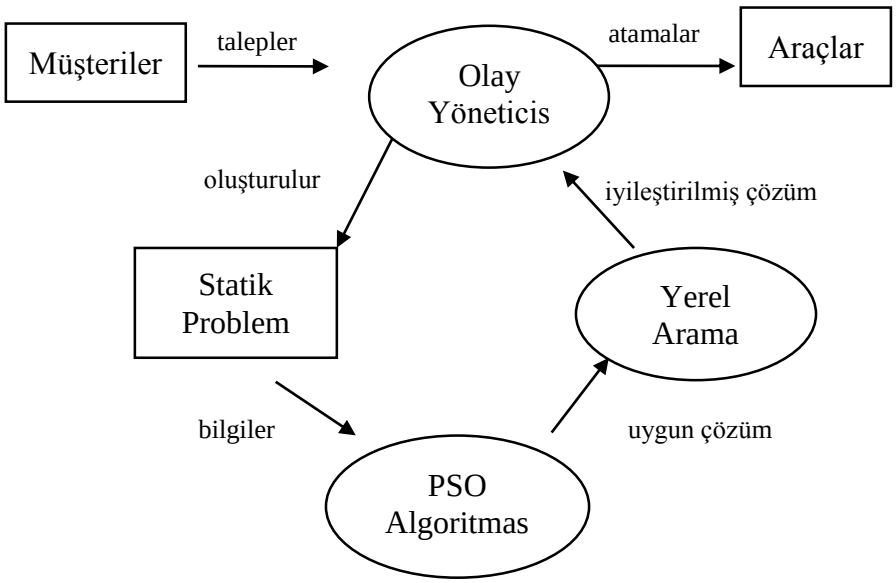
Bu yaklaşım önerildiği çalışmadan sonra akademik literatürde farklı çalışmalarda da kullanılmıştır. Montemanni ve arkadaşlarının önermiş olduğu çözüm yapısında, problemin çözüm algoritması aşamasında metasezgisel bir yöntem olan Karınca Kolonisi Algoritması kullanılmıştır. Hanshar ve Ombuki-Berman tarafından yapılan çalışmada (Hanshar ve Ombuki-Berman M. 2007) ise Genetik Algoritma çözüm safhasında kullanılmıştır. Son yıllarda yapılmış olan Khouadjia ve arkadaşları tarafından yapılan çalışmada (Khouadjia, ve diğerleri 2012) ise PSO ve tek çözüm

tabanlı Değişken Komşuluk Arama (Variable Neighborhood Search) algoritmaları kaşılaştırmalı olarak önerilmiştir.

Tez çalışmamızda olay yöneticisi ve çözüm algoritması bileşenli yapı kullanılmıştır. Çözüm algoritmasının bulduğu sonuçlar yerel arama teknikleri kullanılarak daha iyi bir hale getirilmiştir. Genel olarak üç basamakta ilerleyen yapının işleyişi

Şekil 4-2'de gösterilmektedir.

Önerilen yaklaşımın performansını ölçmek adına literatürde referans aldığımız çalışmalar ile aynı test problemlerini kullanmaktayız. Bu bölümün diğer kısımlarında çözüm stratejisinin temeli olan olay yöneticisinin nasıl işlediğinden bahsedilecektir.



Şekil 4-2 Çözüm Yapısının İşleyişi

4.3.1 Olay Yöneticisi

Olay yöneticisi, gerçek hayat ile çözüm algoritması arasında bağlantı görevi üstlenmektedir. Probleme dair bilgileri çözüm algoritmasına iletir ve çözüm algoritmasının bulduğu rotaların bilgilerini tutar. Böylelikle karar verici, olay yöneticisi mekanizması ile veri girişi ve sonuç raporlanması işlemini yapmaktadır.

Gerçek hayat problemi göz önüne alındığında müşteri talepleri, anlık olarak olay yöneticisine iletilir. Olay yöneticisi, planlama periyodunun her bir alt aralığı içerisinde dinamik girdilerden statik problemler oluşturur. Oluşturulan bu statik problem PSO ile çözülür ve uygun çözümler elde edilir. Elde edilen uygun çözümler yerel arama yöntemleri kullanılarak daha kaliteli bir hale getirilir. En son olarak da optimize edilmiş rotalar olay yöneticisine gönderilir. Böylelikle karar verici sonuçlara ulaşır ve gerçekleştirir.

Problemin çözümü için bir çalışma günü eşit zaman aralıklarına bölünmektedir. Montemanni'nin yaptığı incelemeye göre zaman aralığını çok fazla artırmak her zaman çok yararlı olmamaktadır (Montemanni, ve diğerleri 2005). Birbirine yakın mantıkta çözüm önerisi getirdiğimiz düşünülerek zaman aralığı sayısındaki seçimimizi Montemanni'nin seçimiyle yakın seçmek mantıklı olacaktır.

Gerçek bir problemde ilk zaman aralığı bir önceki günden kalan taleplerle başlatılmalıdır. Her bir statik problemde araçlar en son bulundukları müşterilerin pozisyonlarından başlatılarak geçerli olan taleplere en uygun çözüm önerisi araştırılmaktadır. Her bir aracın başlangıç zamanı bir önceki zaman diliminde en son hizmet verdikleri müşteriden ayrılma zamanı olarak kabul edilmektedir. Ayrıca her bir aracın her zaman diliminde geriye kalan kapasitesi hesaplanarak ve bu kapasiteye göre çözüm aranmaktadır. Her araç bir önceki çözüm diliminde gitmeye karar verdiği son lokasyona doğru devam edecek şekilde bir sonraki çözüm uzayına aktarılır.

PSO algoritması araçların kapasitelerini de göz önünde bulundurarak rotaları belirlemektedir. Algoritmanın işleyişi esnasında birden fazla kez çalıştırılması sonucu bazı istenmeyen durumlar ile karşılaşılmıştır. Bu problemler ilerleyen bölümlerde ele alınmış ve çözüm önerileri getirilmiştir. Bunların ortadan kaldırılması adına ilk etapta, olay yöneticisine bir karar fonksiyonu eklenmiştir. Bu karar fonksiyonu ile araçlar için belirlenmiş rotaların gerçeklenmesine olay yöneticisi karar vermektedir. PSO algoritması araçlara kapasitelerinden fazla müşteri atanmamasının kontrolünü yapmaktadır, ancak araçların kapasitelerinin etkin bir biçimde kullanılmasını kontrol etmekte zayıf kalmaktadır.

Gerçek hayatta maliyetler çok büyük önem taşıdığından, bir aracın olabildiğince etkin bir şekilde kullanılması istenmektedir. Bu nedenle PSO ve yerel arama yöntemleri kullanılarak belirlenen rotalar üzerinde, olay yöneticisi yeni yola çıkacak her bir aracın kapasitesinin yüzde kaçını kullanığını kontrol eder. PSO algoritmasının çalışması esnasında bulunan çözümler içerisinde bazı araçlar kapasitelerini etkin bir şekilde kullanmadan tekrar depoya dönebilirler. Bu tip alt rotalar maliyetleri artırmaktadır. Yola çıkacak araçların kapasitelerinin verimli bir şekilde kullanıldığına emin olmak adına belirli bir oranda kapasite kullanım şartı getirilebilir. Geliştirilen kural doğrultusunda araçlara yola çıkma talimatı verilmektedir. Çalışmamızda %75 ve üzerindeki kapasite kullanımı verimli olarak kabul edilmektedir. Planlanmış rotaya göre, kapasitesinin %75 inden daha azını kullanan araçların rotalarına başlamalarına izin verilmez, yeni müşterilerin ortaya çıkması beklenir.

Yola çıkmış bir araç için sonraki zaman dilimlerinde, PSO tarafından kapasitenin verimli kullanımına uygun bir rota bulunamayıp depoya dönmesi atanabilir. Bu durumda, karar fonksiyonu bu araçların depoya dönmelerini engelleyip en son bulundukları lokasyonda beklemelerine karar verir. Karar fonksiyonu kullanımı ile çözümler gerçek hayata çok daha yakın bir hale gelmektedir.

4.3.2 Parçacık Gösterimi

Parçacık sürü optimizasyonu algoritmasının performansını büyük ölçüde etkileyen unsurlardan biri de parçacık gösterimidir. Çözüm uzayında her bir parçacığın bulunduğu bir pozisyon vardır. Bu pozisyonlar ilgilenilen problem için parçacığın temsil ettiği çözümlerini kodlamaktadırlar. Çözüm uzayında gezinen iki parçacığın pozisyon vektörlerinin birbirleri ile uzaklığının, temsil ettikleri çözüm değerlerinin birbirleri ile uzaklıkları ile ilişkili olması beklenir. Bu durum parçacık pozisyon vektörü gösteriminin çözüm uzayını iyi temsil ediyor olması demektir.

Bir parçacığın pozisyon vektörünün gösterimi için literatürde farklı yaklaşımlar önerilmektedir. Açık uçlu ARP için PSO algoritması çözüm önerisi getirilen çalışmada (MirHassani ve Abolghasem 2011), yazarlar reel sayılardan oluşan pozisyon vektörü kullanmışlardır. Pozisyon vektörünün boyutu müşteri sayısı ile eşit

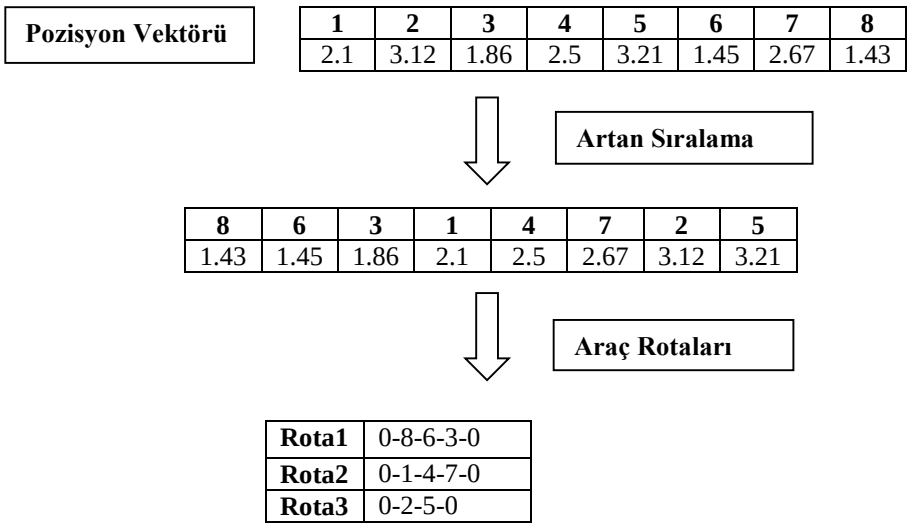
olarak seçilmektedir. Pozisyon vektöründen, temsil ettiği çözüm belirli yöntemler ile okunmaktadır. MirHassani ve Abolghasem tarafından yapılan çalışmada, önce pozisyon vektörü elemanları büyükten küçüğe sıralanmaktadır. Pozisyon vektörünün her bir elemanına karşılık gelen müşteri de aynı sıralamada yer değiştirir. Daha sonra ilk sıradaki müşteri uygun olan ilk araca atanır. Son olarak "tek nokta yer değiştirme" (*one-point moving*) yerel arama yöntemi ile çözüm iyileştirilmiştir.

Açık uçlu ARP için bir diğer PSO önerisi Wang ve arkadaşları tarafından (Wang, ve diğerleri 2006) yapılmıştır. Bu çalışmada yine reel sayılardan oluşan pozisyon vektörü kullanılmıştır. Reel sayıların tam kısımları müşterilerin hangi araçlara atanacağının belirlenmesinde kullanılmıştır. Tam kısımları aynı olan reel ayıların temsil ettikleri müşteriler aynı araçlardan hizmet görmektedir. Reel sayıların ondalıklı kısımları ise müşterilerin atandıkları araçlarda hangi sırada hizmet göreceğinin belirlenmesinde kullanılmıştır. Yine literatürde çok bilinen yerel arama yöntemlerinden; "en yakındakini ekleme" (*nearest insertion*), GENI ve 2-opt yerel arama yöntemleri çözüm iyileştirmekte kullanılmıştır.

Bu tez çalışmasında kullanılan parçacık gösterimi Wang ve arkadaşlarının (2006) önerdiği gösterime dayanmaktadır. Pozisyon vektörünün boyutu müşteri sayısı kadardır. *n* tane müşteriden oluşan bir rotalama problemi için reel sayılardan oluşmuş *n* boyutlu bir vektör kullanılmaktadır. Pozisyon vektörünün temsil ettiği çözümü vektörden okuma yönteminde, ilk olarak vektör elemanları küçükten büyüğe sıralanmaktadır. Daha sonra aynı tam kısma sahip olan reel sayıların temsil ettikleri müşteriler aynı araçlara atanır. Reel sayıların ondalıklı kısımları ise müşterilerin hizmet sıralarının belirlenmesine yardımcı olmaktadır.

Dinamik ARP için idealde kaç araç kullanılacağının önceden belirlenmesi mümkün değildir. Seçmiş olduğumuz parçacık gösteriminde araç sayısı ile ilgili olarak yalnızca müşteri sayıları dikkate alınmaktadır, kaç tane araç kullanılacağı dinamik olarak değişebilmektedir. Buna göre, en fazla müşteri sayısı kadar araç ile optimizasyona başlanmaktadır. Yani kullanılabilir araç sayısı bu şekilde belirlenmektedir. Optimizasyon esnasında araç sayısı anlık olarak değişmektedir.

Seçilen parçacık gösteriminin 8 müşteri için parçacık gösterimi ve gösterimden çözüm okunması aşaması Şekil 4-3'de detaylı bir şekilde gösterilmektedir.



Şekil 4-3 Parçacık Pozisyon Gösterimi ve Çözüm Belirlenmesi

Başlangıç Pozisyonunun Belirlenmesi: Çözüm uzayını olabildiğince iyi tarayabilmek için başlangıçta parçacıklar çözüm uzayına rasgele dağıtılmıştır. Bir parçacığın pozisyon vektörünün her bir bileşeni bir müşterinin hangi araç tarafından hizmet edildiğini ifade etmektedir. Bu sebeple pozisyon vektöründeki her bileşen $[1, m]$ aralığında olmak zorundadır. Ayrıca *m.* araç tarafından en son sırada hizmet edilen bir müşteri bir sonraki iterasyonda ilk araç tarafından ilk sırada hizmet görebilmek için hız vektörünün ilgili bileşeninin $-m$'e çok yakın bir değer olması gerekmektedir. Bu nedenle her bir parçacığın pozisyon vektörünün her bir bileşeni için $[1, m]$ aralığında, hız vektörünün bileşenleri için ise $[-m, m]$ aralığında rasgele reel sayılar üretilir. Burada *m*, DARP 'nin çözümü için kullanılabilecek maksimum araç sayısını yani maksimum alt rota sayısını göstermektedir. *m*, karar verici tarafından önceden belirlenmesi gereken bir parametredir. Bu parametre için her bir statik problemde müşteri sayısı üst limit olacağı gözden kaçırılmamalıdır.

4.4 Yerel Arama Metotları

Yerel arama yöntemleri bir çözümden başlayarak ardışık olarak çözümü belirli bir kural doğrultusunda adım adım iyileştirmek için kullanılan, basit kurallara dayalı sezgisel algoritmalardır. Genellikle problem türüne özgü olarak algoritmaların işleyiş

biçimleri belirlenmektedir. Araç rotalama problemi göz önüne alındığında ise her bir araç için belirlenmiş bir tur çözümü teşkil etmektedir. Bu noktada PSO algoritması ile elde edilen bir çözümün, yerel arama yöntemi kullanılarak ardışık olarak iyileştirilmesi amaçlanmaktadır.

Rotayı iteratif olarak iyileştirmek amaçlı kullanılan yerel arama yöntemleri iki kategoride ele alınmaktadır; rota-içi (intra-route) ve rotalar-arası (inter-route) olarak isimlendirilmektedir (Woodruff 1998).

Rota-içi ve rotalar-arası yerel arama yöntemlerinin temelde nasıl işlediğini bir örnekle anlatmak mümkündür. Örneğin bir araç rotalama problemi için servis hizmeti verilmesi gereken 10 müşterimiz olduğunu düşünelim. PSO algoritması sonucunda ise 2 araç ile 10 müşteriye hangi sırada hizmet verileceği belirlenmiş olsun. Elde edilen çözüm Şekil 4-4' de gösterildiği gibi olsun.

Araç 1: Depo – 3 – 7 – 2 – 6 – 8 – 10 – Depo

Araç 2: Depo – 1 – 4 – 5 – 9 – Depo

Şekil 4-4 PSO sonucu elde edilen örnek çözüm

Rota-içi yerel arama yöntemlerinde bir araca ait rotadaki müşterilerin sıralaması toplam tur maliyetini en iyileyecek şekilde iteratif olarak belirli bir kurala göre değiştirilmektedir.

Rotalar-arası yerel arama yöntemlerinde ise bir aracın rotasında bulunan bir müşteri başka bir aracın rotasına eklenebilir veya karşılıklı olarak iki rota arasında müşteri yer değişikliği yapılabilir. Burada yine toplam tur maliyeti göz önünde bulundurulmaktadır. Bölüm 4.4.1 ve Bölüm 4.4.2 'de DARP için PSO algoritması tarafından bulunan uygun çözümlerin iyileştirilmesi için tez çalışmamızda kullanmış olduğumuz yerel arama metotlarından detaylı bir şekilde bahsedilecektir.

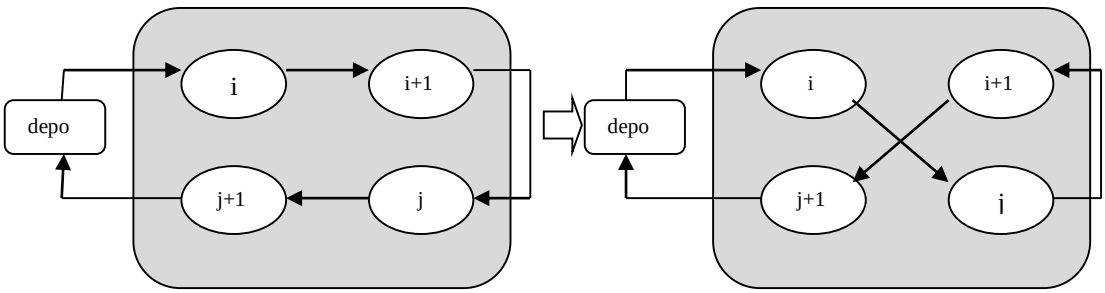
4.4.1 Rota – içi yerel arama metotları

Rota- içi yerel arama metotlarında her bir aracın rotası kendi içerisinde iyileştirilmeye çalışılır. Yani alt-rotalar (*subtours*) içerisinde yerel arama metotları uygulanır. Alt-rotaya ait elemanların sırası tekrar gözden geçirilirken toplam tur maliyeti göz önünde bulundurulur. Birçok yöntem ile bir alt-rotanın diziliminde değişiklik yapılabilir.

Örneğin alt-rota içerisinde bir elemanın yeri değiştirilebilir ya da alt-rotanın belirli bir noktadan sonraki elemanlarının dizilimi tersine çevrilebilir. Buna benzer bir çok sezgisel yöntem bir kural dahilinde uygulanarak maliyetlerin düşürülmesine çalışılabilir (Woodruff 1998).

Tez çalışmamızın bu aşamasında literatürde çok kullanılan ve başarılı sonuçlar elde edilen 2-opt yerel arama algoritması detaylı bir şekilde anlatılmıştır.

2-Opt Yerel Arama Metodu: 2-Opt prosedürü tek bir rotayı ele alır ve rota içerisinde ardışık müşteri çiftlerinden ikisi arasında güzergâh değişimi yapar. Sistematik olarak yapılan güzergâh değişimlerinde toplam tur maliyetlerini en küçük kılan yer değişimi kabul edilir. Ardışık müşteri ikilileri i – (i + 1) ve j – (j+1) olarak seçildiğini varsayalım; bu durumda karşılıklı olarak düğümlerin yer değişimi sonucunda i – j ve (i + 1) – (j+1) kenarları elde edilir. Yani i. müşteriden sonra j. müşteri ve (i+1). müşteriden sonra (j+1). müşteriye servis verilir. (i+1) ile j arasındaki müşterileri temsil eden düğümler ise ters sırada sıralanarak 2-opt metodu uygulanmış olur. Şekil 4-5 ‘de 2-opt metodunun işleyişi görselleştirilmiştir.



Şekil 4-5 2-Opt Prosedürünün işleyişinin görseli

Şekil 4-4 ’de verilen örnek çözümü incelersek, 1. araca ait rota için 3. ve 7. ile 8. ve 10. müşteri ikililerini arasında 2-opt yerel arama metodunu uyguladığımızı varsayalım.

Bu durumda Araç 1 ‘e ait elde edilecek yeni sıralama Şekil 4-6 ‘deki gibi olacaktır:

Araç 1: Depo – 3 – 8 – 6 – 2 – 7 – 10 – Depo

Şekil 4-6 2-Opt prosedürü sonucu

2-opt metodunun uygulanması sonucunda elde edilen rota maliyet açısından optimali vermek zorunda değildir. İteratif olarak tüm noktalar için yöntem uygulanarak maliyet açısından en uygun rota çözüme dahil edilmektedir. 2-opt yerel arama sezgiselinin algoritması Algoritma 2 ile verilmiştir.

Algoritma 2: 2-opt sezgiselinin işleyişi

2opt(*alt rota, i, k*) {
Alt rotanın ilk *i-1* elemanını al ve yeni bir rota oluştur, *yeni-alt rota* olarak isimlendir
Alt rotanın i. elemanından *k.* elemanına kadar olan elemanları ters sırada oluşturulan *yeni alt rotaya* ekle
Alt rotanın k+1. elemanından son elemanına kadar olan elemanları *yeni alt rotaya* ekle
Return *yeni alt rota*
}

4.4.2 Rotalar – arası yerel arama metotları

Rota-içi yerel arama metotları bir aracın toplam tur maliyetlerini optimize etmektedir. Rotalar-arası yerel arama yöntemleri ise buna ek olarak toplam tur maliyetini en iyilerken araç sayısında da değişiklik yapmaya imkân vermektedir. Böylelikle minimum sayıda araç kullanmaya olanak sağlamaktadır.

Rotalar arası yerel arama yöntemlerinde, bir müşteri bir alt rotadan diğerine eklenebilir veya alt rotalar arasında birer müşteri yer değiştirebilmektedir. Yer değiştirme işlemleri veya ekleme işlemleri yapılırken, araçların kapasitelerinin göz önünde bulundurulması gerekmektedir. Araç kapasitesi izin verdiği ölçüde ve yer değiştirme ile bir maliyet indirgenmesi veya araç sayısında bir indirgenme söz konusu ise uygulanmaktadır.

Literatürde yer alan gösterim şekline göre, iki rota arasında birer eleman yer değişikliği yapıyorsa (1,1)-yer değiştirme (*exchange*) olarak isimlendirilir. Bir eleman başka bir alt rotaya ekleniyorsa (1,0)-yer değiştirme veya (1,0)-ekleme (*inserting*) olarak isimlendirilmektedir (Woodruff 1998). Bir elamanın alt rotalar arası konum değiştirmesi hesaplama maliyeti olarak çok karmaşık değildir.

Yer değiştirme veya ekleme operatörleri isimlendirilirken kaç elemanın pozisyonunun değiştiğine bakılmaktadır. Örneğin bir alt rotadan iki eleman bir diğer alt rotaya eklenecek ise (2,0)-ekleme, karşılıklı ikişer eleman yer değiştirecekse (2,2)-yer değiştirme olarak isimlendirilir.

Ekleme ve yer değiştirme yerel arama yöntemlerinde alt rotaların tüm elemanları için aynı işlemler yapılmaktadır ve maliyet açısından en ideal durum uygulanmaktadır. Bu nedenle 3 ve daha yukarı elemanın birden değişikliğini içeren yerel arama yöntemleri hesaplama süresini üstel bir şekilde artıracağından pratikte tercih edilmemektedir.

ARP'de müşterilerin hangi sırada hizmet görmesinin yanında, ardışık müşterilerin de hangi müşteriler olduğu önemlidir. Çünkü bir şehirden diğerine yol alınırken aradaki yolun maliyeti de çözümü etkilemektedir. Bu nedenle elde edilmiş bir çözüm önerisinde ardışık iki müşterinin de sıralaması bozulmadan rotalar arası yer değişikliği yapılması söz konusu olabilir. Çizge teorisi (*Graph Theory*)'ne göre iki şehir arasındaki yol, kenar (*edge*) olarak isimlendirilmektedir. Bir kenarın rotalar arası yer değiştirmesi operatörü ise 1-kenar ekleme (*one edge insertion*) ile isimlendirilmektedir.

Tez çalışmamızda rotalar-arası yerel arama yöntemlerinden (1,0)- yer değiştirme ve 1-kenar ekleme yerel arama yöntemleri kullanılmıştır. Bu bölümde rotalar-arası yerel arama tekniği olarak kullanmış olduğumuz sezgisel algoritmalardan bahsedilecektir.

(1,0)-yer değiştirme: Tek düğümü yer değiştirme, (1,0)-ekleme şeklinde de isimlendirilen bu sezgisel yöntem ARP probleminin çözüm kalitesini artırmak için kullanılmaktadır. Literatürde yer alan en basit yer değiştirme metodu (1,0) – yer değiştirme şeklindedir. Yöntemin işleyiş kuralı; bir rotadaki bir elemanın başka bir rotaya eklenmesidir. Bir alt rotanın tüm elemanları sırası ile diğer alt rotaların içerisine eklenir. Her uygulama sonucunda maliyetler hesap edilir. Eğer bulunduğu nokta dışında başka bir noktada maliyet daha küçük oluyorsa yeni yeri o nokta tayin edilir. İteratif olarak tüm düğümler arasında gezdirildikten sonra *i.* düğüm en iyi yerine sabitlenmektedir.

1-kenar ekleme: Bir alt rotanın ardışık iki elemanının başka bir alt rotanın maliyet açısından en uygun yerine eklenmesi şeklinde uygulanmaktadır. Ardışık iki düğüm bir kenarı temsil ettiğinden ismi kenar ekleme olarak kullanılmaktadır.

PSO algoritmasının bulmuş olduğu uygun çözüm sırasıyla 2-opt, (1,0)-yer değiştirme ve 1-kenar ekleme sezgiselleri kullanılarak iyileştirilmektedir. Kullanılmış olan yerel arama metotları birbirleri ile iş birliği içerisinde çalışmaktadırlar. İlk olarak 2-opt sezgiseli her bir alt rotayı kendi içerisinde daha ideal hale getirmektedir. Her bir alt rota için, her müşteri ikilisi için 2-opt işlemi uygulanır. Her seferinde maliyetler hesaplanır. Tüm mümkün yer değiştirmelerin sonucunda en düşük maliyetli yer değiştirme kabul edilerek uygulanır.

2-opt işlemi tamamlandıktan sonra (1,0)-yer değiştirme sezgisel algoritması uygulanır. (1,0)-yer değiştirme operatörü alt rotaların her biri için, tüm elemanlarının sırasıyla diğer alt rotalarda farklı pozisyonlarda denemektedir. Her bir yer değiştirme için yine maliyetler hesaplanır ve en küçük maliyetli yer değiştirme kabul edilerek uygulanır. Son olarak 1-kenar ekleme operatörü ile ardışık müşterilere ait yollar, farklı alt rotalarda denenerek maliyetler dikkate alınarak uygulanır.

Yerel arama bölümünü sonlandırmadan önce bahsedilmesi gereken önemli bir alt başlık bulunmaktadır. Yerel arama yöntemlerinin bu noktaya kadar anlatılmış olanları literatürde daha önce önerilmiş olup, tez çalışmasında kullanılanlarıdır. Yerel arama yöntemleri birer sezgisel algoritmalardır ve sezgisel algoritmalar problem bağımlı, yani probleme özgü kurallar doğrultusunda çalışmaktadırlar. Bu noktada çalışmış olduğumuz problem çözümü esnasında çözüm kalitesini artırmak adına gerekli görülen bir sezgisel algoritma önerilmiştir.

Tez çalışmamızda önerilen PSO algoritmasında bir aracın kapasitesinden daha fazla müşteriye gitmesi durumu ile karşılaşıldığında ceza fonksiyonu yardımıyla bu durum engellenmeye çalışılmıştır. DARP test problemlerinin çözümleri esnasında görülmüştür ki, ceza fonksiyonu kullanılmasına rağmen, araçların cezadan kaçamaması olasıdır. Gerçek hayatta bu tip bir duruma izin verilmemesi için kapasiteyi aşan çözümün düzeltilmesi gerekmektedir. Karşılaşılan bu problem bağımlı sorunu çözebilmek adına sezgisel bir algoritma önerilmiştir. Önerilen sezgisel yöntem her

hangi bir araç için ceza fonksiyonu kullanılmasına rağmen, kapasite aşımı engellenemediğinde yani amaç fonksiyonu yeterince minimize edilemediğinde devreye girmektedir. Kapasite aşımına sebep olan müşterilerin yeni bir araç yola çıkarılarak ona atanmasını sağlamaktadır.

BEŞİNCİ BÖLÜM

5 PSO'NUN DİNAMİK ARAÇ ROTALAMA PROBLEMLERİNE UYGULANMASI

5.1 Giriş

Dinamik araç rotalama probleminin çözümü için önerilen yöntemi test etmek adına literatürde çalışılan problem setleri ele alınmıştır. DARP veri setleri olarak ele alınan problemler akademik çalışmalarda kullanılmış problemlerdir. Tez çalışmamızın bu bölümünde önerilen PSO algoritması ile test verileri çözülmüş ve diğer akademik çalışmalarda elde edilen sonuçlar ile karşılaştırılmıştır.

Aynı problem seti ile çalışıp Karınca kolonisi algoritması (Montemanni, ve diğerleri 2005), Genetik algoritma (Hanshar ve Ombuki-Berman M. 2007) ve PSO algoritması ile çözüm önerisi getiren (Khouadjia, ve diğerleri 2012) çalışmalardaki çözümlere de yer verilerek sonuçlar karşılaştırmalı olarak raporlanmıştır.

Bu bölümde veri setindeki problemlerin özelliklerinden bahsedildikten sonra karşılaştırmalı sonuçlara yer verilecektir. Son olarak gerçek hayat uygulaması için, taşımacılık yapan bir şirketten alınmış veriler üzerindeki çalışmalara yer verilecektir.

5.2 Veri Seti

Önerilen bir algoritmanın bulduğu çözümlerin ne derece başarılı olduğunu ölçmek için bir standart oluşturmak önemlidir. Bu nedenle genellikle her araştırmacının kolaylıkla erişebileceği veriler ile çalışmak gerekmektedir. Test problemleri (*benchmark* olarak da isimlendirilir), genellikle problemi literatüre kazandıran kimseler tarafından rasgele sayılar kullanılarak üretilir.

İlgilenilen problem türüne ait bir çözüm önerisi getirdikten sonra, literatürde aynı problem ile test edilmiş algoritmalar önerilen çözümün kalitesini ölçmek için kullanılır. Karşılaştırmalı sonuçlar bu noktada önem taşımaktadır.

Statik ARP için Christofides ve Beasley (Christofides ve Beasley 1984) (yedi farklı problem), Fisher (Fisher 1995) (iki farklı problem) ve Taillard (Taillard 1993) (on üç farklı problem) tarafından oluşturulan test problemleri, Kilby ve arkadaşları

(Kilby, Prosser ve Shaw 1998) tarafından dinamik hale getirilmiştir. Statik ARP test problemlerinde, her bir müşterinin coğrafik konumu, talep miktarı ve kullanılabilecek araç kapasiteleri bilgileri bulunmaktadır. Kilby test problemlerini dinamik hale getirirken bir takım özellikler eklemiştir.

Öncelikle her bir müşterinin ortaya çıkma zamanı tanımlanmıştır. Çünkü statik ARP de zaman kavramı kullanılmadığından böyle bir bilgiye gerek yoktur. Planlamaya başlarken hiçbir müşteri bilinmiyor varsayıldığından, ortaya çıkma zamanına kadar müşterinin hiçbir bilgisi bilinmemektedir. Ortaya çıkma zamanları üretilirken düzgün rasgele dağılım kullanılmıştır (Kilby, Prosser ve Shaw 1998).

İkinci eklenen bilgi her bir müşteriye ne kadar süre servis edileceğidir. Veri incelendiğinde bir problem için, müşterilere ait servis edilme süreleri eşit seçilmiştir.

En son eklenen bilgi ise, bir çalışma gününün uzunluğudur. İki en uzak konum (müşteri) arasındaki mesafenin sekiz katı olarak belirlenmiştir. Birim zamanda bir birim yol alan araçlarla çalışıldığı göz önüne alınmıştır.

Tez çalışmamızda Kilby nin dinamik hale getirdiği DARP test problemleri çözülmüştür. Aynı test problemleri Montemanni ve diğerleri (Montemanni, ve diğerleri 2005), Hanshar ve diğerleri (Hanshar ve Ombuki-Berman M. 2007) ve Khouadjia ve diğerleri (Khouadjia, ve diğerleri 2012) tarafından da çözüldüğü için elimizde sonuçlarımızı karşılaştırabileceğimiz başka çalışmalar da bulunmaktadır.

Tez çalışmamızda popülasyon tabanlı bir algoritma kullanıldığından, karşılaştırmalı sonuçlarda diğer popülasyon temelli çalışmalara yer verilmektedir. Dinamik araç rotalama problemi için tek çözüm temelli algoritmaların da önerildiği çalışmalar da literatürde yer almaktadır. Ancak karşılaştırmalı sonuçlarda bu çalışmaların verileri dikkate alınmamıştır.

Problemleri üreten araştırmacının ismini ve problemin kaç müşterili bir problem olduğunun bilgisi ile problemler isimlendirilmiştir. Örneğin, "*c50"* bir problem ismidir. Christofides'in literatüre kazandırdığı tek depodan 50 müşteriye hizmet verecek araçların rotalanması problemidir.

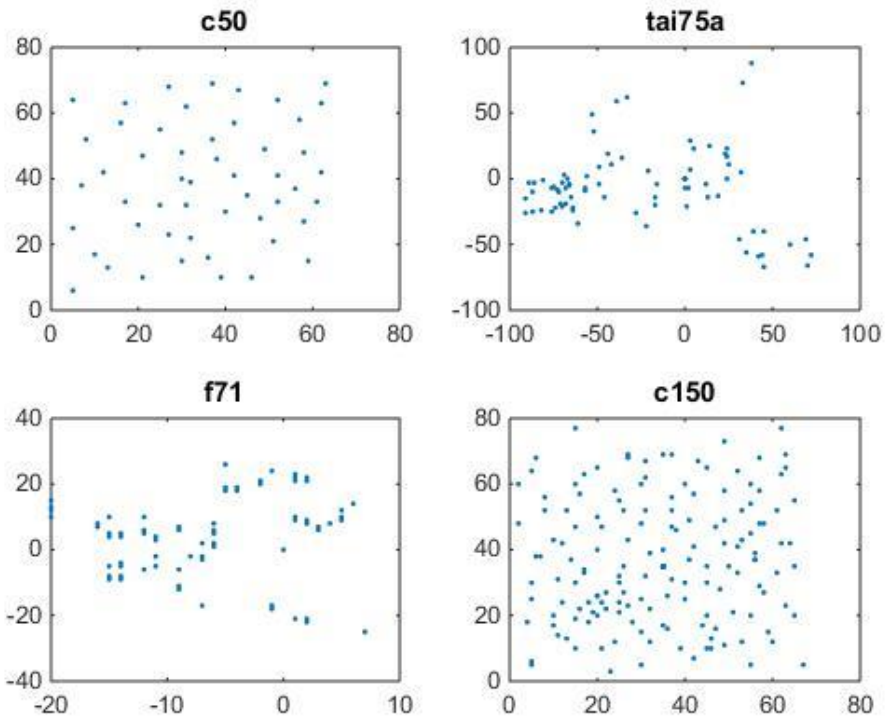
Kullanılan veri setleri üç ana grupta toplanmıştır:

Christofides Veri Seti: Müşteri sayısı 50 ve 199 arasında değişen yedi farklı problemden oluşmaktadır. Müşterilerin coğrafik konumu servis bölgesine düzgün olarak dağılmıştır. Tek depodan servis verilmektedir. Deponun konumu müşterilerin konumlarının orta bölgesine denk gelmektedir. Koordinat sistemi 80x80 boyutlarında bir kare olarak düşünülüp, depo (30,40) konumunda yerleştirilmiştir.

Taillard Veri Seti: Merkezi (0,0) olan ve depo merkezde bulunan 200x200 boyutlu bir kare bölge servis alanı olarak düşünülmüştür. Tek depodan hizmet gören müşteri sayısı 75-150 arasında değişen on üç farklı problem vardır. Servis alanının belirli birkaç bölgesinde toplanan müşteriler kendi merkezleri etrafında düzgün olarak dağılmıştır.

Fisher Veri Seti: Depo koordinatı (0,0) da bulunan, 30x60 boyutunda bir bölge servis alanı olarak belirlenmiştir. Müşterilerin çok büyük bir kısmı depo civarında bulunmaktadır. Depodan uzaklaştıkça müşteriler sayıca azalmaktadır.

Problemler kendi içinde yapısal olarak yukarıda anlatıldığı gibi ayrışmaktadırlar. Üç farklı problem türüne ait örnek problemler Şekil 5-1'de görselleştirilmişlerdir. Böylelikle müşterilerin konumlarının nasıl dağıldığı daha ayrıntılı görülmektedir.



Şekil 5-1 Christofides, Taillard ve Fisher verilerine ait örnek problemler

5.3 PSO Algoritmasının diğer çalışmalar ile karşılaştırılması

Dinamik hale getirilen problem kümeleri, Olay Yöneticisi ve Çözüm Algoritması temelli yapı kullanılarak çözülmüştür. Çözüm algoritması olarak kullanılan PSO parametrelerinin belirlenmesi önemli bir aşamadır. Bunun yanında, çalışma günü ile ilgili önceden belirlenmesi gereken parametreler bulunmaktadır.

Elde edilen sonuçları, diğer çalışmalar ile karşılaştırabilmek için çalışma günü ile ilgili parametreleri aynı seçmek uygun olacaktır. Karınca Kolonisi algoritması ile çözüm getiren Montemanni ve diğerleri (2005), günün kaça bölünmesi ile ilgili bir takım deneyler yapmışlardır. Farklı sayıda alt dilim ile elde edilen sonuçlar karşılaştırılmıştır. Sonuç olarak, bir günü belirli bir sayıdan fazla alt dilime ayırmak sonuçları iyiye götürmemektedir. Çünkü alt zaman dilimleri çoğaltıldıkça süreleri azalmaktadır ve bir zaman diliminden diğerine geçerken çoğunlukla problem bilgisi değişmemektedir. Bu da çözüm algoritmasının gereksiz yere çalıştırılmasına sebep olmaktadır. Bir günü eşit alt dilimlere ayırırken bu durum dikkate alınmıştır ve n_{ts}

parametresi 25 olarak belirlenmiştir. Yani bir çalışma günü 25 alt döneme ayrılıp, her bir döneme ait statik problem çözülmüştür.

T_{co} parametresi ile günün hangi zamanından sonra gelen talepler ertesi güne erteleneceği belirlenmektedir. Aynı problem kümesini çözen çalışmalarda (Montemanni, ve diğerleri 2005) (Khouadjia, ve diğerleri 2012) bu değer günün yarısı olarak belirlenmiştir. Bu nedenle $T_{co} = 0{,}5 * T$ olarak seçmek uygun olmaktadır. Yani öğleden sonra gelen müşteri talepleri ertesi günün ilk zaman diliminde değerlendirilecektir.

Bir planlama günü 25 alt dilime ayrıldığında, günün yarısına kadar 13 zaman dilimi geçmektedir. 13 zaman dilimi boyunca, her bir zaman dilimi öncesinde oluşturulmuş statik problemler PSO algoritması ile çözülür. Günün yarısından sonra gelen talepler ise ertesi günün ilk zaman diliminde çözülür. Bu nedenle 14. iterasyon olarak öğleden sonra gelen taleplerin tamamı bir değerlendirilerek, tek bir statik problem olarak ele alınmaktadır.

Ele alınan araç rotalama problemi simetrik ARP dir, yani *i* şehrinden *j* şehrine gidiş mesafesi (maliyeti) dönüş mesafesi (maliyetine) eşittir. $t \in \{1,2,\dots,14\}$ olmak üzere herhangi bir t alt zaman diliminde bilinen talepler ile oluşturulan tam sayılı programlama modeli aşağıdaki gibidir:

Parametreler:

$V = \{0,1,\dots,n\}$: Düğüm kümesi,

$K = \{1,2,\dots,m\}$: Araç kümesi,

Q: Araç kapasitesi,

c_{ij}: *i* düğümünden *j* düğümüne gitme maliyeti,

d_i: *i* düğümünde yer alan müşterinin talep miktarı,

Karar değişkeni:

$$x_{ij}^{k} = \begin{cases} 1, & k\ aracı\ i\ düğümünden\ j\ düğümüne\ giderse \\ 0, & aksi\ taktirde \end{cases}$$

Amaç fonksiyonu:

$$Minimize \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} c_{ij}x_{ij}^{k} \tag{5-1}$$

Kısıtlar:

$$\sum_{i=0}^{n}\sum_{k=1}^{m} x_{ij}^{k} = 1, \qquad j \in V \setminus \{0\}, \tag{5-2}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ij}^{k} = 1, \qquad i \in V \setminus \{0\}, \tag{5-3}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{i0}^{k} = m, \tag{5-4}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{0j}^{k} = m, \tag{5-5}$$

$$\sum_{i=0}^{n} x_{ih}^{k} = \sum_{j=0}^{n} x_{hj}^{k}, \qquad \forall h \in V, k \in K \tag{5-6}$$

$$\sum_{i=0}^{n} d_i \sum_{j=0}^{n} x_{ij}^{k} \leq Q, \qquad k \in K \tag{5-7}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k} \leq |S| - 1, \qquad \forall S \subseteq V - \{1\}, S \neq \emptyset, k \in K \tag{5-8}$$

Amaç fonksiyonu toplam tur maliyetini minimize edecek şekilde modellenmiştir. (5-2) ve (5-3) numaralı kısıtlar depo dışındaki her bir müşterinin yalnızca bir kez ziyaret edilmesi gerektiğini göstermektedir. (5-4) ve (5-5) numaralı kısıtlar depoya araç sayısı kadar giriş ve çıkış olmasını kontrol etmektedir. Yani *m* adet alt rota oluşturulmaktadır. Kısıt (5-6) ise her bir düğüme giriş ve çıkış aynı araç tarafından yapılması gerektiğini söylemektedir. Kısıt (5-7) bir araca ait müşterilerin toplam talebinin araç kapasitesini aşmamasını kontrol etmektedir. Son olarak (5-8) numaralı kısıt alt tur oluşmasını engellemektedir.

PSO algoritmasına ait seçilen parametreler Tablo 5-1 ‘de verilmektedir

Tablo 5-1 PSO algoritması parametreleri

Parametre Türü	Değer
Popülasyon Büyüklüğü	10
İterasyon Sayısı	1000
Her Bir Problem İçin Tekrar Sayısı	10
Atalet Ağırlığı	1
Kişisel En İyi Pozisyonun Ağırlığı (c_1)	1
Yerel En İyi Pozisyonun Ağırlığı (c_2)	1

Tablo 5-1 ‘den de görüldüğü üzere, popülasyon büyüklüğü 10 olarak belirlenmiştir. Yani 10 farklı koldan çözüm uzayı taranmaktadır. Her bir zaman diliminde üretilmiş problem PSO algoritması çalıştırılarak çözülmektedir. İterasyon sayısı 1000 olarak belirlenmiştir. Test problemlerinin her biri 10 ‘ar kez çözülerek her bir çözüm için sonuçlar kaydedilmiştir. PSO parametrelerinin her biri 1 olarak seçilmiştir. Algoritma, i7 cpu ve 8 gb ram özelliklerindeki bir bilgisayarda çalıştırılmış ve karşılaştırmalı sonuçlar Tablo 5-2 ile verilmiştir.

Karşılaştırmalı sonuçlar raporlanırken, her bir algoritma ile elde edilen çözümler maliyetler açısından karşılaştırılmıştır. Bu durumda tüm algoritmaların amaç fonksiyonları, kullanılan tüm araçların almış oldukları yolların toplamını minimize etmektir. Bu nedenle en küçük amaç fonksiyonu değeri ile tüm müşterilere hizmet veren algoritma başarılı kabul edilmektedir.

Yirmi adet farklı probleme ait amaç fonksiyonu değerlerinde karşılaştırılan dört algoritma için, en iyi değerin bulunduğu hücre koyu renkle verilmiştir. Görülmektedir ki sekiz adet problem için önerilen PSO algoritması literatürde bu güne kadar çalışılmış olan algoritmalardan daha iyi sonuç vermektedir. En iyi sonuçların elde edildiği problem boyutlarına dikkat edilecek olursa, genellikle büyük boyutlu (120, 150 adet gidilecek konumun bulunduğu) problemlerde PSO algoritması başarılı olmaktadır. Buradan yola çıkarak hesaplama karmaşıklığı açısından daha karmaşık ve çözüm uzayının göreli olarak daha büyük olduğu problemlerde başarılı olunmaktadır.

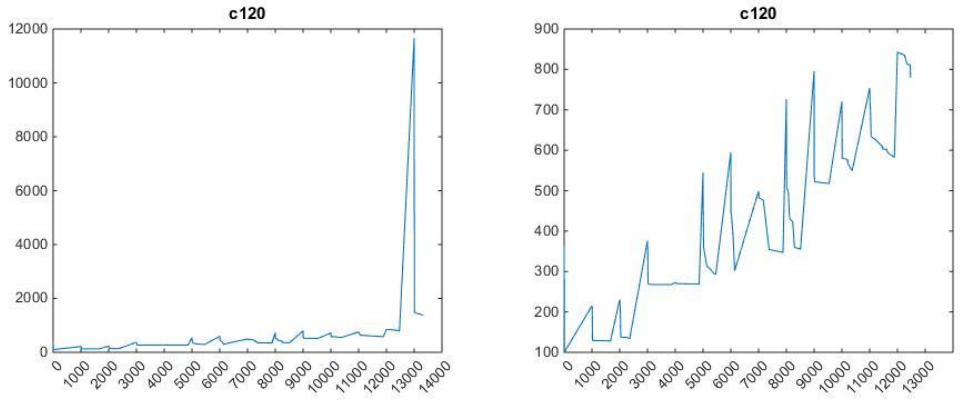
Optimizasyon açısından bu husus önemli olmaktadır ve algoritmanın problem boyutu artsa bile yerel optimaya takılmadan çözüme gittiğini göstermektedir.

Tablo 5-2 Önerilen PSO algoritmasının literatürde bilinen çalışmalar ile karşılaştırılması

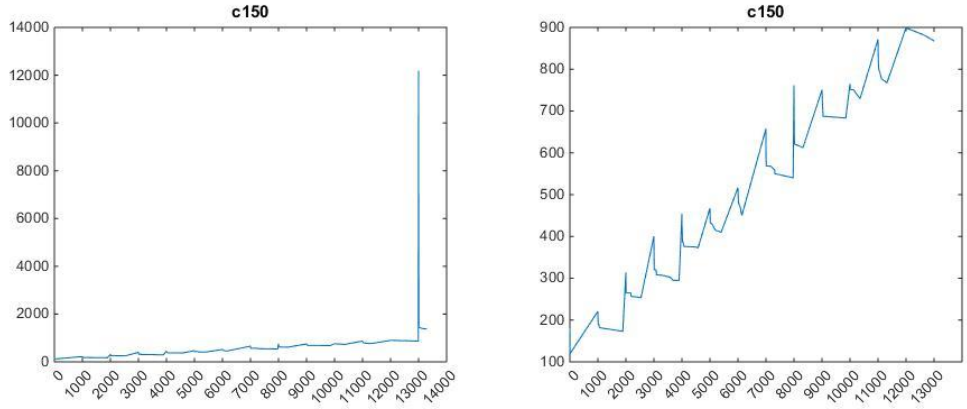
Problem	PSO		DAPSO		ACO		GA	
	En İyi	**Ortalama**	**En İyi**	**Ortalama**	**En İyi**	**Ortalama**	**En İyi**	**Ortalama**
c50	632,93	655,248	**575,89**	632,38	631,30	681,86	570,89	593,42
c75	988,36	1029,915	**970,45**	1031,76	1009,38	1042,39	981,57	1013,45
c100	975,62	1028,567	988,27	1051,5	973,26	1066,16	**961,1**	987,59
c100b	969,9	1018,114	924,32	964,47	944,23	1023,6	**881,92**	900,94
c120	**1267,26**	1348,492	1276,88	1457,22	1416,45	1525,15	1303,59	1390,58
c150	**1227,21**	1363,564	1371,08	1470,95	1345,73	1455,5	1348,88	1386,93
c199	1781,3	1818,18	**1640,40**	1818,55	1771,04	1844,82	1654,51	1758,51
f71	**277,41**	301,259	279,52	312,35	311,18	348,69	301,79	309,94
tai75a	**1770,03**	1859,38	1816,07	1935,28	1843,08	1945,2	1782,91	1856,66
tai75b	**1396,39**	1435,853	1447,39	1484,73	1535,43	1704,06	1464,56	1527,77
tai75c	1455,77	1522,101	1481,35	1664,4	1574,98	1653,58	**1440,54**	1501,91
tai75d	1518,58	1590,723	1414,28	1493,47	1472,35	1529,0	**1399,83**	1422,27
tai100a	2357,66	2471,718	2249,84	2370,58	2375,92	2428,38	**2232,71**	2295,61
tai100b	2286	2374,241	2238,42	2385,54	2283,97	2347,90	**2147,7**	2215,39
tai100c	**1495,2**	1588,892	1532,56	1627,32	1562,30	1655,91	1541,28	1622,66
tai100d	**1722,27**	1879,962	1955,06	2123,9	2008,13	2060,72	1834,6	1912,43
tai150a	3571,34	3746,666	3400,33	3612,79	3644,78	3840,18	**3328,85**	3501,83
tai150b	**2893,76**	3200,968	3013,99	3232,11	3166,88	3327,47	2933,4	3115,39
tai150c	2772,36	2928,112	2714,34	2875,93	2811,48	3016,14	**2612,68**	2743,55
tai150d	3110,11	3258,46	3025,43	3347,6	3058,87	3203,75	**2950,61**	3045,16
Toplam	34469,46	36420,42	34315,87	36892,83	35740,74	37700,46	33673,92	35101,99

Algoritmanın performansını ölçmek için, her bir zaman diliminde ele almış olduğu problem için iterasyonlar boyunca en iyiye nasıl yakınsadığının görselleştirilmesi önemlidir. Performans ölçümü için verilen metriklerin hesaplanmasından önce Şekil 5-2 ile verilen en iyi değere yakınsama grafiği incelenebilir. Bu grafikte özellikle büyük boyutlu iki problem örneği seçilmiştir. Bu noktada yakınsamanın hızı gözlemlenebilir. Her bir zaman diliminde bulunan bir çözüm değerinin iyileştirilme hızı gözlemlenmektedir. Son zaman diliminde günün yarısından sonra gelen talepler birlikte değerlendirildiği için, ilk elde edilen çözüm diğer zaman dilimlerine kıyasla daha büyüktür. Ancak hızlı bir şekilde düşüş olduğu

görülmektedir. Bu açıdan bakıldığında algoritmanın optimal değere yakınsamasının göreli olarak hızlı olduğu sonucuna varılabilir.



Şekil 5-2 c120 Problemi için algoritmanın en iyi değere yakınsaması



Şekil 5-3 c150 Problemi için algoritmanın en iyi değere yakınsaması

Algoritmanın en iyi değere yakınsaması verilirken, c120 ve c150 problemleri örnek olarak seçilmiştir. Her iki problem için de ilk olarak planlama periyodunun tamamının grafiği çizilmiştir ve ikinci olarak ise günün ilk yarısının grafiği çizilmiştir. Daha önce de bahsedildiği gibi, günün yarsından sonra gelen taleplerin tamamı ertesi gün değerlendirildiğinden, toplam maliyet ilk aşamada yüksek çıkmaktadır. Bu nedenle y-eksenindeki değer birden yükselmiştir. PSO algoritmasının iterasyonları neticesinde bu değerler iyileştirilmiştir.

Önerilen algoritmanın performansını değerlendirmek için bir diğer yöntem ise sayısal değerler ile incelemektir. Bu noktada (Weicker 2002) tarafından önerilen "kesinlik, kararlılık ve tepkisellik" metriklerinden kesinlik metriği seçilmiştir. Test problemleri için elde edilen sonuçlar doğrultusunda algoritmaların kesinlik değerleri

hesaplanmıştır. Bu değerler hesaplanırken, her bir problem için planlama periyodunun sonunda (T anında) bulunan çözüm değerleri kullanılmıştır.

Tez çalışmamızda önerdiğimiz PSO algoritması ve diğer çalışmaların kesinlik değerleri

Tablo 5-3 ile verilmektedir. Kesinlik değerini hesaplarken eşitlik (3-3) kullanılmıştır. Problem minimizasyon problemi olduğundan kesinlik ölçüsü eşitlik (5-9) şeklinde hesaplanmıştır. Test problemlerinin statik problemlerden üretildiklerinden daha önce bahsedilmişti. Problemlerin tamamına ait, statik halleri için optimum değerler bilinmektedir. Kesinlik değerleri hesaplanırken Min_F^T değerleri için, statik hallerinin optimum değerleri kabul edilmiştir. Çünkü dinamik versiyonları için bu değerler bir sınır teşkil etmektedirler. Dinamik versiyonları için bu değerlerden daha iyi bir çözüm değeri bulmak söz konusu değildir.

$$accuracy_{F,A}^{T} = \frac{Min_F^T}{F(best_A^T)} \qquad (5\text{-}9)$$

Tablo 5-3 Önerilen PSO Algoritmasının Kesinlik Değerlerinin Diğer Algoritmalar ile Karşılaştırılması

Problem	Kesinlik (Accuracy)			
	PSO	DAPSO	ACO	GA
c50	0,83	0,91	0,83	0,92
c75	0,85	0,86	0,83	0,85
c100	0,85	0,84	0,85	0,86
c100b	0,84	0,89	0,87	0,93
c120	0,82	0,82	0,74	0,80
c150	**0,84**	0,75	0,76	0,76
c199	0,72	0,79	0,73	0,78
f71	0,85	0,85	0,76	0,79
tai75a	0,91	0,89	0,88	0,91
tai75b	**0,96**	0,93	0,88	0,92
tai75c	0,89	0,87	0,82	0,90
tai75d	0,90	0,97	0,93	0,98
tai100a	0,87	0,91	0,86	0,91
tai100b	0,85	0,87	0,85	0,90
tai100c	**0,94**	0,92	0,90	0,91
tai100d	**0,92**	0,81	0,79	0,86
tai150a	0,86	0,90	0,84	0,92

tai150b	**0,92**	0,88	0,84	0,91
tai150c	0,84	0,86	0,83	0,90
tai150d	0,85	0,87	0,86	0,90
Ortalama	0,87	0,87	0,83	0,88

Kesinlik değerlerine bakarak önerilen algoritmanın literatürdeki diğer algoritmalarla karşılaştırıldığında tutarlı bir performans gösterdiği gözlemlenmektedir. Algoritmanın ortalama kesinlik değerleri diğer algoritmalarla yaklaşık olarak aynı seviyededir. Bu durum önerilen algoritmanın DARP test problemleri için iyi sonuçlar ürettiğinin bir göstergesidir.

5.4 Önerilen PSO Algoritmasının Bir Gerçek Hayat Problemine Uygulanması

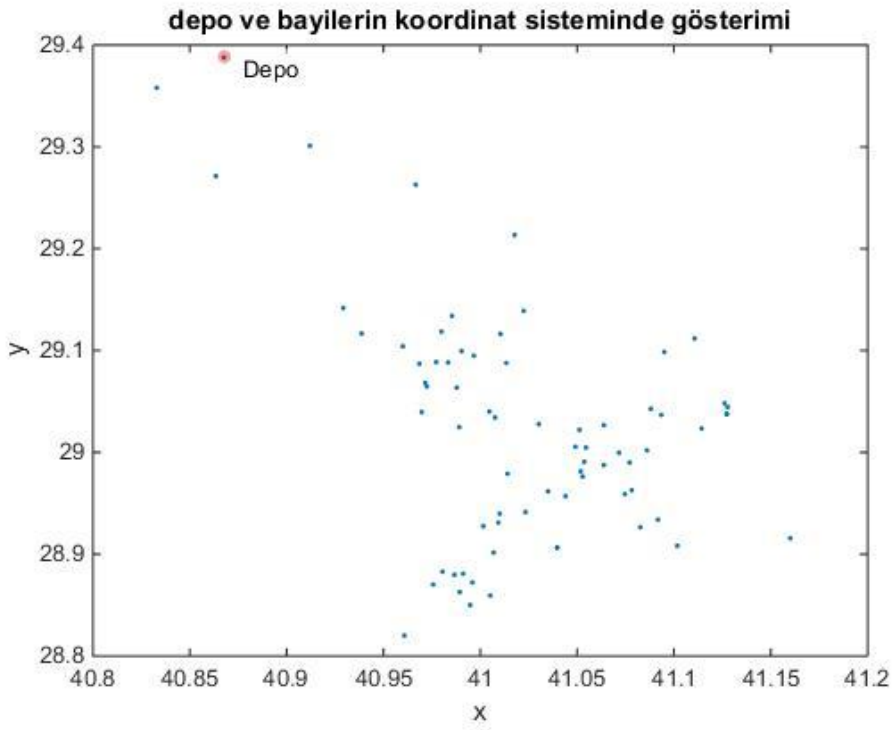
Bu çalışmada DARP için PSO algoritması kullanarak çözüm önerisi geliştirilmiş ve elde edilen sonuçlar literatürdeki diğer çalışmalar ile karşılaştırılmıştır. Literatürde bulunan test problemlerine ek olarak bir gerçek hayat problemi için çözüm önerisinde bulunulmuştur.

Gerçek hayat probleminin ait olduğu firma içme suyu dağıtım sektörüne ait bir firmadır. Uzun zamandır bu sektörde rekabet eden firma rakiplerine nazaran daha iyi bir rotalama yaparak maliyetleri azaltmak istemektedir. Tek bir noktada bulunan depodan bayilere servis hizmeti verilmektedir. Çalışmamızda bu deponun belirtilen bayilere damacana su dağıtımının en uygun rota ile yapılması için bir çözüm önerisi getirilmiştir.

Çeşitli yerlerde toplam 71 adet bayi bulunmaktadır. Firmanın eşit kapasiteli 20 adet aracı bulunmaktadır. İdeal olan rotalamada bu 20 adet aracın en minimal şekilde kullanılması olduğundan, en kötü senaryoda 20 araç ile servis verilmektedir. Eşit kapasitede olan her bir aracın 150 birim taşıma kapasitesi bulunmaktadır.

Bir günlük planlar halinde rotalama yapılmaktadır. Her bir müşterinin talepleri ve taleplerin ortaya çıkma zamanları firmadan temin edilmiştir.

Deponun ve diğer 71 bayinin koordinat sistemi üzerinde dağılımı Şekil 5-4'de gösterildiği gibidir.



Şekil 5-4 Depo ve Bayilerin Koordinat Sisteminde Gösterimi

Firma bilgilerinin gizliliğine sadık kalmak adına, harita üzerinde konum göstermek yerine koordinat sistemi tercih edilmiştir. Bayiler ve depoların koordinat sistemindeki konumları incelendiğinde, geniş bir bölgeye yayılmış servis ağı olduğu gözlemlenmektedir. Depo, geleneksel araç rotalama problemlerinin aksine merkezi bir konumda değildir. Genellikle test problemlerinde, depo servis alanının orta bölgesinde yer almaktadır. Deponun yerleşim alanlarının dışında yer alması, şehir planlaması ve şirket dinamikleri açısından tercih edilmiş olabilir.

Bayilerin depodan talep ettikleri ürün miktarları ve depoyu bilgilendirme zamanlarına ait veriler şirketten tedarik edilmiştir. Dinamik araç rotalama problemi uygulaması olarak önerilen PSO algoritması ile çözümler elde edildikten sonra mevcut durum ile karşılaştırmalar yapılmıştır.

Problemin matematiksel formülasyonu, kapasite kısıtlı araç rotalama problemi olarak modellendiğinde özel olarak probleme ait parametre değerleri:

Düğüm Sayısı (depo dahil) $n = 72$

Özdeş Araçların Kapasitesi $Q = 150$

i düğümünden j düğümüne gitme maliyeti c_{ij}: İki konum arasındaki mesafe Öklid Uzaklıkları[7] hesaplanarak bulunmuştur.

d_i: i düğümünde yer alan müşterinin talep miktarı

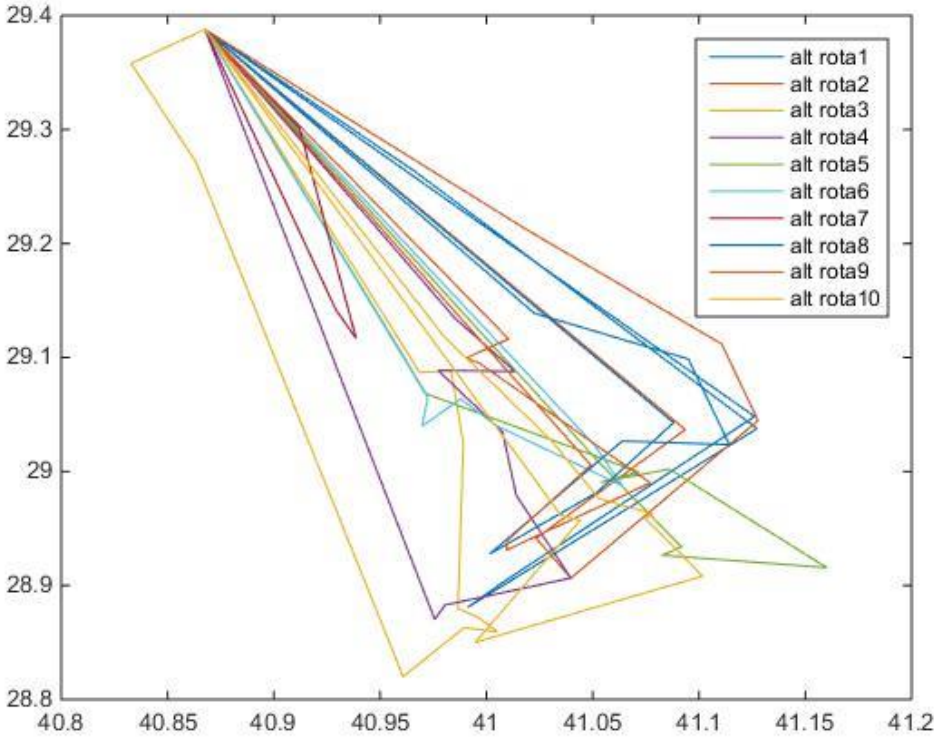
Amaç fonksiyonu olarak rota uzunluğunun maliyeti belirlenmiştir. PSO algoritması kullanılarak çözülen problemde, Olay Yöneticisi ve çalışma gününü eşit zaman aralıklarına bölme yöntemi kullanılmıştır. PSO algoritması parametreleri, test problemlerinin çözümünde kullanıldığı şekildedir. Popülasyon büyüklüğü 10, iterasyon sayısı 1000, atalet ağırlığı 1, öğrenme sabitlerinin her biri 1 olarak seçilmiştir.

Mevcut durumdaki rotalar Şekil 5-5 ile gösterildiği gibidir. Toplamda 10 adet alt rotanın belirlenmiş olduğu mevcut durumda toplam maliyet 1109,84 km olarak hesaplanmıştır.

PSO algoritması sonucunda elde edilen çözüm önerisi Şekil 5-6'de verilmektedir. Alt rota sayısı yine 10 olmakla beraber toplam maliyet 963,99 km olup, %15 oranında iyileştirme gözlenmektedir.

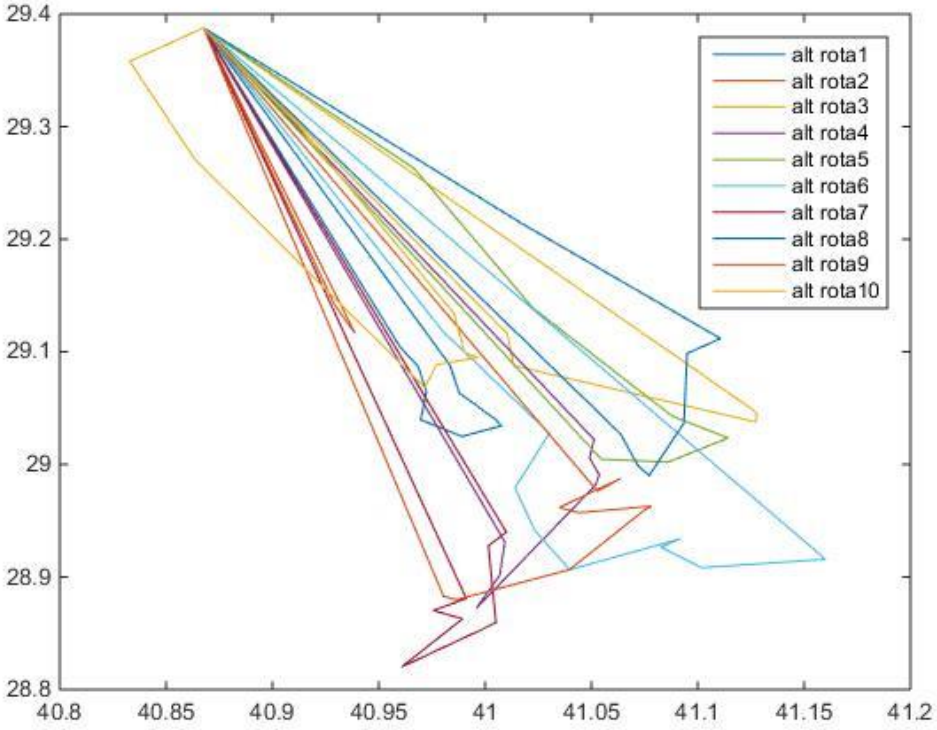
Ulaşım giderlerinin %15 azalması şirket adına çok önemli bir kazanç sayılmaktadır. Bu avantaj aynı zamanda şirketin rakipleri karşısında daha öne çıkmasına olanak sağlamaktadır. Şirket istediği takdirde rotalamalar anlık olarak sürücülere bildirilebilir.

[7] $A = (x_1, y_1) ve$ B $= (x_2, y_2)$ olmak üzere AB arasındaki uzaklık $|AB| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$



Alt-Rota 1:	0-26-34-35-28-24-68-0
Alt-Rota 2:	0-22-4-31-27-46-67-0
Alt-Rota 3:	0-56-53-49-1-39-41-36-43-70-71-0
Alt-Rota 4:	0-37-40-42-38-48-55-60-63-0
Alt-Rota 5:	0-12-11-9-2-25-17-19-52-62-0
Alt-Rota 6:	0-14-10-47-51-50-54-0
Alt-Rota 7:	0-66-64-69-0
Alt-Rota 8:	0-21-13-5-15-23-45-65-0
Alt-Rota 9:	0-16-33-32-18-58-57-59-0
Alt Rota 10:	0-7-20-3-8-30-29-6-44-61-0

Şekil 5-5 Şirketin Belirlemiş Olduğu Rotalar



Alt-Rota 1:	0-62-56-54-50-49-48-47-51-53-0
Alt-Rota 2:	0-66-64-0
Alt-Rota 3:	0-26-27-24-28-60-59- 0
Alt-Rota 4:	0-14-16-17-13-39-34-32-0
Alt-Rota 5:	0-12-25-23-21-65-68-0
Alt-Rota 6:	0-2-8-9-11-31-4-38-44-61-0
Alt-Rota 7:	0-35-37-36-43-3-41-5-33-0
Alt-Rota 8:	0-15-19-18-22-45-46-67-0
Alt-Rota 9:	0-6-10-7-20-29-30-42-1-40-0
Alt Rota 10:	0-69-63-57-58-55-52-70-71-0

Şekil 5-6 PSO Algoritması ile Elde Edilen Rotalar

SONUÇ VE ÖNERİLER

Geleneksel Araç Rotalama Problemi, uzun yıllardır üzerinde çalışılan ve hala geliştirilmeye müsait bir alandır. Öte yandan gerçek hayat problemlerinde, planlama esnasında değişiklikler gerekebilmektedir. Örneğin araç arızaları, yol çalışmaları veya müşterilerin taleplerindeki değişiklikler planlanan rotaların değiştirilmesini gerektirir.

Bu tez çalışmasında ulaştırma sektöründe çok önemli bir yer tutan Dinamik Araç Rotalama Problemleri ele alınmıştır. Dinamik Araç Rotalama Problemleri, kısıtları ve amaç fonksiyonlarına göre alt problemlere ayrılmaktadırlar. Bu doğrultuda ilk olarak DARP türlerinden bahsedilmiştir. İlgilenilen problem Kapasite Kısıtlı DARP olduğundan dolayı bu problem türüne ait matematiksel model verilmiştir.

DARP için literatürde geliştirilmiş olan çözüm yöntemleri; strateji temelli algoritmalar, sezgisel yöntemler ve meta sezgisel yöntemler olarak sınıflandırıldıktan sonra her bir çözüm yöntemi detaylı bir şekilde anlatılmıştır. Çözüm yöntemleri ve problem türlerine göre kapsamlı bir literatür araştırması sunulmuştur.

DARP çözüm önerisi için meta sezgisel algoritmalar içerisinde NP-Zor sınıfına ait optimizasyon problemlerine başarılı bir şekilde uygulanmış olan Parçacık Sürü Optimizasyon algoritmasının yapısı anlatılmıştır. Önerilen çözüm tasarımı planlama periyodunu eşit alt zaman dilimlerine bölmeye dayanmaktadır. Her bir zaman diliminde o ana kadar açığa çıkan problem verileri kullanılarak statik problemler üretilir ve çözümleri elde edilir. Yapı içerisinde PSO algoritmasının işleyişi ve çözüm gösterimi önemli bir yer tutmaktadır.

PSO ile elde edilen çözümleri iyileştirmek adına yerel arama yöntemleri kullanılmıştır. Bir rota kendi içerisinde iyileştirilebileceği gibi rotalar arası elemanların da yer değişikliği yapılabilmektedir. Bu açıdan yaygın olarak kullanılan yerel arama yöntemleri çalışmamızda kullanılmıştır ve son aşamada probleme özgü bir sezgisel algoritma geliştirilmiştir. PSO algoritması ile beraber 2-Opt ve yer değiştirme yerel arama algoritmaları önerilen çözüm tasarımı içerisinde kullanılarak DARP için bir çözüm önerisinde bulunulmuştur.

Önerilen çözüm algoritması müşteri sayısı 50 ile 199 arasında değişen 25 farklı test problemi üzerinde çalıştırılarak geliştirilen çözümün kalitesi ve etkinliği

ölçülmüştür. Test problemleri DARP ile ilgili çalışmalarda kullanılan, kolay ulaşılabilen ve optimal değerleri bilinen veri setleridir. Aynı test problemleri için Genetik Algoritma, Karınca Kolonisi Algoritması ve Dinamik Uyarlanabilir PSO algoritması kullanılarak çözüm önerisi getiren çalışmalar bulunmaktadır. Her bir problem için toplam maliyet araçların gitmiş oldukları yol olarak belirlenmiş ve maliyetler açısından karşılaştırmalı sonuçlar raporlanmıştır. Bu noktada önerilen PSO algoritmasının 25 farklı problem içerisinde sekiz problemde bilinen en iyi sonuçlar elde ettiği görülmüştür. En iyi olduğumuz bu sekiz problem, müşteri sayısı bakımından değerlendirildiğinde diğerlerine göre büyük boyutlu problemlerdir (müşteri sayısı 75 ile 199 arasında değişmektedir).

Algoritmanın performansının ölçülmesi için literatürde kullanılan belirli başlı metrikler yer almaktadır. Bunlar içerisinde Kesinlik ölçüsü bir minimizasyon problemi için "Bilinen En İyi" değer ile algoritma tarafından "Bulunan En İyi" değerin oranlanması şeklinde hesaplanmaktadır. Önerilen PSO algoritması test problemleri için kesinlik değerinin 0,72- 0,96 arasında değer almaktadır ki optimal değer bulunduğunda bu sonuç 1 olmaktadır. Bu açıdan değerlendirildiğinde algoritmanın performansı başarılı olarak yorumlanabilir.

Çalışmanın son bölümünde ise bir gerçek hayat problemi ele alınmıştır. Müşterilerinden telefonla talep bildirimi alan ve dağıtım yapan bir firmanın günlük araç rotalama verileri kullanılarak maliyet minimizasyonu yapılmıştır. PSO algoritması ile elde edilen sonuçlar firmanın hali hazırda kullanmış olduğu dağıtım rotaları ile karşılaştırılmıştır. Sonuçlar gidilen yol uzunluğu olarak değerlendirildiğinde %15 daha az maliyetli bir sonuç elde edilmiştir. Firma bünyesinde ortalama %15'e varan tasarruflar sağlanmıştır.

Tez çalışması uluslararası literatürde yeni çalışılmaya başlanmış olan dinamik optimizasyon problemleri içerisinde önemli bir yer tutan DARP ile ilgili karşılaştırılabilir sonuçlar sunmaktadır. Türkçe literatür tarandığında bu konuda daha önce çok az çalışma olduğu görülmüştür. Ayrıca bu kaynaklarda bir gerçek hayat uygulaması bulunmamaktadır. Bu alanda yapmış olduğumuz çalışmalar akademik referans olarak bir kaynak aracı olarak kullanılabilecektir. Tez kapsamında yapılan

çalışmalar bilişim teknolojileri alanında projelendirilmesi ve gerçek hayata geçirilme olasılığı yüksektir.

Görsel bir çıktı üretmesi amacı ile algoritmanın çözümleri harita üzerinde anlık gösterilebilir ve kullanıcılara sunulabilir. Mobil cihazlar üzerinde geliştirilecek bir uygulama ile sonuçlar anlık olarak paylaşılabilir. Trafik bilgisi sisteme entegre edilebilir. Araç arızaları ve benzeri kısıtlar eklenebilir.

Çalışmalar neticesinde görülmüştür ki; geliştirilen algoritma farklı sektörlerdeki firmalarda da kullanılabilir. Finans, enerji, lojistik gibi alanlardaki gerçek hayat problemlerinde uygulanabilir.

İleriki çalışmalarda ise, çözüm algoritmasında kullanılan parametrelerin seçimine ağırlık veren parametre optimizasyonları yapılabilir. Dinamik optimizasyon yeni gelişen bir konu olması sebebiyle henüz çalışılmamış farklı problem türleri içermektedir. Önerilen çözüm tasarımı bu problemlerin çözümü için kullanılabilir. Ek olarak DARP için farklı meta sezgisel yöntemler denenerek çalışma genişletilebilir.

KAYNAKÇA

Alvarenga, Guilherme Bastos, Ricardo Martins De Abreu Silva, Geraldo Robson Mateus: "A hybrid approach for the dynamic vehicle routing problem with time windows", **Fifth International Conference on Hybrid Intelligent Systems**, HIS'05, IEEE, 2005.

Archetti, Claudia, Maria Grazia Speranza: "The split delivery vehicle routing problem: A survey", **The vehicle routing problem: Latest advances and new challenges** içinde, yazan Bruce Golden, S. Raghavan ve Edward Wasil, 2008, Springer, s.103-122.

Arumugam, M. Senthil, M.V.C. Rao: "On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems", **Discrete Dynamics in Nature and Society,** 2006

Attanasio, A., J. F. Cordeau, G. Ghiani, G. Laporte: "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem", **Parallel Computing** C.30, No:3, 2004, s.377-387.

Azi, Nabila, Michel Gendreau, Jean-Yves Potvin: "A dynamic vehicle routing problem with multiple delivery routes", **Annals of Operations Research** C.199, No:1, 2012, s.103-112.

Bansal, J. S., P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, A. Abraham: "Inertia weight strategies in particle swarm optimization", **Proceeding of the Third World Congress on In Nature and Biologically Inspired Computing** (NaBIC), IEEE, 2011, s.633-640.

Bent, Russell, Pascal Van Hentenryck: "Dynamic vehicle routing with stochastic requests", **International Joint Conference on Artificial Intelligence**, 2003, s.1362-1363.

Bent, Russell, Pascal Van Hentenryck: "Waiting and Relocation Strategies in Online Stochastic Vehicle Routing" **Proceeding of the 20th international joint conference on Artificial intelligence**, Morgan Kaufmann Publishers INc., 2007, s.1816-1821.

Bertsimas, Dimitris J., Garrett Van Ryzin: "A stochastic and dynamic vehicle routing problem in the Euclidean plane", **Operations Research,** C.39, No:4, 1991, s.601-615.

Bertsimas, Dimitris J., Garrett Van Ryzin	"Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles", **Operations Research,** C.41, No:1, 1993, s.60-76.
Bianchi, Leonora:	"**Notes on dynamic vehicle routing-the state of the art**", 2000.
Branchini, Rodrigo Moretti, Vinicius Amaral Armentano, Arne Løkketangen:	"Adaptive granular local search heuristic for a dynamic vehicle routing problem", **Computers & Operations Research,** C.36, No:11, 2009, s. 2955-2968.
Branke, Jürgen:	"**Evolutionary optimization in dynamic environments**", Springer Science Business Media, 2012.
Chen, G., X. Huang, J. Jia, Z. Min:	"Natural exponential inertia weight strategy in particle swarm optimization", **In the Proceeding of The Sixth World Congress on Intelligent Control and Automation,** IEEE, 2006, s.3672-3675
Chen, Huey-Kuo, Che-Fu Hsueh, Mei-Shiang Chang:	"The real-time time-dependent vehicle routing problem", **Transportation Research Part E: Logistics and Transportation Review**, C.42, No:5, 2006, s.383-408.
Chen, Zhi-Long, Hang Xu:	"Dynamic column generation for dynamic vehicle routing with time windows", **Transportation Science**, C.40, No:1, 2006, s.74-88.
Chitty, Darren M., Marcel L. Hernandez:	"A hybrid ant colony optimisation technique for dynamic vehicle routing", **Genetic and Evolutionary Computation–GECCO**, Springer Berlin Heidelberg , 2004, s.48-59.
Christofides, Nicos, John E. Beasley:	"The period routing problem", **Networks**, C.14, No:2, 1984, s.237-256.
Colorni, A., M. Dorigo, V. Maniezzo:	"Distributed optimization by ant colonies", **In Proceedings of the first European conference on artificial life**, Elsevier Publishing, 1991, s.134-142.

Cordeau, Jean-Françcois, Gilbert Laporte, Stefan Ropke:	**"Recent models and algorithms for one-to-one pickup and delivery problems",** Springer, 2008
Cordeau, Jean-François, Michel Gendreau, Gilbert Laporte:	"A tabu search heuristic for periodic and multi-depot vehicle routing problems", **Networks**, C.30, No:2, 1997, s.105-119.
Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, Clifford Stein:	**Introduction to Algorithms**, Cilt 3. Cambridge: MIT Press, 2009.
Cruz, Carlos, Juan R. González, David A Pelta:	"Optimization in dynamic environments: a survey on problems, methods and measures", **Soft Computing,** C.15, No:7, 2011, s.1427-1448.
Dan, B., W. Zhu, H. Li, Y. Sang, Y Liu:	"Dynamic optimization model and algorithm design for emergency materials dispatch", **Mathematical Problems in Engineering,** 2013
Dantzig, George B., John H. Ramser:	"The truck dispatching problem", **Management science**, C.6, No:1, 1959, s.80-91.
Du, Timon C., Eldon Y. Li, Defrose Chou:	"Dynamic vehicle routing for online B2C delivery", **Omega**, 2005, C.33, s.30-45.
Eberhart, Russell C., Yuhui Shi:	"Particle swarm optimization: developments, applications and resources", **Proceedings of the Congress on Evolutionary Computation**. 2001, s.81-86.
Eberhart, Russell C., Yuhui Shi	"Tracking and optimizing dynamic systems with particle swarms", **Proceedings of the Congress on Evolutionary Computation,** IEEE, 2001, s.94-100.
Elhassania, M. J., Boukachour Jaouad, Elhilali Alaoui Ahmed:	"A new hybrid algorithm to solve the vehicle routing problem in the dynamic environment", **International Journal of Soft Computing,** C.8, No:5, 2013, s.327-334.

Ercan, Cihan, Cevriye Gencer: "Dinamik İnsansız Hava Sistemleri Rota Planlaması Literatür Araştırması ve İnsansız Hava Sistemleri Çalışma Alanları", **Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi** C.19, No:2, 2013, s.104-111.

Fabri, Anke, Peter Recht: "On dynamic pickup and delivery vehicle routing with several time windows and waiting times", **Transportation Research Part B: Methodological**, C.40, No:4, 2006, s.335-350.

Feng, Y., G. F. Teng, A. X. Wang, Y. M. Yao: "Chaotic inertia weight in particle swarm optimization", **Second International Conference on Innovative Computing, Information and Control,** IEEE, 2007, s.475-475.

Feo, Thomas A., Jonathan F. Bard: "Flight scheduling and maintenance base planning", **Management Science,** 1989, C.35, No:12, s.1415-1432.

Fisher, Marshall: "Vehicle routing", **Handbooks in operations research and management science içinde**, yazan Michael Ball, Tom Magnanti, Clyde Monma, George Nemhauser, C.8, Elsevier, 1995, s.1-33.

Fleischmann, Bernhard, Stefan Gnutzmann, Elke Sandvoß: "Dynamic vehicle routing based on online traffic information", **Transportation Science,** C.38, No:4, 2004, s.420-433.

Gendreau, M., F. Guertin, J. Y. Potvin, E. Taillard: "Parallel tabu search for real-time vehicle routing and dispatching", **Transportation Science**, C.33, No:4, 1999, s.381-390.

Gendreau, M., F. Guertin, J. Y. Potvin, R. Séguin: "Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries", **Transportation Research Part C: Emerging Technologies**, C.14, No:3, 2006, s.157-174.

Gendreau, Michel, Gilbert Laporte, René Séguin: "Stochastic vehicle routing", **European Journal of Operational Research**, C.88, No:1, 1996, s.3-12.

Ghiani, G., F. Guerriero, G. Laporte, R Musmanno: "Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies", **European Journal of Operational Research** C.151, No:1, 2003, s.1-11.

Glover, Fred: "Future paths for integer programming and links to artificial intelligence", **Computers and operations research**, C.13, No:5, 1986, s.533-549.

Haghani, Ali, Soojung Jung: "A dynamic vehicle routing problem with time-dependent travel times", **Computers and Operations Research**, C.32, No:11, 2005, s.2959-2986.

Hanshar, Franklin T., Beatrice Ombuki-Berman M.: "Dynamic vehicle routing using genetic algorithms", **Applied Intelligence,** C.27, No:1, 2007, s.89-99.

Holland, J. H.: **"Adaption in natural and artificial systems",** Ann Arbor MI: The University of Michigan Press, 1975.

Housroum, H., T. Hsu, R. Dupas, G. Goncalves: "A hybrid ga approach for solving the dynamic vehicle routing problem with time windows", **Information and Communication Technologies ICTTA'06,** IEEE, 2006, s.787-792.

Hvattum, Lars M., Arne Løkketangen, Gilbert Laporte: "Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic", **Transportation Science,** C.40, No:4, 2006, s.421-438.

Jih, Wan-Rong, Jane Yung-jen Hsu: "Dynamic vehicle routing using hybrid genetic algorithms", **In Proceedings of International Conference on Robotics and Automation.** IEEE, 1999, s.453-458.

Jun, Qin, Jiangqing Wang, Bo-jin Zheng: "A hybrid multi-objective algorithm for dynamic vehicle routing problems", **In Proceeding of Computational Science,** Springer Berlin Heidelberg, 2008, s.674-681.

Kennedy, James, Russell Eberhart: "Particle Swarm Optimization", **Proceeding of the international conference on neural networks**, IEEE, 1995, s.1942-1948.

Kergosien, Y., C. Lente, D. Piton, J. C Billaut: "A tabu search heuristic for the dynamic transportation of patients between care unit" **European Journal of Operational Research** C.214, No:2, 2011, s.442-452.

Khouadjia, M. R., B. Sarasola, E. Alba, L. Jourdan, E. G. Talbi: "A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests", **Applied Soft Computing,** C.12, No:4, 2012, s.1426-1439.

Kilby, Philip, Patrick Prosser, Paul Shaw: "Dynamic VRPs: A study of scenarios", **University of Strathclyde Technical Report**, 1998, s.1-11.

Laporte, G., M. Gendreau, J. Y. Potvin, F. Semet: "Classical and modern heuristics for the vehicle routing problem", **International transactions in operational research**, 2000, C.7, No:4-5, s.285-300.

Laporte, Gilbert: "The vehicle routing problem: An overview of exact and approximate algorithms", **European Journal of Operational Research**, C.59, No:3, 1992, s.345-358.

Larsen, Allan: "**The Dynamic Vehicle Routing Problem**" Doktora Tezi Institute of Mathematical Modelling, Technical University of Denmark, 2000.

Larsen, Allan, Oli BG Madsen, Marius M Solomon: "The a priori dynamic traveling salesman problem with time windows", **Transportation Science,** C.38, No:4, 2004, s.459-472.

Li, Feiyue, Bruce Golden, Edward Wasil: "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results", **Computers and Operations Research,** C.34, No:10, 2007, s.2918-2930.

Liao, Tsai-Yun, Ta-Yin Hu: "An object-oriented evaluation framework for dynamic vehicle routing problems under real-time information", **Expert Systems with Applications,** C.38, No:10, 2011, s.12548-12558.

Lin, Shen: "Computer solutions of the traveling salesman problem", **Bell System Technical Journal,** C.44, No:10, 1965, s.2245-2269.

Lorini, Sandro, Jean-Yves Potvin, Nicolas Zufferey: "Online vehicle routing and scheduling with dynamic travel times", **Computers and Operations Research**, C.38, No:7, 2011, s.1086-1090.

Lund, Karsten, Oli B.G. Madsen, Jens M. Rygaard: "**Vehicle routing with varying degree of dynamism**". The Department Informatics of Mathematical Modelling, Technical University of Denmark, 1996.

Malik, R. F., T. A. Rahman, S. Z.M. Hashim, R. Ngah: "New particle swarm optimizer with sigmoid increasing inertia weight", **International Journal of Computer Science and Security**, C.1, No:2, 2007, s.35-44.

Mavrovouniotis, Michalis, Shengxiang. Yang: "Ant Colony Optimization with Memory-Based Immigrants for the Dynamic Vehicle Routing Problem", **Congress on Evolutionary Computation (CEC),** IEEE, 2012, s.1-8.

MirHassani, S. A., N. Abolghasem: "A particle swarm optimization algorithm for open vehicle routing problem", **Expert Systems with Application,** C.38, No:9, 2011, s.11547-11551.

Mitrović-Minić, Snežana, Ramesh Krishnamurti, Gilbert Laporte: "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows", **Transportation Research Part B: Methodological**, C.38, No:8, 2004, s.669-685.

Mladenović, Nenad, Pierre Hansen: "Variable neighborhood search", **Computers and Operations Research,** C.24, No:11, 1997, s.1097-1100.

Montemanni, R., L. M. Gamberdella, A. E. Rizzoli, A. V. Donati: "Ant Colony System for a Dynamic Vehicle Routing Problem", **Journal of Combinatorial Optimization**, C.10, No:4, 2005, s.327-343.

Morrison, Ronald W: "Performance measurement in dynamic environments", **GECCO workshop on evolutionary algorithms for dynamic optimization** içinde, 2003, s.5-8.

Nagy, Gabor, Saïd Salhi: "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries", **European journal of operational research,** C.162, No:1, 2005, s.126-141.

Nikabadi, A., M. Ebadzadeh: "Particle swarm optimization algorithms with adaptive Inertia Weight: A survey of the state of the art and a Novel method", **Journal of evolutionary computation,** 2008.

Novoa, Clara, Robert Storer: "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands", **European Journal of Operational Research,** C.196, No:2, 2009, s.509-515.

Osman, Ibrahim Hassan: "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", **Annals of operations research,** C.41, No:4, 1993, s.421-451.

Pankratz, Giselher: "Dynamic vehicle routing by means of a genetic algorithm", **International Journal of Physical Distribution and Logistics Management,** C.35, No:5, 2005, s.362-383.

Papastavrou, Jason D.: "A stochastic and dynamic routing policy using branching processes with state dependent immigration", **European Journal of Operational Research,** C.95, No:1, 1996, s.167-177.

Pavone, M., N. Bisnik, E. Frazzoli, V. Isler: "A stochastic and dynamic vehicle routing problem with time windows and customer impatience", **Mobile Networks and Applications**, C.14, No:3, 2009, s.350-364.

Pavone, Marco, Emilio Frazzoli, Francesco Bullo: "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment", **IEEE Transactions on Automatic Control,** C.56, No:6, 2011, s.1259-1274.

Pillac, V., M. Gendreau, C. Guéret, A. L. Medaglia: "A review of dynamic vehicle routing problems", **European Journal of Operational Research,** C.225, No:1, 2013, s.1-11.

Poli, Riccardo, James Kennedy, Tim Blackwell: "Particle swarm optimization", **Swarm Intelligence**, C.1, No:1, Springer, 2007, s.33-57.

Prins, Christian: "A simple and effective evolutionary algorithm for the vehicle routing problem", **Computers and Operations Research,** C.31, No:12, 2004, s.1985-2002.

Psaraftis, Harilaos N.: "Dynamic vehicle routing: Status and prospects" **Annals of Operations Research**, C.61, 1995, s.143-164.

Psaraftis, Harilaos N., Min Wen, Christos A. Kontovas: "Dynamic vehicle routing problems: Three decades and counting", **Networks**, (online baskı), 2015, http://onlinelibrary.wiley.com/doi/10.1002/net.21628/epdf

Psaraftis, Hariloas N.: "Dynamic vehicle routing problems", **Vehicle Routing: Methods and studies** içinde, düzenleyen: B.L. Golden, A. A. Assad, 1988, s.223-248. North-Holland.

Rizzoli, A. E., R. Montemanni, E. Lucibello, L. M. Gambardella. "Ant colony optimization for real-world vehicle routing problems", **Swarm Intelligence,** C.1, No:2, 2007, s.135-151.

Schilde, M., K. F. Doerner, R. F Hartl: "Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem", **European journal of operational research,** C.238, No:1, 2014, s.18-30.

Shi, Yuhui, Russell C. Eberhart: "Empirical study of particle swarm optimization", **Proceedings of the Congress on Evolutionary Computation**, IEEE, 1999, s.945-1950.

Shi, Yuhui, Russell C. Eberhart: "Parameter selection in particle swarm optimization", **Evolutionary programming VII** içinde, Springer Berlin Heidelberg, 1998a, s.591-600.

Shi, Yuhui, Russell Eberhart: "A modified particle swarm optimizer", **Evolutionary Computation Proceedings, World Congress on Computational Intelligence**, IEEE, 1998b, s.69-73.

Taillard, Éric: "Parallel iterative search methods for vehicle routing problems", **Networks**, C.23, No:8, 1993, s.661-673.

Talbi, El-Ghazali: **"Metaheuristics: From Design to Implementation",** C.74, John Wiley & Sons, 2009.

Taniguchi, Eiichi, Hiroshi Shimamoto: "Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times" **Transportation Research Part C: Emerging Technologies,** C.12, No:3, 2004, s.235-250.

Tian, Y., J. Song, D. Yao, J. Hu: "Dynamic vehicle routing problem using hybrid ant system", **In the Proceedings of Intelligent Transportation Systems,** IEEE, 2003, s.970-974.

Toth, Paolo, Daniele Vigo: "The vehicle routing problem", **Society for Industrial and Applied Mathematics,** 2001.

Van Breedam, Alex: "**An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints**", RUCA, 1994.

van Veen, B., M. Emmerich, Z. Yang, T. Bäck, J. Kok: "Ant Colony Algorithms for the Dynamic Vehicle Routing Problem with Time Windows", **Natural and Artificial Computation in Engineering and Medical Applications** içinde, Springer Berlin Heidelberg, 2013, s. 1-10.

Wang, W., B. Wu, Y. Zhao, D Feng: "Particle swarm optimization for open vehicle routing problem", **Computational Intelligence** içinde, Springer, 2006, s.999-1007.

Weicker, Karsten: "Performance measures for dynamic environments", **Parallel Problem Solving from Nature—PPSN VII** içinde, Springer Berlin Heidelberg, 2002, s.64-73.

Wen, M., J. F. Cordeau, G. Laporte, J. Larsen: "The dynamic multi-period vehicle routing problem", **Computers and Operations Research,** C.37, No:9, 2010, s.1615-1623.

Woodruff, David L.: "**Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search",** Springer, 1998.

Xin, Jianbin, Chen Guimin, Yubao Hai: "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight", **International Joint Conference on Computational Sciences and Optimization,** IEEE, 2009, s.505-508.

Xu, Yingcheng, Li Wang, Yuexiang. Yang: "Dynamic vehicle routing using an improved variable neighborhood search algorithm", **Journal of Applied Mathematics**, 2013.

Yeun, L. C., W. R. Ismail, K. Omar, M. Zirour: "Vehicle Routing Problem: Models And Solutions", **Journal of Quality Measurement and Analysis,** C.4, No:1, 2008, s.205-218.

Yu, Xinjie, Mitsuo Gen: "**Introduction to evolutionary algorithms**", Springer Science and Business Media, 2010.

ÖZGEÇMİŞ

Yonca ERDEM DEMİRTAŞ, 1986 yılında İstanbul'da doğdu. İlköğretimini Yeşilköy Şehit Pilot Muzaffer Erdönmez İlköğretim Okulu'nda tamamladı. 2000 yılında Beşiktaş Sakıp Sabancı Anadolu Lisesi'nde orta öğrenimine başladı ve 2004 yılında orta öğrenimini tamamladı. Aynı yıl Marmara Üniversitesi Fen Edebiyat Fakültesi Matematik Bölümü'nde Lisans öğrenimine başladı. 2008 yılında Matematik bölümünden bölüm beşincisi olarak mezun oldu. 2008 yılında Marmara Üniversitesi Fen Bilimler Enstitüsü Endüstri Mühendisliği (ing.) Anabilim Dalı'nda Yüksek Lisans öğrenimine başladı. "Workforce Planning for Seasonal Demands" başlıklı tezi ile Yüksek Lisans öğrenimini 2011 yılında tamamladı. Aynı yıl İstanbul Üniversitesi Sosyal Bilimler Enstitüsü Sayısal Yöntemler Anabilim Dalı'nda Doktora öğrenimine başladı. 2008 Aralık ayından itibaren İstanbul Üniversitesi İşletme Fakültesi Sayısal Yöntemler Anabilim Dalı'nda Araştırma Görevlisi olarak hizmet vermektedir. Evli ve bir çocuk annesidir.