Süleyman Demirel Üniversitesi
İktisadi ve İdari Bilimler
Fakültesi Dergisi
Y.2017, C.22, S.3, s.807-823.

Suleyman Demirel University
The Journal of Faculty of Economics
and Administrative Sciences
Y.2017, Vol.22, No.3, pp.807-823.

DİNAMİK ARAÇ ROTALAMA PROBLEMLERİ İÇİN YENİ BİR ÇÖZÜM ÖNERİSİ[1]

A NOVEL APPROACH FOR SOLUTION OF DYNAMIC VEHICLE ROUTING PROBLEMS

Yonca ERDEM DEMİRTAŞ[*], Erhan ÖZDEMİR[**]

[*] Arş.Gör. Dr., İstanbul Üniversitesi, İşletme Fakültesi, Sayısal Yöntemler ABD, yncerdem@istanbul.edu.tr

[**] Emekli Prof. Dr., İstanbul Üniversitesi, İşletme Fakültesi, Sayısal Yöntemler ABD

ÖZ

Araç Rotalama Problemi (ARP) üzerinde çok uzun zamandır çalışılan bir optimizasyon problemidir. Bir ARP'de tüm problem girdileri önceden bilinir ve problem boyunca değişmezler. Her ne kadar ARP, NP-Zor sınıfında iyi bilinen statik bir problem olsa da gerçek hayatta benzeri problemler dinamik bir şekilde değişmektedir. Bu tip problemler Dinamik ARP (DARP) olarak isimlendirilmektedir. 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. Bu çalışmada, DARP incelenmiş ve Parçacık Sürü Optimizasyonu (PSO) yöntemi ile bir çözüm önerilmiştir. Literatürde çalışılan test problemleri PSO ile çözülmüş ve sonuçlar literatürde yer alan diğer yöntemler ile karşılaştırılmıştır. Ele alınan bir gerçek hayat probleminin çözümü için önerilen PSO algoritması uygulanarak elde edilen sonuçlar hali hazırda kullanılan rotalar ile karşılaştırılmaktadır.

Anahtar Kelimeler: *Dinamik Araç Rotalama Problemi, Meta Sezgisel Algoritmalar, Parçacık Sürü Optimizasyonu, Yerel Arama Teknikleri*

Jel Kodları: *C61, L9.*

ABSTRACT

Vehicle Routing Problem (VRP) is a well-known NP-Hard optimization problem that has been studied for many years. In the VRP, all information about the problem is known at the beginning of the solution procedure. It is basically a static problem, although many VRP problems change over time in the real world. These problems are called as Dynamic VRP (DVRP) in the literature. In most cases, the real world information can change or appear after the solution process begins. These characteristics make the DVRP harder than static VRP. In this study 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. The propose algorithm is used to solve a real world DVRP and the obtained results are compared with the route that is used by the factory.

Keywords: *Dynamic Vehicle Routing Problem, Meta Heuristics, Particle Swarm Optimization, Local Search Techniques.*

Jel Codes: *C61, L9.*

[1] Bu çalışma Prof.Dr. Erhan ÖZDEMİR danışmanlığında yürütülen ve Dr. Yonca ERDEM DEMİRTAŞ tarafından hazırlanan "Dinamik Araç Rotalama Problemine Parçacık Sürü Optimizasyonu Algoritması Çözüm Önerisi" Başlıklı Doktora tezinden yararlanılarak hazırlanmıştır.

1. GİRİŞ

Günümüz gerçek hayat problemlerinin birçoğu zaman faktörünü dikkate aldığımız müddetçe geçerlidir ve optimum çözüm veya çözüm uzayı zamana bağlı değişebilmektedir. Literatürde yer alan Araç rotalama problemlerine ait çalışmalar genellikle statik optimizasyon problemleri üzerine yoğunlaşmıştır. Statik versiyonları dahi 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 (Cormen v.d., 2009: 1049-1050).

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 yaygın olarak kullanılmaktadır. Dolayısıyla Dinamik ARP (DARP) için de analitik yöntemler etkisiz kalmakta ve meta sezgisel yöntemler önerilmektedir.

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. ARP gerçek hayat uygulaması çok yaygın olan bir kombinatoryal optimizasyon problemidir. Literatüre kazandırıldığı 1959 (Dantzig ve Ramser, 1959) tarihinden itibaren geniş bir kitle tarafından üzerinde çalışılmaktadır. Problemin zaman kavramını dâhil eden ve daha karmaşık bir yapıya sahip olan dinamik versiyonu Dinamik Araç Rotalama problemidir.

Dinamik ARP literatüre 1988 (Psaraftis, 1988) yılında tanıtılmıştır. Çalışmada dinamik ve statik araç rotalama problemleri arasındaki farklara değinilmiştir. Ayrıca bir kargo şirketinin araçlarının dinamik olarak rotalama problemini çözen bir algoritma önerilmiştir. Psaraftis 1995 yılında DARP ile ilgili geniş bir derleme çalışması yaparak bu alandaki literatüre katkı sağlamıştır (Psaraftis, 1995).

ARP için literatürde yer alan test problemleri Kilby v.d. (1998) tarafından geliştirilerek DARP için test problemleri üretilmiştir. Dinamik hale getirilen test problemlerine, çalışma günü uzunluğu, taleplerin ortaya çıkış zamanları, her bir talebin servis süresi gibi zamana bağlı parametreler eklenmiştir.

Montemanni v.d. (2005) aynı test problemlerini kullanarak Karınca Kolonisi Algoritması ile çözüm önerisi sunmuştur. Bu çalışmada, planlama periyodu eşit uzunlukta zaman aralıklarına bölünerek her bir zaman aralığı içerisinde o ana kadar ortaya çıkan problem girdileri ile statik problem üretilerek çözüm algoritması çalıştırılmıştır. Planlama periyodunun kaç eşit alt aralığa bölünmesi gerektiği ile ilgili test çalışmaları yapılmıştır.

Bu yaklaşım ilk olarak Montemanni v.d. (2005) tarafından önerilmiştir. Daha sonra Hanshar ve Ombuki-Berman (2007) Genetik Algoritma ve Khouadjia v.d. (2012) Parçacık Sürü Optimizasyonu ile aynı yaklaşımı kullanarak literatüre katkıda bulunmuşlardır.

Çalışmamızda PSO algoritması ile DARP problemine yeni bir çözüm önerisi sunulmaktadır. Bir meta sezgisel algoritmanın performansını belirleyen en önemli özellik çözüm gösterimi ve dolayısıyla çözüm uzayının nasıl tarandığıdır. Bu açıdan bakıldığında yakın zamanda (Khouadjia v.d., 2012) tarafından önerilen benzer bir çalışmayla karşılaştırıldığında özgün bir PSO algoritması önerdiğimizi söyleyebiliriz.

Önerilen yöntemde, çözüm gösterimi için reel sayılardan oluşan bir vektör kullanılmaktadır. PSO algoritmasıyla bulunan çözümler, performans açısından başarılı olduğu bilinen ve yaygın olarak kullanılan yerel arama teknikleriyle birlikte kullanılarak elde edilen sonuçlar karşılaştırılmıştır.

İlerleyen bölümlerde sırasıyla problemin tanımı ve matematiksel gösterimi anlatılacak, ardından 3. Bölümde PSO algoritması açıklanacak ve önerilen yöntemin detayları verilecektir. 4. Bölümde parametrelerin seçimi ifade edilip elde edilen sonuçlar literatürde bilinen diğer yöntemlerle karşılaştırılacaktır. Son olarak 5. Bölümde sonuçlar değerlendirilecek ve olası gelecek çalışmalardan bahsedilecektir.

2. DİNAMİK ARAÇ ROTALAMA PROBLEMİ

2.1. 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 az maliyetli turun bulunmasını hedefleyen bir rotalama problemidir.

ARP 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 v.d., 2009: 1115)

Dinamik ARP literatüre 1988 yılında kazandırılmıştır (Psaraftis, 1988). DARP'de yer alan "dinamik"lik kavramı karar vericiler için, araç rotaları bulunması ve güncellenmesinde 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 DARP 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 (Larsen, 2000). Problemle ilgili bilgiler müşterilerin coğrafi konumları, müşteri sayısı, talep veya arz miktarları ve müşterilerin servis edilme süreleri olarak ele alınmaktadır. ARP türleri için yapılan tüm sınıflandırmalar problemin dinamik versiyonu için genelleştirilebilir. Detaylı bilgi için yapılan çalışmalar incelenebilir (Bianchi, 2000), (Larsen, 2000), (Pillac v.d, 2013), (Kilby v.d., 1998).

Çalışmamızda ele aldığımız DARP türü kapasite kısıtlı araç rotalama probleminin dinamik halidir. Araç kapasite ve sayılarının bilinmesi ile birlikte, müşteri talepleri ve konumları zaman ilerledikçe açığa çıkmaktadır. Problem ele alınırken Montemanni v.d. (2005) tarafından önerilen yaklaşım kullanılmıştır. Buna göre planlama periyodu belirli sayıda zaman dilimine bölünerek, her bir zaman diliminde o ana kadar açığa çıkan problem girdileri ile üretilen ARP, çözüm algoritması ile çözülmektedir. Bu sebeple ARP türlerinden kapasite kısıtlı statik araç rotalama probleminin matematiksel modelinden bahsederek daha sonra problemin dinamik versiyonuna geçiş yapılarak dinamizm değerlendirmesi yapılmaktadır.

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, $A = \{(i,j): i,j \in V, i \neq j\}$ 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:

$V = \{0,1,...,n\}$: Düğüm kümesi,

$K = \{1,2,...,m\}$: Araç kümesi,

Q : Araç kapasitesi,

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

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

$$x_{ij}^k = \begin{cases} 1, \text{k aracı i. şehirden j. şehire gidiyorsa} \\ 0, \text{aksi halde.} \end{cases}$$

Amaç fonksiyonu:

$$Minimize \sum_{i=o}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} c_{ij}x_{ij}^k \quad \textbf{(1)}$$

Kısıtlar:

$$\sum_{i=0}^{n}\sum_{k=1}^{m} x_{ij}^k = 1, \forall j \in V \setminus \{0\}, \quad \textbf{(2)}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ij}^k = 1, \forall i \in V \setminus \{0\}, \quad \textbf{(3)}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{i0}^k = m, \quad \textbf{(4)}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{0j}^k = m, \quad \textbf{(5)}$$

$$\sum_{i=0}^{n} x_{ih}^k = \sum_{j=0}^{n} x_{hj}^k, \forall h \in V, k \in K, \quad \textbf{(6)}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} d_i x_{ij}^k \le Q, \forall k \in K \quad \textbf{(7)}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^k \le |S| - 1, \forall S \subseteq V \setminus \{0\}, \quad S \neq \varnothing, \quad \textbf{(8)}$$

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

Çalışmada ilgilenilen araç rotalama problemi simetrik 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 oluşturulmaktadır.

2.2. 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 v.d (1996) tarafından önerilen metrik en yalın olanıdır ve eşitlik (9) ile ifade edilmektedir.

$$dod = \frac{n_d}{n_s + n_d} \quad \textbf{(9)}$$

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.

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 (10) ile gösterildiği şekilde hesaplanmaktadır.

$$edod = \frac{\sum_{i=1}^{n_s+n_d} (t_i / T)}{n_s + n_d} \quad \textbf{(10)}$$

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.

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.

3. PSO ALGORİTMASI

PSO, 1995 yılında 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 ve Gen, 2010: 328).

PSO algoritmasında, çözüm uzayındaki aramalar popülasyon temelli yapılmaktadır, popülasyona sürü, sürü elemanlarının her birine de parçacık ismi verilmektedir. Her bir parçacık aslında probleme ait bir çözümü temsil etmektedir. Başlangıç popülasyonu rasgele seçilen parçacıklardan oluşmaktadır ve 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.

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:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nD} \end{bmatrix}$$

Pozisyon matrisindeki *i.* satır *i.* parçacığın pozisyonunu temsil eder ve $X_i = [x_{i1}, x_{i2}, ..., x_{iD}]$ olarak ifade edilir. Aynı parçacığın o anki en iyi uygunluk değerini elde ettiği kişisel en iyi pozisyonu $pbest_i = [p_{i1}, p_{i2}, ..., p_{iD}]$ şeklinde gösterilir. Popülasyonun, yani sürünün en iyi pozisyonu olan global en iyi $gbest = [g_1, g_2, ..., g_D]$ ile temsil edilir. Her bir iterasyonda tüm parçacıkların pozisyonu hız vektörüne göre güncellenir, *i.* parçacığın pozisyonundaki değişim miktarını veren hız vektörü ise $V_i = [v_{i1}, v_{i2}, ..., v_{iD}]$ 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 (1995) ın önermiş olduğu sonrasında (Shi ve

Eberhart, 1999)'ın geliştirmiş olduğu formülasyona göre *i.* parçacığın *k.* iterasyonda hız vektörünün *j.* elemanı denklem (11) şeklinde güncellenmektedir:

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 rand_1(p_{ij}^{k} - x_{ij}^{k}) + c_2 rand_2(g_j^{k} - x_{ij}^{k}) \quad \textbf{(11)}$$

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[2]. c_1, parçacığın kendi tecrübelerine göre hareket etmesini, c_2 ise sürüdeki diğer parçacıkların tecrübelerine göre hareket etmesini sağlar. Ayrıca $0 < w < 1$ olmak koşulu ile *w* atalet ağırlığıdır.[3] 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.

Hız vektörünün güncellenmesinden sonra *i.* parçacığın pozisyon vektörünün elamanları denklem (12) 'e göre güncellenir:

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \quad \textbf{(12)}$$

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
}
```

3.1. Önerilen Çözüm Tasarımı

DARP için çözüm önerisi geliştirilen çalışmamızda çözüm algoritması PSO kullanılmıştır. PSO'nun dinamik problemlere uygulanmasına literatürde yakın zamanda yeni yeni karşılaşılmaktadır. Çalışmada çözüm yaklaşımı olarak Montemanni v.d. (2005) tarafından 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. Çözüm yapısının işleyişi Şekil 1 ile verilmektedir.

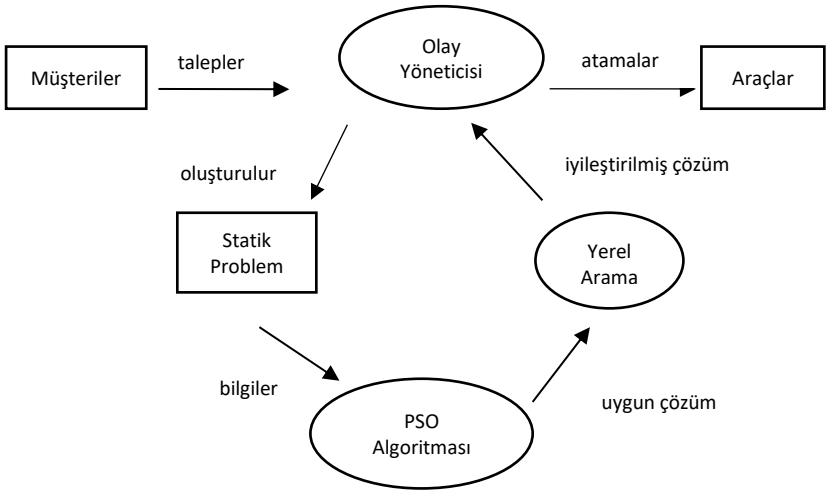
Çözüm stratejisi planlama periyodunun eşit uzunluktaki zaman dilimlerine bölünmesine dayanmaktadır. Tüm planlama periyodunun uzunluğu *T* ile gösterilecek olursa ve n_{ts} alt dönem sayısı olmak üzere; her bir alt dönemin uzunluğu T/n_{ts} olacaktır. Problem alt zaman aralıklarına ayrıldıktan sonra her 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.

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

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

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.

Şekil 1: Çözüm Yapısının İşleyişi



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 meta sezgisel bir yöntem olan Karınca Kolonisi Algoritması kullanılmıştır. Hanshar ve Ombuki-Berman (2007) tarafından yapılan çalışmada ise Genetik Algoritma çözüm safhasında

kullanılmıştır. Son yıllarda yapılmış olan Khouadjia ve arkadaşları (2012) tarafından yapılan çalışmada ise PSO ve tek çözüm tabanlı Değişken Komşuluk Arama algoritmaları karşılaştırmalı olarak önerilmiştir.

3.2. Parçacık Gösterimi

PSO 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 çözümleri 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 v.d. (2006) tarafından 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 sayıların 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 çalışmada kullanılan parçacık gösterimi Wang v.d.'nin (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.

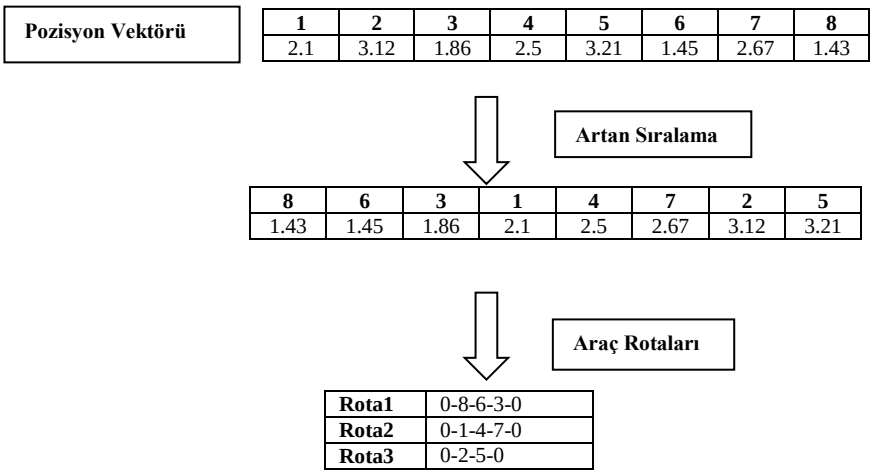
Dinamik ARP için idealde kaç araç kullanılacağının önceden belirlenmesi mümkün değildir. Seçmiş olduğumuz parçacık gösteriminde araç sayısı ile ilgili olarak yalnızca müşteri sayıları dikkate alınmaktadır, kaç tane araç kullanılacağı dinamik olarak değişebilmektedir. Buna göre, en fazla müşteri sayısı 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 nasıl olduğu ve gösterimden çözüm okunması aşaması Şekil 2'de detaylı bir şekilde gösterilmektedir.

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.

Şekil 2: Parçacık Pozisyon Gösterimi ve Çözüm Belirlenmesi



3.3. 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: 259).

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. Çalışmamızın bu aşamasında, literatürde çok kullanılan ve başarılı sonuçlar elde edilen yerel arama metotlarından DARP için PSO algoritması tarafından bulunan uygun çözümlerin iyileştirilmesi için kullandıklarımızdan detaylı bir şekilde bahsedilecektir.

2-Opt Yerel Arama Metodu (Croes, 1958)*:* 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.

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.

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.

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.

4. UYGULAMA

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. 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 v.d., 2005), Genetik algoritma (Hanshar ve Ombuki-Berman, 2007) ve PSO algoritması ile çözüm önerisi getiren (Khouadjia v.d., 2012) çalışmalardaki çözümlere de yer verilerek sonuçlar karşılaştırmalı olarak raporlanmıştır.

4.1. Parametre Seçimi

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 v.d. (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 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. PSO algoritmasına ait seçilen parametreler Tablo 1 ile verilmektedir.

Tablo 1: PSO Algoritması Parametreleri

Parametre Türü	**Değer**
Popülasyon Büyüklüğü	10
İterasyon Sayısı	1000
Her Bir Problem İçin Tekrar Sayısı	10
Atalet Ağırlığı (w)	1
Kişisel En İyi Pozisyonun Ağırlığı (c1)	1
Yerel En İyi Pozisyonun Ağırlığı (c2)	1

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. Statik ARP için Christofides ve Beasley (1984) (7 farklı problem), Fisher (1995) (1 problem) ve Taillard (1993) (12 farklı problem) tarafından oluşturulan test problemleri, Kilby v.d. (1998) tarafından dinamik hale getirilmiştir. Problemler literatüre kazandıran kişinin isminin baş harfi ve içerdiği müşteri sayısı ile birlikte isimlendirilmişlerdir.

4.2. Test Sonuçları

Test problemlerinin her biri 10 'ar kez çözülerek her bir çözüm için sonuçlar kaydedilmiştir. Algoritma, i7 cpu ve 8 gb ram özelliklerindeki bir bilgisayarda çalıştırılmış ve karşılaştırmalı sonuçlar Tablo 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.

20 adet farklı probleme ait amaç fonksiyonu değerlerinde karşılaştırılan dört algoritma için, en iyi değerin bulunduğu hücre koyu renkle verilmiştir. Görülmektedir ki 8 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 2: Elde Edilen Sonuçların Diğer Çalışmalar ile Karşılaştırılması

Problem	PSO		DAPSO[1]		ACO[2]		GA[3]	
	En İyi	Ort.	En İyi	Ort.	En İyi	Ort.	En İyi	Ort.
c50	632,93	655,25	**575,89**	632,38	631,30	681,86	570,89	593,42
c75	988,36	1029,92	**970,45**	1031,76	1009,38	1042,39	981,57	1013,45
c100	975,62	1028,57	988,27	1051,50	973,26	1066,16	**961,10**	987,59
c100b	969,90	1018,11	924,32	964,47	944,23	1023,6	**881,92**	900,94
c120	**1267,26**	1348,49	1276,88	1457,22	1416,45	1525,15	1303,59	1390,58
c150	**1227,21**	1363,56	1371,08	1470,95	1345,73	1455,50	1348,88	1386,93
c199	1781,30	1818,18	**1640,40**	1818,55	1771,04	1844,82	1654,51	1758,51
f71	**277,41**	301,26	279,52	312,35	311,18	348,69	301,79	309,94
tai75a	**1770,03**	1859,38	1816,07	1935,28	1843,08	1945,20	1782,91	1856,66
tai75b	**1396,39**	1435,85	1447,39	1484,73	1535,43	1704,06	1464,56	1527,77
tai75c	1455,77	1522,10	1481,35	1664,40	1574,98	1653,58	**1440,54**	1501,91
tai75d	1518,58	1590,72	1414,28	1493,47	1472,35	1529,00	**1399,83**	1422,27
tai100a	2357,66	2471,72	2249,84	2370,58	2375,92	2428,38	**2232,71**	2295,61
tai100b	2286,00	2374,24	2238,42	2385,54	2283,97	2347,90	**2147,70**	2215,39

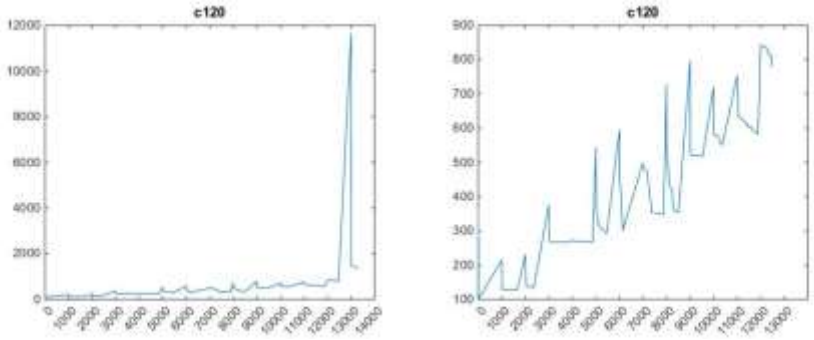
Problem	PSO		DAPSO[1]		ACO[2]		GA[3]	
	En İyi	Ort.	En İyi	Ort.	En İyi	Ort.	En İyi	Ort.
tai100c	**1495,20**	1588,89	1532,56	1627,32	1562,30	1655,91	1541,28	1622,66
tai100d	**1722,27**	1879,96	1955,06	2123,90	2008,13	2060,72	1834,60	1912,43
tai150a	3571,34	3746,67	3400,33	3612,79	3644,78	3840,18	**3328,85**	3501,83
tai150b	**2893,76**	3200,97	3013,99	3232,11	3166,88	3327,47	2933,40	3115,39
tai150c	2772,36	2928,11	2714,34	2875,93	2811,48	3016,14	**2612,68**	2743,55
tai150d	3110,11	3258,46	3025,43	3347,60	3058,87	3203,75	**2950,61**	3045,16

1: Dynamic Adopted PSO (Khouadjia v.d., 2012)
2: Ant Colony Optimization (Montemanni v.d., 2005)
3: Genetic Algorithm (Hanshar ve Ombuki-Berman M., 2007)

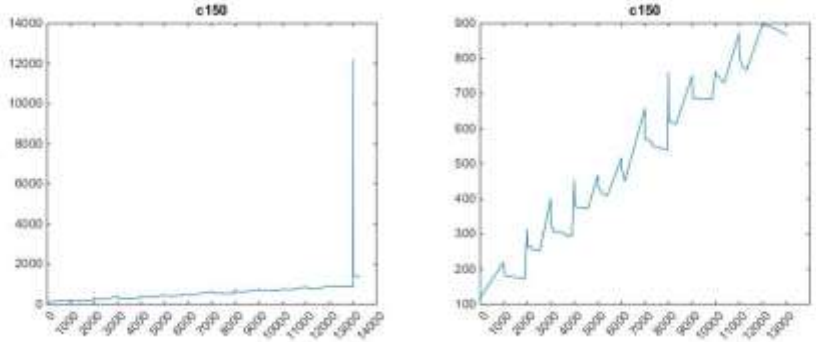
Algoritmanın performansını ölçmek için, her bir zaman diliminde ele almış olduğu problem için iterasyonlar boyunca en iyiye nasıl yakınsadığının görselleştirilmesi önemlidir.

Şekil *3* ve Şekil 4 ile verilen iki örnek problem için en iyi değere yakınsama grafikleri incelenebilir. Bu grafiklerde 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 3: c120 Problemi için Algoritmanın En İyi Değere Yakınsaması



Şekil 4 c150 Problemi için Algoritmanın En İyi 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.

4.3 Önerilen Algoritmanın 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. 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.

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.

Toplamda 10 adet alt rotanın belirlenmiş olduğu mevcut durumda toplam maliyet 1109,84 km olarak hesaplanmıştır.

PSO algoritması sonucunda elde edilen çözüm ile 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.

5. SONUÇ VE ÖNERİLER

Bu çalışmada ulaştırma sektöründe çok önemli bir yer tutan Dinamik Araç Rotalama Problemleri ele alınmıştır. DARP için meta sezgisel algoritmalar içerisinde NP-Zor sınıfına ait optimizasyon problemlerine başarılı bir şekilde uygulanan olan Parçacık Sürü Optimizasyon algoritması önerilmiştir.

Önerilen çözüm algoritması müşteri sayısı 50 ile 150 arasında değişen 20 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.

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.

Çalışmamız 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.

6. TEŞEKKÜR

Bu çalışma İstanbul Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi Tarafından Tez Projesi Kapsamında Desteklenmiştir. Proje numarası: 47943

7. KAYNAKÇA

1. BIANCHI, L. (2000). Notes On Dynamic Vehicle Routing-The State of The Art.
2. CHRISTOFIDES, N. ve BEASLEY, J. E. (1984). "The Period Routing Problem", Networks, 14(2): 237-256.
3. CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. ve STEIN, C. (2009). Introduction to Algorithms (Cilt 3). Cambridge: MIT Yayım.
4. CROES, G. A. (1958). "A Method For Solving Traveling-Salesman Problems", Operations Research, 6(6): 791-812.
5. DANTZIG, G. B. ve RAMSER, J. H. (1959). "The Truck Dispatching Problem", Management Science, 6(1): 80-91.
6. FISHER, M. (1995). "Vehicle Routing", (İçinde) Handbooks In Operations Research and Management, 8: 1-33, Elsevier.
7. HANSHAR, F. T. ve OMBUKI-BERMAN, M., B. (2007). "Dynamic Vehicle Routing Using Genetic Algorithms", Applied Intelligence, 27(1): 89-99.
8. KENNEDY, J. ve EBERHART, R. (1995). "Particle Swarm Optimization", Proceeding of the IEEE international conference on neural networks, 1942-1948.
9. KHOUADJIA, M. R., SARASOLA, B., ALBA, E., JOURDAN, L. ve TALBI, E. G. (2012). "A Comparative Study Between Dynamic Adapted PSO and VNS for the Vehicle Routing Problem with Dynamic Requests", Applied Soft Computing, 12(4): 1426-1439.
10. KILBY, P., PROSSER, P. ve SHAW, P. (1998). "Dynamic VRPs: A Study of Scenarios", University of Strathclyde Technical Report.
11. LARSEN, A. (2000). The Dynamic Vehicle Routing Problem. Doktora Tezi, Institute of Mathematical Modelling, Technical University of Denmark.
12. LUND, K., MADSEN, O. B. ve RYGAARD., J. M. (1996). Vehicle routing with varying degree of dynamism. The Department Informatics of Mathematical Modelling. Technical University of Denmark.
13. MIRHASSANI, S. A. ve ABOLGHASEM, N. (2011). "A Particle Swarm Optimization Algorithm for Open Vehicle Routing Problem", Expert Systems with Application, 38(9): 11547-11551.
14. MONTEMANNI, R., GAMBERDELLA, L. M., RIZZOLI, A. E. ve DONATI, A. V. (2005). "Ant Colony System for a Dynamic Vehicle

Routing Problem", Journal of Combinatorial Optimization, 10(4): 327-343.

15. PILLAC, V., GENDREAU, M., GUÉRET, C., ve MEDAGLIA, A. L. (2013). "A Review of Dynamic Vehicle Routing Problems", European Journal of Operational Research, 225(1): 1-11.
16. PSARAFTIS, H. N. (1988). "Dynamic vehicle routing problems", (İçinde) Vehicle Routing: Methods and Studies: 223-248, North-Holland, Amsterdam.
17. PSARAFTIS, H. N. (1995). "Dynamic Vehicle Routing: Status and Prospects", Annals of Operations Research, 61(1): 143-164.
18. SHI, Y. ve EBERHART, R. C. (1999). "Empirical Study of Particle Swarm Optimization", Proceedings of the IEEE Congress on Evolutionary Computation: 1945-1950.
19. TAILLARD, É. (1993). "Parallel Iterative Search Methods for Vehicle Routing Problems", Networks, 23(8): 661-673.
20. TOTH, P. ve VIGO, D. (2001). "The Vehicle Routing Problem", Society for Industrial and Applied Mathematics.
21. WANG, W., WU, B., ZHAO, Y. ve FENG, D. (2006). "Particle Swarm Optimization for Open Vehicle Routing Problem", (İçinde) Computational Intelligence: 999-1007, Springer.
22. WOODRUFF, D. L. (1998). Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search, Springer.
23. YU, X., GEN, M. (2010). Introduction to Evolutionary Algorithms. Springer Science - Business Media.