AÇIK ARAÇ ROTALAMA PROBLEMİ İÇİN METASEZGİSEL ALGORİTMA TASARIMI ve UYGULAMASI

Erdener ÖZÇETİN

Doktora Tezi

Eskişehir, 2019

AÇIK ARAÇ ROTALAMA PROBLEMİ İÇİN METASEZGİSEL ALGORİTMA TASARIMI ve UYGULAMASI

ERDENER ÖZÇETİN

DOKTORA TEZİ
Endüstri Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Gürkan ÖZTÜRK

Eskişehir
Eskişehir Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Mayıs 2019

Bu tez çalışması Eskişehir Teknik Üniversitesi *BAP Komisyonu tarafından kabul edilen 1601F004 no.lu proje kapsamında desteklenmiştir*

JÜRİ VE ENSTİTÜ ONAYI

Erdener Özçetin'in "Açık Araç Rotalama Problemi için Metasezgisel Algoritma Tasarımı ve Uygulaması" başlıklı tezi 03/05/2019 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca, **Endüstri Mühendisliği** Anabilim dalında Doktora tezi olarak kabul edilmiştir.

Jüri Üyeleri	**Unvanı Adı Soyadı**	**İmza**
Üye (Tez Danışmanı)	: Doç. Dr. Gürkan ÖZTÜRK	.....
Üye	: Prof. Dr. Bülent ÇATAY	.....
Üye	: Doç. Dr. Cihan KALELİ	.....
Üye	: Prof. Dr. Aydın SİPAHİOĞLU	.....
Üye	: Prof. Dr. Onur KAYA	.....

.....

Lisansüstü Eğitim Enstitüsü Müdürü

ÖZET

AÇIK ARAÇ ROTALAMA PROBLEMİ İÇİN METASEZGİSEL ALGORİTMA TASARIMI ve UYGULAMASI

Erdener ÖZÇETİN

Endüstri Mühendisliği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Mayıs 2019

Danışman: Doç. Dr. Gürkan ÖZTÜRK

Bu çalışmada, firmaların daha çok üçüncü parti lojistik hizmetleri kullanması ile birlikte, uygulamada yaygın şekilde ortaya çıkan Açık Araç Rotalama Problemi ele alınmıştır. Bu problem, yüksek boyutlu ve karmaşık eniyileme problemleri sınıfında yer almaktadır. Öncelikle, heterojen araç filosunun bulunduğu bir gerçek hayat problemi için model tabanında matematiksel modeller ile popülasyon temelli bir metasezgisel algoritmanın yer aldığı bir karar destek sistemi geliştirilmiştir. Daha sonra, özellikle büyük boyutlu problemleri etkin şekilde çözmek üzere üç evreli bir Değişken Komşuluk Arama Algoritması önerilmiştir. Önerilen bu yöntemde dört adet rota içi ve dört adet rotalar arası olmak üzere sekiz farklı komşuluk ve sarsma stratejisi kullanılmıştır. Yöntemin performansı literatür test problemleri üzerinde test edilmiş ve başarısı karşılaştırmalı olarak raporlanmıştır. Ayrıca, önerilen değişken komşuluk arama algoritmasının eş zamanlı hesaplamaya uygun olan kısımlarının, grafik işlem birimleri üzerinde paralelleştirilmesi için farklı stratejiler tasarlanmış ve uygulanmıştır. Bu stratejilerin uygulanması ile elde edilen paralel yöntem, seri versiyonuna göre anlamlı şekilde hızlandırılmıştır.

Anahtar Kelimeler: Açık araç rotalama problemi (AARP), karar destek sistemleri, değişken komşuluk arama, grafik işlem birimleri, CUDA.

ABSTRACT

METAHEURISTIC ALGORITHM DESIGN AND APPLICATION FOR OPEN VEHICLE ROUTING PROBLEM

Erdener ÖZÇETİN
Industrial Engineering Program
Eskişehir Technical University, Graduate School, May 2019

Supervisor: Assoc. Prof. Dr. Gürkan ÖZTÜRK

In this study, the Open Vehicle Routing Problem, which is widely employed by the companies in practice with the use of third party logistics services, is discussed. This problem is in the class of high-dimensional and complex optimization problems. First of all, a decision support system based on a mathematical models and a population based meta-heuristic algorithm was developed for a real-life problem with a heterogeneous vehicle fleet. Then, a three-phase Variable Neighborhood Search Algorithm was proposed in order to solve large-scale problems efficiently. In this proposed method, eight different neighborhoods and shaking strategies was used. The performance of the method was tested on literature test problems and its success was reported comparatively. In addition, different strategies were designed and implemented for the parallelization of parts of the proposed variable neighborhood search algorithm that are suitable for simultaneous calculation on the graphics processing units. A significant speed up was observed in the parallel method obtained by the implementation of these strategies compared to the serial version.

Anahtar Kelimeler: Open vehicle routing algorithm (OVRP), decision support systems, variable neigborhood search, graphics processing units, CUDA.

TEŞEKKÜR

Öncelikle tanıştığımdan beri birçok alanda gelişmemde pay sahibi olan, tez çalışmam süresince hiçbir yardımını esirgemeyen ve her karamsarlığa düştüğümde beni güçlendiren danışman hocam Doç. Dr. Gürkan Öztürk'e sonsuz teşekkürlerimi sunarım. Ayrıca tez çalışmam süresince desteklerini esirgemeyen Prof. Dr. Bülent Çatay'a teşekkürü bir borç bilirim. Bununla birlikte çalışmalarım süresince desteklerini hissettiğim arkadaşlarım Cenk İçöz, Emre Çimen ve Gökmen Zararsız'a teşekkür ederim.

Son olarak bugünlere gelmemde önemli katkıları bulunan aileme ve tüm özverili yaklaşımından dolayı eşim Serap'a sonsuz teşekkür ederim.

Erdener ÖZÇETİN

Mayıs 2019

03/05/2019

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Erdener ÖZÇETİN

İÇİNDEKİLER

Sayfa

JÜRİ VE ENSTİTÜ ONAYI ii

ÖZET iii

ABSTRACT iv

TEŞEKKÜR v

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ vi

İÇİNDEKİLER vii

ŞEKİLLER DİZİNİ xi

TABLOLAR DİZİNİ xiii

1. GİRİŞ 1

2. METASEZGİSEL ALGORİTMALAR ve ARAÇ ROTALAMA PROBLEMLERİ 5

2.1. Metasezgisel Algoritmalar 5

2.1.1. Çözümlerin gösterimi 6

2.1.2. Amaç fonksiyonu 6

2.1.3. Kısıtların ele alınması 7

2.1.4. Parametre ayarları 7

2.1.5. *S*-metasezgiseller 7

2.1.6. *P*-metasezgiseller 8

2.1.7. Metasezgisel algoritmaların melezleştirilmesi ve melez genetik algoritma 8

2.1.8. Değişken komşuluk arama 9

2.2. Araç Rotalama Problemleri 10

2.2.1. Son zamanlarda ön plana çıkan ARP alanları ve çözüm yaklaşımlarındaki yenilikler 11

2.2.2. Açık araç rotalama problemi 14

2.2.3. AARP için çözüm yaklaşımları 14

2.2.4. Sabit heterojen filolu açık araç rotalama problemi 18

3. GERÇEK HAYAT PROBLEMİ ve GELİŞTİRİLEN KARAR DESTEK SİSTEMİ 20
3.1. Problemin Özellikleri 20
3.1.1. Uzaklık bilgilerinin elde edilmesi 21
3.1.2. Firma verilerine ön işlem uygulanması 22
3.1.3. Arayüzlerin geliştirilmesi 24
3.1.4. Önerilen melez genetik algoritma 25
3.2. Test Problemlerinin Üretilmesi ve Melez Genetik Algoritmada Uygulanması 29
3.2.1. Test problemlerin üretilmesi 29
3.2.2. Test problemlerinden elde edilen sonuçlar 30
4. AARP İÇİN ÜÇ AŞAMALI DEĞİŞKEN KOMŞULUK ARAMA ALGORİTMASI 32
4.1. Başlangıç Çözümlerin Oluşturulması 32
4.2. Rota İçi Arama için Kullanılan Komşuluklar 33
4.2.1. 2-opt komşuluğu 33
4.2.2. Or-Opt komşuluğu 34
4.2.3. 3-opt komşuluğu 35
4.2.4. Dinamik arama komşuluğu 36
4.3. Rotalar Arası Arama için Kullanılan Komşuluklar 37
4.3.1. *Bire-bir* karşılıklı değişim arama operatörü 37
4.3.2. Yeniden konumlandırma (Relocate) arama operatörü 37
4.3.3. Çoklu-karşılıklı değişim (Cross-Exchange) komşuluğu 38
4.3.4. *2opt** komşuluğu 39
4.4. Sarsma Mekanizmaları 40
4.4.1. Rota içi k-değişim komşuluğu 40
4.4.2. Rota içi çift-köprü (double-bridge) komşuluğu 41
4.4.3. Rotalar arası k-değişim komşuluğu 41
4.4.4. Çıkar-ekle stratejileri 41
4.5. Çatı Algoritma 43

4.5.1. Rota içi arama DKA 45
4.5.2. Rotalar arası arama DKA 46
4.5.3. Ana sarsma mekanizması 46
4.6. Parametrelerin Belirlenmesi 48
4.7. Detaylı Operatör İncelemeleri 51
4.7.1. Başlangıç çözümlerin etkinliğinin araştırılması 51
4.7.2. Dinamik arama yönteminin etkinliğinin araştırılması 52
4.7.3. Çoklu-karşılıklı değişimde sınırlama getirilmesinin araştırılması 52
4.7.4. Ana sarsma mekanizmasının çözüm üzerinde etkisinin incelenmesi 53
4.7.5.Komşulukların ilk iyileştirme yerine en iyi iyileştirme olması durumunda etkinliği 53
4.8. Test Problemlerinden Elde Edilen Sonuçlar 55
4.9. Önerilen Algoritmanın Gerçek Hayat Problemine Uygulanması 57
5. ÖNERİLEN ALGORİTMANIN GRAFİK İŞLEM BİRİMLERİ ÜZERİNDE PARALELLEŞTİRİLMESİ 60
5.1. Metasezgisel Algoritmalar ve GİB 60
5.2. GİB Mimarisi 61
5.3. Paralel Metasezgisel Algoritmalar İle İlgili Literatür Çalışmaları 65
5.4. AARP'nin Fazla Sayıda Çözümü İçin Amaç Fonksiyonun Eş Zamanlı Hesaplanması 66
5.5. DKA'nın GİB'de Ele Alınması 69
5.5.1. Problemin GİB'de gösterimi 69
5.5.2. Önerilen yöntemde kullanılan rota içi arama komşuluk yapılarının gib'de paralelleştirilmesi 71
5.5.3. Önerilen yöntemde kullanılan rotalar arası arama komşuluk yapılarının GİB'de paralelleştirilmesi 72
5.5.4. Algoritmanın GİB üzerinde akışı 73
5.6. Paralel DKA'dan Elde Edilen Sonuçlar 74

6. SONUÇ ve ÖNERİLER 76
KAYNAKÇA 78
ÖZGEÇMİŞ

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1 Değişken komşuluk arama algoritmasının sözde kodları 10
Şekil 3.1 Karar destek sisteminde yer alan butonlar 25
Şekil 3.2 Karar destek sisteminin veri giriş ekranı 25
Şekil 3.3 İlişki matrisi örneği 26
Şekil 3.4 Yakınlık matrisi örneği 27
Şekil 3.5 Çaprazlama için bölgelere ayırma 28
Şekil 3.6 Pozisyona dayalı çaprazlama örneği 28
Şekil 4.1 Başlangıç çözüm oluşturma algoritması 33
Şekil 4.2 2-opt komşuluğu örneği ve Python sözde kodu 34
Şekil 4.3 Or-opt komşuluğu örneği ve Python sözde kodu 35
Şekil 4.4 3-opt değişim seçenekleri 36
Şekil 4.5 Dinamik arama komşuluğu örneği ve Python sözde kodu 36
Şekil 4.6 Bire-bir karşılıklı değişim komşuluğu örneği ve Python sözde kodu 37
Şekil 4.7 Yeniden konumlandırma komşuluğu örneği ve Python sözde kodu 38
Şekil 4.8 Çoklu-karşılıklı değişim komşuluğu örneği ve sözde kodu 39
Şekil 4.9 2opt* değişim komşuluğu örneği ve sözde kodu 40
Şekil 4.10 Rota içi k-değişim komşuluğu düz yönlü örneği 41
Şekil 4.11 Rota içi k-değişim komşuluğu ters yönlü örneği 41
Şekil 4.12 Çatı algoritma sözde kodu 44
Şekil 4.13 Rota içi arama sözde kodları 45
Şekil 4.14 Rotalar arası arama sözde kodları 46
Şekil 4.15 Rota içi arama sözde kodları 47
Şekil 4.16 O3 için başlangıç çözüm karşılaştırması 51
Şekil 4.17 O3 için Dinamik arama incelemesi 52
Şekil 5.1 Akış işlemcisi ve Cuda çekirdeği 64
Şekil 5.2 Özçetin'in çalışmasında elde edilen sonuçlar 65
Şekil 5.3 GİB'ye vektör olarak aktarılan uzaklık matrisi 67
Şekil 5.4 GİB'ye aktarılan toplu çözüm örneği 67
Şekil 5.5 GİB'ye aktarılan toplu rota başlangıç noktaları örneği 67
Şekil 5.6 GİB'de amaç fonksiyonunun eş zamanı hesaplanması 68

Şekil 5.7 GİB için yeni çözüm gösterim şekli 70
Şekil 5.8 İndeks belirleme örneği 73
Şekil 5.9 GİB üzerinde paralel DKA akış işlemleri 73

TABLOLAR DİZİNİ

Sayfa

Tablo 2.1 GSP arama uzayının büyüklüğü 5
Tablo 2.2 Çözüm gösterim şekilleri 6
Tablo 2.3 Literatür özet tablosu-1 15
Tablo 2.4 Literatür özet tablosu-2 16
Tablo 2.5 Literatür özet tablosu-3 17
Tablo 3.1 Kullanılan araç türleri ve kapasiteleri 21
Tablo 3.2 Örnek problem verileri 23
Tablo 3.3 Üretilen test problemlerinin özellikleri 29
Tablo 4.1 Literatür test problemleri 48
Tablo 4.2 C1 için en iyinin çözüme eklenmesi parametresine ait testler 48
Tablo 4.3 C2 için en iyinin çözüme eklenmesi parametresine ait testler 49
Tablo 4.4 C3 için en iyinin çözüme eklenmesi parametresine ait testler 49
Tablo 4.5 C1 için b parametresine ait testler 49
Tablo 4.6 C2 için b parametresine ait testler 49
Tablo 4.7 C3 için b parametresine ait testler 50
Tablo 4.8 C2 için aramalardaki en fazla iyileşmeme durumuna ait testler 50
Tablo 4.9 Parametreler 50
Tablo 4.10 O3 için en fazla çoklu-karşılıklı değişim karşılaştırması 52
Tablo 4.11 C1, C2 ve C3 için ana sarsma mekanizmasının etkisi 53
Tablo 4.12 Aç gözlü değişim ile en iyi değişim karşılaştırması 54
Tablo 4.13 Tüm komşuluklar ayrı ayrı algoritmadan ayrıldığında elde edilen sonuçlar 54
Tablo 4.14 Literatür test problemleri için DKA ile elde edilen sonuçlar-1 55
Tablo 4.15 Literatür test problemleri için DKA ile elde edilen sonuçlar-2 55
Tablo 4.16 Literatür çalışmaları ile elde edilen sonuçların karşılaştırılması-1 56
Tablo 4.17 Literatür çalışmaları ile elde edilen sonuçların karşılaştırılması-2 56
Tablo 4.18 Literatür çalışmaları ile elde edilen sonuçların karşılaştırılması-3 57
Tablo 4.19 Üretilen test problemleri için DKA'dan elde edilen sonuçlar 58
Tablo 4.20 Üretilen test problemleri için melez GA ile DKA'dan elde edilen sonuçların karşılaştırılması-1 58

Tablo 4.21 Üretilen test problemleri için melez GA ile DKA'dan elde edilen sonuçların karşılaştırılması-2 59
Tablo 5.1 GİB Mimarileri 63
Tablo 5.2 AARP için eş zamanlı amaç fonksiyonu değerlendirme 68
Tablo 5.3 Literatür test problemleri için GİB üzerindeki DKA ile elde edilen sonuçlar 74
Tablo 5.4 Seri DKA ile paralel DKA karşılaştırılması 75

1. GİRİŞ

Eniyilemede, verilen bir problem için birçok alternatif arasından en iyisini ya da yeteri kadar iyi olanını bulmak için çaba gösterilir. Eniyileme problemleri, günlük yaşamın her alanında sürekli karşımıza çıkmaktadır. Her birimiz düzenli olarak eniyileme problemlerini çözmek için çaba harcarız. Örneğin; işten eve giderken uğramamız gereken yerleri ve trafiği göz önünde bulundurarak en iyi rotayı izleriz ya da bir kutuya eşyaları yerleştirirken kutunun hacmini, en fazla kullanacak şekilde eşyaların yön ve pozisyonlarını ayarlarız. Günlük problemlerin çoğunun boyutunun, küçük olmasının yanı sıra bu problemleri optimal olarak çözememenin bizler için önemli bir maliyeti bulunmamaktadır. Örneğin, sadece bir sefer doldurmamız gereken bir kutuyu yerleştirirken optimal bir yerleşim olması için çok çaba harcamayız. Diğer taraftan ürettiği ürünleri her gün filosundaki araçlara yükleyen bir firma için bu doldurma işleminin optimal olması büyük önem arz etmektedir. Firmalar, günlük yaptığı planları optimal ya da optimale yakın şekilde gerçekleştirdiklerinde lojistik maliyetlerinden çok büyük kazançlar sağlayabilmektedirler. Bunun yanı sıra dağıtım esnasında doğaya verilen zararlarda da önemli bir iyileşme sağlanabilmektedir.

Problemlerin boyutu büyüdüğünde matematiksel ve/veya sezgisel yaklaşımları bilgisayar desteğiyle tercih etmek, problemin çözümü için kaçınılmaz hale gelmektedir. Eniyileme alanından gezgin satıcı problemi, karesel atama problemi ve araç rotalama problemi gibi kombinatorik eniyileme problemleri çoğu zaman, kesin yöntemlerle polinom zaman içerisinde bütünsel en iyi şekilde çözebilmek mümkün değildir.

Metasezgisel algoritmalar, genel anlamda kombinatorik eniyileme problemleri gibi belirli tiplerdeki eniyileme problemlerin çözümü için kullanılan araçlar olarak gösterilirler. Bu algoritmalar, eniyileme problemleri için geliştirilmiş kesin çözümü garanti etmeyen; fakat kısa sürede en iyi çözüme yakınsayan tekniklerdir. Kesin çözüm veren algoritmaların makul bir sürede çözüm vermediği durumlarda sıklıkla kullanılırlar. Metasezgiseller, farklı alanlarda birçok eniyileme problemine uygulanabilmektedir:

• Mühendislik tasarımları, elektronikte topolojik ve yapısal eniyileme, telekomünikasyon, aerodinamik, akışkanlar dinamiği,

• Biyoenformatikte yapay zeka ve veri madenciliği uygulamaları, hesaplamalı biyoloji ve finans,

• Sistem modelleme, fizik, kimya ve biyolojide benzetim, sinyal ve resim işleme,

• Rotalama problemleri, çizelgeleme ve üretim problemleri, lojistik ve taşıma problemleri, tedarik zinciri yönetimi vb. [1].

Metasezgisel algoritmalar oluşturulurken ele alınan probleme makul zamanda, güvenilir ve optimal ya da optimale yakın sonuçlar vermesi istenmektedir. Algoritmalar yüksek performans gösterebilmesi için parametre ayarlarına ihtiyaç duymaktadır. Bu algoritmalar her ne kadar bütünsel en iyiyi bulmayı garanti etmese de birçok alanda büyük boyutlu ve karmaşık problemlere yapılan uygulamalarda, pratik ve hızlı bir yaklaşım sundukları için tercih edilmektedir.

Araç rotalama problemleri kombinatorik eniyileme problemleri arasında yer almaktadır. Problemde temel olarak verilen kısıtlar altında bir araç filosunun müşterilere hizmet için en iyi rotalarının bulunması şeklinde tariflenebilmektedir. Problemin kapasite kısıtlı, zaman pencereli, açık ve önce dağıt sonra topla gibi çeşitli türleri bulunmaktadır. Problemin fazla sayıda alt türünün bulunmasının sebebi gerçek hayatta fazlasıyla karşılaşılır olmasından kaynaklanmaktadır. Bu durum problemin ekonomik boyutunun önemini ortaya koymaktadır. Gerçek hayatta etkin gerçekleştirilmeyen dağıtım planları firmalara fazladan katlanılmasını gerektiren büyük maliyetler ortaya çıkarmaktadır. Bununla birlikte etkin olmayan dağıtım planları araçlardan fazladan salınan gazlar sebebiyle doğaya zarar vermektedir.

Açık araç rotalama probleminde, araçlar bir depodan başlayarak müşterilere hizmet sunmakta ve rotalarında bulunan son müşteriye de hizmet sunduktan sonra depoya dönmemektedir. Firmaların ürünlerinin dağıtımı için kendi araç filolarının olmadığı durumlarda açık araç rotalama problemiyle sıklıkla karşılaşılmaktadır. Firma ürünlerinin dağıtımını gerçekleştirmek üzere üçüncü parti bir lojistik şirketiyle anlaşmakta ve yapılan anlaşmaya göre ürünler üçüncü parti şirketin araçlarıyla gerçekleştirilmektedir. Firmalar ve üçüncü parti lojistik şirketleriyle sıklıkla yapılan anlaşmalara göre ürünlerin dağıtımı tamamlandıktan sonra firma araçlara müdahale edememektedir. Dağıtım işlemini tamamlayan araçların nasıl değerlendirileceği kiralama hizmetini sağlayan şirketin takdirine bırakılmaktadır.

Tez çalışmasında, açık araç rotalama probleminin bir gerçek hayat uygulaması ele alınmıştır. Ele alınan gerçek hayat probleminde, firmanın dağıtımlarında kullanılacak araçları kiralamak üzere üçüncü parti bir lojistik şirketiyle anlaşması bulunmaktadır. Firmanın anlaşma gerçekleştirdiği araç türleri farklı kapasitelerde olan kamyon ve tırdır. Diğer bir deyişle firmanın kullandığı araç filosu homojen değildir. Problemin bir başka

yönü, firmanın iki farklı fabrikasında üretilen ürünlerin birlikte dağıtılmasıdır. Var olan durumda firma iki fabrikanın ürünlerinin dağıtım planlarını ayrı ayrı gerçekleştirmektedir. Bu durum firma adına fazladan maliyet ortaya çıkarmaktadır. Çalışmada iki farklı fabrikada üretilen ürünlerin birlikte dağıtılması ele alınmaktadır. Bununla beraber firma ve üçüncü parti lojistik şirketi arasındaki anlaşmada rotalardaki son noktalara göre bir ana maliyetin yanı sıra, rotadan sapmalardan doğan bir ek kilometre maliyet, boşaltma sayısından doğan bir maliyet ve belirli sayıda boşaltmadan sonra ortaya çıkan sürücü konaklama maliyeti bulunmaktadır. Bütün bu açılar değerlendirildiğinde ele alınan gerçek hayat probleminin sıradan problemlerden ayrışmakta ve problemin araştırılma ihtiyacı doğmaktadır.

Çalışmada gerçek hayat probleminin çözümü için model tabanında matematiksel modeller ile popülasyon temelli bir metasezgisel algoritmanın yer aldığı bir karar destek sistemi geliştirilmiştir. Bu aşamadan sonra elde edilen çözümlerin kalitesinin karşılaştırılabilmesi için bir değişken komşuluk arama algoritması önerilmiştir. Önerilen algoritmanın literatür test problemleri üzerindeki başarısı raporlanmıştır. Bununla birlikte, melez genetik algoritmayla elde edilen sonuçlar ile karşılaştırmalar gerçekleştirilmiştir. Daha sonra zaman alıcı bazı işlemlerin grafik işlem birimleri üzerinde eş zamanlı hesaplamalarla kısa sürelerde ele alınabilmesinin üzerinde durulmuştur. Çalışmanın temel motivasyonları arasında şunlar bulunmaktadır:

- Gerçek hayat probleminin çözümü için bir karar destek sisteminin geliştirilmesi,
- Literatür test problemlerinde algoritmanın geçerliliğinin gösterilmesi,
- Geliştirilen algoritmanın elverişli kısımlarının grafik işlem birimleri üzerinde paralel çalışmaya uygun şekilde tasarlanarak zamandan tasarruf elde edilmesidir.

Çalışmanın ikinci bölümünde, metasezgisel algoritmalar ve araç rotalama problemleriyle ilgili genel bilgilerin yanı sıra son zamanlarda araç rotalama problemlerindeki literatür çalışmalarına yönelik derinlemesine incelemeler bulunmaktadır. Üçüncü bölümde ise, çalışmada incelenen gerçek hayat probleminin yapısı ve bu problemin çözümüne yönelik geliştirilen karar destek sistemi tüm detaylarıyla açıklanmaktadır. Çalışmanın dördüncü bölümünde, karar destek sisteminin model tabanında karşılaştırılmaların yapılabilmesi amacıyla geliştirilen değişken komşuluk arama algoritması açıklanmakta ve algoritmanın literatür problemleri karşısında başarısı tartışılmaktadır. Yine aynı bölümde, geliştirilen değişken komşuluk arama algoritmasının tez çalışmasında ele alınan gerçek hayat problemi temel alınarak

üretilen test problemleri üzerindeki başarısı incelenmektedir. Beşinci bölümde, önerilen değişken komşuluk arama algoritmasının eş zamanlı hesaplamaya uygun olan bölümlerini, grafik işlem birimleri üzerinde ele almak için izlenilen stratejiler detaylarıyla açıklanmakta ve sonuçlara yer verilmektedir. Bununla birlikte önerilen değişken komşuluk arama algoritmasının, paralel versiyonun bire bir karşılığı olan seri versiyonu ile adil karşılaştırmalar yapılmış ve hızlanma faktörleri sunulmuştur. Son bölümde sonuç ve önerilerinin yanı sıra gelecek çalışmalar için öngörüler bulunmaktadır.

2. METASEZGİSEL ALGORİTMALAR ve ARAÇ ROTALAMA PROBLEMLERİ

Tez çalışmasının bu bölümünde, metasezgisel algoritmalar ve araç rotalama problemleri hakkında bazı bilgiler ve literatür çalışmaları yer almaktadır.

2.1. Metasezgisel Algoritmalar

Bir eniyileme problemi (*OP*), bir ya da daha çok amaç fonksiyonunun enküçüklenmesi ya da enbüyüklenmesi olarak tarif edilebilir ve şu şekilde formüle edilir:

$$(OP) = \begin{cases} min\ F(X) = (f_1(x), f_2(x), \dots, f_n(x)) \\ s.t.\ \ x \in S \end{cases} \quad (2.1)$$

Eniyileme problemlerinin çözümü için problemin karmaşıklığına bağlı olarak kesin ya da sezgisel algoritmalar kullanılabilmektedir. Dal sınır algoritması, kesme düzlemi yöntemi ve dinamik programlama gibi kesin yöntemler, eniyileme problemlerinin çözümünde en iyi çözümün bulunmasını garanti eden yöntemlerdir. Ancak problemlerin boyutunun büyümesi ile birlikte bu yöntemleri kullanmak mümkünlüğünü yitirmeye başlamaktadır. Tablo 2.1'de gezgin satıcı problemi (GSP) için problem boyutu artırıldığında çözüm uzayındaki büyüme izlenmektedir. Çözüm uzayının büyümesi, en iyi çözümün bulunmasını zorlaştırmaktadır. Hatta bazı durumlarda şu anki yöntemler ve teknolojilerle imkansız hale gelmektedir. Bu sebepten dolayı, problemlerin çözümü için, probleme özgü sezgisel algoritmalar ya da probleme uyarlanabilen jenerik metasezgisel algoritmalar tercih edilmektedir. Metasezgisel algoritmalar, tek bir çözümün iteratif olarak iyileştirilmesi (yasaklı arama, tavlama benzetimi vb.) temelli olabildiği gibi popülasyon (genetik algoritma, karınca kolonileri eniyilemesi) temelli de olabilir. Bunlardan tek çözüm temelli olanlara *S*-metasezgiseller, popülasyon temelli olanlara *P*-metasezgiseller denmektedir. *P*-metasezgisellerin birden çok çözümle arama yapması sebebiyle keşfetme özellikleri fazladır. *S*-metasezgisellerin ise derinlemesine arama yetenekleri güçlüdür. Algoritmalara hem keşfetme hem de derinlemesine arama özelliği kazandırabilmek için *P* ve *S* metasezgiseller, bir arada melez bir şekilde sıklıkla kullanılmaktadır.

Tablo 2.1 *GSP arama uzayının büyüklüğü*

Problem boyutu	Arama uzayının büyüklüğü
5	120
10	3 628 800
100	9.33×10^{157}

İzleyen alt bölümlerde genel olarak metesezgisel algoritmaların genel yapıları açıklanmaktadır. Metasezgisellerin ortak özelllikleri olarak karşımıza çıkan; çözüm gösterim şekilleri, amaç fonksiyonlarının/kısıtların ele alınması ve parametrelerin etkileri ile birlikte *P* ve *S* metasezgisel algoritmalar ile ilgili bilgiler verilmektedir. Bu bilgilerin verilmesinin ardından, bu çalışmada kullanılan melez genetik algoritma ve değişken komşuluk arama algoritmalarına kısaca değinilmiştir.

2.1.1. Çözümlerin gösterimi

Metasezgisel algoritmaların tasarımında ilk adım, çözümün ne şekilde temsil edileceğinin belirlenmesidir. Çözümler için kullanılan gösterim şekilleri, algoritmanın etkinliğinde önemli bir rol oynamaktadır. Algoritmada yer alan bazı operatörler (çaprazlama, yerel arama vb.) çözümlerin gösteriminden etkilenebilmekte ve bu durum elde edilen çözümlerin kalitesine doğrudan yansıyabilmektedir.

Tablo 2.2 *Çözüm gösterim şekilleri*

Permütasyon	*4*	*1*	*2*	*3*
İkili	*0*	*1*	*1*	*0*
Tam sayılı vektör	*4*	*2*	*4*	*1*
Reel sayılı vektör	*0.52*	*3.44*	*0.77*	*2.1*
Rassal anahtarlı	*0.41*	*0.15*	*0.22*	*0.76*

Literatürde yaygın bir şekilde kullanılan gösterim türlerinden biri permütasyon tabanlı gösterimdir. Bunun dışında Tablo 2.2'de görüldüğü gibi ikili gösterim, tam sayılı vektör gösterimi, reel değerlerden oluşan vektör gösterimi ve rassal anahtarlı gösterim ön plana çıkmaktadır.

2.1.2. Amaç fonksiyonu

Bir metasezgisel algoritma tasarlanırken amaç fonksiyonu ele alınan her çözümle ilişkilendirilmelidir. Amaç fonksiyonun her bir değeri, arama uzayında bir gerçek değere karşılık gelmelidir. Amaç fonksiyonunun doğru tanımlanmaması, algoritmanın arama uzayında çalışan operatörlerinin kabul edilemeyecek çözümlere erişmesine sebep olabilmektedir.

Metasezgisel algoritmalarda kimi zaman gerçek problemi temsil eden amaç fonksiyonları da kullanılabilmektedir. Burada dikkat edilmesi gereken bazı hususlar bulunmaktadır. Buna göre temsili amaç fonksiyonu, gerçek amaç fonksiyonu değerine kolay dönüştürülebilir olmalıdır. Bununla birlikte temsili amaç fonksiyonu, problemin uzayında yer alan tüm çözümleri temsil edebilecek nitelikte olması gerekmektedir. Eğer

kullanılan amaç fonksiyonu çözüm uzayındaki tüm çözümleri temsil edebilir durumda değilse, kullanılan yöntemin gücü de anlamsız olmaktadır.

2.1.3. Kısıtların ele alınması

Eniyileme problemlerinde yer alan kısıtların, metasezgisel algoritma tasarımında ele alınması bir başka önemli konudur. Kısıtlar, eşitlik ya da eşitsizlik halinde ve doğrusal ya da doğrusal olmayan durumda bulunabilmektedir. Kısıtlar, algoritmaya aktarılırken hem çözümlerin gösterimi hem de amaç fonksiyonunun değerlendirme aşaması rol oynamaktadır. Kısıtlar ele alınırken çeşitli stratejiler kullanılmaktadır. Bu stratejilerden bazıları; ret stratejileri, ceza stratejileri ve onarma stratejileridir. Ret stratejilerinde, algoritma arama yaparken yalnızca uygun çözümlerle çalışılmakta ve kısıtları ihlal eden çözümler doğrudan elenmektedir. Ceza stratejilerinde uygun olmayan çözümler ele alınabilmekte; ancak bu çözümler ile karşılaşıldığında, amaç fonksiyonuna bir ceza maliyeti eklenmektedir. Onarma stratejilerinde ise kısıtları ihlal eden bir çözüm ile karşılaşıldığında çözümün uygun hale getirilmesi için çeşitli yollara başvurulmaktadır.

2.1.4. Parametre ayarları

Metasezgisel algoritmaların başarısının önemli bir bölümünü, parametrelerin isabetli ayarlanması oluşturmaktadır. Çoğunlukla parametrelere ait sabitlenmiş literatür değerleri bulunmamaktadır. Bazı öneriler mevcut olmakla beraber, parametreler bazen problemin türü bazen de ele alınan örneğin yapısına göre değişebilmektedir. Parametreler statik, dinamik ve adaptif olmak üzere üç farklı şekilde kullanılmaktadır. Statik parametreler, algoritmanın çalışması sürecinde sabit kalmaktadır. Dinamik parametreler, bir başlangıç değeri ile başlayıp algoritma sonlanıncaya kadar belirli çarpanlarla güncellenmektedir. Adaptif parametreler ise algoritma süresince çözümü iyileştiren ve iyileştirmeyen koşullara göre uyum sağlayan parametrelerdir.

2.1.5. *S*-metasezgiseller

S-metasezgiseller, tek bir çözüm üzerinde belirli stratejilere göre değişiklikler yaparak en iyi çözüme yakınsayan algoritmalardır. Ele alınan bir çözümden bir aday çözümler kümesi oluşturulmakta ve bu kümede ulaşılan en iyi çözüm, var olan çözümün yerine geçmektedir. Bu işlem bir durdurma noktasına ulaşıncaya dek devam etmektedir. *S*-metasezgisellerin hafızalı ya da hafızasız olma imkanları bulunmaktadır. Yasaklı arama, tavlama benzetimi ve yerel arama algoritmaları en sık kullanılan *S*-metasezgiseller arasında yer almaktadır. Bütün *S*-metasezgisellerin yaygın arama prosedürü, başlangıç çözümünün belirlenmesine ve komşuluk tanımının yapılmasına bağlı olmaktadır. *S*

türünde bir metasezgisel tasarlanırken komşuluk önemli bir kavramdır. Komşulukların yapısı algoritmanın başarısı açısından kritik önem taşımaktadır. Eğer tanımlanan komşuluk yapısı probleme göre uygun değilse, algoritma çalışırken problemin en iyi değerine yakınsamada güçlüklerle karşılaşabilmektedir.

2.1.6. *P*-metasezgiseller

Popülasyon temelli metasezgiseller bir dizi başlangıç çözümden oluşan popülasyonla aramaya başlamaktadır. Daha sonra tekrarlı olarak yeni nesil çözümler oluşturmaktadır. Yeni nesil çözüm popülasyonu oluşturulurken var olan çözüm popülasyondan bazı bilgiler çeşitli stratejilerle aktarılmaktadır. Bir durdurma koşulu sağlanana dek bu durum tekrar etmektedir. *P*-metasezgisellerin neredeyse tamamı doğadan esinlenilerek türetilmiştir. Bunlardan en yaygınları, evrimsel algoritmalardan olan genetik algoritma (GA), parçacık sürü eniyilemesi ve karınca kolonisi eniyilemesidir.

P-metasezgiselleri birbirlerinden ayıran özellik yeni nesil çözümlerin nasıl oluşturulacağıyla ilgilidir. Bu türden algoritmalarda yeni nesil çözümler oluşturulurken var olan popülasyondaki bilgilerin bir kısmı çeşitli stratejilerle yeni nesle aktarılmaktadır. Bu durum GA'da seçim ve çaprazlama operatörleriyle sağlanırken karınca kolonisi eniyilemesinde feromon matrisinin güncellenmesiyle ortaya çıkmaktadır.

P-metasezgisellerde başlangıç çözümlerin bulunduğu popülasyon oluşturulurken çeşitli yöntemler bir arada kullanılmaktadır. Bu durum sayesinde popülasyondaki çözümler arama uzayının çok farklı yerlerine erişebilmektedir. Bu sebepten dolayı P-metasezgisellerin keşfetme yetenekleri oldukça fazladır. Keşfetme gücünü ortaya koyan durum isabetli seçilmiş başlangıç çözüm oluşturma yöntemleridir. Doğru bir biçimde seçilmeyen başlangıç çözüm oluşturma stratejileri birbirine çok benzeyen fazla sayıda çözümün popülasyonda yer almasına sebebiyet verebilmektedir. Bu durumda algoritma keşfetme yeteneğinde kayıplar yaşamaktadır [2].

2.1.7. Metasezgisel algoritmaların melezleştirilmesi ve melez genetik algoritma

Eniyileme alanında algoritmaların birbirleriyle melezlenmesinde anlamlı bir artış gerçekleşmiştir. Geliştirilen melez algoritmayla birçok problem ve problemin gerçek hayat uygulaması ele alınmış ve çözümlere ulaşılmıştır [3]. Melezleştirme stratejileri olarak P-metasezgiseller, S-meteasezgiseller, matematiksel programlama, kısıt programlama ve makine öğrenmesi teknikleri kullanılarak son derece etkili algoritmalar üretilmektedir.

Metasezgisel algoritmalar geliştirilirken hem çözüm uzayını iyi keşfetmesi hem de güçlü bir derinlemesine arama yapması arzulanmaktadır. *P*-metasezgiseller, çözüm uzayını keşfetmede başarılıyken, uzayı derinlemesine aramada aynı başarıyı gösterememektedir. *S*-metasezgiseller ise uzayı derinlemesine aramada başarılıyken keşif başarıları düşük kalmaktadır. Araştırmacılar her iki avantajdan da faydalanabilmek adına algoritmaların güçlü yanlarını bir araya getirerek melezleştirmektedirler.

Popülasyon temelli bir yaklaşım olan GA, genellikle çözüm uzayının derinlemesine aranmasında yeteri kadar etkili olamamaktadır. Bu durumun üstesinden gelebilmek için genellikle yerel arama teknikleriyle algoritma melezleştirilmektedir. Böylelikle algoritmaya derinlemesine arama özelliği de kazandırılmaktadır. Bu şekilde güçlendirilmiş GA, genellikle melez genetik algoritma olarak adlandırılmaktadır. GA ve melezleştirme teknikleriyle ilgili daha fazla bilgiye Talbi'nin [1] kitabından ulaşılabilmektedir.

2.1.8. Değişken komşuluk arama

Değişken komşuluk arama (DKA) ilk olarak Hansen ve Mladenovic [4] tarafından 1997 yılında önerilmiştir. Bu metasezgisel yöntem, temel olarak komşuluk yapılarının arama esnasında sistematik değişimine dayanmaktadır. DKA'da çözümün iyileştirilmesi şu gerçekliklere dayanmaktadır:

- Bir komşuluktan bir diğerine geçerken ilk komşuluğun yerel en iyi noktasının bulunması gerekli değildir.
- Bir global en iyi çözüm tüm komşuluklar için yerel en iyi çözümdür.
- Farklı komşuluklardan elde edilen yerel en iyi çözümler göreceli olarak birbirlerine yakındır.

Üçüncü maddeye göre, yerel en iyi nokta global en iyi çözüme dair bazı bilgiler barındırmaktadır. Yerel en iyi çözümde gerçekleştirilecek olan bazı değişiklikler çözümü global en iyi noktaya götürebilmektedir. Öte yandan bu değişiklikleri akıllıca gerçekleştirmek gerekmektedir. Bu noktada DKA, Şekil 2.1'deki gibi komşulukları sistematik olarak değiştirerek global en iyi noktanın bulunmasını hedeflemektedir. Algoritma her bir iterasyonda çözüme kaldığı yerden devam etmemektedir. Bu durum, sarsma (shaking) olarak isimlendirilen bir operatörle sağlanmaktadır. Sarsma operatörü sayesinde bir iterasyon sonunda farklı bir aday çözüme geçilmekte ve sonraki iterasyona elde edilen yeni aday çözüm ile başlanmaktadır.

k_{max} belirle $k = 1, \dots, k_{max}$ *için komşuluk yapılarını belirle* (N_k)

Durdurma koşulunu ve yerel arama algoritmalarını belirle

Bir başlangıç çözüm oluştur x

k=1 olarak belirle

k= k_{max} *olana kadar*

Sarsma: x' rasgele komşu çözümü x'in k. komşuluğundan oluştur $(x' \in N_k(x))$

Yerel arama uygula (x'), elde edilen çözümü yerel eniyi olarak x'' olarak belirle

Eğer x'' var olandan daha iyi bir çözümse x=x'' olarak belirle ve (4)'e dön, diğer durumlarda k=k+1 olarak belirle

Şekil 2.1 *Değişken komşuluk arama algoritmasının sözde kodları*

DKA yönteminin en önemli avantajlarından biri fazla sayıda parametreye ihtiyaç duymamasıdır. Bu durum yoğun parametre duyarlılığını ortadan kaldırmakta ve algoritmanın tutarlılığını güçlendirmektedir. Öte yandan algoritmada yerel arama ve sarsma mekanizmaları arasındaki geçişlerdeki işleyiş, algoritmanın yapısını kritik derecede etkilemektedir. Özellikle sarsma kısmında çözümün orantısız bir biçimde bozulması algoritmanın, *çoklu başlangıçlı yerel arama algoritması* gibi çalışmasına sebebiyet verebilmektedir. Bir diğer yandan çözümün yeteri kadar sarsılmaması da yerel en iyi noktalardan kurtulmaya olanak tanımamaktadır. Algoritma tasarlanırken karar verilmesi güç olan bir diğer konu ise, yerel aramalardan hangi durumlarda ya da kaç tekrar sonunda vazgeçilip sarsma yapılacağına karar verilmesidir. Yerel arama algoritmalarına fazla şans verildiği durumlarda, algoritma derinlemesine arama bakımından güçlenirken aynı yerel en iyi nokta çevresinde fazla zaman harcamasına sebebiyet verebilmektedir. Burada algoritmanın derinlemesine arama ve keşfetme özellikleri arasında dengenin iyi kurulması büyük önem arz etmektedir.

2.2. Araç Rotalama Problemleri

Araç rotalama problemi (ARP), verilen kısıtlar altında bir araç filosunun müşterilere hizmet için en iyi rotalarının bulunmasıdır. Tedarik zinciri yönetiminde mal ve hizmetlerin dağıtımı ve toplanması odağında ele alınmaktadır. Np-zor sınıfı, problemler arasında yer alan ARP, literatürde en sık çalışılan eniyileme problemlerinden birisidir. [5] ve [6] bütün bu araştırmalara öncü olan çalışmalardır. Problemin sıklıkla çalışılmasının esas sebebi, uygulamada önemli bir yer tutmasıdır. Firmalar, dağıtımdan kaynaklı operasyonel maliyetlerini en aza indirerek yoğun rekabet ortamına ayak uydurmaya çalışmaktadırlar. Günümüzde ARP'ye türüne bağlı olarak kesin çözüm yöntemleri, en fazla 50-100 boyutunda problemler için işlevsel durumdadır [7]. Bu

sebeple, çalışmalar en iyi çözüme yakın çözümler sunan sezgisel ve metasezgisel algoritmalara yönelmiştir.

Literatürde ARP'nin *kapasite kısıtlı*, *zaman pencereli* ve *açık* gibi birçok çeşidi bulunmaktadır. ARP denildiğinde genellikle ilk akla gelen kapasite kısıtlı ARP'dir. Kapasite kısıtlı ARP'de aynı kapasiteye sahip olan bir araç filosu, bir depodan hareket ederek talepleri önceden bilinen müşterilere hizmet vermektedir. Her müşteri sadece bir araçtan hizmet almakta ve her müşterinin talebi karşılanmaktadır. Araçlar kapasitelerini aşamamaktadır ve hizmet sonunda depoya geri dönmektedir. Bu kısıtlar altında en az maliyetli rotaların bulunması hedeflenmektedir.

Bu çalışmada geliştirilen yöntemler öncelikle AARP'nin çözümü için olmakla birlikte küçük değişiklikler ile son zamanlarda ortaya çıkan diğer ARP türleri için de etkin çözüm yaklaşımları geliştirilmesinde kullanılabilir.

2.2.1. Son zamanlarda ön plana çıkan ARP alanları ve çözüm yaklaşımlarındaki yenilikler

ARP, uygulamadaki ihtiyaçlara göre çeşitlenmekte ve her geçen gün yeni boyutlar kazanmaktadır. Gerek teknolojik yenilikler, gerekse sanayinin ihtiyaçları çok farklı türde ARP'nin ortaya çıkmasına sebep olmuştur. Bununla beraber etkin kararları kısa zamanda verme ihtiyacı, araştırmacıları ARP'nin çözüm yöntemleri üzerinde çeşitli çalışmalara yöneltmiştir. Son zamanlarda ARP, gerçek örneklerden yola çıkılarak farklı kısıtlar altında ele alınmaktadır. Çeşitli kısıtların bir araya gelmesi ile ortaya çıkan ARP'ye, zengin ARP (Rich VRP) ismi verilmiştir. Lahyani ve ark. [8] ile Caceres-Cruz ve ark. [9], zengin ARP konusunda sınıflandırma ve literatür çalışmalarının incelemesini ele almışlardır. Yıldırım ve Çatay [10], *MathAnt* ismini verdikleri yöntemde kümelere ayırarak zaman pencereli ARP için zaman temelli karınca sistemi ve tam sayılı programlama kullanarak paralel bir metasezgisel algoritma önermişlerdir. Kalina ve ark. [11], aynı tür problem için ajan anlaşması (Agent Negotiation) kullanarak polinom zamanlı bir algoritma önermiş; fakat literatürdeki önemli yöntemlerin başarısına ulaşamamışlardır. Bektaş ve Lysgaard [12] ise zaman pencereli ARP'de rotaların alt ve üst zaman sınırlarının olduğu durumu ele almışlardır. Problem için çeşitli matematiksel modeller sunulmuş ve çözüme yönelik kesin bir yöntem önerilmiştir. Jabali ve ark. [13], lojistik hizmet sağlayıcıların zaman dayatması ile ortaya çıkan zaman pencereli ARP üzerinde çalışmışlardır. Yazarlar problemin çözümü için doğrusal programlama modeli

ile yerel arama algoritmasını bir arada kullanmışlardır. Spliet ve Desaulniers [14], kesikli zaman penceresine araç rotalarının atanması ve eniyilemesini iki basamaklı stokastik bir eniyileme problemi olarak ele almış ve bir dal-fiyat algoritması önermişlerdir. Yine zaman pencereli ve bölünebilir talepli ARP için McNabb ve ark. [15], yerel arama algoritmalarındaki taşıma operatörlerini karşılaştırmışlardır.

Gaur ve Singh [16], ilk olarak Kara ve arkadaşlarının [17] yayınladığı akaryakıt tüketiminin sabit olmadığı kümülatif ARP için bir sütun türetme modeli önermişlerdir. Yang ve Sun [18], elektrikli araçlar için pil değiştirme istasyonlarının belirlenmesini ve rotalarının planlanmasını birlikte ele almış ve SIGALNS isimli iki fazlı yasaklı arama ve düzenlenmiş tasarruf algoritmasını içeren bir algoritma önermişlerdir. Yang ve ark. [19], elektrikli araçlar için elektrik fiyatlarını da dikkate alan Partheno öğrenmeli GA önermişlerdir. Goeke ve Schneider [20], elektrikli ve elektrikli olmayan araçlardan oluşan karma filo ve zaman pencereli ARP için adaptif geniş komşuluk arama yöntemi üzerinde çalışmışlardır.

Bauer ve Lysgaard [21], rüzgar tribünlerinden dağılan kabloların düzenini bir AARP olarak ele almış, matematiksel programlama ve tasarruf algoritmasını kullanarak çözümler önermişlerdir. Dayarian ve ark. [22], süt toplama sistemi için heterojen filolu, çok depolu ve birçok kaynak kısıtlı ARP olarak sütun oluşturma temelli bir matematiksel model önermiş ve dal-fiyat yöntemi ile çözüm araştırmışlardır.

Dayarian ve ark. [23], çok periyotlu ARP için küme belirleme modeli oluşturmuş ve dal-fiyat algoritması ile çözüm önermişlerdir. Schneider ve ark. [24], ara noktalarda ürün boşaltmalardan ya da akaryakıt dolumundan dolayı durmaların olduğu ARP için adaptif bir değişken komşuluk arama algoritması geliştirmişlerdir. Li ve ark. [25], eş zamanlı dağıtma, toplamanın ve çok deponun bulunduğu ARP için adaptif komşuluk seçiminin olduğu bir yerel arama algoritması önermişlerdir. Avcı ve Topaloğlu [26], eş zamanlı dağıtma ve toplamanın olduğu ARP için adaptif bir yerel arama algoritmasını literatüre kazandırmışlardır.

Şahinyazan ve ark. [27], mobil kan toplama araçlarının rotalanmasını, toplanan kan miktarının en büyüklenmesinin yanı sıra maliyetlerin en küçüklenmesi olarak çok amaçlı ele almışlardır. Çok amaçlı yapıya matematiksel programlama ve sezgisel bir yaklaşım ile çözüm aramışlardır.

Talarico ve ark. [28], bir örnekte oluşturulan rotaların birbirine benzememesi temelli bir ARP modeli üzerinde çalışmışlardır. Çalışmada k-benzemez ARP ile alternatif çözümlerin ortaya konulmasını kolaylaştırdığı vurgulanmıştır.

Dimitrakos ve Kyriakidis [29], taleplerin rassal olduğu ve müşteri sıralarının önceden belirlendiği, hem dağıtım hem toplamanın olduğu ARP üzerinde çalışmışlardır. Euchi ve ark. [30], dinamik toplama ve dağıtımın olduğu ARP'nin çözümü için karınca kolonisi eniyilemesi ile 2-opt yerel arama algoritmasını bir arada kullanmışlardır. Wang ve ark. [31], eş zamanlı dağıtım ve toplamanın olduğu ARP için paralel bir tavlama benzetimi algoritması geliştirmişlerdir.

Bortfeldt ve ark. [32], üç boyutlu dikdörtgenler prizması şeklindeki parçaların yüklenmesi kısıtı altındaki ARP için iki melez algoritma üzerinde çalışmışlardır. Wei ve ark. [33] ise, iki boyutlu yükleme kısıtı eşliğinde klasik ARP için değişken komşuluk arama algoritması sunmuşlardır. Domingez ve ark. [34], ARP'yi iki boyutuyla birlikte ele almışlardır. Yazarlar geri dönüş yüklemeleriyle birlikte ARP'yi araç yükleme kısıtlarını da göz önünde bulundurarak modellemişlerdir. Problemin çözümü için bir büyük komşuluk arama algoritması önermişlerdir.

Azimi ve ark. [35], literatüre yeni bir tür ARP kazandırmışlardır. Hastanelerden toplanan kan örnekleri merkezi bir laboratuvara getirilmekte ve uygun cihazlar ile tahlil edilmektedir. Burada istenmeyen durum gelen örneklerin laboratuvarda trafiğe yol açmasıdır. Cihazlar belirli sayıda örneklerle çalışabilmekte ve örnekleri saklamak için laboratuvarların kapasiteleri belirlidir. Eğer aynı cihazda analiz edilecek örnekler yığılma şeklinde laboratuvara ulaşırsa kanlar bozulabilir ve tahlil edilemez duruma gelebilirler. Yazarlar bu gerçek hayat problemi için senkronize olmayan ARP modeli geliştirmiş, matematiksel programlama ve sezgisel yöntem ile probleme çözüm aramışlardır.

Uchoa ve ark. [36], var olan klasik ARP test problemlerinin, gerçek problemleri tam yansıtmadığı ve var olan yöntemlerle kolaylıkla çözülebildikleri gerekçe gösterilerek müşteri sayısı 100 ile 1000 arasında değişen yeni test problemleri sunmuşlardır.

Sze ve ark. [37], adaptif değişken komşuluk arama yöntemiyle büyük komşuluk arama yöntemini melezleştirmiştir. Yazarlar yöntemin gücünü literatür problemleri üzerinde test ederek göstermişlerdir.

Literatürde teslimat noktası fazla olan kapasite kısıtlı ARP üzerinde çalışmalar da güncelliğini korumaktadır. Geçmişten günümüze kadar büyük boyutlu problemlerin ele alındığı önemli çalışmalar şöyledir: Golden ve ark. [38], başlangıç çözümü Clarke ve

Wright'ın tasarruf algoritması ile oluşturarak özel komşuluk tanımlarıyla bir algoritma geliştirmiştir. Aynı çalışmada yazarlar, Xu ve Kelly'nin ekleme (Insertion) algoritması kullanılarak ağ akışı temelli bir yasaklı arama algoritması sunmuşlardır. Bir başka çalışmada Tarantilis ve Kiranoudis [39], adaptif hafıza tabanlı bir yasaklı arama algoritması geliştirmişlerdir. Diğer bir çalışmada Toth ve Vigo [40], özel bir komşuluk kavramıyla yasaklı arama algoritması sunmuşlardır. Önemli makalelerden birinde Li ve ark. [41], literatüre yeni büyük boyutlu ARP test problemleri kazandırmışlardır. Mester ve Braysy [42], çalışmalarında aktif yönlendirmeli bir değerlendirme stratejisi ile büyük boyutlu problemleri ele almışlardır. Kytöjoki ve ark. [43], etkin bir değişken komşuluk algoritması sunmuş ve gerçek hayatta karşılaşılan büyük boyutlu ARP'ler üzerinde hesaplamalı karşılaştırmalar yapmışlardır. Bu çalışmada ele alınan problemlerin boyutu 20,000 müşteriyi bulmaktadır.

2.2.2. Açık araç rotalama problemi

Açık araç rotalama probleminde (AARP), klasik araç rotalama probleminden farklı olarak depodan çıkan araçlar son müşterilerine ulaştıktan sonra depoya dönmemektedir. AARP ilk olarak Sariklis ve Powel [44] tarafından önerilmiştir. AARP maddeler halinde şu şekilde açıklanmaktadır:

- Her müşteri mutlaka ve sadece bir kez ziyaret edilerek müşterinin talebi karşılanır.
- Araçlar bir depodan çıkar ve müşterilerden birinde hizmeti sonlandırırlar.

AARP gerçek hayatta, özellikle firmaların kendi araç filolarına sahip olmadığı durumlarda ortaya çıkmaktadır. Firmalar, lojistik destek sağlayan şirketlerle anlaşmalar düzenlemekte ve ürünlerinin müşterilere teslimatından sonra araçların nasıl değerlendirileceği lojistik şirketinin takdirine çoğunlukla bu anlaşmalara göre bırakılmaktadır.

2.2.3. AARP için çözüm yaklaşımları

AARP için çalışmalarda sunulan çözüm yaklaşımları sezgisel yöntemler üzerinde yoğunlaşmaktadır. Tablo 2.3, Tablo 2.4 ve Tablo 2.5'te bu çalışmalardan ön plana çıkanlar özet olarak verilmektedir.

Tablo 2.3 *Literatür özet tablosu-1*

Yazarlar	Algoritma	Notlar
Sariklis ve Powell [44]	Önce kümele, sonra rotala	Algoritmanın ilk fazında araç kapasitesine göre kümeleme ve yapılan kümelemenin dengelenmesi gerçekleştirilmektedir. İkinci fazda ise kümeler en küçük kapsayan ağaç problemi şeklinde ele alınmakta ve açık rotalar oluşturulmaktadır. Uygun olmayan rotaları uygun hale getirmek başa için ceza maliyetleri verilmektedir.
Brandao [45]	Yasaklı arama	Algoritmanın ilk adımında en yakın komşu sezgiseli ve *k*-yakın komşu sezgiseli gibi yöntemlerle başlangıç çözümler belirlenmektedir. Yasaklı arama algoritmasında bir müşterinin bir rotadan alınarak başka bir rotaya yerleştirilmesi ve iki farklı rotadaki müşterilerin karşılıklı değiştirilmesi kullanmaktadır.
Tarantilis ve ark. [46]	Adaptif hafızalı yasaklı arama	Algoritma iki aşamadan oluşmaktadır. İlk fazda ağırlıklandırılmış kazançlar ile rota havuzu oluşturulmakta ve standart yasaklı arama algoritması ile çözümler iyileştirilmektedir. İkinci fazda ise "kemik" olarak isimlendirilen aynı rotada olması muhtemelen olan noktalar ayrılmaktadır. Elde edilen bu bilgiyle çözümler güncellenmekte ve yasaklı arama uygulanmaktadır.
Tarantilis ve ark. [47] Tarantilis ve ark. [48]	Eşik kabulü	Eşik kabulü tavlama benzetimi yönteminin bir deterministik halidir. Eşik değeri *T* amaç fonksiyonu değerindeki kabul edilen en büyük artıştır. İlk çalışmada *T*'nin arama sırasında artışına izin verilmektedir. Diğer çalışmada ise bir liste halindeki *T* değerleri kullanılmaktadır. Yazarlar 2-opt, bir müşterinin bir rotadan alınarak başka bir rotaya yerleştirilmesi ve iki farklı rotadaki müşterilerin karşılıklı değiştirilmesine yönelik arama yöntemlerini kullanmaktadır.

Tablo 2.4 *Literatür özet tablosu-2*

Yazarlar	Algoritma	Notlar
Fu ve ark. [49]	Yasaklı arama	Çalışmada geliştirilen "Öncelikle Uzaktaki" sezgiseli ile başlangıç çözümler oluşturulmaktadır. Bu yöntem ile henüz rotalanmamış depoya en uzaktaki müşteriden başlayarak araç dolana dek rotaya ekleme yapılmaktadır. Yasaklı arama sezgiselinde ise dört farklı komşuluk yapısı ile eniyi çözüm aranmaktadır.
Pisinger ve Ropke [50]	Adaptif büyük komşuluk arama	Adaptif büyük komşuluk arama yönteminde her tekrarda çözümü bozmaya ve sonrasında tekrardan onarmaya yönelik yöntemler seçilir. Daha iyi çözümlere ulaşmak için bozma ve onarmaya yönelik çeşitli sezgiseller kullanılmaktadır. Yeni çözümün kabulü için tavlama benzetimi yöntemi kullanılmaktadır.
Li ve ark. [51]	Record to Record Seyehat-Yasaklı arama algoritması	Yazarlar büyük boyutlu problemler türetmiş ve daha önce önerdikleri bir yöntemi AARP'ye uygulamışlardır. Yöntemde süpürme algoritması ile başlangıç çözüm oluşturulmaktadır. Yasaklı arama çatısı altında çeşitli komşuluk yapılarıyla eniyi çözüm aranmaktadır.
Flezsar ve ark. [52]	Değişken komşuluk arama	Çalışmada yazarlar dengeli rotalar oluşması için homojen araçlara en uzun boşaltma süresi tanımlamışlardır. Bununla beraber noktalar arası seyahat süresi ve her noktada talebin boşaltılma süresi mevcuttur. Bu bilgiler eşliğinde her araç toplam seyahat süresini aşamamaktadır. Çalışmada değişken komşuluk arama yöntemi hem seyahat süreli probleme hem de sıradan AARP'ye uygulanmıştır. Yazarlar algoritmanın gücünün rota içerisinden alınıp karşılıklı değiştirilerek çözümü iyileştiren geriye dönük rota parçalarından (reversing route segments) kaynaklandığını vurgulamaktadır.

Tablo 2.5 *Literatür özet tablosu-3*

Yazarlar	Algoritma	Notlar
Salari ve ark. [53]	Tam sayılı lineer programlama temelli sezgisel	Çalışmada başlangıç çözüm olarak geçmiş çalışmalarda bulunan en iyi sonuçlar kullanılmıştır. Rassal bir şekilde bozulan çözümler tam sayılı lineer programlama modeliyle onarılmaktadır. Çalışmada gösterilen yöntem ile literatürdeki bazı problemler için bilinen en iyi çözümler geliştirilmiştir.
Repoussis ve ark. [54]	Evrimsel algoritma	Çalışmada klasik bir evrimsel algoritma üzerinde çalışılmıştır. Çaprazlama operatörü olarak çoklu birey çaprazlaması gerçekleştirilmiştir. Amaç fonksiyonu iki kademeli olarak ele alınmıştır. İlk kademede gerekli araç sayısı ele alınmaktayken; ikinci kademede toplam uzaklık en küçüklenmektedir. Söz konusu çalışmada özellikle büyük boyutlu bazı literatür problemleri için bilinen en iyi çözümler güncellenmiştir.
Zachariadis ve Kiranoudis [55]	Statik hareket tanımlayıcı yasaklı arama	Çalışmada araç sayısı ve toplam uzaklık en küçüklenmeye çalışılmaktadır. İkiden fazla müşterinin yer aldığı rota segmentlerinden oluşan komşuluk yapıları ile aramalar gerçekleştirilmiştir. Tüm komşuluklar yerine tanımlanan yapılar incelenerek klasik yerel arama algoritmalarının karmaşıklığından kaçınılmıştır. Bu strateji sayesinden literatürdeki bazı problemlerin bilinen en iyi çözümlerden iyileşmeler sağlanmıştır.
Yu ve ark. [56]	Melez genetik algoritma ve yasaklı arama	Çalışmada bir gerçek hayat problemi ele alınmış ve AARP olarak modellenmiştir. Çatı algoritma olarak melezleştirilmiş bir genetik algoritma kullanılırken yerel aramada yasaklı aramadan faydalanılmıştır.
Şevkli ve Güler [57]	Salınımlı değişken komşuluk arama	Çalışmada bir salınım stratejisi geliştirilerek yerel en iyi noktalardan kaçınılmaya çalışılmıştır. Rota içi ve rotalar arası aramalar birbirini takip eden değişken komşuluk aramalar şeklinde ele alınmıştır.

2.2.4. Sabit heterojen filolu açık araç rotalama problemi

Firmaların kendi araç filolarının olmaması ya da söz konusu filoların yetersiz kaldığı durumda AARP ile sıklıkla karşılaşılmaktadır. Firmaların ürünlerinin taşınması için anlaşma yaptığı şirketlerin filolarının homojen olmaması mümkündür. Problemdeki "sabit" sözcüğü araç sayısının sınırlandırılmış olmasından gelmektedir. Bununla birlikte filoda yer alan araç kapasitelerinin farklı olması, araçlardan doğan sabit ve değişken maliyetlerin de farklılaşması anlamına gelmektedir.

Sabit heterojen filolu açık araç rotalama problemi (SHFAARP) ilk olarak Li ve ark. [58] tarafından çalışılmıştır. Çalışmada araçların sabit ve değişken maliyetleri de göz önünde bulundurulmuştur. Buna ek olarak rotaların dengeleri olabilmesi için tüm araçlara yönelik toplam katedebilecekleri mesafeyle ilgili bir üst sınır getirilmiştir. Problemin çözümü için yasaklı arama ile melezleştirilmiş çoklu başlangıçlı adaptif hafıza programlama metasezgiseli önerilmiştir. Yazarlar kendi stratejileriyle ürettikleri test problemlerine çözümü önerdikleri yöntemle aramışlardır.

SHFAARP'yi ele alan bir diğer çalışma ise Yousefikhoshbakht ve ark. [59] tarafından yapılmıştır. Yazarlar daha önceden oluşturulan literatür problemi olmadığı için Taillard'ın heterojen filolu araç rotalama probleminin test problemlerinden türeterek SHFAARP için test problemleri elde etmişlerdir. Çalışmada, yöntem olarak adaptif hafızalı melezleştirilmiş yasaklı arama kullanılmıştır.

Yousefikhoshbakht ve ark. [60] bir başka çalışmada SHFAARP için düzenlenmiş sütun türetme yöntemi önermişlerdir. Başlangıç çözümünün üretilmesi için düzenlenmiş süpürme algoritması ile birlikte ikili değiştirme (swap), ekleme ve 2-opt teknikleri birlikte kullanılmıştır. Daha sonra başlangıç çözümünden yola çıkarak sütun türetme yöntemiyle çözüm aramışlardır.

SHFAARP'nin matematiksel modeli ise şu şekildedir:

Parametreler:

$I, J = \{1, 2, \dots, n\}$: müşteriler kümesi

$K = \{1,2, \dots, m\}$: araçlar kümesi

r_k : k. aracın kapasitesi

d_j : j. müşterinin talebi

c_{ijk} : k. aracın i. müşteriden j. müşteriye gitme maliyeti

Karar değişkenleri:

$$x_{ijk} = \begin{cases} 1, & \text{eğer } k.\text{ araç } i.\text{müşteriden } j.\text{ müşteriye giderse} \\ 0, & d.d \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{eğer } k.\text{ araç } j.\text{ müşteriye uğrarsa} \\ 0, & d.d \end{cases}$$

Amaç fonksiyonu:

$$\min z = \sum_i \sum_j \sum_k c_{ijk} x_{ijk} \quad (2.2)$$

Kısıtlar:

$$\sum_{i,i\neq j} x_{ijk} = y_{jk}, \quad \forall j,k \ , j > 1 \quad (2.3)$$

$$\sum_{j,i\neq j,j>1} x_{ijk} \leq y_{ik}, \quad \forall\, i,k \ \ i > 1 \quad (2.4)$$

$$\sum_j \sum_k x_{1jk} \leq m \quad (2.5)$$

$$\sum_{j,j>1} d_j y_{jk} \leq r_k, \ \forall k \quad (2.6)$$

$$\sum_i \sum_k x_{ijk} = 1 \ \forall j > 1 \quad (2.7)$$

$$u_{ik} - u_{jk} + r_k x_{ijk} \leq r_k - d_j, \ i \neq j, i > 1 \text{ ve } j > 1 \text{ ve } d_j + d_i \leq r_k. \quad (2.8)$$

(2.2) ile toplam maliyetin en küçüklenmesi amaçlanmaktadır. (2.3) numaralı kısıt kümesi, $y_{jk} = 1$ olması durumunda, *j*. müşteriye *k*. aracın uğramasını garanti etmektedir. Kısıt $\sum_i x_{ijk} = 1$ ise, aynı aracın sadece bir müşteriden *j*. müşteriye gitmesini sağlayacaktır. (2.4) numaralı kısıt kümesi ile eğer *k*. araç *i*. müşteriye uğramış ise *k*. araç *i*. müşteriden yalnız bir müşteriye gidebilmesi ya da o müşteride rotasını sonlandırması sağlanmaktadır. Araç sayısının ve her bir aracın kapasitelerinin aşılmaması için sırasıyla (2.5) kısıtı ve (2.6) kısıt kümesi ile ifade edilmiştir. Her bir müşteriye mutlaka sadece bir aracın, yalnızca bir kez uğraması (2.7) numaralı kısıt kümesi ile sağlanmakta ve alt turların oluşması (2.8) numaralı kısıt kümesi ile engellenmektedir.

3. GERÇEK HAYAT PROBLEMİ ve GELİŞTİRİLEN KARAR DESTEK SİSTEMİ

Tez çalışmasında SHFAARP'nin bir gerçek hayat uygulaması ele alınmıştır. Problemin çözümüne yönelik çözüm yaklaşımlarının araştırılmasının yanı sıra etkin bir karar destek sisteminin oluşturulmasının üzerinde durulmuştur. Bu bölümün amacı, problemin verilerinin ön işlemlerden geçirilerek problemin çözümünün anlaşılır bir biçimde sunan kullanıcı dostu bir karar destek sistemi geliştirilmesidir. İzleyen alt bölümlerde firmanın problemi ve problemin aşamalarıyla birlikte geliştirilen karar destek sistemi açıklanmaktadır.

3.1. Problemin Özellikleri

Literatürde de ele alındığı gibi AARP'nin farklı alanlarda gerçek hayat uygulamaları bulunmaktadır. Bunlardan biri de firmaların üçüncü parti taşıma şirketleriyle olan anlaşmalarına göre ürünlerinin taşınmasından doğmaktadır. Bu çalışmada ele alınan gerçek hayat problemi bu grupta yer almaktadır. Problemin çözümünü arayan firma, üçüncü parti bir lojistik firmasının filosuyla dağıtım işlemlerini gerçekleştirmektedir. Yapılan anlaşma doğrultusunda firma rotası planlanan araçların geri dönüşü için herhangi bir maliyet altına girmemektedir. İşletme bünyesinde farklı türden ürünlerin üretildiği iki farklı fabrika (*F1* ve *F2*) bulunmaktadır. Söz konusu iki fabrika için de firmanın lojistik şirketiyle farklı anlaşmaları bulunmaktadır.

Firma iki fabrikada üretilen ürünlerin birlikte dağıtım planlarını oluşturmaktadır. Buna göre söz konusu anlaşmaya göre firmanın maliyetleri değişkenlik göstermektedir. Firmanın *F2* fabrikasında üretilen ürünler *F1*'de üretilenlere göre daha ağır; fakat hacim bakımından daha küçüktür. *F1*'de üretilen ürünler ise hacim bakımından *F2*'de üretilenlerden daha büyük; fakat ağırlık bakımından daha hafiftir. Üçüncü parti lojistik firması, kullanılan bir araçta *F2*'den bir ürünün taşınması durumunda tüm maliyetlerin ilgili plana göre yapılmasını istemektedir. Eğer sadece *F1*'den gönderilen ürünler söz konusu ise lojistik firmasıyla anlaştıkları ayrı bir plan bulunmaktadır. Buna göre maliyet planları şöyledir:

a- Eğer F2'den yük bulunmuyorsa:

Toplam Maliyet=Taşınan araç tipine göre son nokta maliyeti

+ Ek km * taşınan araç tipine ve son noktaya göre ek km maliyeti

+ (boşaltma sayısı – 1) * taşınan araç tipine göre ek boşaltma maliyeti

+ [boşaltma sayısı 3'ten büyük ise araç tipine göre sürücü konaklama maliyeti]

b- Eğer F2'den yük bulunuyorsa:

Toplam Maliyet=(Toplam tonaj) * (ton başına son nokta maliyeti)

+ (Ek km) *(sabit ek km maliyeti) * (toplam tonaj)

+ (boşaltma sayısı – 1) * (taşınan araç tipine göre ek boşaltma maliyeti)

+ (boşaltma sayısı 3'ten büyük ise araç tipine göre sürücü konaklama maliyeti)

*F2'*den yük bulunması (*b*) durumunda taşınan yükün ağırlığına bağlı olarak tanımlanan maliyet bileşeni, araçların yüklendiği toplam ağırlık miktarı ile son nokta için tanımlanmış olan ağırlık maliyetinin çarpılması ile elde edilmektedir. Ek mesafeye bağlı maliyet, toplam tur uzunluğu ile tur üzerindeki son nokta uzaklık farkının (km cinsinden katedilen ek mesafe) ek km maliyeti ile çarpılması ile elde edilmektedir. Bu iki maliyetin yanı sıra boşaltma maliyeti, ek boşaltma sayısına bağlı olarak hesaplanmaktadır. Boşaltma sayısı üçten büyük ise sürücü konaklama maliyeti araç tipine bağlı olarak toplam maliyete eklenmektedir.

*F2'*den yük bulunmaması (*a*) durumunda ise son nokta maliyetleri ele alınan rotanın son uğrayacağı yere ve kullanılan araç tipine göre değişiklik göstermektedir. Burada ele alınan son nokta maliyetlerinde toplam tonajın bir önemi bulunmamaktadır. Bunun yanı sıra bu durumda maliyet fonksiyonunda ek kilometre maliyeti de son uğrayacağı yere ve kullanılan araç tipine göre belirlenmektedir.

Tablo 3.1 *Kullanılan araç türleri ve kapasiteleri*

Araç Türü	Palet Sayısı	Tonaj
Kamyon	30	16
Tır	64	26

Firma dağıtım planını oluştururken Tablo 3.1'deki gibi kamyon ve tır olmak üzere iki farklı türden araç kullanmaktadır. Üçüncü parti lojistik şirketiyle yapılan anlaşmaya göre iki tip araçtan da belirli sayıda araç, firma tarafından kullanılabilir durumdadır. Bu yüzden firmanın problemi bir tür SHFAARP'dir. Öte yandan kullanılan araç tipine bağlı olarak son nokta maliyetleri de değişkenlik göstermektedir.

3.1.1. Uzaklık bilgilerinin elde edilmesi

Problemin önemli girdilerinden biri olan uzaklık bilgilerinin elde edilebilmesi için harita programlarından yararlanılmıştır. Bu programın ihtiyacı gerçek kara yolu uzaklıklarıyla problemin çözülme ihtiyacıdır. Doğrudan koordinatlara bağlı öklid uzaklıkları üzerinden bir uzaklık matrisi oluşturulması halinde hesaplamaların gerçekçi olduğundan bahsedilmesi ve adil karşılaştırmalar yapılabilmesi olanaklı olmamaktadır.

Firmanın müşterilerine ait posta kodları kullanılarak Google Maps API ile posta koduna göre sorgulama yapan bir Excel VBA kodu yazılmıştır. Posta kodlarından koordinatların elde edilmesi tamamlandıktan sonra, karar destek sisteminin model tabanına girdi olacak uzaklık matrisinin hesaplanma aşaması gelmektedir. Bu aşamada Python programlama diliyle açık kaynak kodlu harita veri tabanlarından olan OpenStreetMap API kullanılmış ve uzaklık matrisinin elde edilmesi için bir program yazılmıştır. OpenStreetMap'in açık kaynak kodlu olması ve herhangi bir ücretlendirme sisteminin olmaması sayesinde sınırsız sayıda sorgu kolaylıkla gerçekleştirilebilecektir.

3.1.2. Firma verilerine ön işlem uygulanması

Firma, ele alınan gerçek hayat probleminde planlamalarını günlük olarak gerçekleştirmektedir. Firmanın verilerine göre araçların bir kısmının tam dolu olarak tek bir teslimat noktasına gönderilebileceği anlaşılmaktadır. Tablo 3.2'de yer alan örnekte 5. ve 8. müşteride tonaj bakımından, 2. müşteride ise palet bakımından bu durum ortaya çıkmaktadır. Burada tek bir fabrikadan çıkan ve tam dolu olarak aynı teslimat noktasına gidecek bir aracı planlamak kolaylıkla ele alınabilir durumdadır. Öte yandan aynı teslimat noktasına *F1* ve *F2*'den taleplerin birleştirilerek ve iki araç tipinden birine karar vererek göndermek ayrı bir problemdir.

Problemin ele alınmasına öncelikli olarak tek noktadan dolu gidecek araçlardan başlanılmıştır. İlk bakışta dolu gidecek araçların direkt kapasitelerine göre doldurulması akla gelmektedir. Bu yol kullanıldığında kullanılan araç sayısı fazlalığı ve dolayısıyla ortaya çıkan ekstra araç maliyetleri problemi ortaya çıkmaktadır. Çünkü *F1*'den olan ürünler ağırlık bakımından kapasiteyi sınırlandırırken *F2*'den olan ürünler hacim bakımından kapasiteyi sınırlandırmaktadır. Dolayısıyla olabildiğince *F1* ve *F2* ürünlerinin belirli oranlarda yüklenmesi çok daha etkili olmaktadır.

Bu durumun en verimli şekilde ele alınabilmesi için her talep noktasından olan taleplere aşağıda yer alan matematiksel model uygulanmaktadır. Bu planlama göz önüne alınırken en fazla miktarda talebin teslim edilebildiği ve maliyet bakımından avantajlı olan *tır* kullanılmaktadır.

Tablo 3.2 *Örnek problem verileri*

NO	Posta Kodu	Müşteri	F1		F2		Koordinat	
			ton	palet	ton	palet	x	y
1	17000	ÇANAKKALE	3	9	0	0	40,35	27,97
2	22800	KEŞAN	21,50	70	4,04	4	39,69	29,16
3	10200	BANDIRMA	3,61	14	0	0	36,58	36,18
4	16400	İNEGÖL	2,23	13	1,57	2	36,59	30,57
5	17000	ÇANAKKALE	0	0	110,75	145	41,05	28,82
6	45000	MANİSA	0	0	0,16	1	40,96	40,03
7	27000	GAZİANTEP	8,05	13	0	0	37,31	40,73
8	31200	HATAY	0	0	71,80	73	39,99	32,70
9	34180	ISTANBUL AVRUPA	0	0	1,02	2	41,09	41,10
10	34540	ISTANBUL AVRUPA	7,04	12	0	0	38,36	38,33
11	48300	FETHİYE	0	0	13,13	13	40,32	36,55
...	...	...	...	...	...	...	...	...
145	34007	ISTANBUL AVRUPA	0	0	0,20	1	41,04	29,00

Parametreler:

$i = \{1, 2, \dots, n\}$: müşteriler kümesi

$f_{1palet}(i)$: *i.* müşterinin *F1* olan talebin palet miktarı

$f_{2palet}(i)$: *i.* müşterinin *F2* olan talebin palet miktarı

$f_{1tonaj}(i)$: *i.* müşterinin *F1* olan talebin tonaj miktarı

$f_{2tonaj}(i)$: *i.* müşterinin *F2* olan talebin tonaj miktarı

$f_{1oran}(i)$: $\frac{f_{1tonaj}(i)}{f_{1palet}(i)}$

$f_{2oran}(i)$: $\frac{f_{1tonaj}(i)}{f_{1palet}(i)}$

Karar değişkenleri:

$x_1(i) = i.$ müşteriden planlanacak *F1* için talep katsayısı

$x_2(i) = i.$ müşteriden planlanacak *F2* için talep katsayısı

Amaç fonksiyonu:

$$\max z(i) = \frac{x_1(i)\, f_{1oran}(i)}{tır\ tonaj\ kap.} + \frac{x_2(i) f_{2oran}(i)}{tır\ palet\ kap.} \tag{3.1}$$

Kısıtlar:

$$x_1(i) f_{1oran}(i) + x_2(i) f_{2oran}(i) \leq tır\ tonaj\ kapasitesi \tag{3.2}$$

$$x_1(i) + x_2(i) \leq tır\ palet\ kapasitesi \tag{3.3}$$

$$x_1(i) \leq f_{1palet}(i) \tag{3.4}$$

$$x_2(i) \leq f_{2palet}(i) \tag{3.5}$$

Modelde yer alan kısıtlardan (3.2) ile her iki fabrikadan alınan toplam siparişin tır tonaj kapasitesini aşmamasını sağlanmaktadır. Benzer şekilde (3.3) ile tır palet kapasitesinin aşılmaması sağlanmaktadır. (3.4) ve (3.5) ile her iki karar değişkeniyle ilgili fabrikadan olan taleplerin palet sayısı ile sınırlandırılmaktadır. (3.1)'de yer alan amaç fonksiyonuyla hem tonaj hem de palet bakımından dengeli ve etkin şekilde tırların yüklenmesini sağlayacak oranlar toplamı en büyüklenmektedir. Elde edilen tam sayılı karar değişkenleri ile her talep noktasına dolu tırlardan kaçar tane çıkarılabildiği şu şekilde hesaplanmaktadır:

Eğer $x_1(i) == 0$, tamF1=0 *Aksi Taktirde*: tamF1=tamsayı$(f_{1palet}(i)/\ x_1(i))$

Eğer $x_2(i) == 0$, tamF2=0 *Aksi Taktirde*: tamF2=tamsayı$(f_{2palet}(i)/\ x_2(i))$

tamF1 ve *tamF2* değerlerinden küçük olanı kadar tır i. talep noktasından her bir araçta F1'den $x_1(i)$ kadar palet, F2'den $x_2(i)$ kadar palet eklenerek dolu gidecek tırlar planlanmaktadır. Arta kalan talepler ise ayrı bir veri dosyasına kaydedilmektedir.

3.1.3. Arayüzlerin geliştirilmesi

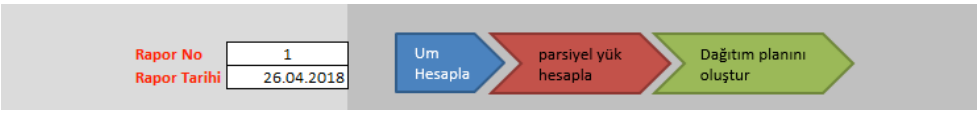
Karar destek sistemi geliştirilirken önemli öğelerden biri arayüzlerin oluşturulmasıdır. Ele alınan gerçek hayat problemi için geliştirilen karar destek sisteminde sadece birkaç arayüz bulunmaktadır. Kullanıcı tercihlerini ve raporların görüntülenmesini Excel üzerinden gerçekleştirmektedir.

Kullanıcının Şekil 3.1 ve Şekil 3.2'de ilk gördüğü arayüzde, problem verilerini girebilmektedir. Bunun yanı sıra üç adet buton bulunmaktadır. Bu butonlardan *um hesapla* ile o veriler için bir uzaklık matrisi oluşturulmaktadır. Uzaklık matrisinin oluşturulması butonuna basıldıktan sonra elde edilen gerçek kara yollarına ait uzaklık matrisi kullanıcının görmediği bir sayfaya yazılmaktadır. Kullanıcı rapor tarihi ve rapor numarasını değiştirmediği taktirde, bu butonu tekrar kullanmayı denemesi durumunda daha önce aynı verilerle bir uzaklık matrisi oluşturulduğuna dair bir uyarıyla karşılaşmaktadır.

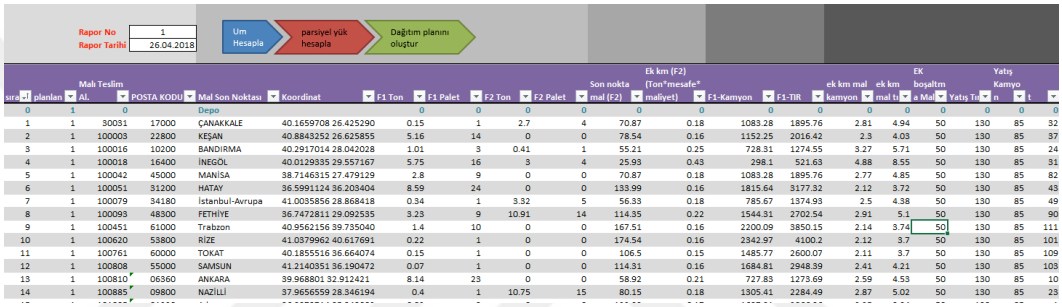
Kullanıcı, uzaklık matrisi hesaplama butonuna bastıktan sonra raporların doğru bir biçimde oluşturulması için *parsiyel yük hesapla* butonu kullanmaktadır. Bu buton ile birlikte tam dolu gidecek araçlar Bölüm 3.1.2'deki gibi gerçekleştirilmektedir. Kullanıcı bu butonu kullanmadan *dağıtım planını oluştur* butonunu kullanırsa yine bir uyarıyla karşılaşmaktadır. Bu uyarıda ise kullanıcının henüz tam dolu gidecek araçları planlamadığı bildirilmektedir. *Dağıtım planını oluştur* butonuylaysa SHFAARP izleyen

bölümde açıklanan karar destek sisteminin model tabanıyla çözülerek bir başka Excel sayfasına raporlar oluşturulmaktadır.

Oluşturulan raporda dağıtım planına ilişkin maliyet bilgilerinin yanı sıra, rota numaralarının üzerine tıklanması durumunda rotaların haritadaki durumu görüntülenebilmektedir.



Şekil 3.1 *Karar destek sisteminde yer alan butonlar*



sıra	planlan	Malı Teslim Al.	POSTA KODU	Mal Son Noktası	Koordinat	F1 Ton	F1 Palet	F2 Ton	F2 Palet	Son nokta mal (F2)	Ek km (F2) (Ton*mesafe* maliyet)	F1-Kamyon	F1-TIR	ek km mal kamyon	ek km mal tı	EK boşaltm a Mal	Yatış Te	Yatış Kamyo n	t	
0	1	0		Depo		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	30031	17000	ÇANAKKALE	40.1659708 26.425290	0.15	1	2.7	4	70.87	0.18	1083.28	1895.76	2.81	4.94	50	130	85	32	
2	1	100003	22800	KEŞAN	40.8843252 26.625855	5.16	14	0	0	78.54	0.16	1152.25	2016.42	2.3	4.03	50	130	85	37	
3	1	100016	10200	BANDIRMA	40.2917014 28.042028	1.01	3	0.41	1	55.21	0.25	728.31	1274.55	3.27	5.71	50	130	85	24	
4	1	100018	16400	İNEGÖL	40.0129335 29.557167	5.75	16	3	4	25.93	0.43	298.1	521.63	4.88	8.55	50	130	85	31	
5	1	100042	45000	MANİSA	38.7146315 27.479129	2.8	9	0	0	70.87	0.18	1083.28	1895.76	2.77	4.85	50	130	85	82	
6	1	100051	31200	HATAY	36.5991124 36.203404	8.59	24	0	0	133.99	0.16	1815.64	3177.32	2.12	3.72	50	130	85	43	
7	1	100079	34180	İstanbul-Avrupa	41.0035856 28.868418	0.34	1	3.32	5	56.33	0.18	785.67	1374.93	2.5	4.38	50	130	85	49	
8	1	100093	48300	FETHİYE	36.7472811 29.092535	3.23	9	10.91	14	114.35	0.22	1544.31	2702.54	2.91	5.1	50	130	85	90	
9	1	100451	61000	Trabzon	40.9562156 39.735040	1.4	10	0	0	167.51	0.16	2200.09	3850.15	2.14	3.74	50	130	85	111	
10	1	100620	53800	RİZE	41.0379962 40.617691	0.22	1	0	0	174.54	0.16	2342.97	4100.2	2.12	3.7	50	130	85	101	
11	1	100761	60000	TOKAT	40.1855516 36.664074	0.15	1	0	0	106.5	0.15	1485.77	2600.07	2.11	3.7	50	130	85	109	
12	1	100808	55000	SAMSUN	41.2140351 36.190472	0.07	1	0	0	114.31	0.16	1684.81	2948.39	2.41	4.21	50	130	85	103	
13	1	100810	06360	ANKARA	39.968801 32.912421	8.14	23	0	0	58.92	0.21	727.83	1273.69	2.59	4.53	50	130	85	10	
14	1	100885	09800	NAZİLLİ	37.9656559 28.346194	0.4	1	10.75	15	80.15	0.18	1305.41	2284.49	2.87	5.02	50	130	85	23	

Şekil 3.2 *Karar destek sisteminin veri giriş ekranı*

3.1.4. Önerilen melez genetik algoritma

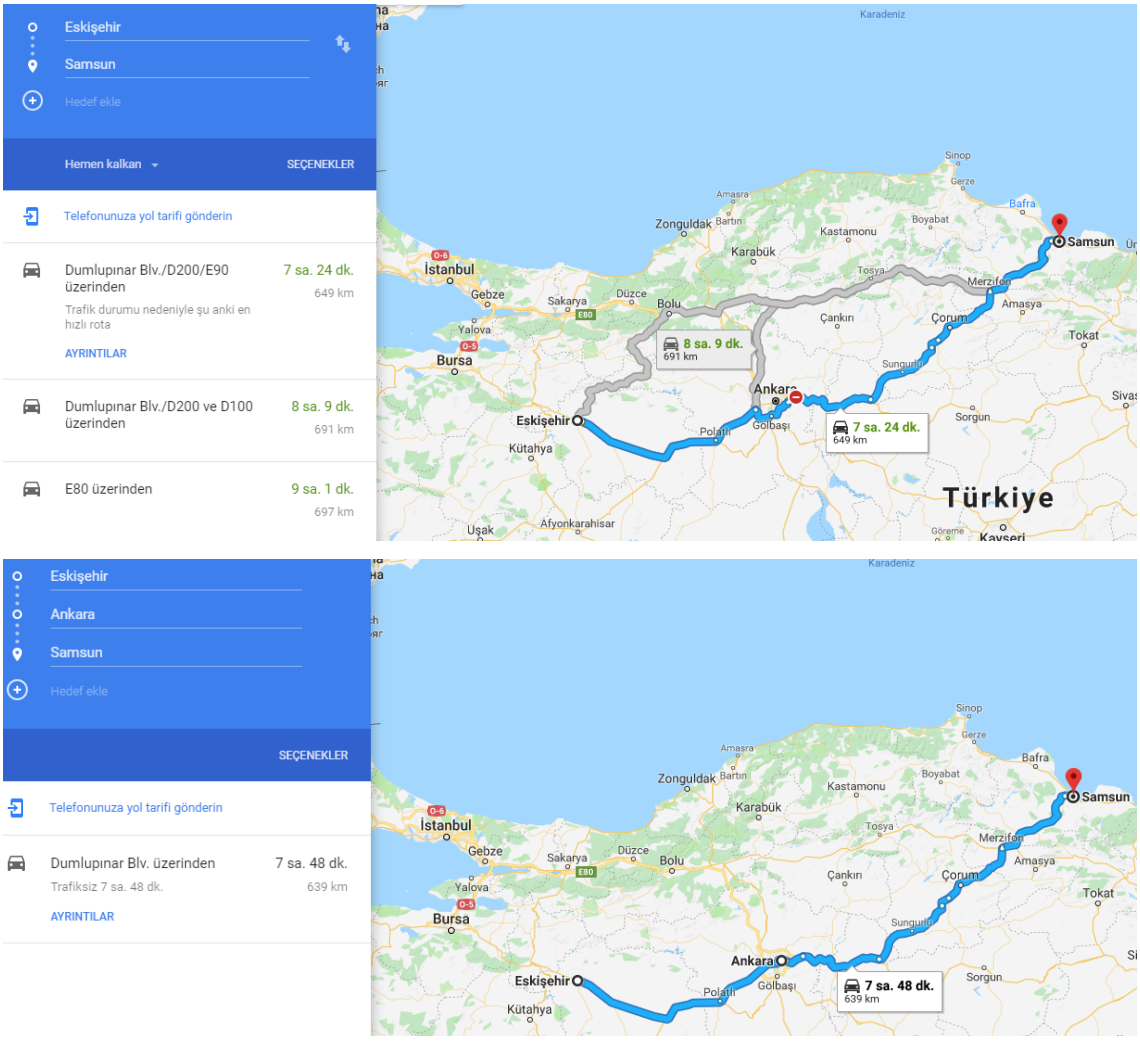
Dolu gidecek araçlar planlanlandıktan sonra kalan talepler için problem ele alınırken öncelikle, uygulanması göreceli olarak daha kolay ve kullanımı yaygın olan melez genetik algoritma tercih edilmiştir. Daha önceden de bahsedildiği gibi popülasyon temelli bir algoritma olan GA'nın keşfetme özelliğinin yanı sıra derinlemesine arama özelliğini de güçlendirebilmek adına yerel arama algoritmalarıyla melezleştirme işlemleri yapılmaktadır.

Geliştirilen karar destek sisteminin model tabanında etkin ve hızlı sonuçlar üreten bir algoritma geliştirmek adına benzer melezleştirme stratejisinden faydalanılmıştır. Buna göre algoritmada seçim, çaprazlama, mutasyon ve elitizm operatörlerinin yanı sıra bir yerel arama algoritma operatörü de bulunmaktadır.

Önerilen melez genetik algoritmada 1000 birey bulunmaktadır. Diğer bir deyişle 1000 farklı başlangıç çözümle problem ele alınmaktadır. Başlangıç çözümlerinin yarısı rassal üretilirken diğer yarısında izleyen paragraflarda açıklanan *ilişki* ve *yakınlık* matrisi

kullanılmaktadır. Bu matrisler kullanılarak oluşturulan başlangıç çözümler ilgisi olmayan talep noktalarının aynı rotada planlanmasının önüne geçilmektedir.

İlişki matrisi: İkili değişkenler ile bu matris oluşturulurken depodan bir noktaya gidilmesi durumunda yol üzerinde bulunan destinasyonların belirlenmesi amaçlanmıştır. Matrisin oluşumu için izlenilen yolu açıklayan örnek şöyledir: Depo Eskişehir'deyken *Ankara-Samsun* arasındaki ilişki *Eskişehir-Samsun – (Eskişehir-Ankara + Ankara-Samsun)<tolerans* koşulu gerçekleştiği durumda 1, diğer durumlarda 0 olmaktadır. Söz konusu matris asimetriktir. Matriste *Eskişehir-Samsun* 1 değeri alırken *Samsun-Eskişehir* 0 değeri almaktadır. Bu örnekteki durum Ankara'nın Eskişehir'den Samsun'a gidilirken bir kara yolu üzerinden olmasından kaynaklanmaktadır.



Şekil 3.3 *İlişki matrisi örneği*

Yakınlık matrisi: Bu matris ile her teslimat noktası belirli bir uzaklıkta (örneğin 100 km) bulunan diğer destinasyonları belirlenmektedir. Bu uzaklıktaki destinasyonların

yakınlık matrisindeki değerleri 1 olmaktadır. Örneğin; Şekil 3.4'te görülen örnekte Eskişehir-Bilecik matriste 1 değerini almaktadır. Bu matris simetriktir.

Şekil 3.4 *Yakınlık matrisi örneği*

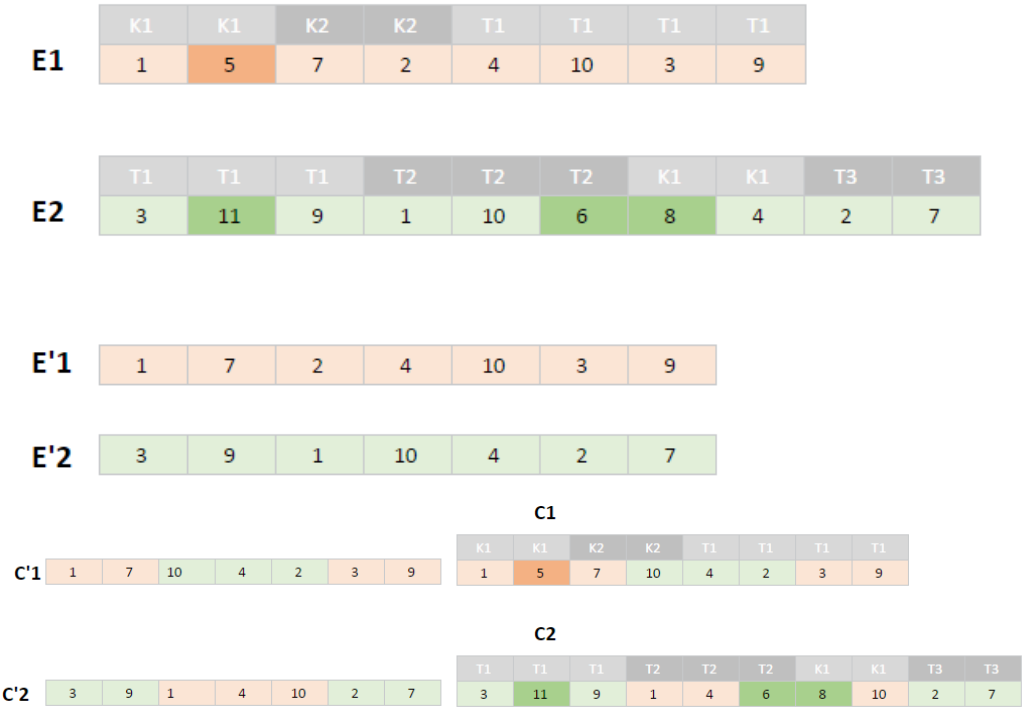
Bireylerin ilk kez amaç fonksiyonlarının değerlendirilmesinden sonra sırasıyla seçim, çaprazlama, mutasyon, yerel arama ve elitizm operatörleri kullanılmaktadır. Aynı sırada en fazla tekrar sayısı kadar tüm operatörler kullanılmakta ve elde edilen sonuç geri döndürülmektedir. Amaç fonksiyonun ilk kez değerlendirilmesi ve her tekrarda birer kez olmak üzere kapasite bakımından uygun olmayan çözümler cezalandırılmaktadır.

Önerilen yaklaşımda kullanılan seçim operatörü, temel olarak bir turnuva seçim yöntemidir. Çaprazlama operatörüne göndermek üzere, yeni nesli oluşturacak bireyler bu adımda seçilmektedir. Popülasyondan rasgele seçilen, herhangi iki birey amaç fonksiyonu değerleri birbirleriyle karşılaştırılmaktadır. Amaç fonksiyonu değeri daha iyi olan birey ön çalışmalar sonucu elde edilen değere göre 0.85 olasılıkla yeni nesli oluşturmak üzere çaprazlama operatörüne gönderilmektedir.

Çaprazlama operatörü, seçim operatöründen gelen bireylerin taşıdıkları bilgileri birbirleri ile değiştirmesini sağlayarak bir sonraki nesle aktarıldığı adımdır. Seçim operatörü uygulandıktan sonra oluşan ara popülasyon bireylerinin %80'i çaprazlanmakta, kalanlar ise yeni nesle doğrudan aktarılmaktadır. Bu operatör çalıştığında öncelikli olarak rotalardaki son noktaları konumlarına göre Şekil 3.5'teki gibi altı bölgeye ayırmaktadır.

Şekil 3.5 *Çaprazlama için bölgelere ayırma*

Bölgeler rotaların son nokta elemanlarına göre gruplandıktan sonra tüm bölgelere ayrı ayrı çaprazlama uygulanmaktadır. Bu işlemin amacı çaprazlama sonrası istenmeyen rotaların oluşmasının önüne geçilmesidir. Örneğin; Doğu Karadeniz son noktalı bir rotanın, Batı Akdeniz son noktalı bir rota ile çaprazlaması gerçekleştirilmemektedir. Çaprazlama tekniği olarak tek nokta, iki nokta ve pozisyona dayalı çaprazlama teknikleriyle ön çalışmalar yapılmış ve daha iyi sonuçlar vermesi sebebiyle pozisyona dayalı çaprazlamanın kullanılmasına karar verilmiştir.



Şekil 3.6 *Pozisyona dayalı çaprazlama örneği*

Bölge bazında çaprazlama şu şekilde yapılmaktadır: *E1* ve *E2* iki bireyin aynı bölgeden olan kısımlarını göstermektedir. Bu ebeveynlerden her ikisinde de ortak olan genleri ile eşit uzunluğa sahip alt diziler oluşturulur (*E'1, E'2*). *E'1* ve *E'2* kullanılarak tek nokta çaprazlama yöntemiyle çocuk bireyler oluşturulur (*C'1, C'2*). Elde edilen yeni bireylerden, ebeveynlerde ortak olmayan genleri sabit tutarak çocuk bireyler güncellenir (*C1, C2*). Şekil 3.6'da bu durumu açıklayan bir örnek bulunmaktadır.

Çaprazlama operatöründen sonra bireylerin tümü mutasyon operatörüne gelmektedir. Burada öncelikle bireyden iki gen rassal olarak seçilmektedir ve bu iki genin yerlerinin değişmesinde ortaya çıkan amaç fonksiyonu değeri ile değişiklikten önceki amaç fonksiyonu arasındaki fark negatif ise değişiklik kabul edilmektedir.

Yerel arama operatörü ise temel olarak bir bireydeki ikili değişikliklere bakarak o bireyin iyileştirilmesi amacıyla kullanılmıştır. Bu operatör sayesinde algoritmaya derinlemesine arama özelliği katılmaktadır.

3.2. Test Problemlerinin Üretilmesi ve Melez Genetik Algoritmada Uygulanması

Karar destek sisteminin model tabanında yer alan melez genetik algoritmanın performansının ortaya konulabilmesi için firmanın verileri esas alınarak bazı test problemleri türetilmiştir. Melez genetik algoritma, türetilen test problemleri ile birlikte çalıştırılmış ve sonuçlar raporlanmıştır.

3.2.1. Test problemlerin üretilmesi

Firmanın iki ayrı fabrikasından elde edilen ham verilerden, tam dolu gidecek araçların birlikte planlanması gerçekleştirildikten sonra geriye kalan siparişlerden bazı test problemleri üretilmiştir. Problemler üretilirken gerçek veriler belirli katsayılarla çarpılarak maliyet ve talep verileri türetilmiştir.

Tablo 3.3 *Üretilen test problemlerinin özellikleri*

Problem adı	n	Maks. Kamyon sayısı	Maks Tır sayısı
EO1	117	15	35
EO2	117	15	35
EO3	40	15	35
EO4	95	15	35
EO5	90	15	35
EO6	80	15	35
EO7	110	15	35
EO8	92	15	35

Gerçek uygulamada firmanın çalıştığı üçüncü parti şirkette her iki araç türünden de belirli sayılarda bulunmaktadır. Öte yandan bu araç sayıları istisnalar hariç firmanın talebini karşılayacak durumdadır. Gerçeğine uygun olması bakımından üretilen test problemlerinde bu duruma dikkat edilmiştir. Bununla beraber firmaya lojistik destek sağlayan üçüncü parti şirketten kiralanarak kullanılan filonun yaklaşık üçte birinin tır olduğu bilindiğinden maksimum kamyon ve tır sayısı belirlenirken bu durum göz önünde bulundurulmuştur.

Bir diğer durum ise talep noktalarının sadece bir kısmında her iki fabrikadan da sipariş bulunmasıdır. Test problemleri oluşturulurken buna dikkat edilmiştir. Kimi talep noktalarından yalnızca *F1* ya da *F2*'den sipariş bulunurken kimi talep noktalarından ise her iki fabrikadan da sipariş bulunmaktadır. Üretilen test problemleri Tablo 3.3'te görülmektedir.

3.2.2. Test problemlerinden elde edilen sonuçlar

İki ayrı fabrikadan elde edilen veriler kullanılarak üretilen test problemleriyle karar destek sisteminin model tabanında yer alan melez GA'dan elde edilen sonuçlar Tablo 3.4'teki gibidir. Firmanın çalışma öncesi iki fabrikaya gelen taleplere bir arada planlama yapmaması sebebiyle elde edilen maliyetleri gerçek maliyetler ile karşılaştırmak mümkün olmamaktadır. Bununla beraber, çözümlerin etkinliğinin karşılaştırılması karar destek sisteminin güvenilirliği açısından son derece kıymetli olduğu bilinmektedir.

Tablo 3.4. *Melez genetik algoritma ile elde edilen sonuçlar*

Problem	n	Kamyon sayısı	Tır sayısı	Ortalama	St. Sapma	En iyi değer	Ort. süre
EO1	117	11	25	83447.5	300.45	82344.3	273.4
EO2	117	9	35	86215.8	281.46	85647.7	248.6
EO3	40	7	17	37141.5	182.89	36740.5	155.2
EO4	95	9	29	83512.4	251.76	82815.5	200.5
EO5	90	11	26	71313.5	207.22	70752.9	198.7
EO6	80	15	22	64235.2	176.56	63995.5	182.5
EO7	110	14	31	81600.1	243.78	81062.2	260.1
EO8	92	9	27	64786.2	248.78	64428.0	203.5

Test problemlerinden elde edilen sonuçların yer aldığı tabloda üçüncü ve dördüncü sütunda yer alan kamyon ve tır sayısı, en iyi değer elde edilirken kullanılan araç sayılarını göstermektedir.

Üretilen test problemleri için elde edilen sonuçların değerlendirilmesi ve melez genetik algoritmanın etkinliğinin ortaya konulabilmesi gerekmektedir. Bu sebeple karşılaştırma yapılabilmesi için bir başka algoritmaya ihtiyaç duyulmuştur. İzleyen bölümde üzerinde çalışılan bu algoritma tüm detaylarıyla açıklanmaktadır.

4. AARP İÇİN ÜÇ AŞAMALI DEĞİŞKEN KOMŞULUK ARAMA ALGORİTMASI

Önceki bölümde, karar destek sistemi ve model tabanında kullanılan popülasyon temelli melez GA sunulmuştu. Bu bölümde ise karşılaştırmaların yapılabilmesi için bir DKA algoritması üzerinde çalışılmıştır. DKA'nın tercih edilmesindeki temel mantık, algoritmanın arama esnasında komşulukları sistematik olarak değiştirmesi ve bu değişikliklere bağlı olarak çözüm uzayının daha etkin aranabilmesidir.

Algoritma geliştirilirken rota içi arama DKA, rotalar arası arama DKA ve sarsma mekanizması şeklinde üç ana bölüm yer almaktadır. Literatürde Sevkli ve ark. [57] benzer bir strateji izlemişler ve başarılı sonuçlar elde etmişlerdir. Tez çalışmasında literatürdeki bu çalışmadan özgün olarak ayrılarak, farklı komşuluklar ve farklı komşuluk değiştirme stratejileriyle beraber farklı sarsma mekanizmaları ele alınmıştır.

Bu bölümde karar destek sisteminin model tabanının iyileştirilmesi için geliştirilen algoritma tüm detaylarıyla birlikte açıklanmaktadır. Bununla birlikte, algoritmanın performansı literatürden AARP test problemleri üzerinde incelenmektedir.

4.1. Başlangıç Çözümlerin Oluşturulması

Başlangıç çözümlerin oluşturulmasında öncelikli olarak farklı yöntemler incelenmiştir. İncelenen başlangıç çözümü oluşturma teknikleri arasında Solomon'un klasik ekleme (insertion) [61] yöntemi, stokastik klasik ekleme yöntemi, süpürme yöntemi ve önce *k*-ortalamalar tekniğiyle kümele daha sonra klasik ekleme ile rotaları oluşturma yer almaktadır. Karşılaştırılan yöntemlerin tamamında başlangıç çözüm oluşturulurken kapasite ve araç kısıtından dolayı dışarıda kalan talep noktaları olması halinde çıkar-ekle teknikleriyle başlangıç çözüm olurlu hale getirilmektedir. Tüm bu ön çalışmalar sonucunda önce *k*-ortalamalar tekniğiyle kümele daha sonra klasik ekleme ile rotaları oluşturma stratejisinin daha iyi noktalardan aramaya başlamaya olanak sağladığı anlaşılmıştır. Buna göre problem öncelikle en küçük araç sayısı kadar kümelere ayrılmaktadır. Bu aşamadan sonra her küme içerisindeki talepler büyükten küçüğe sıralanmaktadır. Her küme için büyük talepliden başlamak üzere araç kapasitesi kontrol edilerek küme içerisinde kapasiteyi aşan talep noktası olup olmadığı kontrol edilmektedir. Kapasiteyi aşan talep noktaları olması durumunda, bu noktalar ilgili kümeden ayrılarak ayrı bir listeye alınmaktadır.

Ayrılan tüm talep noktaları bir araya getirilmekte ve büyükten küçüğe sıralanmaktadır. Daha sonra, talebi büyük olan müşteriden başlamak üzere kapasitesi uygun kümelere sırasıyla atamalar gerçekleştirilmektedir. Açıkta kalan müşteri olması durumunda çıkar-ekle yöntemiyle boşta kalan talep noktası kalmayacak şekilde kümelere atama yapılmaktadır. Kullanılan bu yöntemde izlenilen adımlar Şekil 4.1'de yer alan algoritma adımları gibidir.

Adım 0: Problemi oku
Adım 1: Problemin koordinatları aracılığıyla k-ortalamalar tekniği kullanılarak düğümlere kümeleme uygula.
Adım 2: Her küme içerisinde talepleri büyükten küçüğe sırala, her küme için kapasiteyi aşan talepleri ayır.
Adım 3: Ayrılan tüm talepleri büyükten küçüğe sırala, büyükten başlamak üzere kapasitesi uygun kümelere sırasıyla ata.
Adım 4: Açıkta kalan varsa rotalardan elemanlar sökülüp tekrar eklenerek boşta kalan talep noktası kalmayacak şekilde kümelere atama yap.
Adım 5: Her küme bir rota olacak şekilde klasik ekleme yöntemi ile rotaları kur.

Şekil 4.1 *Başlangıç çözüm oluşturma algoritması*

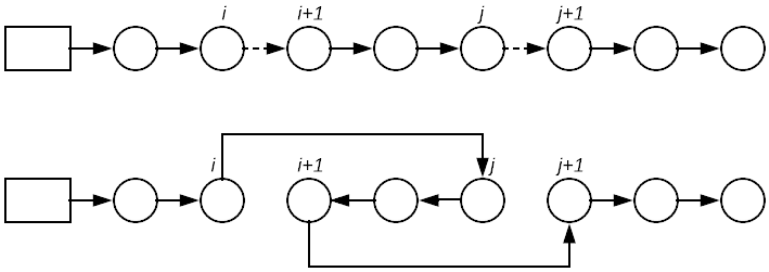
Böylelikle değişken önerilen DKA'nın kullanacağı başlangıç çözüm elde edilmektedir. Bu uygulama sayesinde DKA'nın arama süresince çözümün araç sayısı ve kapasite bakımından uygunluğunu asla bozmayacak bir yapı elde edilmektedir.

4.2. Rota İçi Arama için Kullanılan Komşuluklar

Önerilen yöntemde rota içi arama algoritmaları olarak literatürde sıklıkla gücü vurgulanan *2-opt*, *3-opt* ve *Or-Opt*'un yanı sıra *Dinamik Arama* tekniği kullanılmaktadır.

4.2.1. 2-opt komşuluğu

Rota içerisinde 2-opt komşuluğu her bir rotada $j \geq i + 2$ olmak üzere $(i, i + 1)$ ve $(j, j + 1)$ bağlantılarının kırılıp (i, j) ve $(i + 1, j + 1)$ bağlantılarının oluşturulması test edilmektedir. Değişiklik öncesi maliyeti d_1 olası değişiklik durumunu d_2 göstermektedir. Değişikliğin maliyeti iyileştirebilecek olması durumunda diğer bir deyişle $d_2 - d_1 < 0$ olması durumunda rotada $(i + 1, j + 1)$ bağlantısı ters çevrilmektedir. Şekil 4.2'de 2-opt komşuluğu için bir örnek ve iyileşme durana dek çalışacak şekilde planlanmış 2-opt komşuluğunun sözde Python kodu görülmektedir.

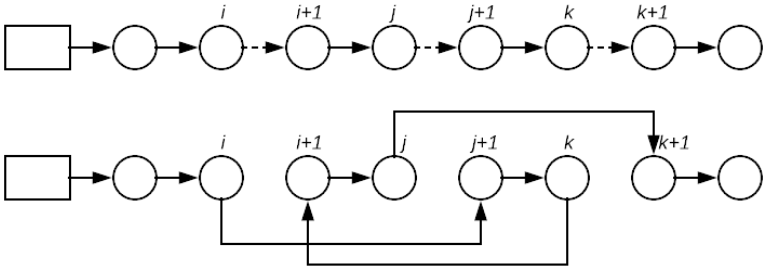



```
for Tüm_Rotalar
    for i in (0,Rota_Uzunluğu-2)
        for j in (i+2, Rota_Uzunluğu)
            if delta<0
                 rota[i + 1:j + 1] = rota[i + 1:j + 1][::-1]
                 rota.z[it] += d2 - d1
                 i=i-2
                 break j
```

Şekil 4.2 *2-opt komşuluğu örneği ve Python sözde kodu*

4.2.2. Or-Opt komşuluğu

Rota içerisinde Or-opt komşuluğu her bir rotada $j \geq i + 2$ ve $k \geq j + 2$ olmak üzere $(i, i + 1)$, $(j, j + 1)$ ve $(k, k + 1)$ bağlantılarının kırılıp $(i, j + 1)$ ve $(k, i + 1)$, $(j, k + 1)$ bağlantılarının oluşturulması test edilmektedir. Değişiklik öncesi maliyeti d_1 olası değişiklik durumunu d_2 göstermektedir. Değişikliğin maliyeti iyileştirebilecek olması durumunda yani $d_2 - d_1 < 0$ olması durumunda rotada yukarıda bahsedilen değişiklikler gerçekleştirilmektedir. Şekil 4.3'te Or-opt komşuluğu için bir örnek ve iyileşme durana dek çalışacak şekilde planlanmış Or-opt komşuluğunun Python sözde kodu görülmektedir.




```
for Tüm_Rotalar
    for i in (0,Rota_Uzunluğu-4)
        for j in (i+2, Rota_Uzunluğu-2)
           for k in (j+2, Rota_Uzunluğu)
              if delta<0
                   a=rota[:i+1]
                   b=rota[i+1:j+1]
                   c=rota[j+1:k+1]
                   d=rota[k+1:]
                   rota= a+b+c+d
                   rota.z[it] += delta
                   i=i-1
                   break j,k
```

Şekil 4.3 *Or-opt komşuluğu örneği ve Python sözde kodu*

4.2.3. 3-opt komşuluğu

Rota içerisinde 3-opt komşuluğu her bir rotada $j \geq i+2$ ve $k \geq j+2$ olmak üzere ortaya çıkan var olan durum ile birlikte 8 farklı durumda değişiklik test edilmektedir.$v_a = rota[i]$, $v_b = rota[i+1]$, $v_c = rota[j]$, $v_d = rota[j+1]$, $v_e = rota[k]$, $v_f = rota[k+1]$, değişime aday olan ayrıtları göstermektedir. Bu ayrıtlar üzerinden 8 farklı değişiklik maliyeti izleyen denklemlerdeki gibi hesaplanmaktadır. Burada *um* uzaklık matrisini ifade etmektedir.

$$m_0 = um[v_a, v_b] + um[v_c, v_d] + um[v_e, v_f]$$
$$m_1 = um[v_a, v_c] + um[v_b, v_d] + um[v_e, v_f]$$
$$m_2 = um[v_a, v_b] + um[v_c, v_e] + um[v_d, v_f]$$
$$m_3 = um[v_a, v_b] + um[v_c, v_e] + um[v_d, v_f]$$
$$m_4 = um[v_a, v_d] + um[v_e, v_b] + um[v_c, v_f]$$
$$m_5 = um[v_a, v_e] + um[v_d, v_b] + um[v_c, v_f]$$
$$m_6 = um[v_a, v_d] + um[v_e, v_c] + um[v_b, v_f]$$
$$m_7 = um[v_a, v_e] + um[v_d, v_c] + um[v_b, v_f]$$

Ele alınan değişimlerden 0. durum, var olan durumu ifade etmektedir. 1., 2. ve 7. durumlar ise 2-opt değişiklikleridir. Bu maliyetlerin tamamı hesaplanmakta ve en küçüğü bulunmaktadır. Eğer en küçük maliyet m_0 ise herhangi bir değişiklik yapılmamaktadır.

Aksi durumda Şekil 4.4'teki gibi değişiklikler yapılmaktadır. Maliyetteki iyileşme ise $m_{enkucuk} - m_0$ kadar ortaya çıkmaktadır.

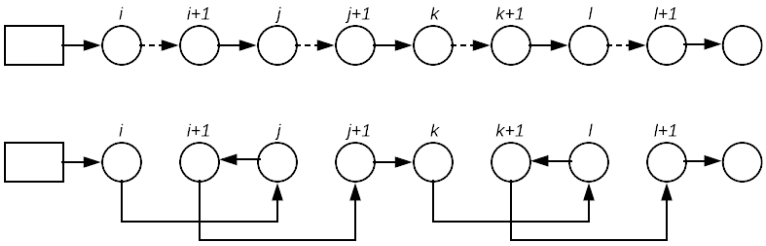
$a = rota[:i]$
$b = rota[i+1:j]$
$c = rota[j+1:k]$
$d = rota[k+1:]$

En küçük maliyet	Yeni Çözüm
m_1	$a + b^{-1} + c + d$
m_2	$a + b + c^{-1} + d$
m_3	$a + b^{-1} + c^{-1} + d$
m_4	$a + c + b + d$
m_5	$a + c^{-1} + b + d$
m_6	$a + c + b^{-1} + d$
m_7	$a + c^{-1} + b^{-1} + d$

Şekil 4.4 *3-opt değişim seçenekleri*

4.2.4. Dinamik arama komşuluğu

Dinamik arama komşuluğu (Dyna-Search) komşuluğu ilk olarak Congram tarafından önerilmiştir [62]. Söz konusu komşulukta her bir rotada $j \geq i+2$, $k \geq j+2$ ve $l \geq k+2$ olmak üzere $(i, i+1)$, $(j, j+1)$, $(k, k+1)$ ve $(l, l+1)$ bağlantılarının kırılarak (i, j) ve $(j+1, i+1)$, (k, l) ve $(l+1, k+1)$ bağlantılarının oluşturulması test edilmektedir. Değişiklik öncesi maliyeti d_1 olası değişiklik durumunu d_2 göstermektedir. Değişikliğin maliyeti iyileştirebilecek olması durumunda yani $d_2 - d_1 < 0$ olması durumunda rotada yukarıda bahsedilen değişiklikler gerçekleştirilmektedir.




```
for Tüm_Rotalar
    for i in (0,Rota_Uzunluğu-6)
        for j in (i+2, Rota_Uzunluğu-4)
           for k in (j+2, Rota_Uzunluğu-2)
              for l in (j+2, Rota_Uzunluğu)
                   if delta<0
                       a = rota[:i + 1]
                       b = rota[i + 1:j + 1]
                       c = rota[j + 1:k + 1]
                       d = rota[k + 1:l+1]
                       e = rota[l + 1:]
                       rota= a+b^-1+c+d^-1+e
                       rota.z[it] += delta
                       i=i-1
                       break j,k,l
```

Şekil 4.5 *Dinamik arama komşuluğu örneği ve Python sözde kodu*

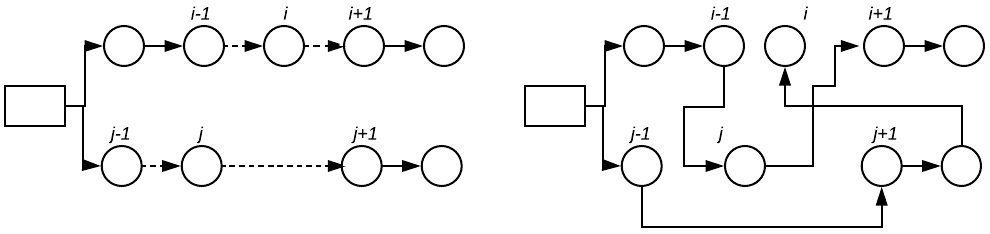
Şekil 4.5'te dinamik arama komşuluğu için bir örnek ve iyileşme sonlanana dek çalışacak şekilde planlanmış dinamik komşuluğunun Python sözde kodu görülmektedir.

4.3. Rotalar Arası Arama için Kullanılan Komşuluklar

Önerilen yöntemde kullanılan rotalar arası komşuluk yapıları izlenilen alt bölümlerde anlatılmaktadır. Rotalar arası arama operatörlerinde tüm ikili rotalar karşılıklı olarak değişiklikler ya da yalnızca bir rotadan diğerine olan aktarmalar ele alınmaktadır.

4.3.1. *Bire-bir* karşılıklı değişim arama operatörü

Bire-bir karşılıklı değişim (Exchange) arama operatöründe karşılıklı iki rotadan ilk rotada *i.* ve diğer rotada *j.* müşterinin karşılıklı aktarılmasının kapasite bakımından uygun olması durumunda ilkinde $(i-1,i)$, $(i,i+1)$ ve ikinci rotada $(j-1,j)$, $(j,j+1)$ bağlantıları kırılmaktadır. Daha sonra ilk rotada *i.* müşterinin ikinci rotada eklenebileceği en iyi yer ve ikinci rotada *j.* müşterinin ilk rotada eklenebileceği en iyi yerin maliyetleri mevcut durumla karşılaştırılmakta ve maliyet iyileştirilmesinin sağlanması durumunda ilk rotadan *i.* eleman ikinci rotada en iyi yere, ikinci rotadan *j.* eleman ise ilk rotada en iyi yere eklenmektedir. Şekil 4.6'da söz konusu komşuluk için bir örnek ve Şekil 4.9'da sözde kodu bulunmaktadır.



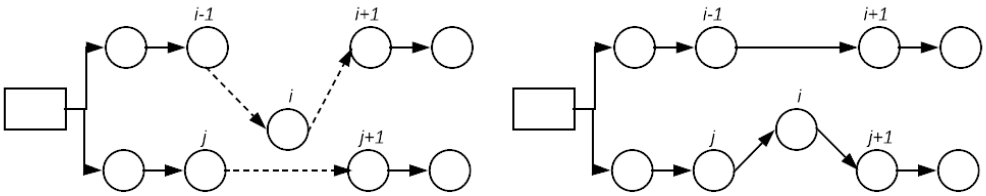

```
for s in (0,Rota_Sayisi-1)
    for t in (s+1,Rota_Sayisi)
        for i in (1, Rota_Uzunluğu)
           for j in (1, Rota_Uzunluğu)
               if (s'den i, t'den j) değişikliği uygunsa
                  if delta<0
                     s'den i'yi t'den en iyi yere
                     t'den j'yi s'den en iyi yere
                    break i,j
```

Şekil 4.6 *Bire-bir karşılıklı değişim komşuluğu örneği ve Python sözde kodu*

4.3.2. Yeniden konumlandırma (Relocate) arama operatörü

Yeniden konumlandırma arama operatöründe bir rotadan bir elemanın sökülüp bir başka rotaya yerleştirilmesi söz konusudur. Karşılıklı iki rotadan ilkinde $(i-1,i)$, $(i,i+1)$ ve ikinci rotada $(j,j+1)$ bağlantılarının kırılıp $(i-1,i+1)$, (j,i) ve

$(i, j+1)$ bağlantılarının değişikliği test edilmektedir. İlk rotada *i.* müşterinin, ikinci rotaya geçişi talep olarak uygunsa ve söz konusu bağların kırılıp yeniden oluşturulması ile maliyet iyileştirilmesi sağlanıyorsa bu değişiklik gerçekleştirilmektedir. Şekil 4.7'de söz konusu komşuluk için bir örnek ve Python sözde kodu bulunmaktadır.

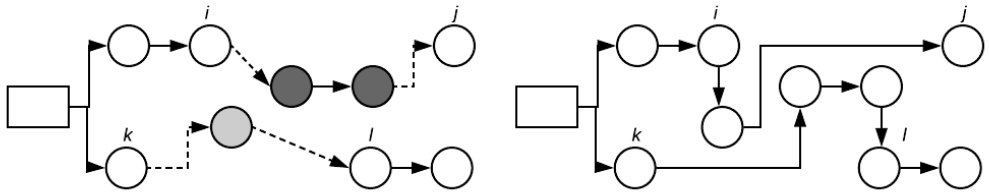



```
for i in (0,Rota_Sayisi)
    for j in (0,Rota_Sayisi)
        if i!=j
           for s in (1, Rota_Uzunluğu[i])
              for t in (0, Rota_Uzunluğu[j])
                 if (i'den s, j'ye aktarılması) uygunsa
                   if delta<0
                      değişiklikleri uygula
                       break s,t
```

Şekil 4.7 *Yeniden konumlandırma komşuluğu örneği ve Python sözde kodu*

4.3.3. Çoklu-karşılıklı değişim (Cross-Exchange) komşuluğu

Çoklu-karşılıklı değişim (Cross Exchange) komşuluğu ilk olarak Taillard ve ark. [63] tarafından önerilmiştir. Karşılıklı iki rotadan çeşitli büyüklükte artarda olan müşteri kümelerinin maliyet iyileştirmesi durumunda taşınması durumudur. Bu komşulukta aramanın çok etkili olması beklenmekle beraber; komşuluğun karmaşıklığının da fazla olması durumunun göz ardı edilmemesi gerekmektedir. Bu sebeple komşuluğa bazı sınırlandırmalar getirilmesi etkili olabilmektedir. Tez çalışmasında bir rotadan en az 1, en fazla ise 5 ardışık müşterinin karşılıklı değişimine izin verilmiştir. Öte yandan bu değişimlerden 1-1 değişimler ile ilgilenilmemiştir. Burada 1-1 müşteri değişiminin dahil edilmemesinin sebebi söz konusu değişimin 4.4.1'deki operatörde çok daha etkin bir şekilde sağlanması ve çoklu-karşılıklı değişim komşuluğunun kısmen karmaşıklığının azalmasıdır.



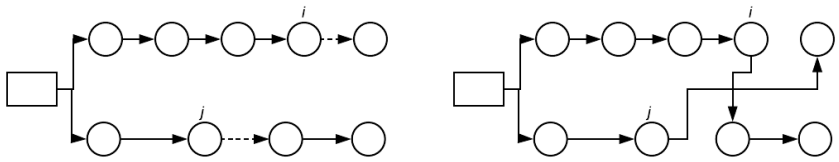

```
for s in (0,Rota_Sayisi)
    for t in (s+1,Rota_Sayisi)
         if s!=t
           for i in (1, Rota_Uz[s]-1)
               for k in (i+1, Rota_Uz[s])
                  for j in (0, Rota_Uz[t]-1)
                    for l in (0, Rota_Uz[t])
                      if (s'den i-k arasının t'ye aktarılması ve
t'den j-l arasının s'ye aktarılması) uygunsa
                         if delta<0
                            değişiklikleri uygula
                           break i,j,k,l
```

Şekil 4.8 *Çoklu-karşılıklı değişim komşuluğu örneği ve sözde kodu*

Tez çalışmasında çoklu-karşılıklı değişim komşuluğu ele alınırken maliyet iyileştirmesi kontrolü iki yönde gerçekleştirilmektedir. Diğer bir deyişle ilk rotadan değişime uygun (*i-k)* arasındaki müşterilerin ikinci rotaya eklenirken hem aynı yönde eklenmesinin maliyeti hem de ters yönde (*k-i)* eklenmesinin maliyeti kontrol edilmektedir. Bu durum ikinci rotadan birinci rotaya olan aktarım için de aynı şekilde gerçekleştirilmekte ve maliyeti iyileştiren yönlerden daha iyisi tercih edilip bağlantılar yapılmaktadır. Şekil 4.8'de bu komşuluğun gösterildiği bir örnek ve bu komşuluğun Python sözde kodları yer almaktadır.

4.3.4. *2opt** komşuluğu

2opt* komşuluğunda AARP'de karşılıklı iki rotanın belirli noktalarından başlamak üzere kapasite geçişinin uygun ve maliyetin azalması şartıyla kuyruklarının yer değiştirmesi şeklinde uygulanmaktadır. Şekil 4.9'da 2opt* komşuluğunu gösteren bir örnek ve komşuluğun Python sözde kodları bulunmaktadır.




```
for s in (0,Rota_Sayisi-1)
    for t in (s+1,Rota_Sayisi)
        for i in (1, Rota_Uzunluğu)
           for j in (1, Rota_Uzunluğu)
               if (s'den i'den sonrakilerin, t'den j'den
sonrakilerin) değişikliği uygunsa
                   if delta<0
                      s'den i'den sonrakileri t'den j'den sonraya bağla
                      t'den j'den sonrakileri s'den i'den sonraya bağla
                      break i,j
```

Şekil 4.9 *2opt* değişim komşuluğu örneği ve sözde kodu*

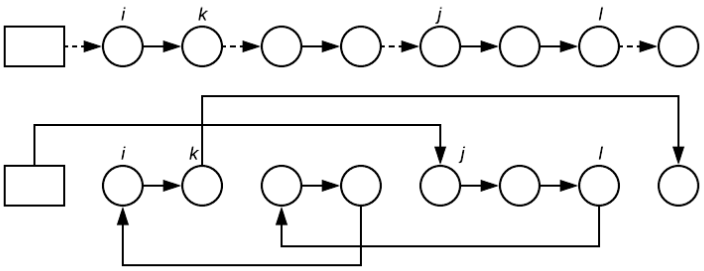
4.4. Sarsma Mekanizmaları

DKA yönteminin temelinde, arama yapılan komşulukların sistematik değiştirilmesiyle beraber çözümün arama uzayında başka bir noktaya taşınması bulunmaktadır. Algoritmada bu taşıma işlemi sarsma mekanizmalarıyla gerçekleştirilmektedir. Sarsma mekanizmalarının temel mantığı, arama yapılırken algoritmayı yerel en iyi noktalardan kurtarmaktır. Algoritmanın başarısı için son derece önemli olan bu mekanizmalar dikkatli tercih edilip kullanılmalıdır. Sarsma mekanizmalarının etkili bir biçimde planlanmaması durumunda çözümün çok az ya da çok fazla bozulmasına sebebiyet verebilmektedir. Çözümün az bozulması yerel en iyi noktalardan kurtulmayı zorlaştırırken; çok bozulması ise algoritmanın yapısından çıkarak rasgele arama şekline bürünmesine sebep olmaktadır.

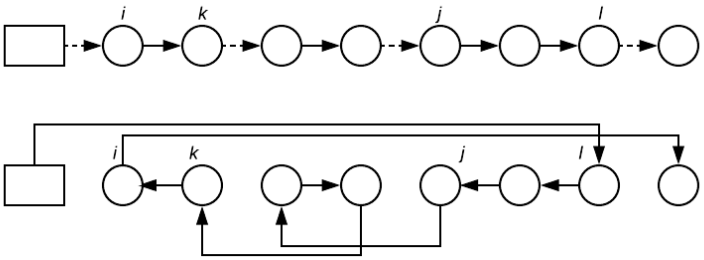
Tez çalışmasında hem rota içi arama DKA'da ve rotalar arası arama DKA'da hem de çatı algoritmada çeşitli sarsma yöntemleri kullanılmıştır. İzleyen alt başlıklarda bu operatörler ve ele aldıkları komşuluklar açıklanmaktadır.

4.4.1. Rota içi k-değişim komşuluğu

Rota içi değişim için kullanılan bu komşuluk da her bir rota için rasgele iki segment belirlenmekte ve değişim gerçekleştirilmektedir. Bu operatör rota içi arama kısmında kullanılmakta ve bağlanma yönüne göre iki farklı sarsma operatörü olarak görev almaktadır. Şekil 4.10'da söz konusu sarsma komşuluğuna dair bir örnek görülmektedir. Bu örnekte segmentler aynı yönde yer değiştirerek bağlanmaktadır. Şekil 4.11'de bu sarsma komşuluğuna dair bir başka örnek görülmektedir. Bu örnekte segmentler ters yönde yer değiştirerek bağlanmaktadır.



Şekil 4.10 *Rota içi k-değişim komşuluğu düz yönlü örneği*



Şekil 4.11 *Rota içi k-değişim komşuluğu ters yönlü örneği*

4.4.2. Rota içi çift-köprü (double-bridge) komşuluğu

Rota içi değişim için kullanılan bu komşuluk da her bir rota için rasgele üç nokta seçilerek bu noktaların denk geldiği yerlere göre rota dört bölmeye ayrılmakta ve bölmeler tekrardan 1-4-3-2 sırasıyla bağlanmaktadır. Bu komşuluk rota içi k-değişim komşuluğunun özel bir hali olarak ortaya çıkmaktadır.

4.4.3. Rotalar arası k-değişim komşuluğu

Rotalar arası arama kısmında kullanılan bu komşulukta, karşılıklı olarak ele alınan iki rotadan rasgele iki segmentin maliyetine bakılmaksızın değiştirilmesi gerçekleştirilmektedir. Burada karşılıklı iki segmentin talep geçişlerinin kapasite bakımından uygun olmaması durumunda en fazla on kez farklı rasgele segmentlerle değişiklik aranmaktadır.

4.4.4. Çıkar-ekle stratejileri

Algoritmanın ana sarsma kısmında çıkar-ekle stratejileri kullanılmaktadır. Buna göre, çözüm içerisinden en maliyetli ya da rasgele talep noktalarının çıkarılıp eklenmesi sağlanmaktadır. Burada kullanılan çıkarma ve ekleme stratejiler şöyledir:

- En maliyetlileri çıkarıp, en iyi yere ekleme
- Rasgele çıkarıp, en iyi yere ekleme
- En maliyetlileri çıkarıp, en iyi ikinci yere ekleme
- Rasgele çıkarıp, en iyi ikinci yere ekleme

- En maliyetlileri çıkarıp, en iyi üçüncü yere ekleme
- Rasgele çıkarıp, en iyi üçüncü yere ekleme
- En maliyetlileri çıkarıp, en iyi dördüncü yere ekleme
- Rasgele çıkarıp, en iyi dördüncü yere ekleme

Kullanılan stratejilerin etkinliği Ropke'nin [64] doktora tez çalışmasında gösterilmiştir. Operatörde işlemler sırasında sekiz stratejiden biri rulet tekeri tekniğine göre seçilmektedir. Tekniklerin seçilme olasılıklar ön çalışmalar sonucunda çözümü daha fazla bozan stratejilere daha az şans verecek şekilde planlanmıştır. Buna göre tekniklerin seçilme olasılığı sırasıyla 0.2, 0.2, 0.15, 0.15, 0.10, 0.10, 0.05 ve 0.05'tir. Bununla birlikte algoritmanın başında belirtilen (*a-b)* gibi bir aralıktan, çözümü çoğunlukla daha az bozmak için üçgen dağılıma uygun şekilde *(a, a, b)* parametreleriyle üretilen sürekli rassal değişkenin tam sayı kısmını alarak *t* kadar talep noktası çözümden çıkarılmaktadır.

Sürekli dağılımlardan olan üçgen dağılımda *(x, y, z)* parametreleri bulunmaktadır. Dağılımın ortalaması ve standart sapması sırasıyla Denklem 4.1 ve Denklem 4.2'deki gibi hesaplanmaktadır. Parametrelerden *x* ve *z* üçgenin her iki ucunu gösterirken *y* parametresi mod değerini göstermektedir. Çalışmada *x=a, z=b* değeri olarak belirlenirken *y=a* olarak belirlenmiştir. Burada amaç daha çok *a* değerine yakın rassal değişkenlerin elde edilmesinin istenmesidir. Örnek olarak (*a, b)=(7, 16*) olduğu durumda üçgen dağılıma göre *(7, 7, 16)* parametreleriyle rassal değişkenler türetildiğinde $\mu=10$ ve $\sigma=2.11$ olmaktadır.

$$\mu = \frac{x+y+z}{3} \tag{4.1}$$

$$\sigma = \sqrt{\frac{x^2+y^2+z^2-xy-xz-yz}{18}} \tag{4.2}$$

Çözümden çıkarılma işlemi gerçekleştirildikten sonra talepleri büyükten küçüğe sıralanmakta ve talebi en büyükten başlamak üzere çözüm içerisinde seçilen stratejiye uygun olarak atanmaktadır. Talebi büyük olandan başlanılmasının sebebi çözümün uygunluğunun olabildiğince korunmasıdır. Burada her ne kadar büyük talepliler öncelikli olarak yerleştirilse de yerinden ayrılan müşterilerin farklı rotalara eklenebilmesi durumu çözümün uygunluktan çıkmasına sebebiyet verebilmektedir. Bu durumda çözüm geldiği şekliyle herhangi bir değişiklik yapılmadan geri döndürülmektedir.

Burada bir talep noktası eklenmek üzere ele alındığında öncelikle o noktanın hangi maliyetlerle çözümün uygunluğunu bozmadan nerelere ekleneceği listelenmektedir. Seçilen stratejiye uygun olarak listeden ekleneceği yer belirlenmektedir. Eğer liste uzunluğu seçilen stratejideki ekleme yerini karşılamıyorsa listedeki son nokta ekleme yeri olarak tercih edilmektedir. Örneğin, listenin uzunluğu 3 olduğu durumda sekizinci strateji ile rasgele çıkarılan bir talep noktası en iyi dördüncü yere eklenmek istediğinde bu durum o talep noktasının eklenebilecek üç yerinin bulunmasından dolayı gerçekleştirilememekte ve en iyi üçüncü yere ekleme sağlanmaktadır.

4.5. Çatı Algoritma

Başlangıç çözümün oluşturulmasının ardından algoritma üç ana öğeden oluşmaktadır. Bu öğeler *rota içi arama DKA, rotalararası arama DKA* ve *sarsma mekanizması* kısmıdır. Algoritma belirli tekrarda çalıştırılmaktadır. Her tekrarın başında o iterasyona başlanılan çözüm *sTut'* a ve en iyi çözüm ise *sBestTut* değişkenlerinde saklanmaktadır. Daha sonra ilk olarak *rota içi arama DKA* çağrılmaktadır. Bu operatöre *sTut* ve bir arama deseni gönderilmektedir. Bu operatör içerisinde *sTut* ve *sBest* mümkün olduğu taktirde güncellenmektedir. Operatöre gönderilen arama deseni ise rasgele bir vektör oluşmaktadır. Örneğin arama deseni (3, 4, 1, 2) olduğunda komşulukların değişme sırası 3opt, Dinamik arama, 2-opt ve Or-opt şeklinde olmaktadır. Şekil 4.12'de çatı algoritmanın sözde kodları bulunmaktadır.

```
s=Başlangıç çözüm
sBest=s
while(iter<max_iter):
      sBestTut=sBest
      sTut=s
      iterTut=s

      s'=Rotaiçiarama(sTut, aramadeseni1)
      if f(s')<f(sBest):
            sBest=s'
      if f(s')<f(sTut):
            sTut=s'

      s''=Rotalararasiarama(sTut, aramadeseni2)
      if f(s'')<sBest:
            sBest=s''
      if f(s'')<sTut:
            sTut=s''

      Sarsma Mekanizması
      if f(sBest)==f(sBestTut):
            bestFoldIter+=1
            if bestFoldIter==bestFold:
                  s=sBest
                  bestFoldIter=0
            else:
                  s=Stut
            if f(s)== f(iterTut)or f(s)== f(sBest):
                  s=Ana Sarsma(s)
      else:
            s=sBest
            bestFoldIter=0
```

Şekil 4.12 *Çatı algoritma sözde kodu*

Rota içi arama operatöründen elde edilen ve mümkünse iyileştirilen *sTut* ve *sBest* güncellenerek *rotalar arası DKA*'ya gönderilmektedir. Benzer şekilde bu operatörde de bir rasgele arama deseni kullanılmaktadır. Örneğin arama deseni [2, 4, 3, 1] olduğunda komşulukların değişme sırası çoklu karşılıklı değişim, *Bire-bir değişim, Yeniden konumlandırma* ve *2opt** şeklinde olmaktadır. Bu operatörde çalışırken de mümkün olduğu durumda *sTut* ve *sBest* güncellenmektedir.

Bu aşamadan sonra *sBest*'in amaç fonksiyonu değerinin *sBestTut*'un değeri ile aynı olup olmadığı kontrol edilmektedir. Eğer *sBest* daha iyi bir noktada ise sarsma mekanizmaları devreye girmeden doğrudan *s=sBest* olacak şekilde sonra tekrara geçilmektedir. Öte yandan *f(sBest)=f(sBestTut)* olduğu durumda sarsma mekanizmaları önceden tanımlı bazı parametrelere bağlı olarak çalışmaktadır.

İzleyen alt başlıklarda algoritmanın tüm operatörlerinin detayları bulunmaktadır.

4.5.1. Rota içi arama DKA

Bu operatörde her rota için ayrı ayrı olmak üzere bir DKA çalıştırılmaktadır. Her rota için belirli sayıda iyileşme olmayana dek arama devam etmektedir. Her tekrara yerel minimum noktalardan kaçınmak için rota içi sarsma ile başlanmaktadır. Sarsma aşamasında var olan üç farklı rota içi sarsma yöntemi *sk* parametresine bağlı olarak seçilmektedir. Daha sonra *lk* değerine bağlı olarak arama deseni vektöründen komşuluk belirlenmekte ve arama iyileşme duruncaya dek o komşulukta gerçekleştirilmektedir. Her tekrarda eğer en iyi çözümde iyileşme olduysa *sBest* güncellenmektedir. Bununla beraber *sTut*'a göre iyileşme olması durumunda *sTut* güncellenmekte ve aramaya sonraki komşuluktan devam edilmektedir. Bir komşulukta arama esnasında *sTut*'un amaç fonksiyonu değerinde iyileşme olmaması durumunda da sonraki komşuluğa geçilmektedir ancak aramaya sarsma mekanizmasından *sk* için 1 ile 3 arasında rasgele bir teknik seçilmekte ve sarsma gerçekleştirildikten sonra devam edilmektedir.

Rota içi DKA'da kullanılan sarsma algoritmaları Bölüm 4.4.1 ve 4.4.2'de açıklandığı gibi *aynı yönde k-değişim, ters yönde k-değişim* ve *çift-köprü* algoritmalarıdır. Rota içi aramada herhangi bir sarsma algoritması çağrıldığında bir rota içerisinde yapılmaya çalışılan değişiklik beş kez tekrar edilmektedir.

```
s=sTut
for it in len(rotalar)
      sk=0, lk=0, iyilesmememesay=0
      secim=aramadeseni1[lk]
      while(iyilesmememesay<max_rota_içi_iyilesmeme):
            if secim==1
                  s''=opt2(s',it)
            if secim==2
                  s''=orOpt(s',it)
            if secim==3
                  s''=opt3(s',it)
            if secim==4
                  s''=dinamikArama(s',it)
            if f(s'')<f(sBest):
                  sBest=s''
            if f(s'')<f(sTut):
                  sTut=s''
            else
                  iyilesmememesay+=1
                  sk= randomint(1,3)
                  s=sTut
                  s=rotaicisarsma(s,sk,it,3)
            lk=lk%4+1
            secim=aramadeseni1[lk-1]
```

Şekil 4.13 *Rota içi arama sözde kodları*

4.5.2. Rotalar arası arama DKA

Bu operatörde tüm rotalar ikişer ikişer birbirleriyle ele alınmak üzere DKA çalıştırılmaktadır. Her rota ikilisi için belirli sayıda iyileşme olmayana dek arama devam etmektedir. Bir *lk* değerine bağlı olarak arama deseni vektöründen komşuluk belirlenmekte ve arama iyileşme duruncaya dek o komşulukta gerçekleştirilmektedir. Her tekrarda, eğer en iyi çözümde iyileşme olduysa *sBest* güncellenmektedir. Bununla beraber *sTut*'a göre iyileşme olması durumunda *sTut* güncellenmektedir. Herhangi bir komşulukta aramanın sonlanmasından sonra iyileşme olmasına bakılmaksızın komşuluk güncellenmekte ve *lk = lk mod 4 + 1, secim=aramadeseni2[lk-1]* ile sonraki komşuluğa geçilmektedir. Ayrıca çözümün iyileşmediği durumda sarsma mekanizması *sTut* çözüme aktarılarak çalıştırılmaktadır.

```
s=sTut
for it in len(rotalar)-1
      for jt in (it+1,len(rotalar))
            iyilesmememesay=0
            sk=0, lk=0
                secim=aramadeseni2[lk]
            while(iyilesmememesay<max_iyilesmeme):
                  if secim==1
                        s''=opt2star(s',it,jt)
                  if secim==2
                        s''=coklukarsiliklidegisim(s',it,jt)
                  if secim==3
                        s''=yenidenkonumlandirma1_0(s',it,jt)
                  if secim==4
                        s''=degisim1_1(s',it,jt)
                  if f(s'')<f(sBest):
                        sBest=s''
                  if f(s'')<f(sTut):
                        sTut=s''
                  else
                        iyilesmememesay+=1
                        s=sTut
                        s=rotalararasi_sarsma(s,it,jt,3)
                  lk=lk%4+1
                  secim=aramadeseni2[lk-1]
```

Şekil 4.14 *Rotalar arası arama sözde kodları*

Rotalar arası DKA'da sarsma gerçekleştirilirken Bölüm 4.4.3'de açıklanan rotalar arası k-değişim tekniği kullanılmaktadır. Her ikili rota için rasgele oluşturulan farklı segmentlerde üç kez kapasite bakımından uygun olan değişiklikler uygulanmaktadır.

4.5.3. Ana sarsma mekanizması

DKA'da farklı komşuluklar her ne kadar akıllıca ve sistematik olarak değiştirilse de algoritma yerel en iyi noktalarda takılabilmektedir. Bu durumlardan kurtulmak için çözümün belirli ölçülerde farklı noktalara taşınması gerekmektedir. Burada dikkatli

olunması gereken nokta çözümün fazlaca bozularak aramanın tamamen rassal aramaya döndürülmesinin önüne geçilmesidir. Önerilen algoritmada bir tekrarda çözümün iyileştirilememesi durumunda bu operatör devreye girmektedir. Öncelikli olarak en iyi çözümün kaç tekrarda bir çözüm olarak algoritmaya verileceği *bestFoldIter==bestFold* ile kontrol edilmektedir. Bu durumun sağlanması halinde *s=sBest* olarak devam edilmektedir.

```
if f(sBest)==f(sBestTut):
            bestFoldIter+=1
            if bestFoldIter==bestFold:
                  s=sBest
                  bestFoldIter=0
            else:
                  s=Stut
            if f(s)== f(iterTut)or f(s)== f(sBest):
                  s'=Sarsma(s)
else:
            s=sBest
            bestFoldIter=0
```

Şekil 4.15 *Rota içi arama sözde kodları*

Eğer çözümün iyileşmeme sayısı, kaç tekrarda iyileşme olmazsa en iyi çözümün algoritmaya aktarılacağı (*bestFoldIter==bestFold)* koşulu sağlanmıyorsa algoritmaya *s=sTut* olarak devam edilmektedir. Bu aşamadan sonra çözüm iterasyon başındakine göre iyileşmemişse (*f(s)== f(iterTut))* ya da çözüm en iyi çözümle aynıysa (*f(s)==f(sBest))* sarsma mekanizması çalışmaya başlamaktadır. Sarsma mekanizmasında Bölüm 4.4.4'te anlatıldığı gibi rulet tekeri tekniğine göre bir strateji seçilmekte ve işlemler gerçekleştirilmektedir.

Sarsma mekanizması çalışırken (*a, b*) parametreleri bulunmaktadır. Burada *a* çözümün en az ne kadar bozulacağını *b* çözümün en fazla ne kadar bozulacağını göstermektedir. (*a, b*) aralığından bir rassal sayı seçilirken b'ye yakın olanların çözümün daha az sıklıkla fazlaca bozulmasının istenmesinden dolayı (*a,a,b*) parametreleriyle *üçgen dağılıma* göre rassal üretilen belirli sayıda talep noktasının çözümden seçilen stratejiye göre koparılmasına karar verilmektedir.

Sarsma mekanizmasında rulet tekeri stratejisiyle seçilen teknik ve üçgen dağılımdan rassal üretilen *t* değerine göre bir deneme yapılmakta ve dönen çözümün uygun olması durumundan ikinci bir denemeyi yapmaksızın algoritma kaldığı yerden çalışmaya devam etmektedir. Öte yandan ilgili teknik ve *t* değeriyle uygun çözüm dönmemesi durumunda yeni bir *t=int(triangular(a,a,b))* belirlenmekte ve aynı teknik ile sarsma denenmektedir. İkinci seferde uygun çözüm elde edilemediği durumda algoritma

kaldığı yerden çalışmaya çözümde herhangi bir değişiklik yapmaksızın devam etmektedir.

4.6. Parametrelerin Belirlenmesi

Algoritmada parametrelerin belirlenmesi amacıyla bir dizi ön çalışma gerçekleştirilmiştir. Çalışmada tercih edilen AARP literatüründe sıklıkla kullanılan test problemleri Tablo 4.1'de yer almaktadır. Bu problemlerden bazıları ön çalışmalarda kullanılmıştır. Tablolarda yer alan *n* problem büyüklüğünü, *Q* araç kapasitesini, *Min araç* en az araç sayısını göstermektedir.

Tablo 4.1 *Literatür test problemleri*

C serisi	n	Q	Min araç	F serisi	n	Q	Min araç	O serisi	n	Q	Min araç
C1	50	160	5	**F11**	71	30000	4	**O1**	200	900	5
C2	75	140	10	**F12**	134	2210	7	**O2**	240	550	9
C3	100	200	8					**O3**	280	900	7
C4	150	200	12					**O4**	320	700	10
C5	199	200	16					**O5**	360	900	8
C11	120	200	7					**O6**	400	900	9
C12	100	200	10					**O7**	440	900	10
								O8	480	1000	10

Öncelikle etkisi incelenen en iyi çözümün sunulması amacıyla kullanılan *BestTut* parametresidir. Bu parametre ile en iyi çözümün kaç tekrarda bir çözüme entegre edileceği kararlaştırılmaktadır. Parametrenin belirlenmesi için 10, 25, 50, 75 ve en iyi çözümün hiç entegre edilmemesinin çözüm üzerinde yarattığı etki *C1*, *C2* ve *C3* problemlerinde incelenmiş ve sonuçlar tablolarda raporlanmıştır. Bu incelemeler yapılırken sarsma mekanizmasındaki (*a,b*) değerleri *a=n/20*, *b=n/10* olarak sabit tutulmuştur. Buna göre 25 tekrarda bir en iyi çözümün algoritmaya entegre edilmesine karar verilmiştir.

Tablo 4.2 *C1 için en iyinin çözüme eklenmesi parametresine ait testler*

BestTut	Ortalama	St. Sapma	En iyi Değer	Ortalama Süre(sn)
10	422.67	4.89	416.06	190.1
25	**418.37**	**3.41**	**416.06**	**147.1**
50	419.08	3.43	416.06	195.5
75	421.21	4.72	416.06	227.5
∞	419.93	3.88	416.06	233.2

Tablo 4.3 *C2 için en iyinin çözüme eklenmesi parametresine ait testler*

BestTut	**Ortalama**	**St. Sapma**	**En iyi Değer**	**Ortalama Süre(sn)**
10	576.02	3.87	567.14	319.9
25	**575.10**	4.67	**567.14**	**286.1**
50	577.91	**3.28**	567.14	316.5
75	577.79	4.76	569.79	409.0
∞	576.43	4.59	571.06	439.9

Tablo 4.4 *C3 için en iyinin çözüme eklenmesi parametresine ait testler*

BestTut	**Ortalama**	**St. Sapma**	**En iyi Değer**	**Ortalama Süre(sn)**
10	649.06	3.23	643.48	**430.5**
25	**647.08**	2.78	**641.80**	490.0
50	648.42	3.50	642.34	556.0
75	648.29	3.63	642.50	620.1
∞	648.88	**2.38**	646.24	671.9

Daha önce de bahsedildiği gibi sarsma mekanizması çalışırken (*a, b*) parametreleri bulunmaktadır. Burada *a* çözümün en az ne kadar bozulacağını *b* çözümün en fazla ne kadar bozulacağını göstermektedir. Algoritma boyunca ön çalışmalar neticesinde çözümden en az *n/20* (%5) elemanın bozulmasıyla birlikte en fazla ne kadar elemanın bozulmasının *BestTut =25* parametresiyle birlikte *n/10*, *n/7.5* ve *n/5* seviyelerinde çözüme etkisi incelenmiştir.

Tablo 4.5 *C1 için b parametresine ait testler*

BestTut =25 (a,b)=(n/20,x)	**Ortalama**	**St. Sapma**	**En iyi Değer**	**Ortalama Süre(sn)**
n/10	418.37	3.41	416.06	147.1
n/7.5	**417.98**	3.12	**416.06**	**125.5**
n/5	418.49	**3.02**	416.06	153.3

Tablo 4.6 *C2 için b parametresine ait testler*

BestTut =25 (a,b)=(n/20,x)	**Ortalama**	**St. Sapma**	**En iyi Değer**	**Ortalama Süre(sn)**
n/10	575.10	4.67	567.14	286.1
n/7.5	**574.74**	**3.31**	**567.14**	**259.6**
n/5	578.66	4.96	567.14	293.5

Tablo 4.7 *C3 için b parametresine ait testler*

BestTut =25 ***(a,b)=(n/20,x)***	**Ortalama**	**St. Sapma**	**En iyi Değer**	**Ortalama Süre(sn)**
n/10	647.08	**2.78**	641.80	**490.0**
n/7.5	**646.48**	3.35	**640.75**	501.5
n/5	646.88	4.82	642.45	541.3

Üç problem üzerinde yapılan testler sonucunda n/7.5 seviyesinin diğer seviyelere göre daha iyi sonuç verdiği ortaya konmuştur. Bu aşamaya kadar olan testlerde rota içi en fazla iyileşmeme sayısı 24, rotalar arası en fazla iyileşmeme sayısı 12 olarak kullanılmıştır. Bu değerlerin özellikle çözüm süresine anlamlı yansımalarının olması amacıyla *24-12* değerlerinin yanı sıra *16-8* ve *12-6* seviyelerinde bir problem üzerinden testler gerçekleştirilmiştir.

Tablo 4.8 *C2 için aramalardaki en fazla iyileşmeme durumuna ait testler*

BestTut=25 ***(a,b)=(n/20,*** **n/7.5*)** ***Rota içi –Rotalar arası***	**Ortalama**	**St. Sapma**	**En iyi Değer**	**Ortalama Süre(sn)**
24-12	574.74	**3.31**	567.14	259.6
16-8	**574.87**	3.45	**567.14**	203.8
12-6	576.41	4.92	567.73	**170.2**

Test edilen seviyelerden 24-12 ve 16-8'den elde edilen sonuçların ortalaması birbirlerine oldukça yakın çıkmaktadır. Öte yandan 16-8 seviyesinde süre anlamlı bir şekilde azalmıştır. Bu yüzden algoritmada rota içi en fazla iyileşmeme sayısı her bir rota için 16, rotalar arası en fazla iyileşmeme sayısı ise her ikili farklı rota için 8 olarak belirlenmiştir.

Tablo 4.9 *Parametreler*

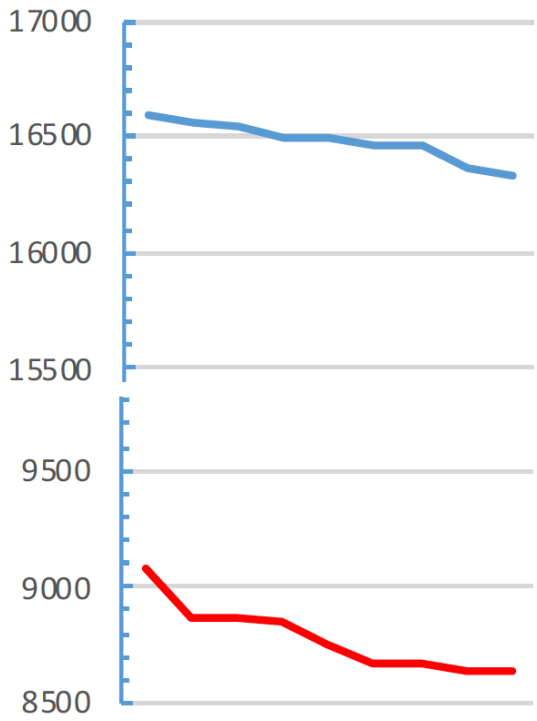
Tekrar sayısı	250
En iyi çözümün kaç tekrarda bir algoritmaya sunulacağı (BestTut)	25
Rota içi en fazla iyileşmeme	16
Rotalararası en fazla iyileşmeme	8
a (minimum bozulma)	n/20
b (maksimum bozulma)	n/7.5

4.7. Detaylı Operatör İncelemeleri

Tez çalışmasının bu alt bölümünde algoritmanın bazı operatörlerinde kullanılan tekniklerin seçilen test problemleri üzerindeki etkisi araştırılmıştır. Özellikle etkisi incelenen durumlar arasında başlangıç çözümlerin algoritmanın etkinliği üzerindeki etkisi, dinamik arama tekniğinin etkinliği, çoklu-karşılıklı değişimde komşuluk büyüklüğünün sınırlandırılmasının etkisi ve ana sarsma mekanizmasının elde edilen sonuçlar üzerindeki etkisi araştırılmıştır.

4.7.1. Başlangıç çözümlerin etkinliğinin araştırılması

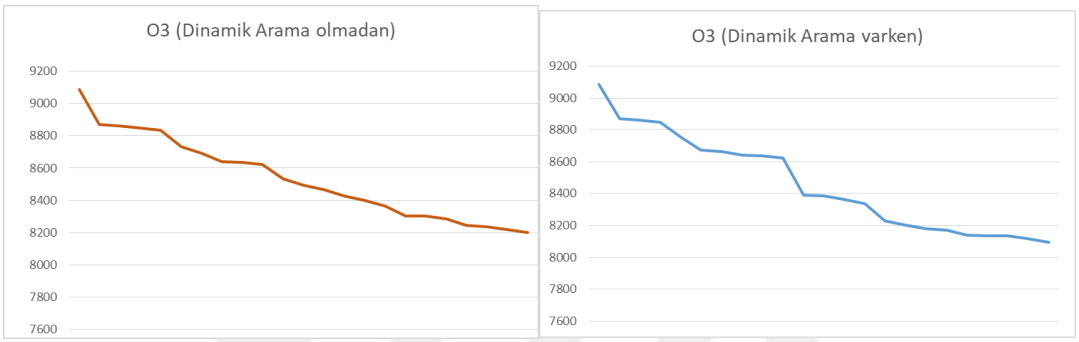
Algoritmada başlangıç çözümlerin oluşturulmasında k-ortalamalar tekniğine göre araç sayısına göre kümeler oluşturulduktan sonra klasik ekleme algoritmasına göre işlemler gerçekleştirilmektedir. Öte yandan bu başlangıç çözümlerin kümeleme olmaksızın klasik ekleme algoritmasıyla oluşturulduğunda nasıl bir farklılık çıktığı araştırılmaya değerdir. Burada kısmen boyutu daha küçük olan problemlerde anlamlı bir zaman kaybı beklenmezken, problem boyutu büyüdükçe kümelemenin etkinliği ortaya çıkmaktadır. Bu durumun ortaya konulması amacıyla *O3* test problemi üzerinde ilk 120 saniye içerisindeki yakınsama durumu 10 tekrar olarak çalıştırılmış ve hesaplar raporlanmıştır. Şekil 4.6'da kümelemenin etkinliği açık bir şekilde anlaşılmaktadır.



Şekil 4.16 *O3 için başlangıç çözüm karşılaştırması*

4.7.2. Dinamik arama yönteminin etkinliğinin araştırılması

Algoritmada dikkat çeken operatörlerden biri sıklıkla kullanımı olmayan ve büyük komşulukları ele alan *Dinamik Arama* yöntemedir. Yöntem büyük komşuluklara bakmasına karşın algoritmaya belirli bir zaman yükü getirmektedir. Bu durumun gerçekten katlanılabilir bir halde olup olmadığı incelenmiştir. Yine bu durumun araştırılması amacıyla *Dinamik Arama* olmadan ve Dinamik arama ile *O3* test problemi üzerinde ilk 120 saniye içerisindeki yakınsama durumu 10 tekrar olarak çalıştırılmış ve karşılaştırma raporlanmıştır.



Şekil 4.17 *O3 için Dinamik arama incelemesi*

Şekil 4.17'de görüldüğü gibi *Dinamik Arama* komşuluğunun anlamlı bir getirisi söz konusudur. Bununla beraber komşuluğun etkinliğinin anlaşılabilmesi için rota uzunluklarının çok kısa olmaması gerekmektedir.

4.7.3. Çoklu-karşılıklı değişimde sınırlama getirilmesinin araştırılması

Karşılıklı iki rotadan çeşitli büyüklükte artarda olan müşteri kümelerinin maliyet iyileştirmesi durumunda taşınması durumu olarak özetlenen teknikte en az 1, en fazla 5 müşteri ile sınırlandırma yapıldığı önceki alt bölümlerde açıklanmıştı. Bu durumun temel sebebi karmaşıklığı çok fazla olan yöntemin kısmen daha kısa sürelerde çalışmasını sağlamaktır. Bu sınırlandırmanın çözümlerin kalite ve süresi üzerindeki etkisinin araştırılmasına ihtiyaç duyulmuştur. Bu sebepler sırasıyla en fazla *10, 15* ve *20* büyüklüğünde ardışık müşteri değişimi *O3* test problemi üzerinde ilk 120 ve 1200 saniye içerisindeki etkisi incelenmiştir.

Tablo 4.10 *O3 için en fazla çoklu-karşılıklı değişim karşılaştırması*

	En fazla=5	**En fazla=10**	**En fazla=15**	**En fazla=20**
120 sn. içerisinde aldığı ortalama değer	8117.55	8169.89	8179.89	8200.96
1200 sn. içerisinde aldığı ortalama değer	7921.11	8010.25	8029.12	8042.56

Tablo 4.10'da bulunan sonuçlar incelendiğinde daha fazla ardışık segment değişiminin algoritmanın karmaşıklığını artırdığından aynı sürelerde daha kötü ortalamalar ile sonuçlar elde edildiği anlaşılmaktadır.

4.7.4. Ana sarsma mekanizmasının çözüm üzerinde etkisinin incelenmesi

Sarsma mekanizmaları kısmında bahsedildiği gibi ana sarsma mekanizmasında sekiz farklı stratejiye göre çıkar-ekle uygulanmaktadır. Araştırmalar esnasında söz konusu stratejilerin algoritmaya etkisinin incelenmesine ihtiyaç duyulmuştur. C1, C2 ve C3 test problemlerinde 10 tekrarda elde edilen ortalama değerler ana sarsma varken elde edilen sonuçlarla karşılaştırılmıştır.

Tablo 4.11 *C1, C2 ve C3 için ana sarsma mekanizmasının etkisi*

Problem	n	Q	Min araç	BKS	Ana sarsma varken ortalama	Ana sarsma yokken ortalama
C1	50	160	5	416.06	417.98	425.89
C2	75	140	10	567.14	574.87	598.43
C3	100	200	8	639.74	646.48	659.22

Tablo 4.11'deki sonuçlara göre ana sarsma mekanizmasının etkinliği ortaya çıkmaktadır. Ana sarsma mekanizması olmaksızın algoritma yerel en iyi noktalardan kurtulma konusunda yeterli olamamaktadır.

4.7.5. Komşulukların ilk iyileştirme yerine en iyi iyileştirme olması durumunda etkinliği

Daha önceki alt kısımlarda açıklandığı gibi algoritmada tüm komşuluklar açgözlü olarak tasarlanmıştır. Bir diğer deyişle herhangi bir iyileştirmenin olduğu an bu değişiklikler gerçekleştirilmekte ve aramaya geri dönerek devam edilmektedir. Öte yandan ilk iyileştiren değişiklik yerine o tekrar içerisinde mümkün olan en iyi değişikliğin gerçekleştirmesi stratejisi de kullanılabilmektedir.

Bu durumun etkinliğinin araştırılması için test problemlerinin önemli bir bölümü kullanılmıştır. Karşılaştırmalar sonucunda Tablo 4.12'de görüldüğü gibi açgözlü strateji tüm test problemleri için ortalama olarak en iyi değişim stratejisine göre daha iyi sonuçlar vermektedir.

Tablo 4.12 *Aç gözlü değişim ile en iyi değişim karşılaştırması*

Problem	Aç gözlü ve iyileşme sonlanana dek			En iyi değişim ve iyileşme sonlanana dek		
	Ort. Maliyet	St. Sapma	Ort. Süre (sn)	Ort. Maliyet	St. Sapma	Ort. Süre (sn)
C1	417.98	3.12	116.5	418.55	3.36	106.3
C2	574.87	3.45	203.8	576.12	3.46	182.5
C3	646.48	3.35	480.5	650.13	3.48	400.2
C4	744.51	5.42	780.3	747.22	6.12	770.2
C5	894.22	6.55	1285.4	899.12	7.15	1100.1
C11	699.22	5.88	610.9	705.22	6.15	570.1
C12	534.90	1.22	185.4	538.83	1.85	170.1
F11	178.87	2.55	412.2	180.20	3.13	357.9
F12	797.42	6.42	985.4	805.19	6.88	945.6

4.7.6. Tüm komşulukların etkinliklerinin incelenmesi

Bu alt bölümde tüm komşuluklar sırasıyla algoritmadan ayrıldığında ortaya çıkan durum incelenmiştir. Benzer analiz Bölüm 4.7.2'de dinamik arama komşuluğu için detaylı olarak gerçekleştirilmiştir. Diğer komşulukların etkilerini görmek üzere *O*3 test problemi üzerinden, sadece ilgili komşuluğun çıkarılması ile ortaya çıkan farklar elde edilmiştir. Bu etkileri incelemek üzere, tüm testler 10'ar kez tekrar edilmiş ve 120 sn. içerisindeki sonuçlar Tablo 4.13'te raporlanmıştır.

Tablo 4.13 *Tüm komşuluklar ayrı ayrı algoritmadan ayrıldığında elde edilen sonuçlar*

	2-opt	Or-opt	3-opt	Dinamik arama	Bire-bir değişim	Yeniden kon.	Karşılıklı çoklu-değişim	2opt*
Amaç fonksiyonu değeri	8158.44	8169.89	8256.52	8201.10	8195.16	8198.23	8303.01	8178.13
Yüzde değişim	0,814	0,956	2,026	1,341	1,268	1,306	2,601	1,058

Herhangi bir komşuluk ayrılmadığında *O*3 problemi için elde edilen ortalama amaç fonksiyonu değeri 8092.55'tir. Buradan tüm komşulukların çözüme anlamlı bir katkı sağladığı anlaşılmaktadır. Elde edilen sonuçlara göre katkısı en fazla olduğu görülen komşuluklar %2.6 ile karşılıklı çoklu değişim ve %2.03 ile 3-opt komşuluğudur. 2-opt komşuluğunun ise göreceli olarak katkısının daha az olduğu görülmektedir.

4.8. Test Problemlerinden Elde Edilen Sonuçlar

Çözüm kalitesi ve süre bakımından en iyi değerleri veren parametrelerle algoritmanın test problemleri üzerinde 10'ar tekrar çalıştırılmasıyla elde edilen sonuçlar izleyen tablolarda verilmiştir. Kullanılan iş istasyonunun ortamının Intel Xeon E5-2630 2.4 Ghz işlemci ve 32 GB RAM bulunmaktadır. Bununla birlikte algoritma Python 2.7'ye uygun olarak kodlanmıştır. İş istasyonunun gücünden tam olarak faydalanmak için problem okunduktan sonra tüm ikili değişiklerin maliyetleri büyük bir, çok boyutlu değişkene kaydedilmektedir. Örneğin, bu değişkende (*[3][1][4][5][7][9])* değeri çağrıldığında bir çözümde 3 ve 4, 1'in öncesi ve sonrasındaki teslimat noktaları 5 ve 9 ise 7'nin öncesi ve sonrasındaki teslimat noktalarına karşılık gelmekte ve 1, 7 değişikliğinin maliyeti sonuç olarak döndürülmektedir.

Tablo 4.14 *Literatür test problemleri için DKA ile elde edilen sonuçlar-1*

Problem	n	Q	Min araç	BKS	Ortalama	St. sapma	En iyi bulunan	Ortalama süre (sn)
C1	50	160	5	416.06	417.98	3.12	**416.06**	116.5
C2	75	140	10	567.14	574.87	3.45	**567.14**	203.8
C3	100	200	8	639.74	646.48	3.35	**639.74**	480.5
C4	150	200	12	733.13	744.51	5.42	737.95	780.3
C5	199	200	16	871.27	894.22	6.55	879.13	1285.4
C11	120	200	7	682.12	699.22	5.88	690.02	610.9
C12	100	200	10	534.24	534.90	1.22	**534.24**	185.4
F11	71	30000	4	176.99	178.87	2.55	177.00	412.2
F12	134	2210	7	769.55	797.42	6.42	779.12	985.4

C ve *F* serisi test problemlerinden elde edilen değerlere göre önerilen DKA çözüm kalitesi ve süresi bakımından rekabetçi bir yapıya sahiptir. *O* serisi test problemleri ise daha büyük yapıdadır.

Tablo 4.15 *Literatür* ***test problemleri için DKA ile elde edilen sonuçlar-2***

Problem	n	Q	Min araç	BKS	Ortalama	St. sapma	En iyi bulunan	Ortalama süre (sn)
O1	200	900	5	6018.52	6135.98	12.23	6018.52	3432
O2	240	550	9	4544.46	4609.25	11.45	4581.10	4155
O3	280	900	7	7721.16	7870.96	15.66	7830.86	5853
O4	320	700	10	7215.48	7400.16	17.25	7374.22	9325
O5	360	900	8	9180.93	9255.23	18.68	9194.80	10857
O6	400	900	9	9773.83	9992.24	24.44	9951.60	13395
O7	440	900	10	10326.57	10610.75	27.88	10507.81	14750
O8	480	1000	10	12389.43	12701.34	22.55	12524.49	15602

O serisi test problemlerinde bulunan düğüm sayısı son derece fazladır. Bu durum problemlerde bilinen en iyi değere yakınsamayı dahi çok zor bir hal aldırmaktadır. Buna rağmen önerilen algoritma tutarlı bir şekilde bilinen en iyi değerlere yakınsamaktadır. Algoritmanın performansının literatürdeki diğer bazı çalışmalar ile karşılaştırmaları ise izleyen tablolarda yer almaktadır.

Tablo 4.16 *Literatür çalışmaları ile elde edilen sonuçların karşılaştırılması-1*

Problem	**A**		**B**		**C**		**Geliştirilen DKA**		
	Maliyet	**Süre(sn)**	**Maliyet**	**Süre(sn)**	**Maliyet**	**Süre(sn)**	**En iyi Maliyet**	**Ort. Maliyet**	**Süre(sn)**
C1	416.1	88.8	**416.06**	120	**416.06**	6.2	**416.06**	417.98	116.5
C2	574.5	167.5	**567.14**	360	**567.14**	31.3	**567.14**	574.87	203.8
C3	641.6	325.3	**641.76**	850	**639.74**	39.5	**639.74**	646.48	480.5
C4	740.8	870.2	**733.13**	1790	**733.13**	128.6	737.95	744.51	780.3
C5	953.4	1415	897.93	1240	924.96	380.6	879.13	894.22	1285.4
C11	683.4	696	**682.12**	730	682.54	121.6	690.02	699.22	610.9
C12	535.1	233.6	**534.24**	800	**534.24**	32.9	**534.24**	534.90	185.4
F11	177.4	398.1	**177.00**	690	**177.00**	19.5	**177.00**	178.87	412.2
F12	781.2	1000.2	770.17	2370	769.66	158.2	779.12	797.42	985.4

A: Yasaklı arama [45]
B: Adaptif büyük komşuluk arama [50]
C: Record to Record seyahat algoritması [51]

Tablo 4.17 *Literatür çalışmaları ile elde edilen sonuçların karşılaştırılması-2*

Problem	**D**		**E**		**F**		**Geliştirilen DKA**		
	Maliyet	**Süre(sn)**	**Maliyet**	**Süre(sn)**	**Maliyet**	**Süre(sn)**	**En iyi Maliyet**	**Ort. Maliyet**	**Süre(sn)**
C1	416.06	17.6	**416.06**	1.78	**416.06**	0.46	**416.06**	417.98	116.5
C2	567.14	29.0	**567.14**	6.44	**567.14**	46.44	**567.14**	574.87	203.8
C3	639.74	239.6	**639.74**	15.67	**639.74**	12.09	**639.74**	646.48	480.5
C4	733.13	585.0	**733.13**	27.77	**733.13**	63.71	737.95	744.51	780.3
C5	905.96	292.1	883.50	1579.45	**871.27**	596.95	879.13	894.22	1285.4
C11	682.12	231.6	**682.12**	23.54	**682.12**	53.08	690.02	699.22	610.9
C12	534.24	163.7	**534.24**	5.74	**534.24**	2.94	**535.24**	534.90	185.4
F11	178.09	140.2	**177.00**	4.41	**176.99**	74.58	**177.00**	178.87	412.2
F12	769.66	1237.5	**769.55**	40.51	770.17	749.38	779.12	797.42	985.4

D: Çoklu başlangıçlı DKA [52]
E: Set partition temelli ietratif yerel arama [65]
F: Çok fazlı salınımlı DKA [57]

Literatürde göze çarpan altı farklı çalışmada elde edilen AARP'ye yönelik sonuçlar ile tez çalışmasında önerilen DKA'dan elde edilen sonuçlar karşılaştırılmıştır.

Önerilen algoritma diğer geliştirilen metasezgisel yöntemlerle rekabet edebilir sonuçlar ortaya koymaktadır.

Tablo 4.18 *Literatür çalışmaları ile elde edilen sonuçların karşılaştırılması-3*

Problem	A		B		C		Geliştirilen DKA		
	Maliyet	Süre (sn)	Maliyet	Süre (sn)	Maliyet	Süre (sn)	En iyi Maliyet	Ort. Maliyet	Süre (sn)
O1	**6018.52**	365	**6018.52**	1271	**6018.52**	7.9	**6018.52**	6135.98	3432
O2	4584.55	440	4544.46	1247	4568.85	2232	4581.10	4609.25	4155
O3	7732.85	493	7721.16	2009	7731.00	174	7830.86	7870.96	5853
O4	7297.61	574	7215.48	2663	7260.89	7200	7374.22	7400.16	9325
O5	9197.61	767	9180.93	5702	9200.182	3451	9194.80	9255.23	10857
O6	9803.80	977	9773.83	5066	9790.00	5779	9951.60	9992.24	13395
O7	10374.97	935	10326.57	6141	10357.40	10205	10507.81	10610.75	14750
O8	12429.59	1127	12389.43	6653	12392.00	14262	12524.49	12701.34	15602

A: Record to Record seyahat algoritması [51]
B: Set partition temelli ietratif yerel arama [65]
C: Çok fazlı salınımlı DKA [57]

Daha büyük boyutlu problemlerin yer aldığı *O* serisinde karşılaştırmalar oldukça iyi sonuçlar elde edilen çalışmalar ile yapılmıştır. Buna göre, çözüm süreleri bakımından avantajlı durumda bulunmayan algoritma çözüm kalitesi bakımından tutarlı ve rekabetçi bir yapı ortaya koymaktadır.

4.9. Önerilen Algoritmanın Gerçek Hayat Problemine Uygulanması

Geliştirilen karar destek sisteminin model tabanında yer alan melez GA'ya göre maliyet bakımından oldukça iyi getirisinin olması beklentisiyle bir önceki bölümde anlatılan DKA geliştirilmiştir. Bu bölümde karar destek sisteminde çözüm aranan gerçek hayat probleminden türetilen test problemleri üzerinde önerilen iki algoritmanın performansları karşılaştırılmıştır.

DKA gerçek hayat problemine uygulanırken kümeleme tekniği kullanılmamış ve başlangıç çözümler sıradan ekleme (insertion) [61] algoritmasına göre oluşturulmuştur.

Tablo 4.19 *Üretilen test problemleri için DKA'dan elde edilen sonuçlar*

Problem	n	Kamyon sayısı	Tır sayısı	Ortalama	St. Sapma	En iyi değer	Ort. süre
EO1	117	7	16	68213.1	134.25	67871.8	283.2
EO2	117	11	21	80542.2	151.22	80146.1	294.3
EO3	40	3	13	36378.9	142.13	36130.2	70.3
EO4	95	7	21	76544.5	145.12	76330.0	275.4
EO5	90	5	20	64465.7	137.22	64122.4	250.1
EO6	80	7	15	60265.4	140.45	59918.4	160.9
EO7	110	9	20	76325.4	147.52	75983.2	253.5
EO8	92	4	13	58920.2	143.20	58632.2	188.8

Hem melez genetik algoritmayla karşılaştırma yapabilmek hem de mümkünse daha iyi sonuçlar elde edebilmek üzere geliştirilen DKA gerçek hayat probleminde melez genetik algoritmaya göre çok daha iyi sonuçlar vermiştir. DKA uygulamasıyla birlikte türetilen tüm test problemlerinde maliyet bakımından melez genetik algoritmaya göre ortalama yaklaşık %9.3 iyileştirme bulunmuştur.

Tablo 4.20 *Üretilen test problemleri için melez GA ile DKA'dan elde edilen sonuçların karşılaştırılması-1*

Problem	Melez Genetik Algoritma			DKA		
	Ort. Maliyet	En iyi Maliyet	Süre (sn)	Ort. Maliyet	En iyi Maliyet	Süre (sn)
EO1	83447.5	82344.3	273.4	68213.1	67871.8	283.2
EO2	86215.8	85647.7	248.6	80542.2	80146.1	294.3
EO3	37141.5	36740.5	155.2	36378.9	36130.2	70.3
EO4	83512.4	82815.5	200.5	76544.5	76330.0	275.4
EO5	71313.5	70752.9	198.7	64465.7	64122.4	250.1
EO6	64235.2	63995.5	182.5	60265.4	59918.4	160.9
EO7	81600.1	81062.2	260.1	76325.4	75983.2	253.5
EO8	64786.2	64428.0	203.5	58920.2	58632.2	188.8

DKA uygulamasıyla birlikte melez genetik algoritmaya göre kullanılan araç sayısında da anlamlı bir azalma gerçekleşmiştir. Sonuç olarak karşılaştırma amacıyla geliştirilen DKA kullanılarak melez GA'ya göre çok daha az maliyetli sonuçlar elde edilmiştir.

Tablo 4.21 *Üretilen test problemleri için melez GA ile DKA'dan elde edilen sonuçların karşılaştırılması-2*

Problem	Melez Genetik Algoritma		DKA	
	Kamyon	Tır	Kamyon	Tır
EO1	11	25	7	16
EO2	9	35	11	21
EO3	7	17	3	13
EO4	9	29	7	21
EO5	11	26	5	20
EO6	15	22	7	15
EO7	14	31	9	20
EO8	9	27	4	13

5. ÖNERİLEN ALGORİTMANIN GRAFİK İŞLEM BİRİMLERİ ÜZERİNDE PARALELLEŞTİRİLMESİ

Grafik işlem birimleri (Graphics Processing Units, GPUs) (GİB) günümüz bilgisayarlarının ekran kartları üzerinde bulunan işlemcilerdir. Günümüzde GİB'ler eş zamanlı hesaplamaya uygun yapıları sayesinde algoritmaların oldukça hızlı çalışmasına olanak tanıyabilmektedir. Çalışmanın bu bölümünde, çözüm sürelerinin anlamlı bir şekilde azaltılabilmesi için geliştirilen algoritmanın GİB üzerinde paralel hesaplamaya uygun olarak ele alınışı açıklanmaktadır.

5.1. Metasezgisel Algoritmalar ve GİB

Metasezgisel algoritmalar, son derece gelişmiş arama teknikleriyle çözüm uzaylarını aramalarına rağmen, problemlerin boyutunun büyümesi, algoritmaların en iyi çözüme yakınsayamamasına sebep olabilmektedir. Bu yüzden paralelleştirme, büyük problemlerin çözümü için yeni yöntemlerin geliştirilmesine olanak sağlayan oldukça önemli bir araçtır.

Paralel hesaplama, yıllar boyunca büyük boyutlu zor problemlerin çözümü için başvurulan kaçınılmaz bir yol olmuştur. Paralel metasezgisel algoritmaların tasarımı ve kodlanması, hesaplamanın yapılacağı platformdan doğrudan etkilenmektedir. Bilgisayar sistemlerinde görselleştirme aracı olarak kullanılan GİB bu platformlardan biridir.

Günümüzde gelişmiş ve çok güçlü ekran kartları üzerinde yüzlerce bulunan GİB, kitleler halindeki veriyi tamamen paralel bir yapıya dönüştürerek görselleştirmektedir [66]. Bununla birlikte GİB, yalnızca verilerin görselleştirmesi için kullanılmakla kalmayıp özellikle, yoğun aritmetik işlem gerektiren hesaplamalarda da kullanılabilmektedir. Bu tür hesaplamalar, GİB üzerinde gerçekleştirilerek işlem süreleri kayda değer şekilde azaltılabilmektedir; fakat merkezi işlem birimlerinden (MİB) farklı olarak GİB, verilerin paralel hesaplamaya uygun hale getirilmesine ihtiyaç duymaktadır. Bu durum, GİB kullanımında dikkat edilmesi gereken en önemli adım ve üzerinde en çok emek sarf edilen kısımdır.

Önceleri GİB'ler yalnızca grafiğe dayalı uygulamalarda kullanılırken 2006 yılında nVIDIA firmasının CUDA™ (Compute Unified Device Architecture) geliştirme ortamını tanıtması, GİB'lerin hesaplamalı bilimlerde kullanımı için milat olarak kabul edilebilir [67]. Son yıllarda GİB'lerin daha da gelişmesi, paralel hesaplama alanındaki çalışmaları bu yöne odaklandırmıştır. Söz konusu teknoloji sayesinde karmaşık algoritmalar onlarca kat hızlanmıştır.

Her ne kadar metasezgisel algoritmalar, anlamlı bir şekilde hesapsal karmaşıklığı azaltarak en iyi çözümü arama işlemini gerçekleştirse de problemlerin boyutunun büyümesi, çözüm sürelerinin artmasına ve çözüm kalitesinin azalmasına sebep olmaktadır. Kombinatorik eniyileme problemlerinde problem boyutunun büyümesi, çözüm uzayının üssel olarak büyümesine sebep olmaktadır. Bununla birlikte amaç fonksiyonunun hesaplanması ve diğer birçok hesaplama daha fazla işlem gerektirmektedir. Bu sebeplerden dolayı metasezgisel algoritmalara paralel bir tasarım kazandırılması ön plana çıkmaktadır. Daha geniş anlamda paralel ve dağıtılmış hesaplama, metasezgisel algoritmaların tasarımı ve kodlanmasında kullanılması şu amaçları gütmektedir:

- Aramanın hızlandırılması,
- Elde edilen çözüm kalitesinin iyileştirilmesi,
- Tutarlılığın artırılması,
- Büyük boyutlu problemlerin çözülmesi.

Bu amaçlar eşliğinde metasezgisel algoritmalarda paralelleştirme üç ana boyutta ortaya çıkmaktadır;

- *Çözüm seviyesinde paralelleştirme:* Bu tip paralelleştirme, tek bir çözümün değerlendirmesinin parçalara bölünerek eş zamanlı değerlendirmesini içermektedir.
- *İterasyon seviyesinde paralelleştirme:* Birden çok çözümün eş zamanlı ele alınması ya da bir çözüm üzerindeki komşuluk karşılaştırmanın parçalanarak eş zamanlı ele alınabilmesi anlamına gelmektedir.
- *Algoritmik seviyede paralleştirme:* Daha tutarlı sonuçların elde edilebilmesi için farklı metasezgisel algoritmaların eş zamanlı olarak işlemesidir.

5.2. GİB Mimarisi

GİB ekran kartları üzerindeki işlemcilerdir. Bu işlemciler, MİB ‘ye göre daha az güçte fakat daha fazla sayıdadır. Son yıllarda piyasaya sürülen ekran kartlarının üzerinde oldukça gelişmiş özellikleriyle bulunan GİB, genel olarak bilgisayar ortamında görselleştirme aracı olarak kullanılmaktadır. Öte yandan GİB yalnızca görselleştirme işlemlerinde değil, hesaplamalı bilimlerde de yükselen bir eğilimle ilgi görmektedir.

Eniyileme problemlerinin çözüm sürelerinin kısalması motivasyonuyla metasezgisel algoritmaların GİB üzerinde geliştirilmesi yaygınlaşmaktadır. Bununla birlikte söz konusu teknoloji geleneksel programlama bilgi ve becerilerinin doğrudan kullanılmasının önünde bir engeldir. Bu teknolojiyi kullanarak uygulama geliştirmek için GİB mimarisinin çok iyi şekilde anlaşılması gerekmektedir.

GİB üzerinde hesapsal işlemlerin yapılması konusunda 2006 yılında nVIDIA firmasının CUDA™ geliştirme ortamını tanıtması bu alanda dönüm noktası olarak kabul edilmektedir. O günden beri binlerce uygulama ve makale literatüre katılmış, milyonlarca bilgisayar ise CUDA kullanılabilir hale gelmiştir [67]. Günümüzde sadece bilgisayarlarda değil, televizyonlarda, akıllı cep telefonlarında ve tabletlerde de GİB teknolojisin yaygın şekilde kullanıldığı görülmektedir.

nVIDIA bugüne kadar yonga kümeleri için altı farklı mimari sunmuştur. İlki 2006 yılında sunulan G80, ikincisi GT200, üçüncüsü Fermi, dördüncüsü Kepler, beşincisi Maxwell ve sonuncusu Turing teknolojisidir. Mimarilere ait temel bilgiler Tablo 5.1'de yer almaktadır. Tüm mimariler GİB üzerinde programlama yapmaya olanak sağlamaktadır. CUDA, C, C++ ve Fortran gibi programlama dillerinde yazılan kodların GİB üzerinde paralel olarak işlenmesini sağlayan yazılımdır. Mimarinin daha iyi anlaşılabilmesi için bilinmesi gereken bazı terimler şöyledir:

CUDA Çekirdeği (CUDA Core): Yonga kümesi üzerinde yer alan işlemciler.

Akış İşlemcisi (Streaming Multiprocessor): CUDA çekirdeklerinin kümelendiği işlemci.

Cihaz Belleği (Device Ram): Yonga kümesinin bağlı olduğu bellek.

Paylaşılan Bellek (Shared Memory): Yonga üzerindeki bilgi paylaşımını sağlayan bellek.

Kernel: Yazılan programın GİB üzerinde işlenmesini sağlayan kod bloğu.

Thread: GİB'de işlenen en küçük iş parçacığı.

Blok: Threadlerden oluşan iş parçacığı topluluğu.

Grid: Bloklardan oluşan iş parçacığı topluluğu.

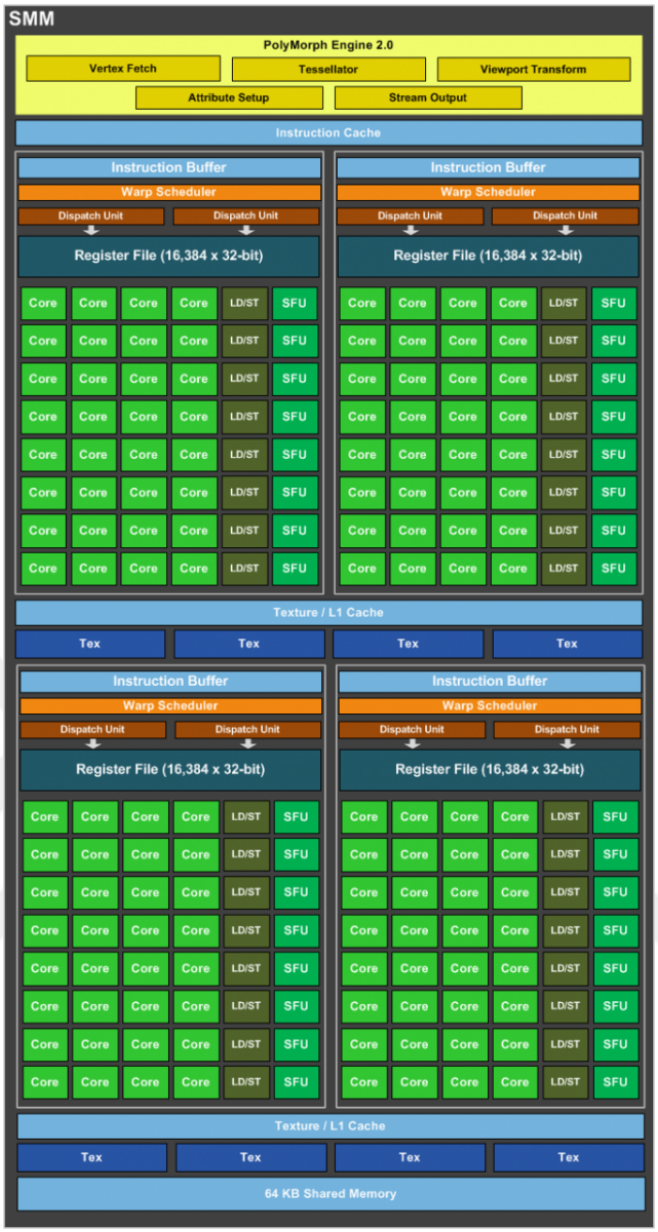
Tablo 5.1 *GİB Mimarileri*

Mimari	**G80**	**GT200**	**Fermi**	**Kepler (GK 108)**	**Maxwell (GM 108)**	**Turing (TU 102)**
Transistör	681 Milyon	1.4 Milyar	3.0 Milyar	3.54 Milyar	5.2 Milyar	18.6 Milyar
CUDA Çekirdeği	128	240	512	1536	2048	4608

Tablo 5.1'de de bahsedildiği gibi Maxwell mimarisinde 2048 CUDA çekirdeği bulunmaktadır. Bu çekirdekler, her birinde 128'er tane olmak üzere 16 adet akış işlemcisi içerisinde kümelenmiştir. Maxwell 96 KB paylaşılan hafızayla, bloklar arası bilgi paylaşımına olanak sağlamaktadır.

Şekil 5.1'de akış işlemcilerinden birinin yapısı bulunmaktadır. 128 adet çekirdek barındıran akış işlemcisinin özel matematiksel işlemleri yapmak üzere kullanılan 32 adet SFU (Special Function Unit) birimi bulunmaktadır. Her bir çekirdekte ise tam sayılı işlemler için INT (Integer Unit), ondalıklı işlemler için FP (Floating Point Unit) birimleri bulunmaktadır.

CUDA, sıklıkla kullanılan birçok programlama dili ile Microsoft Visual Studio ortamında ya da diğer derleyici ortamlarında program yazılmasını mümkün kılmaktadır. Yazılan kodların bir kısmı hala MİB üzerinde işlenirken, bir kısmı da kernel sayesinde GİB üzerinde işlenmektedir. Bir kernel çalıştırılmadan önce, grid yapısına dair parametrelere ihtiyaç duymaktadır. Grid yapısı, iş parçacıklarının GİB üzerinde eş zamanlı yürütülmesi açısından büyük önem taşımaktadır. Maxwell mimarisinde her akış işlemcisinde en fazla 32 blok eş zamanlı olarak çalışabilmektedir. Her blokta bulunabilecek thread sayısı ise 1024 ile sınırlandırılmıştır.



Şekil 5.1 *Akış işlemcisi ve Cuda çekirdeği*

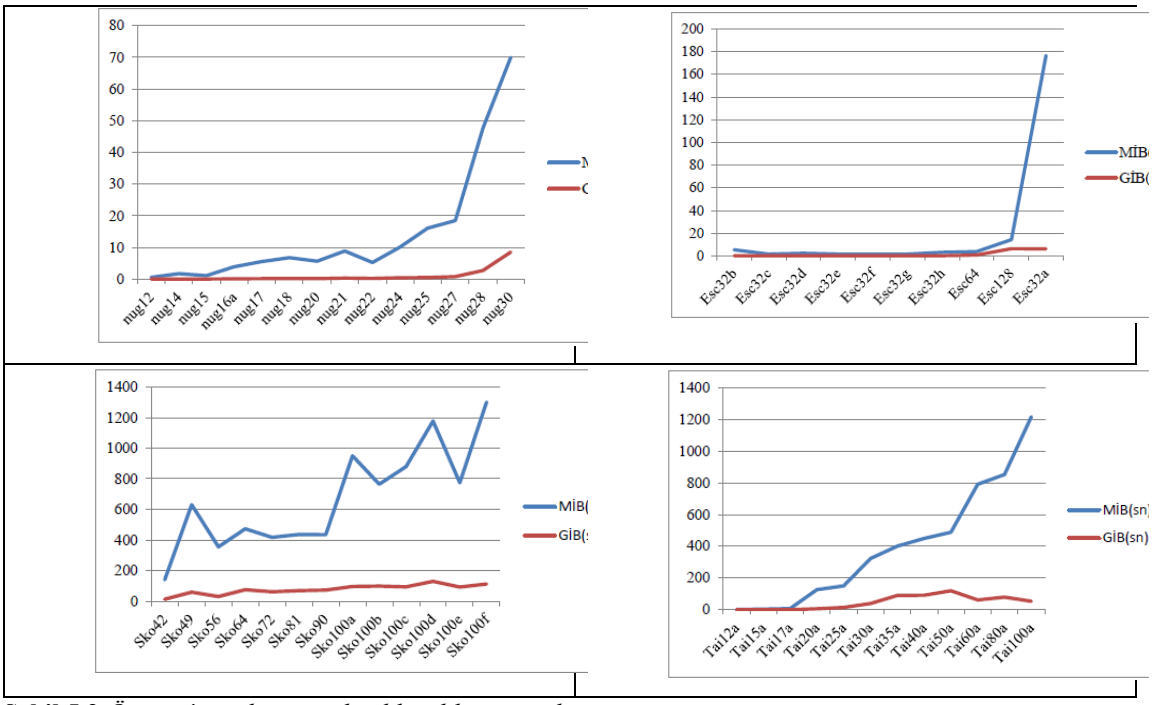
Bir grid yapısı tanımlandığında threadler Warp çizelgeleyeci (Warp Scheduler) tarafından 32'şerli gruplar olarak akış işlemcisine gönderilmektedir. Maxwell mimarisinde her akış işlemcisinde dörder adet Warp çizelgeleyici bulunmaktadır. GİB üzerinde programlama yaparken Warp çizelgeleyicilerinin çalışma mantığını göz önünde bulundurmak yazılan programın etkinliği için büyük önem taşımaktadır. Çizelgeleyici threadleri CUDA çekirdeklerine dağıtmaktadır. Grid yapısı, en fazla (2^{31}-1,65535,65535) blok ve en fazla her blok için (1024,1024,64) thread içerecek şekilde tanımlanabilmektedir.

5.3. Paralel Metasezgisel Algoritmalar İle İlgili Literatür Çalışmaları

Teknolojik gelişmeler, hem MİB'lere hem de GİB'lere çok farklı bir boyut kazandırmıştır. Son yıllarda ise MİB'lerde fiyat performans oranına katkı sağlayacak önemli gelişmeler olmamaktadır. Öte yandan GİB'ler hızlı adımlarla güçlenmeye devam etmektedir.

Yıllar boyunca paralel yaklaşımlar ve uygulamaları Aiex ve ark. [68], Alba ve ark. [69], Andre ve ark. [70], Crainic ve ark. [71], Randall ve ark. [72] çalışmalarında olduğu gibi güncelliğini korumuştur. Bu çalışmaların bazılarında paralel MİB'ler kullanılırken bazılarında birçok iş istasyonu işbirliği içerisinde kullanılmıştır. Son zamanlarda ise eniyileme problemleri için paralel metasezgisel algoritma tasarımları GİB üzerinde yoğunlaşmıştır.

Özçetin [73], çalışmalarında karesel atama problemi (KAP) için GİB üzerinde paralel bir evrimsel algoritma sunmuşlardır. Algoritma içerisinde amaç fonskiyonun hesaplanması ve yerel arama operatörü en zaman alıcı parçalar olarak belirlenmiş ve bu kısımlar GİB'ye uygun şekilde paralelleştirilmiştir. Algoritma çalışmada ele alınan 59 test probleminin 43'nde bilinen en iyi değerlere ulaşmıştır. Çözüm sürelerinin ölçümü sonucunda GİB'de elde edilen çözümler, MİB'e göre 51 kata kadar, ortalama 17 kat daha hızlı olduğu gözlemlenmiştir. Şekil 5.2'de elde edilen sonuçların grafikleri yer almaktadır.



Şekil 5.2 *Özçetin'in çalışmasında elde edilen sonuçlar*

Tsutsui ve Fujimoto [74], çalışmalarında KAP için GİB üzerinde paralel bir GA sunmuşlardır. Çalışmada, GİB üzerinde geliştirilen algoritmanın MİB'e göre 3 ile 12 kat arası daha hızlı sonuçlar elde edilmiştir. Aynı yazarlar bir başka çalışmada [75], KAP'ın çözümü için karınca kolonileri eniyilemesi ve 2-opt yerel arama algoritmasının biraraya getirilmesinden oluşan bir yöntem sunmuşlardır. Elde edilen sonuçlara göre GİB üzerindeki çözüm süreleri, MİB'e göre ortalama 24.6 kat daha hızlı olduğu vurgulanmıştır. Czapinski [76], çalışmalarında CUDA ortamında KAP için çoklu başlangıçlı YA geliştirmiştir. Çalışmada, KAP için geliştirilen literatür problemlerinden bazıları ele alınmış ve %1'lik çözüm uzaklıklarıyla 70 kata kadar daha kısa sürede sonuçlar elde edilmiştir.

Cecilia ve ark. [77], GSP için karınca kolonileri eniyilemesi geliştirmiş ve 20 kata kadar daha hızlı çözümler elde etmişlerdir. Aynı problem için yapılan bir başka çalışmada Delevacq ve ark. [78], karınca kolonileri eniyilemesi ile 23.6'ya kadar çıkan hızlanma elde etmişlerdir.

Shulz [79], ARP için yerel arama algoritmalarının GİB uygulamaları üzerine bir çalışma sunmuştur. Hızlanma faktörlerine ait bilgi verilmeyen çalışmada özellikle, GİB üzerinde işlem yaparken darboğazlardan birisi olan hafıza işlemlerinin azaltılabileceği üzerinde durulmuştur. Aynı problem üzerine çalışan Groer ve ark. [80], yerel arama tekniklerini GİB üzerinde uygulamışlardır. Bir diğer çalışmada Szymon ve Dominik [81], çok kriterli ARP için GİB üzerinde paralel bir yasaklı arama algoritması geliştirmişlerdir.

5.4. AARP'nin Fazla Sayıda Çözümü İçin Amaç Fonksiyonun Eş Zamanlı Hesaplanması

Algoritma geliştirme çalışmaları başlandığında ilk olarak GİB'nin potansiyel gücünü ortaya konulması amaçlanmıştır.

Popülasyon temelli algoritmalar ve bazı çoklu başlangıçlı tek nokta arama algoritmalarında en iyi çözüm fazla sayıda çözüm ile gerek aralarında bilgi alışverişi olarak gerekse bilgi alışverişi olmaksızın gerçekleştirilmektedir. Kombinatorik eniyileme problemlerinin çoğunda olduğu gibi AARP'de de amaç fonksiyonun hesaplanması zaman alıcıdır. Bu iki noktadan yola çıkarak GİB üzerinde fazla sayıda çözümün eş zamanlı değerlendirilmesinden elde edilecek kazanç araştırılmıştır. Bu bağlamda AARP için fazla sayıda çözümün amaç fonksiyonun eş zamanlı olarak GİB üzerinde hesaplanmasına yönelik bir teknik önerilmiştir.

Uzaklık matrisi aşağıdaki Şekil 5.3 görüldüğü gibi bir vektör haline dönüştürülmüş ve GİB hafızasına kopyalanmıştır.

um(i,j)	0-0	0-1	0-2	0-3	0-4	1-0	1-1	1-2	1-3	1-4	2-0	2-1	2-2	2-3	2-4	…
indeks	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…

Şekil 5.3 *GİB'ye vektör olarak aktarılan uzaklık matrisi*

Permütasyon temelli çözümlerin tümü aşağıdaki şekilde görüldüğü bir vektör haline getirilerek GİB hafızasına transfer edilmiştir. Söz konusu permütasyon temelli çözümler rassal olarak oluşturulmaktadır. Şekil 5.4'te toplu şekilde çözümlerin nasıl aktarıldığına yönelik bir örnek bulunmaktadır.

	Çözüm-1				Çözüm-2				Çözüm-3				…
sol_gpu	2	3	1	4	4	1	2	3	3	2	4	1	
indeks	0	1	2	3	4	5	6	7	8	9	10	11	…

Şekil 5.4 *GİB'ye aktarılan toplu çözüm örneği*

Bununla birlikte her çözüm için rota başlangıç noktaları ikili değişkenler ile tarif edilmiş ve tüm çözümler için aşağıdaki şekildeki gibi bir araya getirilmiştir. Rota başlangıç noktaları permütasyonun en solundan başlayarak verilen araç kapasitesi dolana dek 0 yeni bir araca başlandığı noktada 1 değeri alacak şekilde GİB'ye gönderilmeden önce hesaplanmaktadır. Şekil 5.5'te toplu şekilde rota başlangıç noktalarının nasıl aktarıldığına yönelik bir örnek bulunmaktadır.

	Ç1 için araç baş.				Ç2 için araç baş.				Ç3 için araç baş.				…
rotabas_gpu	1	0	0	1	1	1	0	0	1	0	1	0	
indeks	0	1	2	3	4	5	6	7	8	9	10	11	…

Şekil 5.5 *GİB'ye aktarılan toplu rota başlangıç noktaları örneği*

Yukarıdaki bilgilere ek olarak uygun olmayan çözümlerden doğabilecek ceza maliyetinin hesaplanabilmesi için her çözümün araç sayısı bir vektör halinde GİB'ye gönderilmektedir. Bütün bu bilgiler eşliğinde grid yapısı her bir blokta *1024* thread olacak şekilde blok sayısı artırılarak hesaplamalar gerçekleştirilmiştir.

*idx=blockIdx.x*blockDim.x+threadIdx.x* olmak üzere eş zamanlı hesaplamanın yapıldığı kernelin sözde kodu Şekil 5.6'daki gibidir. Kernel içerisinde tek döngü kullanılmaktadır. Bu döngüde indeks depo hariç şehir sayısına kadar hareket etmektedir. Döngü içerisinde ilk adımda tüm çözümler için permütasyonun *i.* elamanı ile ikili değişkenlerden oluşan rota başlangıç vektörünün *i.* elemanı çarpılmakta ve uzaklık matrisinde karşı gelen değer ilgili çözümün amaç fonksiyonuna eklenmektedir. İkili

değerlerden oluşan vektörde rota başlangıcı olmayan elemanlardan 0 değeri geleceği için çarpım işleminin sonucu 0 olmakta ve bu değer (um[0]=0) deponun kendine olan uzaklığı olduğu için amaç fonksiyonuna herhangi bir ekleme yapılmamaktadır. Öte yandan bu değer 1 olduğunda depo ile rotanın ilk elemanın uzaklığı amaç fonksiyonu değerine eklenmektedir. Döngü içerisinde sonraki aşamada ise ikili değerlerden oluşan vektörde rota başlangıcı olmayan elemanların ilgili çözümün önceki elemanı ile bağlantısının uzaklığı amaç fonksiyonu değerine eklenmektedir.

```
for(i<sehirs)
                    sum[idx]+=um[rbas[idx*sehirs+i]*sol[idx*sehirs+i]];
                    if (rbas[idx*sehirs+i]==0)
                          sum[idx]+=um[city_from*(sehirs+1)+city_to];
```

Şekil 5.6 *GİB'de amaç fonksiyonunun eş zamanı hesaplanması*

Üç test problemi için elde edilen ölçüm süreleri Tablo 5.2'deki gibidir.

Tablo 5.2 *AARP için eş zamanlı amaç fonksiyonu değerlendirme*

	P-n16-k8 CPUsn	P-n16-k8 GPUsn	A-n80-k10 CPUsn	A-n80-k10 GPUsn	E386-47t CPUsn	E386-47t GPUsn
1024*1	0.022	0.131	0.126	0.085	0.356	0.097
1024*2	0.037	0.087	0.154	0.088	0.620	0.100
1024*4	0.089	0.113	0.300	0.089	1.412	0.104
1024*8	0.139	0.098	0.529	0.094	2.500	0.106
1024*16	0.360	0.146	1.162	0.096	5.528	0.143
1024*32	0.541	0.091	2.285	0.136	10.027	0.152
1024*64	1.108	0.136	4.383	0.142	20.497	0.210
1024*128	2.472	0.142	8.887	0.151	42.560	0.351
1024*256	4.413	0.156	20.770	0.225	84.241	0.676
1024*512	9.541	0.257	34.871	0.365	174.168	1.378
1024*1024	18.793	0.304	73.872	0.649	372.455	3.282

Yapılan testler sonucunda önerilen amaç fonksiyonu hesaplama tekniği ile çok sayıda çözümün çok kısa sürede değerlendirilebileceği gösterilmiştir. Ortaya çıkan sonuç ile birlikte GİB'de akıllıca tasarlanmış bir algoritma ile birlikte kombinatorik eniyileme problemleri için etkin çözümler, MİB'ye göre oldukça kısa sürede elde edilebilmektedir. Bunun taşıdığı bir diğer anlam ise elde edilen zaman kazancının büyük problemler için daha fazla arama yapılabilmesi olanağının sağlanmasıdır.

5.5. DKA'nın GİB'de Ele Alınması

Bu bölümde önerilen algoritmanın GİB'de paralel olarak ele alınmasında kullanılan stratejiler detaylı olarak açıklanmaktadır. Burada algoritmanın tamamının değil yalnızca yoğun zaman alıcı ve GİB'de paralel olarak hesaplamaya uygun olan operatörler üzerinde durulmuştur. Algoritmada birbirine bağlı işlemlerin fazla sayıda olduğu ve fazla sayıda durum sınaması yapılan operatörlerin zaman katkısı olmayacağı öngörülerek GİB'de ele alınmamıştır.

Başlangıç çözüm MİB'de oluşturulmaktadır. Problemin gerekli bilgileriyle birlikte başlangıç çözümler GİB'nin genel hafızasına (global memory) taşınmaktadır. Bir kez GİB'ye veri transfer edildikten sonra GİB'de ele alınan operatörler işlendikten sonra diğer operatörler için çözüm tekrar MİB'ye taşınmaktadır. Sonraki tekrarlarda aynı süreç devam etmektedir.

Geliştirilen DKA'nın daha önceden de bahsedildiği gibi üç temel ana hattı bulunmaktadır. Bunlar, rota içi arama DKA, rotalar arası arama DKA ve sarsma mekanizmasıdır. Burada rota içi arama DKA ve rotalar arası arama DKA GİB'de ele alınırken sarsma mekanizmasının üzerinde durulmamıştır. Diğer bir deyişle sarsma mekanizmasındaki tüm işlemler hala MİB üzerinde gerçekleştirilmektedir.

Geliştirilen DKA ile GİB üzerinde ele alınan DKA arasında tüm komşuluklar bakımından temel bir farklılık bulunmaktadır. Geliştirilen DKA'da tüm komşuluklar açgözlü bir biçimde indeksler başa dönerek ve iyileşme sonlanana dek gerçekleştirilirken GİB üzerinde tüm komşuluklarda eldeki indeksler için tüm fark değerleri ortaya konulduktan sonra en iyi değişiklik gerçekleştirilmekte ve iyileşme sonlanana dek bu durum sürdürülmektedir. Bu farklılığın temel sebebi GİB'nin etkinliğinden faydalanmaktadır. Bu durumdan dolayı seri versiyon ile adil karşılaştırmanın yapılabilmesi için Bölüm 4.7.5'teki sonuçlar kullanılmıştır.

GİB üzerinde uygulanan DKA'nın operatörleri ve problemin ele alınış şekli izleyen alt bölümlerde açıklanmaktadır.

5.5.1. Problemin GİB'de gösterimi

GİB'de çözümlerin nasıl gösterileceği algoritmanın eş zamanlı çalışması bakımından son derece kritik öneme sahiptir. Çözümlerin GİB'de gösterilmesinde aşağıda detaylarıyla anlatılan kolaylıkları sağlayan yeni bir permütasyon gösterimi önerilmiştir. Bu permütasyon gösterimindeki temel amaç rota uzunluklarının standart hale getirilmesidir.

Gösterim şeklinde kullanılan kukla (dummy) talep noktaları (Ø) ile tüm rotaların eş uzunluğa sahip olması sağlanmaktadır. Depo dahil tüm talep noktalarının Ø'ya olan uzaklığı 0, Ø'nın tüm talep noktalarına olan uzaklığı ise çok büyük bir sayıdır (M). Aynı zamanda Ø'nın kendine olan uzaklığı 0'dır. Bununla birlikte, Ø'nın talebi bulunmamaktadır.

Problem okuması yapıldığında bir araca en fazla kaç farklı müşterinin talebinin karşılanabileceği bilgisi elde edilmektedir. Depo ve kukla talep noktası hariç talepler küçükten büyüğe sıralanarak bir araçla en fazla kaç farklı talep noktasına hizmet edilebileceği elde edilmektedir. Elde edilen bu sayıya 2 eklenerek l sayısı elde edilmektedir. Bu değere 2 eklenmesinin sebebi AARP'ye uyumlu şekilde her rotada ilk elemanların depo son elemanların ise kukla talep noktalarından oluşturuluyor olmasıdır.

Tüm rotaların uzunluğu l kadar olmakla beraber permütasyonun toplam uzunluğuna l sayısının araç sayısıyla çarpımı sonucunda ulaşılmaktadır. Elde edilen gösterim permütasyon artı (π^+) (permutation plus) olarak isimlendirilmektedir. Depo dahil 8 büyüklüğünde 2 araca sahip bir problem için örnek π^+ Şekil 5.6'daki gibidir. Şekil incelenirken bazı konulara dikkat edilmelidir. Buna göre, talepler küçükten büyüğe sıralanarak araç kapasitesine göre bir araca en fazla 4 farklı yük yüklenebildiği varsayılmaktadır. Bu yüzden l sayısı 6 olarak belirlenmektedir. π^+ içerisinde her l kadar çözüm parçası bir rotayı temsil etmektedir. Rotalarda kapasitenin aşılmasına izin verilmemektedir. Örnekte başlangıç çözümün rassal olarak oluşturulmaktadır. Önerilen π^+'da amaç fonksiyonu $\sum_{i=0}^{n-1} um[\pi^+[i], \pi^+[i+1]]$ şeklinde hesaplanabilmektedir.

0	3	1	4	5	Ø	0	6	2	7	Ø	Ø

Şekil 5.7 *GİB için yeni çözüm gösterim şekli*

Uygulamada kukla taleple noktaları uzaklık matrisinde son sütun ve satır olarak eklenmektedir. Buna göre programlama esnasında Ø'ların yerine bu örnekte 8 sayısı yer almaktadır.

π^+'nın önerilmesindeki temel gereklilik GİB'de çözümün eş zamanlı hesaplamalar yapılabilecek hale getirilmesidir. Algoritmanın seri kısmı tasarlanırken operatörlerin paralelleştirilmesi ve GİB'ler üzerinde eş zamanlı hesaplamalar yapabilmesine uygun adımlar üzerinde çalışılmıştır. Burada amaçlanan geliştirilen seri DKA'da MİB'nin avantajlarından faydalanılırken, geliştirilen paralel DKA'da GİB'nin avantajlarından faydalanmaktadır.

Bölüm 4'te anlatılan DKA'daki gibi başlangıç çözümler MİB'de oluşturulduktan sonra GİB'ye aktarılmadan π^+ olarak düzenlenmektedir. Tüm araçlar elden geçirildikten sonra π^+'ta boş kalan noktalar Ø'lar ile doldurulmaktadır. Algoritmanın tüm operatörleri rota sonlarında kaça adet Ø yer alırsa alsın onların rotada talep noktalarının önüne geçmesini engellemektedir. MİB ve GİB arasındaki her transferde çözüm gösteriminde ilgili düzenleme gerçekleştirilmektedir.

5.5.2. Önerilen yöntemde kullanılan rota içi arama komşuluk yapılarının gib'de paralelleştirilmesi

Önerilen yöntemde kullanılan rota içi 2-opt, 3-opt, Or-opt ve Dinamik Arama komşuluklarında blok seviyesinde bir paralelleştirme gerçekleştirilmektedir. Bir diğer deyişle, her araç için eş zamanlı olarak ilgili komşulukta arama gerçekleştirilmektedir.

Tüm komşuluklarda benzer bir strateji izlenilmesi sebebiyle yalnızca 2-opt komşuluğu üzerinden GİB üzerindeki işleyiş detaylı olarak açıklanmaktadır.

2-opt komşuluğunun GİB'de paralel olarak işlenmesinde tüm rotalarda aynı anda arama yapılmaktadır. Tüm rotalarda eş zamanlı olmak üzere her bir rotada $j \geq i+2$ olmak üzere $(i, i+1)$ ve $(j, j+1)$ bağlantılarının kırılıp (i, j) ve $(i+1, j+1)$ bağlantılarının oluşturulması test edilmektedir. Uygun olanlarda gerçekleştiren $(i+1, j+1)$ arasının ters dönüştürülmesi yine eş zamanlı gerçekleştirilmektedir.

π^+'ta arama yapılırken rotaların başında yer alan depo ve en sonunda yer alan Ø değişiklik testin dışında tutulmaktadır. Bununla birlikte π^+'ta son nokta dışında da Ø yer alabilmektedir. 2-opt komşuluğuna bakılırken rota içerisindeki Ø'nın bir talep noktasının önüne geçmesi maliyet matrisinde Ø'nın talep noktalarına olan uzaklığının büyük bir sayı olması sayesinde engellenmektedir. Bunun dışında bir durum daha söz konusu olmaktadır. O da test edilirken rota içerisindeki iki Ø'nın birbirlerine olan uzaklığının 0 olması gerekçesiyle Ø'lar arasında gereksiz değişikliklerin ortaya çıkmasıdır. Bundan korunmak amacıyla $d_2 - d_1 < 0$ karşılaştırılması yapılmadan önce d_2'ye $d_2 = d_2 + (1 - sgn(d_2)) * M$ formülündeki gibi güncelleştirme getirilmektedir. d_2'nin güncellenmesi için GİB'de signum fonksiyonun direkt kullanımı olmamasından dolayı veri tepi değiştirme taktiği ile aynı sonuç elde edilmektedir.

5.5.3. Önerilen yöntemde kullanılan rotalar arası arama komşuluk yapılarının GİB'de paralelleştirilmesi

GİB üzerinde ele alınan paralel DKA'da tıpkı seri versiyonunda olduğu gibi yöntemde ele alınan rotalar arası arama komşuluk yapılarında rotalar arasındaki tüm olası değişimleri eş zamanlı olarak ortaya dökmektedir. Bu operatörlerde hem ardışık indeksli rotalara eş zamanlı bakılmakta hem de her rotadaki tüm olası değişikliklerin durumu kısmi bir paralellikle hesaplanmaktadır. Verilen indekslere göre tüm olası değişiklikler arasından en iyileri ardışık araçlar için belirlenmektedir. Bulunan en küçük maliyetler söz konusu değişikliğin uygun olup olmadığı (0,1) durumu ile çarpılmaktadır. Elde edilen son değerlerden 0'dan küçük olanlar için değişiklik yapılmakta ve bu işlem tüm aramalarda gerçekleştirilmektedir.

Önerilen paralel yöntemde araçlar arası arama esnasında (R=Rota Sayısı) $tamsayı(\frac{araç\ sayısı}{2})$ kadar karşılaştırma eş zamanlı olarak ele alınmaktadır. Bunun sebebi eş zamanlı işlemde bir aracın birden fazla araç ile karşılaştırılıp anlamlı değişikliklerin uygulanmasının mümkün olmamasıdır. Örneğin, 3 rotadan oluşan bir problem ile çalışılsın ve GİB üzerinde R1-R2, R1-R3 ve R2-R3 aramaları en iyi *bire-bir* değişikliği gerçekleştirme şeklinde uygulansın. Bu durumda, R1-R2 ve R1-R3 karşılaştırmalarında R1'den bir elemanın hem R2'den hem R3'ten bir elemanla değişim durumu ortaya çıkabilir ve algoritmanın hatalı çalışmasına sebebiyet verir. GİB literatüründe bu durum *Bank Conflict* olarak ifade edilmektedir. Bu durumla karşılaşmadan tüm araçların karşılaştırılması için önerilen tekniğin belirli kez tekrar etmesi gerekmektedir. Bu durumda da her tekrarda hangi araçların karşılaştırılacağını bildiren bir indeks kümesine ihtiyaç duyulmaktadır. İndeks kümesinde ardışık olarak adı geçen rotalar ikili karşılaştırmaya eş zamanlı olarak alınmaktadır. Her araç en fazla bir kez kullanılacak şekilde rotalar arası karşılıklı birer eleman değişikliği ele alınmıştır. Bir eş zamanlı işlemde bir aracın birden fazla ele alınamamaktadır (bkz. Şekil 5.7)

Tekrar S.						
1	1	2	3	4	5	6
2	1	3	2	4	5	6
3	1	4	2	5	3	6
4	1	5	2	6	3	4
5	1	6	2	3	4	5
6	1	2	3	5	4	6

Şekil 5.8 *İndeks belirleme örneği*

Önerilen yöntemde rotalar arası arama komşuluk yapılarında ardışık indekslere bağlı olarak bir kerede $tamsayı(\frac{araç\ sayısı}{2})$ karşılaştırma eş zamanlı olmak üzere rotalar arasındaki tüm olası değişimler ele alınmaktadır. Buna göre eğer araç sayısı çift ise verilen indeks permütasyonuna karşılık gelen tüm ardışık araçlar eş zamanlı olarak ele alınmakta, eğer araç sayısı tekse sondaki araç arama dışında tutulmaktadır. Ardışık araçlar arasında tüm değişiklikler karşılaştırılmakta ve çözümü iyileştiren en iyi değişiklik varsa gerçekleştirilmektedir. Burada ele alınan araçların indeksleri doğru tanımlandığında her tekrarda $tamsayı(\frac{araç\ sayısı}{2})$ kadar araç karşılaştırması eş zamanlı gerçekleştirilerek, yalnızca sınırlı sayıda adımda tüm araçların karşılaştırılması sağlanmaktadır.

5.5.4. Algoritmanın GİB üzerinde akışı

Bu alt bölümde paralel algoritmanın genel işleyişi hakkında açıklayıcı bilgilere yer verilmektedir. Paralel algoritmanın GİB üzerindeki genel akışı şekil 5.7'deki gibidir.

Adım 0: Parametreleri ve problemi oku.

Adım 1: MİB'de başlangıç çözümü oluştur.

Adım 2: En fazla tekrar sayısına ulaşılıp, ulaşılmadığını kontrol et, ulaşıldıysa Adım 9'a git, ulaşılmamışsa Adım 3'ten devam et.

Adım 3: Gösterim şeklini GİB'ye uygun hale getirip, GİB'ye çözümü kopyala.

Adım 4: GİB'de eş zamanlı rota içi arama DKA'yı çalıştır.

Adım 5: GİB'de eş zamanlı rotalar arası arama DKA'yı çalıştır.

Adım 6: Gösterim şeklini MİB'ye uygun hale getirip, MİB'ye çözümü kopyala.

Adım 7: MİB'de sarsma mekanizmasını çalıştır

Adım 8: Adım 2'ye git.

Adım 9: Sonuçları raporla ve bitir.

Şekil 5.9 *GİB üzerinde paralel DKA akış işlemleri*

Algoritma tasarlanırken GİB'nin etkin bir biçimde kullanılmasına özen gösterilmiştir. Bu yüzden sadece eş zamanlı hesaplamaya uygun operatörler ele alınmıştır. Bununla beraber rota içi arama DKA ve rotalar arası arama DKA'da yer alan sarsma operatörleri GİB içerisinde seri olarak gerçekleştirilmektedir. Buna ek olarak algoritmada GİB ve MİB arasında ana sarsma mekanizmasının çalıştırılması için veri transferi gerçekleştirilmektedir. Bu durum da GİB'nin etkinliğini olumlu yönde etkilememektedir. Burada belirtilmesi gereken algoritmanın başında bir kez problemin verileriyle birlikte indeksler MİB'den GİB'ye transfer edilmektedir. Algoritmanın süreci tamamlana dek transferi yapılan tek öğe çözüm, o tekrarın en iyi çözümü ve en iyi çözümdür.

5.6. Paralel DKA'dan Elde Edilen Sonuçlar

Çözüm kalitesi ve süre bakımından en iyi değerleri veren parametreler Bölüm 4'te belirlendiği için bu bölümde yapılan testlerde aynı parametrelere sadık kalınmıştır. GİB üzerindeki paralel algoritma test problemleri üzerinde 10'ar tekrar çalıştırılmasıyla elde edilen sonuçlar raporlanmıştır. Kullanılan iş istasyonunun ortamının Intel Xeon E5-2630 2.4 Ghz işlemci, 32 GB RAM ve Maxwell mimarisine uygun GTX980 ekran kartı bulunmaktadır. Bununla birlikte algoritma kodlanırken Python 2.7 kullanılmıştır. Bununla birlikte Python'da GİB kullanımı için PyCuda [82] kütüphanesinden faydalanılırken kerneller C++ olarak kodlanmıştır.

Tablo 5.3 *Literatür test problemleri için GİB üzerindeki DKA ile elde edilen sonuçlar*

Problem	n	Q	Min araç	BKS	Ortalama	St. sapma	En iyi bulunan	Ortalama süre (sn)
C1	50	160	5	416.06	418.60	3.23	416.06	26.5
C2	75	140	10	567.14	575.10	3.51	567.14	36.8
C3	100	200	8	639.74	650.47	3.32	640.16	88.5
C11	120	200	7	682.12	703.12	5.56	691.12	90.9
C12	100	200	10	534.24	537.13	1.27	534.24	44.4
F11	71	30000	4	176.99	180.88	2.46	177.15	41.2
F12	134	2210	7	769.55	804.56	6.88	780.13	145.4

Burada kullanılan test problemlerinde π^+ sayısının Maxwell mimarisine göre bir blokta bulunan en fazla thread olan 1024 sayısını geçmemesine özen gösterilmiştir. Seçilen bu test problemleriyle elde edilen çözümler ve süreleri Tablo 5.3'te yer almaktadır.

Paralel algoritmadan elde edilen sonuçların seri versiyon ile adil bir karşılaştırılması yapılabilmesi amacıyla seri programda Bölüm 4.7.5'te elde edilen sonuçlar kullanılmıştır. Bu bölümde elde edilen sonuçlar, paralel versiyonda olduğu gibi açgözlü iyileştirmeye göre değil en iyi değişikliklere yönelik gerçekleştirilmektedir.

Tablo 5.4 *Seri DKA ile paralel DKA karşılaştırılması*

Problem	**Seri DKA**			**GİB'de paralel DKA**			
	Ort. Maliyet	**St. Sapma**	**Ort. Süre (sn)**	**Ort. Maliyet**	**St. Sapma**	**Ort. Süre (sn)**	**Hızlanma Faktörü**
C1	418.55	3.36	106.3	418.60	3.23	26.5	4.01
C2	576.12	3.46	182.5	575.10	3.51	36.8	4.95
C3	650.13	3.48	400.2	650.47	3.32	88.5	4.52
C11	705.22	6.15	570.1	703.12	5.56	90.9	7.76
C12	538.83	1.85	170.1	537.13	1.27	44.4	5.92
F11	180.20	3.13	357.9	180.88	2.46	41.2	8.68
F12	805.19	6.88	945.6	804.56	6.88	145.4	6.50

Tablo 5.4'te görülen karşılaştırma sonuçlarına göre kısmi olarak GİB üzerinde çalışan paralel DKA adil karşılaştırmanın yapıldığı seri versiyonuna göre 7 test problemi için ortalama olarak 6.05 kat daha hızlı sonuçlar elde etmektedir. Burada hızlanmanın sınırlı olmasının gerekçeleri bulunmaktadır. Bunlardan ilki algoritmanın en zaman alıcı kısmı olan ana sarsma mekanizmasının seri olarak çalışması ve bu kısmın seri kısımda da en zaman alıcı bölüm olarak bulunmasıdır. Bir diğer durum ise paralel rota içi ve rotalar arası DKA çalışırken, bu kısımlarda yer alan sarsma mekanizmaları GİB üzerinde seri bir şekilde çalışmakta ve bu durum GİB'nin performansını olumsuz etkilemektedir. Bunların yanı sıra her tekrarda GİB ve MİB arasında veri transferi çözüm sürelerini olumsuz anlamda etkilemektedir.

6. SONUÇ ve ÖNERİLER

Tez çalışmasında, firmaların daha çok üçüncü parti lojistik hizmetleri kullanması ile birlikte, uygulamada yaygın şekilde ortaya çıkan açık araç rotalama problemi ele alınmıştır. Problemin heterojen filolu, karmaşık maliyetli ve iki farklı fabrikadan ürünlerin konsolidasyonu olan bir gerçek hayat örneği için karar destek sistemi geliştirilmiştir. Geliştirilen karar destek sisteminde gerçek uzaklık matrisleri elde edilmiştir. Firmadan elde edilen verilere belirli çarpanlar uygulanarak boyutları 40 ve 117 arasında değişen 8 farklı test problemi geliştirilmiştir. Karar destek sisteminin model tabanında matematiksel modellerin yanı sıra bir melez genetik algoritma önerilmiş ve test problemlerinden elde edilen sonuçlar raporlanmıştır.

Daha sonra, özellikle büyük boyutlu problemleri etkin şekilde çözmek üzere üç evreli değişken komşuluk arama algoritması önerilmiştir. Önerilen bu yöntemde dört adet rota içi ve dört adet rotalar arası olmak üzere sekiz farklı komşuluk ve sarsma stratejisi kullanılmıştır. Yöntemin performansı literatür test problemleri üzerinde test edilmiş ve başarısı karşılaştırmalı olarak raporlanmıştır. Bunun yanı sıra, aynı algoritmanın etkinliği türetilen gerçek hayat test problemleri üzerinde de vurgulanmıştır. Buna göre önerilen üç evreli değişken komşuluk arama algoritması gerçek hayat test problemlerinde melez genetik algoritmaya göre oldukça iyi sonuçlar vermektedir.

Ayrıca, önerilen değişken komşuluk arama algoritmasının eş zamanlı hesaplamaya uygun olan kısımlarının, grafik işlem birimleri üzerinde paralelleştirilmesi için farklı stratejiler tasarlanmış ve uygulanmıştır. Bu stratejilerin uygulanması ile elde edilen paralel yöntem, seri versiyonuna göre anlamlı şekilde hızlandırılmıştır. Buna göre, paralel yöntem seri versiyonuna göre ortalama 6.05 kat, en fazla 8.68 kat daha hızlı çalıştığı ortaya konulmuştur.

Çalışmada geliştirilen değişken komşuluk arama algoritması gelecek çalışmalar için diğer araç rotalama problemi türlerine ve diğer kombinatorik eniyileme problemlerine uygulanabilir durumdadır. Bununla beraber grafik işlem birimlerinde eş zamanlı komşuluk hesaplamaları için kullanılan stratejilere başka problemlerin çözümü için de elverişli haldedir. Araç rotalama problemleri gibi kapasite, araç sayısı vb. gibi kısıtları bulunmayan problemlerde, bu stratejilerin grafik işlem birimleri üzerinde uygulanmasıyla hızlanma faktörlerinin daha çarpıcı olarak ortaya çıkması beklenmektedir. Ayrıca metasezgisel algoritmaların yanı sıra kesin çözüm veren algoritmaların grafik işlem birimleri üzerinde

uygulanarak kombinatorik eniyileme problemlerinin çözülmesi konularındaki çalışmaların akademik literatürde ve uygulama tarafında yer bulabileceği öngörülmektedir.

KAYNAKÇA

[1] E.-G. Talbi, Metaheuristics, New Jersey: Wiley, 2009.

[2] H. Maaranen, K. Miettinen ve A. Penttinen, On initial populations of a genetic algorithm for continuous optimization problems, *Journal of Global Optimization,* cilt 37, pp. 405-436, 2007.

[3] E.-G. Talbi, A taxonomy of hybrid metaheuristics, *Journal of Heuristics,* cilt 8, pp. 541-564, 2002.

[4] N. Mladenovic ve P. Hansen, Variable Neighborhood Search, *Computers & Operations Research,* cilt 24, no. 11, pp. 1097-1100, 1997.

[5] G. Dantzig ve J. Ramser, The Truck Dispatching Problem, *Management Science,* cilt 6, pp. 80-91, 1959.

[6] G. Clarke ve J. Wright, Scheduling of Vehicle Routing Problem from a Central Depot to a Number of Delivery Points, *Operations Research,* cilt 6, no. 1, pp. 568-581, 1964.

[7] S. N. Kumar ve R. Panneerselvam, A Survey on the Vehicle Routing Problem and Its Variants, *Intelligent Information Management,* cilt 4, pp. 66-74, 2012.

[8] R. Lahyani, M. Khemakhem ve F. Semet, Rich vehicle routing problems: From a taxonomy to a definition, *European Journal of Operational Research,* cilt 241, pp. 1-14, 2015.

[9] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera ve A. A. Juan, Rich Vehicle Routing Problem: Survey, *ACM Computing Survey,* cilt 47, no. 2, 2015.

[10] M. U. Yıldırım ve B. Çatay, A parallel matheuristic for solving the vehicle routing problems, *Computer-based Modelling and Optimization in Transportation*, Switzerland, Springer International Publishing , 2014, pp. 477-489.

[11] P. Kalina, J. Vokrinek ve V. Marik, Agents Toward Vehicle Routing Problem With Time Windows, *Journal of Intelligent Transportation Systems,* cilt 19, pp. 3-17, 2015.

[12] T. Bektaş ve J. Lysgaard, Optimal vehicle routing with lower and upper bounds on route durations, *Network,* cilt 65, no. 2, pp. 166-179, 2015.

[13] O. Jabali, R. Leus ve T. V. Woensel, Self-imposed time windows in vehicle routing problems, *OR Spectrum,* cilt 37, pp. 331-352, 2015.

[14] R. Spliet ve G. Desaulniers, The discrete time window assignment vehicle routing problem, *European Journal of Operational Research,* cilt 244, pp. 379-391, 2015.

[15] M. McNabb, J. Weir, R. Hill ve S. Hall, Testing local search move operators on the vehicle routing problem with split deliveries and time windows, *Computers & Operations Research,* cilt 56, pp. 93-109, 2015.

[16] D. Gaur ve R. Singh, Cumulative Vehicle Routing Problem: A Column Generation Approach, *Algorithms and Discrete Applied Mathematics* , Kanpur, Springer, 2015, pp. 262-274.

[17] İ. Kara, B. Kara ve M. Yetiş, Cumulative Vehicle Routing Problems, *Vehicle Routing Problem*, Vienna, I-Tech Education and Publishing KG, 2008, pp. 85-98.

[18] J. Yang ve H. Sun, Battery swap station location-routing problem with capacitated electric vehicles, *Computers & Operations Research,* cilt 55, pp. 217-232, 2015.

[19] H. Yang, S. Yang, E. Cao, M. Lai ve Z. Dong, Electric Vehicle Route Optimization Considering Time-of-Use Electricity Price by Learnable Partheno-Genetic Algorithm, *IEEE Transactions on Smart Grid,* cilt 6, no. 2, pp. 657-666, 2015.

[20] D. Geoeke ve M. Schneider, Routing a mixed fleet of electric and conventional vehicles, *European Journal of Operational Research,* cilt 245, pp. 81-99, 2015.

[21] J. Bauer ve J. Lysgaard, The offshore wind farm array cable layout problem: a planar open vehicle routing problem, *Journal of the Operational Research Society,* cilt 66, pp. 360-368, 2015.

[22] I. Dayarian, T. G. Crainic, M. Gendreau ve W. Rei, A column generation approach for a multi-attribute vehicle routing problem, *European Journal of Operational Research,* cilt 241, pp. 888-906, 2015.

[23] I. Dayarian, T. G. Carinic, M. Gendreau ve W. Rei, A branch-and-price approach for a multi-period routing problem, *Computers & Operations Research,* cilt 55, pp. 167-184, 2015.

[24] M. Schneider, A. Stenger ve J. Hof, An adaptive VNS algorithm for vehicle routing problems with intermediate stops, *OR Spectrum,* cilt 37, pp. 353-387, 2015.

[25] J. Li, P. M. Pardalos, H. Sun, J. Pei ve Y. Zhang, Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups, *Expert Systems with Applications,* cilt 42, pp. 3551-3561, 2015.

[26] M. Avcı ve Ş. Topaloğlu, An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries, *Computers & Industrial Engineering,* cilt 83, pp. 15-29, 2015.

[27] F. G. Şahinyazan, B. Y. Kara ve M. R. Taner , Selective vehicle routing for a mobile blood donation system, *European Journal of Operational Research,* cilt 245, pp. 22-34, 2015.

[28] L. Talarico, K. Sörensen ve J. Springael, The k-dissimilar vehicle routing problem, *European Journal of Operational Research,* cilt 244, pp. 129-140, 2015.

[29] T. Dimitrakos ve E. Kyriakidis, A single vehicle routing problem with pickups and deliveries random demands and predefined customer order, *European Journal of Operational Research,* cilt 244, pp. 990-993, 2015.

[30] J. Euchi, A. Yassine ve H. Chabchoub, The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach, *Swarm and Evolutionary Computation,* cilt 21, pp. 41-53, 2015.

[31] C. Wang, D. Mu, F. Zhao ve J. Sutherland, A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows, *Computers & Industrial Engineering,* cilt 83, pp. 111-122, 2015.

[32] A. Bortfeldt, T. Hahn, D. Mannel ve L. Mönch, Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints, *European Journal of Operational Research,* cilt 243, pp. 82-96, 2015.

[33] L. Wei, Z. Zhang, D. Zhang ve A. Lim, A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints, *European Journal of Operational Research,* cilt 243, pp. 798-814, 2015.

[34] O. Domingez, D. Guimarans , A. Juan ve I. Nuez, A Biase d-Randomise d Large Neighbourhood Search for the two-dimensional Vehicle Routing Problem with Backhauls, *European Journal of Operational Research,* cilt 255, pp. 442-462, 2016.

[35] Z. Naji-Azimi, M. Salari, J. Renaud ve A. Ruiz, A practical vehicle routing problem with desynchronized arrivals to depot, *European Journal of Operational Research,* cilt 255, pp. 58-67, 2016.

[36] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal ve A. Subramanian, New benchmark instances for the Capacitated Vehicle Routing Problem, *European Journal of Operational Research,* cilt 257, pp. 845-858, 2017.

[37] J. Sze, S. Salhi ve N. Wassan, A hybridisation of adaptive variable neighbourhood search and large neighbourhood search: Application to the vehicle routing problem, *Expert Systems With Applications,* cilt 65, pp. 383-397, 2016.

[38] B. Golden, E. A. Wasil, J. P. Kelly ve I.-M. Chao, The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results, *Fleet management and logistics*, Boston, MA: Kluwer, 1998, pp. 33-56.

[39] C. Tarantilis ve C. Kiranoudis , BoneRoute: an adaptive memory-based method for effective fleet management, *Annals of Operations Research,* cilt 41, pp. 115-227, 2002.

[40] P. Toth ve D. Vigo, The granular tabu search and its application to the vehicle-routing problem, *Informs Journal on Computing,* cilt 15, no. 4, 2003.

[41] F. Li, B. Golden ve E. Wasil, Very large-scale vehicle routing: new test problems, algorithms, and results, *Computers & Operations Research,* cilt 32, pp. 1165-1179, 2005.

[42] D. Mester ve O. Braysy, Active-guided evolution strategies for large-scale capacitated vehicle routing problems, *Computers & Operations Research,* cilt 34, pp. 2964-2975, 2007.

[43] J. Kytöjoki, T. Nuortio, O. Braysy ve M. Gendreau, An efficient variable neighborhood search heuristic for very large scale vehicle routing problems, *Computers & Operations Research,* cilt 34, pp. 2743-2757, 2007.

[44] D. Sariklis ve S. Powell, A heuristic method for the open vehicle routing problem, *Journal of the Operational Research Society,* cilt 51, no. 5, pp. 564-573, 2000.

[45] J. Brandao, A tabu search algorithm for the open vehicle routing problem, *European Journal of Operational Research,* cilt 157, pp. 552-564, 2004.

[46] C. Tarantilis, D. Diakoulaki ve C. Kiranoudis, Combination of geographical information system and efficient routing algorithms for real life distribution operations, *European Journal of Operational Research,* cilt 152, pp. 437-453, 2004.

[47] C. Tarantilis, G. Ioannou, Kiranoudis C. ve G. Prastacos, A threshold accepting approach to the open vehicle routing problem, *RAIRO Operations Research,* cilt 38, pp. 345-360, 2004.

[48] C. Tarantilis, G. Ioannou , C. Kiranoudis ve G. Prastacos, Solving the open vehicle routing problem via a single parameter metaheuristic algorithm, *Journal of the Operational Research Society,* cilt 56, pp. 588-596, 2005.

[49] Z. Fu, R. Eglese ve L. Li, A new tabu search heuristic for the open vehicle routing problem, *Journal of the Operational Research Society,* cilt 56, pp. 267-274, 2005.

[50] D. Pisinger ve S. Ropke, A general heuristic for vehicle routing problems, *Computers & Operations Research,* cilt 34, pp. 2403-2435, 2007.

[51] F. Li, B. Golden ve E. Wasil, The open vehicle routing problem: Algorithms, large-scale test problems, and computational results, *Computers & Operations Research,* cilt 34, pp. 2918-2930, 2007.

[52] K. Fleszar, I. Osman ve K. Hindi, Avariable neighbourhood search for the open vehicle routing problem, *European Journal of Operational Research,* cilt 195, pp. 803-809, 2009.

[53] M. Salari, P. Toth ve A. Tramontani, An ILP improvement procedure for the Open Vehicle Routing Problem, *Computers & Operations Research,* cilt 37, pp. 2106-2120, 2010.

[54] P. Repoussis , C. Tarantilis, O. Braysy ve G. Ionannou, A hybrid evolution strategy for the open vehicle routing problem, *Computers & Operations Research,* cilt 37, pp. 443-455, 2010.

[55] E. Zachariadis ve C. Kiranoudis , An open vehicle routing problem metaheuristic for examining wide solution, *Computers & Operations Research,* cilt 37, pp. 712-723, 2010.

[56] S. Yu, C. Ding ve K. Zhu, A hybrid GA–TS algorithm for open vehicle routing optimization of coal mines material, *Expert Systems with Applications,* cilt 38, p. 10568–10573, 2011.

[57] A. Sevkli ve B. Guler, A multi-phase oscillated variable neighbourhood search algorithm fora real-world open vehicle routing problem, *Applied Soft Computing,* cilt 58, pp. 128-144, 2017.

[58] X. Li, S. Leung ve P. Tian, A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem, *Expert Systems with Applications,* cilt 39, pp. 365-374, 2012.

[59] M. Yousefikhoshbakht , F. Didehvar ve F. Rahmati, Solving the heterogeneous fixed fleet open vehicle routing problem by a combined metaheuristic algorithm, *International Journal of Production Research,* cilt 52, no. 9, pp. 2565-2575, 2014.

[60] M. Yousefikhoshbakht , A. Dolatnejad, F. Didehvar ve F. Rahmati, A Modified Column Generation to Solve the Heterogeneous Fixed Fleet Open Vehicle Routing Problem, *Journal of Engineering,* 2016.

[61] M. M. Solomon, Algorithms for the Vehicle Routing and Scheduling Problems with Time Window, *Operations Research,* cilt 35, no. 2, pp. 254-265, 1987.

[62] C. R. K. , Polynomially searchable exponential neighborhoods for sequencing problems in combinatorial optimization, Faculty of Mathematical Studies University of Southampton, Southampton, UK, 2000.

[63] E. Taillard, . P. Badeau, . M. Gendreau, . F. Guertin ve J.-Y. Potvin, A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows, *Transportation Science,* cilt 31, no. 2, pp. 101-195, 1997.

[64] S. Ropke, Heuristic and exact algorithms for vehicle routing problems, DTU, Denmark, 2006.

[65] A. Subramanian, E. Uchoa ve L. Ochi, A hybrid algorithm for a class of vehicle routing problems, *Computers & Operations Research,* cilt 40, no. 10, pp. 2519-2531, 2013.

[66] T. S. Crow, *Evolution of the Graphical Processing Unit,* University of Nevada, 2004.

[67] Developer Centers: CUDA Zone, nVIDIA, [Çevrimiçi]. Available: http://developer.nvidia.com/cuda/what-cuda. [Erişildi: 1 Ekim 2018].

[68] R. M. Aiex, S. Binato ve G. R. Mauricio, Parallel grasp with path-relinking for job shop scheduling, *Parallel Computing,* cilt 29, no. 4, pp. 393-430, 2003.

[69] E. Alba, F. Luna, A. J. Nebro ve J. M. Troya, Parallel heterogeneous genetic algorithms for continuous optimization, *Parallel Computing,* cilt 30, pp. 699-719, 2004.

[70] D. Andre ve J. R. Koza, Parallel genetic programming: a scalable implementation using the transputer network architecture, *Advances in genetic programming: volume 2,* pp. 317-337, 1996.

[71] T. G. Crainic ve M. Genderau, Cooperative parallel tabu search for capacitated network design., *J. Heuristics,* cilt 8, no. 6, pp. 601-627, 2002.

[72] M. Randall, A parallel implementation of ant colony optimization, *J. Parallel Distrib. Comput.,* cilt 62, no. 9, pp. 1421-1432, 2002.

[73] E. Özçetin, Karesel Atama Problemi için Grafik İşlem Birimleri Üzerinde Paralel Bir Evrimsel Algoritma, Eskişehir, 2013.

[74] S. Tsutsui ve N. Fujimoto, Solving Quadratic Assignment Problems by Genetic Algorithms with GPU Computation: A Case Study, *GECCO*, Montreal, 2009.

[75] S. Tsutsui ve N. Fujimoto, Fast QAP Solving by ACO with 2-opt Local Search on a GPU, *IEEE Section Congres*, San Francisco, 2011.

[76] M. Czapinski, An effective Parallel Multistart Tabu Search for Quadratic Assignment Problem CUDA platform, *J. Parallel Distrib. Comput. ,* cilt 73, pp. 1461-1468, 2013.

[77] J. M. Cecilia, J. M. Garcia, A. Nisbet, M. Amos ve M. Ujaldon, Enhancing data parallelism for Ant Colony Optimization on GPUs, *J. Parallel Distrib. Comput. ,* cilt 73, pp. 42-51, 2013.

[78] A. Delevacq, P. Delisle, M. Gravel ve M. Krajecki, Parallel Ant Colony Optimization on Graphics Processing Units, *J. Parallel Distrib. Comput. ,* cilt 73, pp. 52-61, 2013.

[79] C. Shulz, Efficient local search on the GPU—Investigations on the vehicle routing problem, *J. Parallel Distrib. Comput.,* pp. 14-31, 2013.

[80] C. Groer, B. Golden ve E. Wasil, A Parallel Algorithm for the Vehicle Routing Problem, *INFORMS Journal on Computing,* cilt 23, pp. 315-330, 2011.

[81] J. Szymon ve Z. Dominik, Solving Multi-criteria Vehicle Routing Problem by Parallel Tabu Search on GPU, *Procedia Computer Science,* cilt 18, pp. 2529-2532, 2013.

[82] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov ve A. Fasih, PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation, *Parallel Computing,* cilt 38, no. 3, pp. 157-174, 2012.

[83] N. Melab, T.-V. Luong, K. Boufaras ve E.-G. Talbi, Towards ParadisEO-MO-GPU: a Framework for GPU-based Local Search Metaheuristics, *11th International Work-Conference on Artificial Neural Networks*, Malaga, 2011.

[84] T. V. Luong, N. Melab ve E.-G. Talbi, Parallel Hybrid Evolutionary Algorithms on GPU, *IEEE World Congress on Computational Intelligence*, Barcelona, 2010.

[85] Modified genetic algorithms for solving fuzzy flow shop scheduling problems and their implementation with CUDA, *Expert Systems with Applications,* cilt 39, pp. 4999-5005, 2012.

[86] M. Czapiński ve S. Barnes, Tabu Search with two approaches to parallel flowshop evaluation on CUDA platform, *J. Parallel Distrib. Comput.,* cilt 71, pp. 802-811, 2011.

[87] J. C. Bean, Genetic Algorithms and Random Keys for Sequencing and Optimization, *ORSA Journal on Computing,* cilt 6, no. 2, pp. 154-160, 1994.

ÖZGEÇMİŞ

Adı – Soyadı: Erdener ÖZÇETİN

Yabancı dil: İngilizce, YDS 2014: 90

e-posta: erdenerozcetin@hitit.edu.tr

Doğum Yeri ve Yılı: Ankara/ 1985

Eğitim ve Mesleki Bilgisi

- **Yüksek Lisans:** 2009-2013, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Tez adı: Karesel Atama Problemi İçin Grafik İşlem Birimleri Üzerinde Paralel Bir Evrimsel Algoritma (2013)
- **Lisans:** 2004-2019, Anadolu Üniversitesi, Fen Fakültesi, İstatistik Bölümü (%30 İngilizce)
- **Lisans-yandal:** 2006-2009, Anadolu Üniversitesi, İktisadi ve İdari Bilimler Fakültesi/İşletme Bölümü/İşletme Pr. (İngilizce)
- **Görev:** Araştırma Görevlisi, 2016- Devam ediyor, Hitit Üniversitesi, Mühendislik Fakültesi Endüstri Mühendisliği Bölümü
- **Görev:** Araştırma Görevlisi, 2010- 2016, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı
- **Görev:** Araştırma Görevlisi, 2009- 2010, Hitit Üniversitesi, Mühendislik Fakültesi Endüstri Mühendisliği Bölümü

Projelerde Yaptığı Görevler

- Büyük Boyutlu Araç Rotalama Problemlerinin Çözümü İçin Paralel Algoritma Tasarımı ve Uygulaması, Yükseköğretim Kurumları tarafından destekli bilimsel araştırma projesi, Araştırmacı, 2016- Devam ediyor (ULUSAL)
- 1505 Eczacıbaşı Yapı Gereçleri A Ş İçin Ürün Dağıtımında Kullanılan Araç Rotalarının Optimizasyonu 5140008, Tübitak Projesi, Bursiyer, 2014-2018 (ULUSAL)
- Veri Madenciliği ve Kombinatorik Optimizasyon Problemlerinin Çözümü için Algoritmaların Geliştirilmesi ve Uygulamalar, Yükseköğretim Kurumları tarafından destekli bilimsel araştırma projesi, Araştırmacı, 03/06/2015 - 25/11/2016 (ULUSAL)

- Yükseköğretim Kurumlarında Karşılaşılan Ders Çizelgeleme Problemi İçin Web Temelli Bir Çözüm Yaklaşımı, Yükseköğretim Kurumları tarafından destekli bilimsel araştırma projesi, Araştırmacı, , 04/04/2014 - 26/02/2016 (ULUSAL)
- Kombinatorik Optimizasyon Problemleri için Geliştirilen Metasezgisel Algoritmaların Paralelleştirilmesi, BAP, Araştırmacı, 2011-2012 (ULUSAL)

Eserler

- Özçetin Erdener, Öztürk Gürkan (2018). A Parallel Iterated Local Search Algorithm on GPUs for Quadratic Assignment Problem. International Journal of Engineering Technologies, 4(2), 123-127. (Yayın No: 4521969)
- Özçetin Erdener, Öztürk Gürkan (2016). A Hybrid Genetic Algorithm for the Quadratic Assignment Problem on Graphics Processing Units. Anadolu University Journal of Science and Technology-A Applied Sciences and Engineering, 17(1), 167-180., Doi: 10.18038/btda.15399 (Yayın No: 2959463)
- Özçetin Erdener (2016). Yükseköğretimde Etkili Teknoloji Kullanımı. ISVET2016 (Özet Bildiri/Sözlü Sunum)(Yayın No:2959747)
- Özçetin Erdener, Öztürk Gürkan, Kamışlı Öztürk Zehra, Kasımbeyli Refail, Kasımbeyli Nergiz (2015). A Mathematical Model for the Real Life Open Vehicle Routing Problem. 27th Conference on Operational Research (Özet Bildiri/)(Yayın No:2013585)
- Özçetin Erdener,Öztürk Gürkan (2014). A Parallel Iterated Local Search Algorithm On GPUs For Quadratic Assignment Problem. IFORS 2014 (Özet Bildiri/)(Yayın No:2014173)
- Kamışlı Öztürk Zehra, Sağır Müjgan, Özçetin Erdener, Kasımbeyli Nergiz, Alegöz Mehmet (2014). A Real Life Multi Objective Course Timetabling Model With Anp And Conic Scalarization Process. ISAHP 2014 (/)(Yayın No:2015039)
- Özçetin Erdener,Öztürk Gürkan (2014). A Parallel K means Algorithm on GPUs. ISDS 2014 (Özet Bildiri/)(Yayın No:2015201)
- Özçetin Erdener, Öztürk Gürkan (2013). A Parallel Hybrid Genetic Algorithm for Quadratic Assignment Problem on GPUs. European Conference on Operational Research ,EURO MMXIII (/)(Yayın No:598732)
- Kamışlı Öztürk Zehra, Özçetin Erdener, Öztürk Gürkan, Kasımbeyli Nergiz, Kasımbeyli Refail (2013). A Real Life Open Vehicle Routing Problem With an

Alternative Cost Structure. 26th EURO-INFORMS Conference (/)(Yayın No:598731)

- Özçetin Erdener, Öztürk Gürkan (2013). Using GPUs for Combinatorial Optimization Problems. Yöneylem Araştırması ve Endüstri Mühendisliği 33. Ulusal Kongresi (YAEM 2013) ve Uluslararası IIE KonferansI (Özet Bildiri/)(Yayın No:598733)
- Özçetin Erdener, Öztürk Gürkan (2013). A Tabu Search Algorithm for Multi Objective Open Vehicle Routing Problem. 22nd International Conference on Multiple Criteria Decision Making (Özet Bildiri/)(Yayın No:598734)
- Zararsız Gökmen, Özçetin Erdener, İçöz Cenk (2012). Finding Best Curve Fitting Model of Data A Rank Aggregation Approach. 8th International Symposium of Statistics (Özet Bildiri)(Yayın No:643647)
- Özçetin Erdener, Öztürk Gürkan (2012). A Parallel Algorithm for Solving Vehicle Routing Problem on GPUs. European Conference on Operational Research, EURO XXV (Yayın No:598735)
- Özçetin Erdener, Öztürk Gürkan (2012). A Parallel Algorithm for Solving Quadratic Assignment Problem on GPUs. ECCO 2012- 25th Conference of European Chapter on Combinatorial Optimization (Yayın No:598736)
- Lojistikte Coğrafi Bilgi Sistemleri Kullanımı, Bölüm adı:(Lojistik Bilgi Sistemleri) (2016), Özçetin Erdener, Anadolu Üniversitesi, Editör:Güney Yücel, Avdan Uğur, Basım sayısı:1, ISBN:978-975-06-2016-4, Türkçe(Ders Kitabı), (Yayın No: 3283158)
- Lojistikte Coğrafi Bilgi Sistemleri Kullanımı, Bölüm adı:(Lojistikte Coğrafi Bilgi Sistemleri Kullanımı) (2016), Özçetin Erdener, Acar Ilgın, Anadolu Üniversitesi, Editör:Güney Yücel, Avdan Uğur, Basım sayısı:1, ISBN:978-975-06-2016-4, Türkçe(Ders Kitabı), (Yayın No: 3194296)
- Lojistikte Coğrafi Bilgi Sistemleri Kullanımı, Bölüm adı:(Lojistik Ağ Tasarımı) (2016), Özçetin Erdener, Anadolu Üniversitesi, Editör:Güney Yücel, Avdan Uğur, Basım sayısı:1, ISBN:978-975-06-2016-4, Türkçe(Ders Kitabı), (Yayın No: 3283214)
- Encyclopedia of Business Analytics and Optimization, Bölüm adı:(Visualization of High Dimensional Data) (2014). Zararsız Gökmen,İçöz Cenk,Özçetin Erdener,

IGI Global, Editör:John Wang, Basım sayısı:1, ISBN:9781466652026, İngilizce(Ansiklopedi Maddesi), (Yayın No: 2012859)

- Kamışlı Öztürk Zehra, Kasımbeyli Nergiz, Sağır Müjgan, Soyuöz Acar Müge, Özçetin Erdener, Alegöz Mehmet, Ceylan Gürhan (2016). Kullanıcı Tercihlerinin Dikkate Alınması Durumunda Üniversite Ders Çizelgeleme Problemi. Endüstri Mühendisliği Dergisi, 27(1), 2-16. (Kontrol No: 2959551)
- Özçetin Erdener,Öztürk Gürkan (2018). Araç Rotalama Problemi'nin Grafik İşlem BirimleriÜzerinde Ele Alınması. 38. Yöneylem Araştırması ve Endüstri Mühendisliği Ulusal Kongresi (YAEM) (Özet Bildiri/Sözlü Sunum)(Yayın No:4522057)
- Özçetin Erdener,Kamışlı Öztürk Zehra (2015). Optimizasyon Yazılımlarda Çözücü Ayarlarının Önemi. YAEM2015 (Özet Bildiri/)(Yayın No:2013762)
- Zararsız Gökmen, İçöz Cenk, Özçetin Erdener (2012). A Voting Approach for SVM Kernel Selection in Gene Expression Profiling. 14. National Biostatistic Congress (Yayın No:678225)
- Özçetin Erdener (2011). Yolcu Uçaklarının Kokpitlerinde Ergonomik Faktörlerin Uçak Kazaları Açısından İncelenmesi. 17. Ulusal Ergonomi Kongresi (/)(Yayın No:679029)
- Özçetin Erdener, Öztürk Gürkan (2011). Sinterlemede Kullanılacak Malzemelerin Mikro Boyutta Simülasyonu. Yöneylem Araştırması ve Endüstri Mühendisliği 31. Ulusal Kongresi, YA/EM 2011 (Özet Bildiri/)(Yayın No:598737)
- Özçetin Erdener, Çayır Beyzanur, Aras Nil, Öztürk Gürkan (2010). Akademik Personelin Ofislerinin Klimatik Açıdan Analizi. 16. Ulusal Ergonomi Kongresi (Tam Metin Bildiri/)(Yayın No:653542)
- Çayır Beyzanur, Özçetin Erdener, Aras Nil (2010). Akademik Personelin Çalışma Masalarının Antropometrik Açıdan Analizi. 16. Ulusal Ergonomi Kongresi (Yayın No:679743)