EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

MELEZ METASEZGİSEL ALGORİTMALAR İÇİN PARAMETRE KONTROLÜ

Osman GÖKALP

Tez Danışmanı: Prof. Dr. Aybars UĞUR

Bilgisayar Mühendisliği Anabilim Dalı

Sunuş Tarihi: 18.06.2018

Bornova-İZMİR

2018

OSMAN GÖKALP tarafından DOKTORA tezi olarak sunulan "MELEZ METASEZGİSEL ALGORİTMALAR İÇİN PARAMETRE KONTROLÜ" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 18 HAZİRAN 2018 tarihinde yapılan tez savunma sınavında aday oybirliği/~~oyçokluğu~~ ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Prof. Dr. Aybars UĞUR**	.....
Raportör Üye:	**Prof. Dr. M. Serdar KORUKOĞLU**	.....
Üye	**: Dr. Öğr. Üyesi Korhan KARABULUT**	.....
Üye	**: Doç. Dr. Vecdi AYTAÇ**	.....
Üye	**: Dr. Öğr. Üyesi Serkan BALLI**	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Doktora Tezi olarak sunduğum "MELEZ METASEZGİSEL ALGORİTMALAR İÇİN PARAMETRE KONTROLÜ" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

18 / 06 / 2018

İmzası

Osman GÖKALP

ÖZET

MELEZ METASEZGİSEL ALGORİTMALAR İÇİN PARAMETRE KONTROLÜ

GÖKALP, Osman

Doktora Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Prof. Dr. Aybars UĞUR
Haziran 2018, 97 sayfa

Bir metasezgiselin en az bir algoritma ile birleştirilmesi melez metasezgisel olarak tanımlanmaktadır. Parametre değerlerinin algoritmanın çalışma zamanında belirlenmesi ise parametre kontrol olarak adlandırılmaktadır. Bu tez çalışmasının amacı melez metasezgiseller ile parametre kontrol yönteminin birlikte uygulanmasını sağlayarak özgün eniyileme yöntemleri geliştirmektir. Bu kapsamda, sürekli eniyileme ve kombinasyonel eniyileme olmak üzere iki temel alanda yeni algoritmalar geliştirilmiştir.

Sürekli eniyileme için sınır kısıtlamalı problemler üzerinde çalışılmış ve bu alanda yaygın olarak kullanılan ABC, DE ve PSO metasezgiselleri yüksek seviyeli ve nöbetleşe çalışma prensibi ile melezleştirilmiştir. Geliştirilen bu melez algoritmadaki metasezgisellerin çalışma sırası, bir parametre olarak, UCB seçim algoritması tabanlı uyarlanabilir parametre kontrolü ile belirlenmiştir. Gerçekleştirilen deneysel çalışma, algoritma seviyesinde metasezgisel seçiminin parametre kontrolü ile yapılabilirliğini göstermektedir. Kombinasyonel eniyileme alanında ise Kapasiteli Araç Rotalama Problemi için ILS ve VND metasezgisellerini alt seviyeli ve takım çalışması prensibi ile melezleştiren bir algoritma geliştirilmiştir. Bu algoritmada, kabul fonksiyonu için uyarlanabilir parametre kontrolü uygulanmıştır. Gerçekleştirilen deneysel çalışma, önerilen yöntemde parametre kontrolü kullanılmasının istatistiksel olarak anlamlılığını göstermektedir. Ayrıca, ölçülen çalışma süreleri ve ortalama hata yüzdeleri literatürde bu alanda yer alan önemli algoritmalar ile yarışabilir düzeydedir.

Anahtar sözcükler: Metasezgisel, melezleştirme, parametre kontrolü, sürekli eniyileme, kombinasyonel eniyileme, kapasiteli araç rotalama problemi.

ABSTRACT

PARAMETER CONTROL FOR HYBRID METAHEURISTIC ALGORITHMS

GÖKALP, Osman

PhD in Computer Engineering
Supervisor: Prof. Dr. Aybars UĞUR
June 2018, 97 pages

A hybrid metaheuristic is defined as combining a metaheuristic with at least one algorithm. If parameter values are determined at run time of the algorithm, it is called parameter control. The aim of this thesis is to develop novel optimization methods by using hybrid metaheuristics and parameter control methods together. In this context, methods have been developed in two main optimization domains, namely continuous optimization and combinatorial optimization.

For continuous optimization, bound constrained problems have been studied, and ABC, DE, and PSO metaheuristics, which are widely used in this area, were hybridized with high-level and relay principle of operation. As a parameter, the running order of these metaheuristics in this developed hybrid algorithm were determined by the adaptive parameter control based on the UCB selection algorithm. The experimental study shows the feasibility of metaheuristic selection with parameter control at algorithm level. For combinatiorial optimization domain, an algorithm that hybridizes ILS and VND metaheuristics with low-level and teamwork principle of operation was developed for Capacitated Vehicle Routing Problem. In this algorithm, an adaptive parameter control has been applied for acceptance function. The experimental study shows that the usage of parameter control in the proposed method is statistically significant. In addition, measured run-time and mean error rates are comparable to state-of-the-art algorithms in the literature.

Keywords: Metaheuristic, hybridization, parameter control, continuous optimization, combinatorial optimization, capacitated vehicle routing problem.

TEŞEKKÜR

Tez çalışmam boyunca değerli görüş ve önerileriyle doğru yönde ilerlememi sağlayan, her konuda yardım ve ilgisini esirgemeyen değerli hocam ve tez danışmanım Prof. Dr. Aybars UĞUR'a teşekkürlerimi sunarım. Tez izleme komitesi toplantılarındaki kıymetli görüş ve önerilerle çalışmama katkılar sağlayan Prof. Dr. M. Serdar KORUKOĞLU ve Dr. Öğr. Üyesi Korhan KARABULUT hocalarıma teşekkür ederim. Ayrıca, çalışmalarım süresince benden desteğini esirgemeyen bütün hocalarım ve çalışma arkadaşlarıma teşekkürü bir borç bilirim.

BİDEB 2211-A Genel Yurt İçi Doktora Burs Programı kapsamında beni destekleyen Türkiye Bilimsel ve Teknolojik Araştırma Kurumu'na (TÜBİTAK) teşekkür ederim.

Her zaman olduğu gibi tüm öğrenim hayatım boyunca beni destekleyen sevgili annem Sevim GÖKALP'e, babam Fuat GÖKALP'e ve kardeşim Alev ALVER'e teşekkürü bir borç bilirim.

Evliliğimiz boyunca ve onun öncesinde, her konuda olduğu gibi akademik yaşamımda da bana sınırsız destek olan sevgili eşim Özge GÖKALP'e sonsuz teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET vii

ABSTRACT ix

TEŞEKKÜR xi

ŞEKİLLER DİZİNİ xvii

ÇİZELGELER DİZİNİ xviii

KISALTMALAR DİZİNİ xxi

1. GİRİŞ 1

2. METASEZGİSEL ALGORİTMALAR 4

2.1 Tek Çözüm Tabanlı Metasezgisel Algoritmalar 4

2.1.1 Benzetimli tavlama 4

2.1.2 Tabu arama 5

2.1.3 Değişken komşu iniş 6

2.1.4 Değişken komşuluk arama 7

2.1.5 Yinelemeli yerel arama 8

2.1.6 Açgözlü rastgele uyarlanabilir arama prosedürü 9

2.2 Topluluk Tabanlı Metasezgisel Algoritmalar 10

2.2.1 Evrimsel algoritmalar 10

İÇİNDEKİLER (devam)

Sayfa

2.2.2 Sürü zekası 15

2.3 Melez Metasezgisel Algoritmalar 20

3. PARAMETRE KONTROL YÖNTEMLERİ 22

3.1 Kategorik Parametreler İçin Kullanılan Parametre Kontrol Yaklaşımları 24

3.1.1 Kredi atama 24

3.1.2 Operatör seçimi 28

3.2 Sayısal Parametreler İçin Kullanılan Parametre Kontrol Yaklaşımları 30

3.2.1 Başarılı değerlerin ortancası yaklaşımı 31

3.2.2 Uyarlanabilir kabul fonksiyonu 32

4. ÖNCEKİ ÇALIŞMALAR 33

5. YÜKSEK SEVİYELİ VE UYARLANABİLİR METASEZGİSEL SEÇİMİ: HAMS_ABCxDExPSO 37

5.1 Sınır Kısıtlamalı ve Tek Amaçlı Sürekli Eniyileme Problemleri 37

5.2 Algoritma Tasarımı 37

6. ÇOK BAŞLANGIÇLI VE UYARLANABİLİR KABUL FONKSİYONLU ILS-VND MELEZ METASEZGİSELİ: MA_ILSxVND 41

6.1 Kapasiteli Araç Rotalama Problemi 41

6.2 CVRP Çözümlerinin Temsili 41

İÇİNDEKİLER (devam)

Sayfa

6.3 Algoritma Tasarımı 43

6.3.1 Başlangıç çözümü oluşturma aşaması 45

6.3.2 Sarsım aşaması 45

6.3.3 Yerel arama aşaması 47

6.3.4 Uyarlanabilir kabul fonksiyonu 50

6.3.5 Çok başlangıçlı strateji 50

7. DENEYSEL ÇALIŞMALAR 51

7.1 HAMS_ABCxDExPSO İçin Deneysel Çalışma 51

7.1.1 Kullanılan veri seti 51

7.1.2 Deneysel kurulum 52

7.1.3 Deneysel sonuçlar 54

7.1.4 Diğer algoritmalar ile karşılaştırma 61

7.1.5 Sonuç ve değerlendirme 63

7.2 MA_ILSxVND İçin Deneysel Çalışma 64

7.2.1 Kullanılan veri setleri 64

7.2.2 Deneysel kurulum 69

7.2.3 Deneysel sonuçlar 70

İÇİNDEKİLER (devam)

Sayfa

7.2.4 Diğer algoritmalar ile karşılaştırma 81

7.2.5 Sonuç ve değerlendirme 86

8. SONUÇ 87

KAYNAKLAR DİZİNİ 89

ÖZGEÇMİŞ 97

ŞEKİLLER DİZİNİ

Şekil Sayfa

2.1. Çaprazlama örneği. 12

2.2. GP'da kromozom örneği. 13

2.3. Melez metasezgisellerin taksonomisi (Talbi, 2002). 21

3.1. Parametre belirleme yöntemleri taksonomisi. 23

5.1.Geliştirilen HAMS_ABCxDExPSO mimarisinin genel yapısı. 38

6.1. CVRP çözümlerinin temsili a) örnek bir CVRP çözümü, b) eşdeğer büyük tur temsili, c) bir diğer eşdeğer büyük tur temsili. 42

6.2. CE örneği (Taillard et al.'dan 1997). 46

6.3. CE özel durumları a) 2-opt*, b) Or-opt (Taillard et al.'dan 1997). 46

6.4. VND içerisinde kullanılan komşuluk yapıları için kenar değişim örnekleri a) 2-opt*, b) 2-opt, c) swap, d) Or-opt, e) string-exchange (ters çevrim ile) (Irnich et al.'dan, 2006). 48

7.1. HAMS_ABCxDExPSO algoritmasının artan problem boyutuna göre ABC, DE ve PSO karşısındaki kazanç sayıları……...59

7.2. MA_ILSxVND algoritmasının CMT veri seti üzerinde farklı iterasyon sayıları ile çalıştırılması sonucu elde edilen ortalama süre ve OHY değişimi…….72

ÇİZELGELER DİZİNİ

Çizelge Sayfa

7.1. CEC'17 veri seti içerisinde yer alan fonksiyonlar ve özellikleri (1. Bölüm). 51

7.2. CEC'17 veri seti içerisinde yer alan fonksiyonlar ve özellikleri (2. Bölüm). 52

7.3. HAMS_ABCxDExPSO algoritmasının iRace ile ayarlanmış parametre değerleri. 53

7.4. HAMS_ABCxDExPSO algoritmasının D=10 için CEC'17 veri seti üzerinde çalıştırılması ile elde edilen hata değerleri. 54

7.5. HAMS_ABCxDExPSO algoritmasının D=30 için CEC'17 veri seti üzerinde çalıştırılması ile elde edilen hata değerleri. 55

7.6. HAMS_ABCxDExPSO algoritmasının D=50 ve D=100 için CEC'17 veri seti üzerinde çalıştırılması ile elde edilen hata değerleri. 56

7.7. D=10 ve D=30 için ABC, DE ve PSO algoritmalarının ortalama hata değerleri ve fonksiyon bazında HAMS_ABCxDExPSO algoritması ile kıyaslanması..... 57

7.8. D=50 ve D=100 için ABC, DE ve PSO algoritmalarının ortalama hata değerleri ve fonksiyon bazında HAMS_ABCxDExPSO algoritması ile kıyaslanması..... 58

7.9. Önerilen algoritmanın ABC, DE ve PSO algoritmaları ile veri setinin tümü için Friedman skorları ve Wilcoxon işaret sırası testi kullanılarak karşılaştırılması. 59

7.10. AMS_ ABCxDExPSO algoritmasının rastgele seçim ile çalıştırılması sonucu elde edilen ortalama hata değerleri ve UCB seçimli ilk durumu ile kıyaslanması. 60

7.11. SaDE, MEABC ve CMA-ES algoritmalarının ortalama hata değerleri ve fonksiyon bazında HAMS_ABCxDExPSO algoritması ile kıyaslanması. 62

ÇİZELGELER DİZİNİ (devam)

Çizelge Sayfa

7.12. HAMS_ABCxDExPSO algoritmasının SaDE, MEABC ve CMA-ES algoritmaları ile veri setinin tümü için Friedman skorları ve Wilcoxon işaret sırası testi kullanılarak karşılaştırılması. 62

7.13. CMT veri seti problem örnekleri ve özellikleri (Uchoa et al.'dan, 2017). ... 64

7.14. Golden et al. veri seti problem örnekleri ve özellikleri (Uchoa et al.'dan, 2017). 65

7.15. Uchoa et al. veri seti ve özellikleri (1. Bölüm). 67

7.16. Uchoa et al. veri seti ve özellikleri (2. bölüm). 68

7.17. MA_ILSxVND algoritmasının iRace ile ayarlanmış parametre değerleri. .. 69

7.18. MA_ILSxVND algoritmasının CMT veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar. 70

7.19. MA_ILSxVND algoritmasının CMT veri seti üzerinde artan iterasyon sayısı ile çalıştırılması karşısındaki davranışı. 71

7.20. MA_ILSxVND algoritmasının CMT veri seti üzerinde farklı bileşen kombinasyonlarına göre karşılaştırılması (40000 iterasyon). 73

7.21. CMT veri seti için MA_ILSxVND, ILSxVND, M_ILSxVND ve A_ILSxVND algoritmaları arasında gerçekleştirilen Wilcoxon işaret sırası testi ile elde edilen p değerleri. 74

7.22. MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar. 74

7.23. MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde artan iterasyon sayısı ile çalıştırılması karşısındaki davranışı. 75

ÇİZELGELER DİZİNİ (devam)

Çizelge Sayfa

7.24. MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde farklı bileşen kombinasyonlarına göre karşılaştırılması (40000 iterasyon)..... 76

7.25. Golden et al. veri seti için MA_ILSxVND, ILSxVND, M_ILSxVND ve A_ILSxVND algoritmaları arasında gerçekleştirilen Wilcoxon işaret sırası testi ile elde edilen p değerleri..... 77

7.26. MA_ILSxVND algoritmasının Uchoa et al. veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar (1. Bölüm). 79

7.27. MA_ILSxVND algoritmasının Uchoa et al. veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar (2. Bölüm). 80

7.28. Literatür karşılaştırmalarında yer alan algoritmalara ait bilgiler. 82

7.29. MA_ILSxVND algoritmasının literatürde yer alan diğer önemli algoritmalar ile CMT veri seti üzerinde karşılaştırılması. 83

7.30. MA_ILSxVND algoritmasının literatürde yer alan diğer önemli algoritmalar ile Golden et al. veri seti üzerinde karşılaştırılması..... 84

7.31. MA_ILSxVND algoritmasının diğer algoritmalar ile Uchoa et al. veri seti üzerinde karşılaştırılması. 85

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
ABC	Yapay Arı Kolonisi
ACO	Karınca Kolonisi Eniyilemesi
AOS	Uyarlanabilir Operatör Seçimi
AP	Uyarlanabilir Takip
AUC	Eğri Altında Kalan Alan
CE	CROSS Exchange
CVRP	Kapasiteli Araç Rotalama Problemi
CW	Clarke and Wright Sezgiseli
DE	Diferansiyel Gelişim
EA	Evrimsel Algoritmalar
ES	Evrimsel Strateji
GA	Genetik Algoritmalar
GP	Genetik Programlama
GRASP	Açgözlü Rastgele Uyarlanabilir Arama Prosedürü
ILS	Yinelemeli Yerel Arama
PM	Olasılık Eşleme
PSO	Parçacık Sürü Eniyilemesi
RCL	Sınırlandırılmış Aday Listesi
SA	Benzetimli Tavlama

KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
SI	Sürü Zekası
SR	Sıraların Toplamı
TS	Tabu Arama
VND	Değişken Komşu İniş
VNS	Değişken Komşuluk Arama

1. GİRİŞ

Metasezgiseller, eniyileme problemlerinin çözümünde yaygın olarak ve başarı ile uygulanan algoritmalardandır. Adında geçen "meta" ön eki, "üstünde, ötesinde, ardında" gibi anlamlar[1] katarken, sezgisel kelimesi, "deneme yanılma veya sıkı olmayan bir şekilde tanımlanmış kurallar ile bir çözüme ilerlemek" anlamına[2] gelmektedir. Bu bakımdan metasezgiseller, sezgiselleri yöneten, düzenleyen veya oluşturan bir konumda yer almakta ve problemden bağımsız yapısıyla çok çeşitli eniyileme problemlerine uyarlanabilmektedir. Yaygın olarak kullanılan metasezgisellerin bazıları arasında Genetik Algoritmalar (Holland, 1975), Karınca Kolonisi Eniyilemesi (Dorigo, 1992; Dorigo et al., 1999), Benzetimli Tavlama (Kirkpatrick et al., 1983), Tabu Arama (Glover, 1986), Parçacık Sürü Eniyilemesi (Eberhart, 1995) ve Genetik Programlama (Koza, 1992) yer almaktadır.

Eniyileme problemlerinde amaç, alternatif çözümler arasından en iyisini seçmek olup sürekli ve ayrık olmak üzere iki genel kategori altında incelenebilmektedir. Sürekli yapıya sahip problemlerde eniyilenecek problemin çözümü reel sayılardan oluşmaktadır. Bu nedenle, arama uzayında sonsuz sayıda alternatif çözüm bulunmaktadır. Bu tip problemlere örnek olarak mühendislik tasarım problemleri (He and Wang, 2007) verilebilmektedir. Ayrık problemlerde ise çözümler genellikle permütasyonlar ve kombinasyonlardan oluştuğu için sonlu sayılıdır. Yaygın olarak bilinen ayrık eniyileme problemlerinden bazılarına örnek olarak Gezgin Satıcı Problemi, Sırt Çantası Problemi ve Araç Rotalama Problemi verilebilmektedir.

Eniyileme için "No Free Lunch" (NFL) (Wolpert and Macready, 1997) teoremine göre, herhangi iki algoritmanın ortalama performansı, tüm eniyileme problemleri hesaba katıldığında birbirine eşit olmaktadır. Buradan çıkarılabilecek en önemli sonuç, tüm eniyileme problemlerini başarıyla çözen tek bir algoritmanın mümkün olmadığıdır. Bu nedenle, farklı yönlerden üstünlükleri bulunan çeşitli eniyileme algoritmalarının uygun yöntemlerle bir araya getirilerek melez yapılar geliştirilmesi önemli olmaktadır.

Metasezgisel algoritmaların başarısını doğrudan etkileyen faktörlerin arasında parametre seçimi bulunmaktadır. Bu nedenle, çözülecek probleme göre uygun olarak parametre değerlerinin belirlenmesi gerekmektedir. Parametre belirleme

[1] https://www.etymonline.com/word/meta- (Erişim tarihi: 12.03.2018)
[2] https://en.oxforddictionaries.com/definition/heuristic (Erişim tarihi: 12.03.2018)

işlemi eğer eniyileme işleminden önceden yapılıyorsa buna parametre ayarlama adı verilmektedir. Parametre ayarlamanın dezavantajlarından birisi algoritma tasarımcısı tarafından her yeni problem için tekrar ve dikkatlice yapılmasını gerektirmesidir. Bu durum algoritmanın kullanım kolaylığını azaltmaktadır. Bir diğer dezavantajı eniyileme problemlerinin dinamik yapısından kaynaklanmaktadır. Öyle ki, eniyileme sürecinin başlarında ihtiyaç duyulan uygun parametre değerleri ile ilerleyen zamanlarda beklenilen parametre değerleri farklılık gösterebilmektedir. Bu noktada, parametre kontrol adı verilen ve parametre değerlerini eniyileme algoritması çalışırken güncelleyen yöntemler önem kazanmaktadır. Parametre kontrol yöntemleri, eniyileme sürecinden geribildirimler toplayarak parametre değerlerini en uygun düzeyde tutmayı hedeflemektedir.

Tez projesinin amacı, eniyileme problemlerini çözümüne yönelik metasezgisel algoritmaların melezleştirilmesini temel alan ve aynı zamanda parametre kontrol içeren yöntemler geliştirmektir. Melezleştirme ve parametre kontrol yaklaşımlarının beraber kullanımının, geliştirilen algoritmalara avantaj sağlayacağı düşünülmektedir. Bu kapsamda, bir adet sürekli eniyileme alanında, bir adet de ayrık eniyileme alanında olmak üzere iki örnek çalışma gerçekleştirilmiştir.

Sürekli eniyileme alanında geliştirilen melez yöntem, bu alanda en yaygın kullanılan algoritmalar arasında olan Diferansiyel Gelişim (Storn and Price, 1997), Yapay Arı Kolonisi (Karaboga, 2005) ve Parçacık Sürü Eniyilemesi metasezgisellerini içermektedir. Topluluk tabanlı bu üç algoritma yüksek seviyeli bir melezleştirme ile nöbetleşe çalışacak şekilde bir araya getirilmiş olup hangi aşamada hangisinin görev alacağı bir parametre olarak tanımlanmakta ve UCB algoritması ile uyarlanabilir olarak seçilmektedir. Geliştirilen bu yöntem, sürekli eniyileme veri setleri üzerinde denenmiştir. Elde edilen sonuçlara göre önerilen melez algoritma, kendisini oluşturan algoritmaların bireysel performansları ile kıyaslandığında önemli bir iyileştirme getirmektedir.

Ayrık eniyileme alanında geliştirilen melez yöntem Kapasiteli Araç Rotalama Problemine yönelik olup Yinelemeli Yerel Arama (Stützle, 1998) ve Değişken Komşu İniş (Hansen and Mladenovic, 2001) metasezgisellerini melezleştirmektedir. Ayrıca, kabul fonksiyonu aşamasında uyarlanabilir parametre kontrolü ile değeri ayarlanabilen bir eşik parametresi eklenmiştir. Tüm bunlara ek olarak, algoritmanın durağanlığa girdiği zamanlar için yeniden başlatma stratejisi eklenerek arama uzayının keşfi artırılmıştır. Geliştirilen algoritmanın başarımı literatürde yaygın olarak kullanılan iki farklı veri setinde denenmiştir.

Bu tezde yer alan diğer bölümler ve içerikleri şu şekilde özetlenebilmektedir: İkinci bölümde, literatürde yaygın olarak kullanılan metasezgisel algoritmalar, yapılarına göre gruplanarak ayrıntılı olarak açıklanmaktadır. Bu bölümde ayrıca melez metasezgisel algoritmalar tanımlanarak, farklı hiyerarşik bakış açılarına göre sınıflandırılmaktadır. Üçüncü bölümde, parametre kontrol konusu genel olarak tanıtılmakta ve bu amaçla uygulanan bazı önemli yaklaşımlar hakkında bilgi verilmektedir. Dördüncü bölümde, literatürdeki tez konusuna benzer, daha önce yapılmış çalışmalar özetlenmektedir. Beşinci ve altıncı bölümlerde sırasıyla sürekli eniyileme ve Kapasiteli Araç Rotalama Problemlerine yönelik geliştirilen parametre kontrollü melez metasezgisel algoritmalar açıklanmaktadır. Yedinci bölümde, geliştirilen bu iki algoritmanın bazı önemli veri setleri üzerinde ayrıntılı deneysel analizi yer almaktadır. Sekizinci ve son bölümde ise tez süresince elde edilen genel sonuçlar ortaya konulmakta ve tartışılmaktadır.

2. METASEZGİSEL ALGORİTMALAR

Metasezgiseller, zor eniyileme (optimizasyon) problemlerinin çözümünde kullanılmak üzere tasarlanmış olan algoritmalarıdır. Metasezgiselleri klasik sezgisel algoritmalarından ayıran en önemli özellik, belirli bir problem türüne özgü olmamalarıdır. Bir metasezgisel herhangi bir eniyileme problemini çözmek üzere uyarlanabilmektedir. Bir diğer önemli özellikleri de, keşif yeteneklerinin daha yüksek olması nedeniyle, evrensel eni iyi çözüm ya da çözümlere ulaşma şanslarının daha yüksek olmasıdır. İlerleyen alt bölümlerde, literatürde yer alan bazı önemli metasezgisel algoritmalar açıklanacaktır.

2.1 Tek Çözüm Tabanlı Metasezgisel Algoritmalar

Tek çözüm tabanlı metasezgisel algoritmalar tek bir başlangıç çözümünden başlayıp arama uzayında bir yörünge (trajectory) tanımlayıp onu takip etmektedirler (Boussaid, 2013). Yörünge takibi eldeki çözümden sistematik olarak komşu çözümler üretip arama sürecine devam etmek şeklinde özetlenebilmektedir. Bir çözümün komşuluğu ise kısaca, kendisini tanımlayan bileşenlerin belirli ölçüde değiştirilmesi ile elde edilebilecek olan geçerli çözüm kümesi şeklinde tanımlanabilmektedir.

2.1.1 Benzetimli tavlama

Benzetimli Tavlama (Simulated Annealing, SA) metasezgiseli Kirkpatrick et al. (1983) tarafından eniyileme problemlerinin çözümünde kullanılmak üzere geliştirilmiştir. Metalurjideki tavlama işleminden esinlenen bu algoritmanın temelleri, birbirleriyle etkileşim içerisindeki moleküllerin özelliklerini hesaplamaya yönelik Monte Carlo tabanlı bir yaklaşım öneren Metropolis et al. (1953)'a dayanmaktadır.

Metropolis algoritması, maddede yer alan moleküllerinin hareketlerinin benzetimini yapmaktadır. Buna göre, belirli bir sıcaklıkta bulunan moleküller yer değiştirerek daha düşük enerji seviyesine geçmeye çalışmakta ve madde aşamalı olarak soğumaktadır. Denge sıcaklığına kadar soğuduğunda ise yer değiştirmeler sonlanarak kararlı duruma gelmektedir. Bu algoritmada molekül hareketleri iki şekilde gerçekleşebilmektedir. Birinci durumda, eğer molekülün hareketi sistemin enerjisini azaltacak yöndeyse doğrudan kabul edilmektedir. İkinci durumda ise hareket sistemin enerjisini artırmakta ama belirli bir olasılığa göre gerçekleşmesine

izin verilmektedir. Bu olasılık maddenin sıcaklığı ile doğru orantılı olacak şekilde hesaplanmaktadır.

SA algoritması (Algoritma 2.1), Metropolis algoritmasında açıklanan yaklaşımı eniyileme problemlerinin çözümüne uyarlamaktadır. Öyle ki, molekül konumları problemin olası çözümlerini temsil etmektedir. Sistemin enerjisi ise uygunluk fonksiyonun değerine karşılık gelmektedir. Buna göre bir enküçükleme (minimizasyon) problemi için, eldeki çözüm (S) adayının değiştirilmesi ile üretilmiş olan komşu çözüm (S') daha düşük bir uygunluk fonksiyonu değerine sahipse, şimdiki çözüm olarak kabul edilmektedir. Eğer daha kötü bir çözüm üretilmişse sıcaklıkla doğru orantılı olarak bir olasılık değerine göre kabul edilebilmektedir. Sonlanma koşuluna kadar yinelenen algoritmanın her bir adımında, sıcaklık değeri önceden belirlenmiş bir plana göre azaltılmaktadır. Algoritmanın sonlanmasının ardından, o ana kadar elde edilmiş en iyi çözüm (S^*), sonuç çözüm olarak belirlenmektedir.

Algoritma 2.1 SA

$T \leftarrow$ başlangıç sıcaklığı
$S \leftarrow$ başlangıç çözümünü oluştur
$S^* \leftarrow S$
while sonlanma koşulu sağlanmadı **do**
  $S' \leftarrow S$'nin komşuluğunda yeni çözüm üret
  $\delta \leftarrow f(S') - f(S)$
  if ($\delta \leq 0$) **then**
    $S \leftarrow S'$
    if $f(S') \leq f(S^*)$
      $S^* \leftarrow S'$
    end if
  else
    $p \leftarrow$ 0 ile 1 arasında rastgele sayı üret
    $p \leftarrow$ T ve δ değerlerine göre olasılık hesapla
    if r $\leq$ p **then**
      $S \leftarrow S'$
    end if
  end if
  T'yi azalt
end while

2.1.2 Tabu arama

Tabu Arama (Tabu Search, TS) metasezgiseli Glover (1986) tarafından eniyileme problemlerinin çözümü için geliştirilmiştir. TS algoritması, eniyilenecek olan problem üzerinde arama yapabilen bir alt seviye sezgiseli yönlendirmektedir.

Alt seviye sezgisel, eldeki çözümü değişikliğe uğratarak yeni çözümler ararken "Tabu" adı verilen bir listeyi dikkate almaktadır. Bu listede, yakın zamanda gezilmiş olan komşu çözümler veya onlara ait özellikler saklanmaktadır. Eğer yeni üretilecek olan komşu çözüm tabu listesinde yer alıyorsa, bu değişikliğe izin verilmemekte ve başka çözümler denenmektedir. Böylece, arama sürecine bir bellek eklenerek tekrarlayan çözümlerden kaçınılmakta ve keşif yeteneği artırılmış olmaktadır.

Tabu listesinin uzunluğu arama sürecinin keşif gücünü kontrol etmektedir. Kısa uzunluklu listeler az sayıda çözümü belleğinde tuttuğundan önceden gezilmiş çözümler "unutulmakta" ve böylece tekrar kabul edilmesinin önü açılmaktadır. Uzun listeler ise daha eski kayıtları da içerdiğinden, yeni çözümlerin kabul edilmesini teşvik etmektedir.

Algoritma 2.2 TS

$S \leftarrow$ başlangıç çözümünü oluştur
$S^* \leftarrow S$
$TabuListesi \leftarrow \emptyset$
while sonlanma koşulu sağlanmadı **do**
  $S' \leftarrow$ En iyi çözümü { $N(S) \setminus TabuListesi$ } kümesinden seç
  $S \leftarrow S'$
  if $f(S') \leq f(S^*)$
    $S^* \leftarrow S'$
  end if
  $TabuListesi \leftarrow TabuListesi \cup S$
end while

TS Algoritması (Algoritma 2.2) bir adet başlangıç çözümü (S) ile başlamaktadır. Sonlanma koşulu sağlanana kadar her bir iterasyonda, eldeki çözümün komşuluğunda yeni aday çözümler üretilmektedir. Aday çözümler içerisinden tabu listesinde yer alanlar çıkarıldıktan sonra elde kalan en iyi çözüm (S') şimdiki çözüm olarak belirlenmektedir. Algoritmanın sonlanmasının ardından, o ana kadar elde edilmiş en iyi çözüm (S^*), sonuç çözüm olarak belirlenmektedir.

2.1.3 Değişken komşu iniş

Değişken Komşu İniş (Variable Neighborhood Descent, VND) (Hansen and Mladenovic, 2001) metasezgiseli deterministik olarak birden fazla komşuluk yapısı içerisinde arama yapmaktadır (Algoritma 2.3). VND algoritması çalışmaya başlamadan önce bir adet başlangıç çözümü oluşturulmakta ve bu çözüm üzerinde geçerli olacak k>1 adet komşuluk yapısı tanımlanmaktadır. Arama işlemi birinci

komşuluk yapısından başlatılır ve o yapıdan elde edilen en iyi çözüm şimdiki çözümü iyileştirmedikçe bir sonraki komşuluk yapısına geçilir. Eğer herhangi bir anda daha iyi bir çözüm elde edilirse aramaya tekrar baştan (k=1) başlanır. Bir çözüm için eldeki tüm komşuluk yapılarından da sonuç alınamaması durumunda algoritma sonlandırılarak eldeki en iyi çözüm döndürülür.

Algoritma 2.3 VND

$S \leftarrow$ başlangıç çözümünü oluştur
$N_k(k = 1, \dots, n)$ olmak üzere n adet komşuluk yapısını belirle
k ← 1
while k < n **do**
  $S' \leftarrow N_k$ komşuluğunda S'nin en iyi komşusunu seç
  if $f(S') < f(S)$
    $S \leftarrow S'$
    k ← 1
  else
    k ← k + 1
  end if
end while

2.1.4 Değişken komşuluk arama

Değişken Komşuluk Arama (Variable Neighborhood Search, VNS) metasezgiseli (Mladenovic and Hansen, 1997) birden fazla komşuluk yapısı içermektedir. Arama sürecinin gidişatına göre sistematik olarak komşuluk yapıları arasında geçiş yapılmaktadır. VNS, sarsma (shaking) ve yerel arama olmak üzere iki adımı yinelemeli olarak kullanmaktadır.

Algoritma 2.4'te VNS'nin genel işleyişi yer almaktadır. İlkleme aşamasında başlangıç çözümünün (S) oluşturulmasının yanı sıra problemin çözümünde kullanılacak olan komşuluk yapıları (N) da oluşturulmaktadır. Genellikle bu yapılar küçükten (yakın komşuluk üreten) büyüğe (uzak komşuluk üreten) doğru sıralanmaktadır. VNS eniyileme işlemini tekrarlayan sarsma ve yerel arama işlemleriyle sağlamaktadır. Sarsma işleminde, şimdiki çözüm sırası gelen komşuluk yapısı (k. komşuluk) tarafından rastgele değiştirilmekte ve komşu çözüm (S') elde edilmektedir. Ardından komşu çözüm yerel aramadan geçirilerek yerel en iyi nokta (S'') elde edilmektedir. Eğer yerel en iyi çözüm şimdiki çözümü iyileştirmiş ise onun yerini almakta ve bir sonraki iterasyon için başlangıç komşuluk seviyesine geri dönülmektedir (k=1). Aksi takdirde komşuluk seviyesi artırılarak aramaya bir üst seviyeden devam edilmektedir. Bu stratejideki ana fikir, algoritma yerel en iyiye takıldığı zaman komşuluk düzeyini genişleterek keşif yeteneğini artırmak, yeni bir

çözüm kabul edildiği zaman ise komşuluk düzeyini daraltarak o bölge çevresinde aramaya odaklanıp faydalanma yeteneğini artırmaktır. Sonlanma koşulu sağlandığı an bulunmuş olan en iyi çözüm algoritmanın sonucu olarak belirlenmektedir.

Algoritma 2.4 VNS

$S \leftarrow$ başlangıç çözümünü oluştur
$N_k(k = 1, \dots, n)$ olmak üzere n adet komşuluk yapısını belirle
while sonlanma koşulu sağlanmadı **do**
  k ← 1
  while k < n **do**
    $S' \leftarrow N_k$ komşuluğunda S'nin rastgele bir komşusunu seç
    $S'' \leftarrow$ yerelArama(S')
    if $f(S'') \leq f(S)$
      $S \leftarrow S''$
      k ← 1
    else
      k ← k + 1
    end if
  end while
end while

2.1.5 Yinelemeli yerel arama

Yinelemeli Yerel Arama (Iterated Local Search, ILS) algoritması Stützle (1998) tarafından tanımlanmıştır. ILS metasezgiselinin temel yaklaşımı, yerel en iyiye takılma durumunda rastgele yeniden başlatma işlemine başvurmamaktır. Çünkü rastgele başlangıç yapıldığı zaman o ana kadar elde edilmiş olan arama sürecine yönelik bilgiler de sıfırlanmaktadır. ILS algoritması bunun yerine sarsım (perturbation) eylemini gerçekleştirerek var olan çözümün komşuluğunda rastgele yeni bir çözüm üretmektedir. Ardından yerel arama işlemine bu yeni noktadan devam edilerek eniyileme sürecine süreklilik kazandırılmaktadır.

ILS algoritmasının (Algoritma 2.5) ilkleme aşamasında başlangıç çözümü oluşturulup yerel arama işlemine sokulmaktadır. Ardından sonlanma koşuluna kadar girilen döngüde şimdiki çözüm önce sarsım işlemine tabi tutularak komşu çözüm (S') elde edilmekte, sonrasında yerel arama gerçekleştirilerek aday çözüm (S'') oluşturulmaktadır. Aday çözüm o ana kadar bulunmuş en iyi çözümden daha iyiyse yeni en iyi çözüm olarak belirlenmektedir. Döngü içerisindeki son adımda ise elde edilen aday çözüm önceden belirlenmiş kabul kriterine göre ya şimdiki çözüm olarak kabul edilmekte ya da reddedilmektedir. Sonlanma koşulu sağlandığı an bulunmuş olan en iyi çözüm (S^*) algoritmanın sonucu olarak belirlenmektedir.

Dikkat edilirse ILS çatısı hem *sarsım* hem de *kabulKriteri* fonksiyonlarında *geçmişBilgisi*'nden yararlanmaktadır. Buradaki geçmiş bilgisi algoritmanın gidişatını göstermekte olup kendisinden algoritmanın keşif/faydalanma dengesini kurmada yararlanılabilmektedir. Örneğin, arama durağanlığa girdiyse, sarsım fonksiyonu daha geniş ölçekteki komşuluklarda çözüm üretebilmekte veya kabul fonksiyonu iyileştirme yapmayan aday çözümleri de kabul etme olasılığını artırabilmektedir.

Algoritma 2.5 ILS

$S \leftarrow$ başlangıç çözümünü oluştur
$S \leftarrow S$'ye yerel arama uygula
$S^* \leftarrow S$
while sonlanma koşulu sağlanmadı **do**
  $S' \leftarrow$ sarsım(S, *geçmişBilgisi*)
  $S'' \leftarrow$ yerelArama(S')
  if $f(S'') < f(S^*)$
    $S^* \leftarrow S''$
  end if
  $S \leftarrow$ kabulKriteri(S, S'', *geçmişBilgisi*)
end while

2.1.6 Açgözlü rastgele uyarlanabilir arama prosedürü

Açgözlü Rastgele Uyarlanabilir Arama Prosedürü (Greedy Randomized Adaptive Search Procedures, GRASP) ilk kez Feo and Resende (1989) tarafından küme kapsama probleminin çözümüne yönelik geliştirilmiştir. GRASP yapılandırıcı nitelikte bir algoritma olup problemin çözümünü adım adım eklemelerle oluşturmaktadır. GRASP'nin en karakteristik özelliği, çözümün yapılandırmasında hangi parçanın ekleyeceğine karar verirken, doğrudan en iyi alternatifi seçmek yerine belirli sayıdaki iyi alternatifler arasından rastgele seçim yapmasıdır. GRASP metasezgiseline ait sözde kod (Feo and Resende, 1995) Algoritma 2.6'da verilmektedir. Sonlanma koşulu sağlanana kadar her bir iterasyonda sıfırdan bir çözüm yapılandırılmakta (S) ve hemen sonra yerel arama yapılarak yeni aday çözüm (S') elde edilmektedir. Ardından o ana kadar bulunmuş en iyi çözüm bilgisi güncellenmektedir. Algoritmanın sonlanmasının ardından, o ana kadar elde edilmiş en iyi çözüm (S^*), sonuç çözüm olarak belirlenmektedir.

Algoritma 2.6 GRASP

```
while sonlanma koşulu sağlanmadı do
  S ← {}
  while çözüm yapılandırma tamamlanmadı do //yapılandırma aşaması
    RCL ← sınırlandırılmış aday listesi oluştur
    i ← RCL'den rastegele seçim yap
    S ← S ∪ {i}
    aday listesini uyarla
  end while
  S' ← yerelArama(S)
  if f(S') ≤ f(S*)
     S* ← S'
  end if
end while
```

GRASP algoritmasının en önemli noktası sınırlandırılmış aday listesinin (RCL) büyüklüğüdür. RCL, yapılandırma aşamasında sıradaki eklenecek aday çözüm bileşenlerinin tutulduğu listedir. RCL büyüklüğü ya önceden sabit belirli bir p büyüklüğüne eşitlenir ya da α parametresi tarafından $[c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$ aralığında maliyete sahip çözüm elemanlarını içerecek şekilde belirlenmektedir. Burada, c^{min} en az ekleme maliyetine sahip çözüm elemanını, c^{max} ise en yüksek ekleme maliyetine sahip çözüm elemanını temsil etmektedir. A parametresi [0,1] aralığında değer almaktadır ve 0 olması durumunda algoritma saf açgözlü yaklaşıma, 1 olması durumunda da saf rasgele arama yaklaşımına dönüşmektedir. Bu nedenle p ya da α parametrelerinin probleme göre uygun bir şekilde ayarlanması gerekmektedir.

2.2 Topluluk Tabanlı Metasezgisel Algoritmalar

Topluluk (popülasyon) tabalı metasezgisel algoritmalarda birden fazla çözüm adayının aynı anda bulunması söz konusudur. Bu bölümde literatürde başarısı ortaya konulmuş olan bazı önemli topluluk tabanlı algoritmalar hakkında bilgiler verilmektedir. Anlamsal bütünlüğü sağlamak açısından bu algoritmalar Evrimsel Algoritmalar ve Sürü Zekası olmak üzere iki ana grupta toplanmıştır.

2.2.1 Evrimsel algoritmalar

Evrimsel Algoritmalar (Evolutionary Algorithms, EA), biyolojideki evrimsel süreçlerden esinlenen topluluk tabanlı algoritma grubunu ifade eden bir üst başlıktır. EA'ların sahip olduğu genel algoritmik çatı Algoritma 2.7'de (Boussaid, 2003) yer almaktadır. Algoritmanın ilkleme aşamasını oluşturan ilk iki adımda sırasıyla rastgele bireyler (çözüm adayları) üretilerek bir topluluk oluşturulmakta

ve her bir bireyin kalitesi (uygunluk değeri) ölçülmektedir. İlkleme aşamasının tamamlanmasının ardından, sonlanma koşulu sağlanana değin şu adımlar sırası ile tekrarlanmaktadır:

- Ebeveyn çözümler seçilir.
- Seçilen ebeveyn çözümler birleştirilerek yeni yavru çözümler elde edilir.
- Elde edilen yavru çözümler belirli oranda rastgele değiştirilerek mutasyona uğratılır.
- Yeni bireylerin kalitesi ölçülür.
- Eldeki bireylerden bir sonraki nesle aktarılacak olanlar seçilir.

Yukarıda açıklandığı şekliyle işleyen bir EA'dan beklenen, nesiller boyunca güçlü bireylerin (kaliteli çözümlerin) seçilim ve birleşme yoluyla toplulukta varlıklarını sürdürmeleri ve topluluğun ortalama kalitesinin artış eğilimi göstermesidir.

Algoritma 2.7 EA

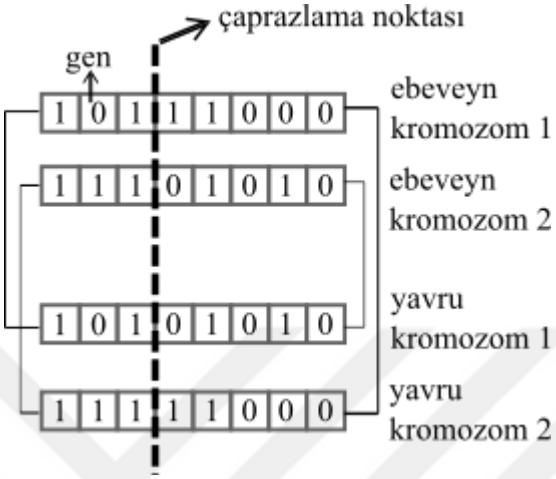
```
Rastgele bireyler ile topluluğu oluştur
Her bir bireyin kalitesini ölç
while sonlanma koşulu sağlanmadı do
  Evebeynleri seç
  Ebeveyn çiftlerini birleştir
  Yavru çözümlere mutasyon uygula
  Yeni bireylerin kalitesini ölç
  Yeni nesil için bireyleri seç
end while
```

2.2.1.1 Genetik algoritmalar

Genetik Algoritmalar (Genetic Agorithms, GA) (Holland, 1975) literatürde en yaygın kullanılan metasezgisel algoritmalardandır. GA, Algoritma 2.7'de açıklanmış olan genel EA çatısına uygun olarak işlemekle beraber bazı farklı terimler içermektedir. Bunlar aşağıda listelenmektedir:

- Kromozom: Topluluktaki bireylere verilen addır. Her bir kromozom bir çözüm adayını temsil etmektedir.
- Gen: Kromozomları oluşturan temel yapı taşlarıdır. Her bir gen bir çözüm adayının bir parçasına karşılık gelmektedir.
- Çaprazlama: Ebeveyn çiftlerinin birleştirilmesi adımına karşılık gelmektedir. Kromozomlar birleştirilerek yeni yavru çözümler üretilmektedir.

Kromozom, gen ve çaprazlama kavramlarını görsel olarak açıklayan bir örnek Şekil 2.1'de yer almaktadır. Bu örnekte ikili kodlama yapısına sahip kromozomlar ve onlardan tek noktalı çaprazlama yoluyla elde edilen yavru bireyler bulunmaktadır. Üçüncü gen konumunun bitiminde seçilen çaprazlama noktasından sonra yer alan gen blokları karşılıklı olarak yer değiştirilmektedir.

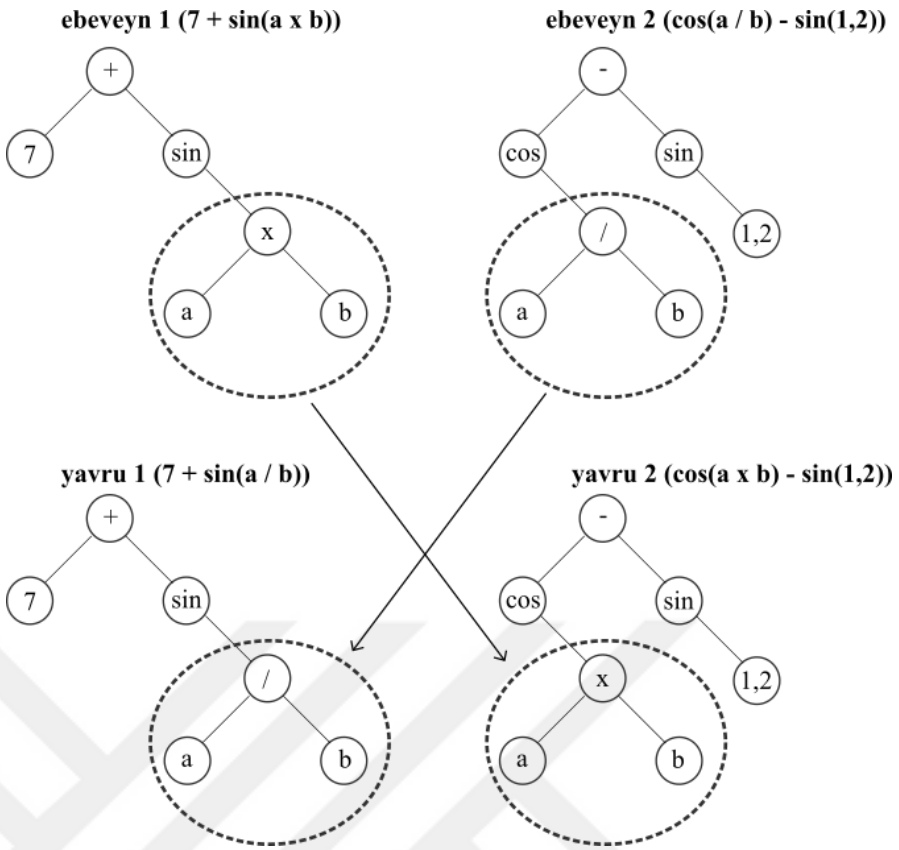


Şekil 2.1. Çaprazlama örneği.

2.2.1.2 Genetik programlama

Genetik Programlama (Genetic Programming, GP) (Koza, 1992), verilen yüksek seviyeli problem ifadelerinden otomatik olarak bilgisayar programlamları üretmeye yarayan bir metasezgiseldir. GP'nin işleyişi GA'ya benzer olup başlangıç topluluğunda yer alan program adayları, nesiller boyunca çaprazlama, mutasyon ve seçilim operatörleri ile iyileştirilmektedir. Öğrenebilen yapısıyla GP, daha çok sınıflandırma ve tahminleme gibi alanlarda kendisine uygulama alanı bulmaktadır.

GP, GA'ya benzer şekilde çalışsa da kromozom yapısı ağaç gibi farklı veri yapıları ile temsil edildiğinden farklı çaprazlama ve mutasyon stratejilerine gereksinim duymaktadır. Şekil 2.2'de yer alan örnekte fonksiyon şeklinde tanımlanmış iki adet ebeveyn birey ve onlardan çaprazlama yoluyla elde edilmiş olan iki yavru birey gösterilmektedir. Örnekteki çaprazlama işlemi, iki adet ebeveyn programdan belirlenen rastgele iki alt ağacın karşılıklı olarak değiş tokuş edilmesi yoluyla yapılmıştır. Böylece iki adet yavru birey üretilmiş olmaktadır.



Şekil 2.2. GP'da kromozom örneği.

GP'de kromozomlar terminal (T) ve fonksiyon (F) olmak üzere iki adet kümeye göre üretilmektedir. T kümesi sabitler (örn. 7) ve bağımlı değişkenlerden (örn. a) oluşurken F kümesi fonksiyonlardan (örn. +) oluşmaktadır. Şekil 2.2'deki örneğe dikkat edilirse ağacın yaprakları tamamen terminallerden oluşmaktadır. GP'nın bir diğer özelliği ise farklı uzunlukta çözümlere sahip olabilmesidir.

2.2.1.3 Evrimsel strateji

Evrimsel Strateji (Evolution Strategy, ES) algoritması (Rechenberg, 1971; Schwefel 1974) doğal evrimsel sürece ait ilkeleri kullanarak eniyileme problemlerine çözüm aramaktadır. Bir ES algoritması (μ,λ) ya da (μ+λ) gösterimleri ile ifade edilebilmektedir. Bu gösterimde μ ebeveyn birey sayısını λ ise üretilecek olan yavru birey sayısını ifade etmektedir. Gösterimde yer alan "+" ve "," işaretleri bir sonraki nesle aktarılacak bireylerin seçilimi aşamasındaki yaklaşımı belirlemektedir. Buna göre, "+" içeren gösterimlerde ebeveyn ve yavru bireylerin tümü arasından seçme işlemi gerçekleştirilirken, "," içeren gösterimlerde sadece yavru bireyler arasından seçim yapılmaktadır.

ES'de mutasyon adım büyüklüğü σ parametresi ile belirlenmektedir. Öyle ki, mutasyonlar ortalaması 0, standart sapması σ olan normal dağılıma göre yapılmaktadır. Literatürdeki çeşitli ES yöntemleri σ parametresinin belirlenme yöntemine göre ayrışmaktadır. Bunlara bir örnek 1/5 başarı kuralıdır. Bu kurala göre başarılı mutasyonların oranı 1/5 olmalıdır; eğer bu oranın altında ise σ değeri azaltılmakta, eğer bu oranın üstünde ise σ değeri artırılmaktadır. Daha güçlü yöntemler ise kendinden uyarlanabilir yaklaşımı benimsemekte ve σ parametresini toplulukta yer alan bireylerin içerisine yerleştirerek otomatik olarak ayarlanmasını sağlamaktadırlar.

2.2.1.4 Diferansiyel gelişim

Diferansiyel Gelişim (Differential Evolution, DE) metasezgiseli (Storn and Price, 1997) sürekli eniyileme problemlerinin çözümüne yönelik kullanılmakta olan en yaygın algoritmalar arasında bulunmaktadır. DE diğer EA'larda olduğu gibi rastgele oluşturulmuş bir başlangıç topluluğunu iterasyonlar boyunca iyileştirmeye çalışmaktadır.

DE'de çözümler vektörler şeklinde temsil edilmekte olup vektörün uzunluğu problem boyutunun uzunluğuna denk olmaktadır. Sonlanma koşuluna kadar her bir iterasyonda, toplulukta yer alan her bir birey (vektör) için mutasyon yoluyla "hedef" vektörü üretilmektedir. Hedef vektör üretimi için çeşitli stratejiler tanımlanmış olup bunlar "DE/a/b" kalıbına uymaktadırlar. Örneğin, "DE/rand/1" stratejisinde (Eşitlik 2.1) topluluktan rastgele seçilen bir bireye yine rastgele seçilen iki bireyin farkı önceden belirlenmiş bir oranda eklenmektedir. Bu stratejinin "DE/rand/2" (Eşitlik 2.2) olması durumunda ise rastgele seçilen bireye rastgele seçilen iki çift bireyin farkları ayrı ayrı eklenmektedir. Bir diğer örnek olarak, "DE/best/1" (Eşitlik 2.3) stratejisinde toplulukta yer alan en iyi bireye rastgele seçilmiş bir çiftin farkı eklenmektedir. Bu denklemlerde yer alan V_i, i. çözüme ait hedef vektörünü, X, çözüm vektörlerini, $r1, r2, r3, r4, r5$ birbirinden ve i'den farklı olacak şekilde [1, topluluk büyüklüğü] arasında seçilen rastgele indisleri, F, diferansiyel ağırlık katsayısını ve X_{eniyi} ise evrensel en iyi çözümü temsil etmektedir.

$$V_i = X_{r1} + F(X_{r2} - X_{r3}) \tag{2.1}$$

$$V_i = X_{r1} + F(X_{r2} - X_{r3}) + F(X_{r4} - X_{r5}) \tag{2.2}$$

$$V_i = X_{eniyi} + F(X_{r1} - X_{r2}) \quad (2.3)$$

Toplulukta yer alan her bir birey için hedef vektörünün belirlenmesinden sonra çaprazlama aşamasına geçilmekte ve bunun sonucunda "deneme" vektörü üretilmektedir. Bu aşamada çaprazlama oranı (CR) parametresi önemli bir rol oynamaktadır. CR [0,1] aralığında belirlenen bir reel sayı olup çaprazlamanın gerçekleşme olasılığını belirlemektedir. Buna göre, binom çaprazlama yöntemi için, bireyin her bir elemanı (boyut) [0,1] aralığında tekdüze rastgele bir sayı belirlenmekte ve eğer bu sayı CR'ye eşit ya da onun altında ise deneme vektörünün ilgili boyutu hedef vektöründen alınmaktadır. Üretilen rastgele sayının CR'den büyük olması durumunda ise bireyin eski değeri saklanmaktadır. Oluşturulan deneme vektörü bireyin eski durumu ile karşılaştırılmakta ve ondan daha iyiyse onun yerine yeni çözüm olarak topluluktaki yerini almaktadır.

2.2.2 Sürü zekası

Sürü zekası (Swarm Intelligence, SI), doğal ortamda sürü oluşturan canlıların kolektif davranışlarından ortaya çıkan problem çözme yeteneğini eniyileme problemlerinin çözümüne uyarlayan algoritmalar için genel bir adlandırmadır. SI'da sürüyü oluşturan bireyler basit etmenler olarak adlandırılmaktadır. Bunun nedeni, bu bireylerin tek başına bir zekası ya da problem çözme yetenekleri olmamasıdır. Ayrıca, bireylerin birbirleriyle etkileşimleri de basit kurallarla sınırlandırılmıştır. SI'nın gücünü ortaya çıkaran ise bu basit birey yapısı ve etkileşim kurallarından kolektif bir "zekanın" ve problem çözme yeteneğinin ortaya çıkabiliyor olmasıdır. Literatürde karıncalar, arılar, kuşlar vb. farklı özelliklerde sürü oluşturan canlılardan esinlenen başarılı metasezgiseller bulunmaktadır.

2.2.2.1 Karınca kolonisi eniyilemesi

Karınca Kolonisi Eniyilemesi (Ant Colony Optimization, ACO), karıncaların doğadaki yiyecek arama davranışlarından esinlenmekte ve bu yaklaşımı eniyileme problemlerinin çözümüne uyarlamaktadır. İlk ACO sistemi Dorigo (1992) tarafından doktora tezi ile ortaya konulmuş olup Karınca Sistemi (Ant System) adı ile adlandırılmıştır. Daha sonraları Dorigo et al. (1999) tarafından ACO metasezgiseli olarak tanımlanmıştır.

ACO metasezgiselinin işleyişine geçmeden önce algoritmada önemli bir yere sahip olan feromon teriminin açıklanması gerekmektedir. Bir kimyasal madde olan

feromon, doğal ortamda karıncalar tarafından üzerinde gezildiği ortama bırakılmaktadır. Bir yerde feromon maddesinin olması oranın çekiciliğini karıncalar için artırmaktadır. Burada, feromon miktarının büyüklüğü de etkili olmakta; daha yoğun feromon bulunan bölgelere yönelme olasılığı daha yüksek olmaktadır. Bırakılacak feromon miktarının büyüklüğünü ise bulunan yiyeceğin kalitesi belirlemektedir. Bir karınca yiyecek bulup onu yuvasına taşırken, geçtiği yol üzerine taşıdığı yiyeceğin kalitesi oranında feromon bırakmaktadır. Böylece kaliteli yiyecek kaynağına giden yollar üzerinde yoğun feromon izleri birikmekte ve diğer karıncaları da o bölgeye çekmektedir. Feromon izi ile ilgili bilinmesi gereken bir diğer önemli nokta da buharlaşma kavramıdır. Dış ortama bırakılan feromonlar zamanla buharlaşmaktadır. Böylece, tükenen bir yiyecek kaynağına giden yol üzerinde feromonlar yenileriyle desteklenmediğinden dolayı zamanla kaybolmakta ve artık karıncaları üzerine çekmeye devam etmemektedir. Bu da karıncaların yiyecek arama davranışlarında değişen koşullara uyum sağlayabilmelerini sağlamaktadır.

ACO metasezgiselinin işleyişine ait sözde kod, Algoritma 2.8'de verilmektedir. İlkleme adımında parametreler belirlenmekte ve feromon tablosu ilk değerleri ile doldurulmaktadır. Bir kombinasyonel eniyileme problemi düşünüldüğünde feromon tablosu, tam bağlantılı çizgenin kenarlarının (ayrıtlarının) ağırlıkları şeklinde temsil edilmektedir. Buna göre çizgede yer alan her bir düğüm çifti arasında belirli bir feromon değeri bulunmaktadır. Ardından, algoritmanın ana döngüsü önceden belirlenmiş olan bir sonlanma koşuluna kadar çalışmaktadır.

Algoritma 2.8 ACO

```
İlkleme
while sonlanma koşulu sağlanmadı do
   Karınca çözümlerini yapılandır
   Yerel arama uygula (seçimlik)
   Feromon değerlerini güncelle
end while
```

Algoritma 2.8'deki ana döngü içerisinde gerçekleştirilen ilk aşama, çözüm adaylarının yapılandırılması adımıdır. Bu aşamada her bir yapay karınca, çizgenin rastgele bir noktasından başlayarak tüm düğümleri bir kez gezecek şekilde yeni bir çözüm inşa etmektedir. Bu durumu matematiksel olarak ifade etmek gerekirse, i. düğümde bulunan k. karınca j. düğüme geçiş yapacağı zaman Eşitlik 2.4'te hesaplanan olasılığa (p_{ij}^{k}) göre karar vermektedir. Burada, $N(s^p)$, seçilebilecek uygun çözüm bileşenlerinin (düğümlerin) kümesini, (i,l), k. karınca tarafından henüz ziyaret edilmemiş kenarları, $\tau_{i,j}$, i ve j düğümleri arasındaki feromon

miktarını, $\eta_{i,j}$, i ve j düğümleri arasındaki sezgisel bilgiyi, α ve β ise kontrol parametrelerini göstermektedir.

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^{p})} \tau_{il}^{\alpha}.\eta_{il}^{\beta}}, \forall c_{ij} \in N(s^{p}) \quad (2.4)$$

Ana döngünün ikinci aşaması olan yerel arama adımı isteğe bağlı olup eldeki çözümler önceden belirlenmiş bir yerel arama yöntemi ile iyileştirilmeye çalışılmaktadır.

Üçüncü ve son aşamada ise elde edilmiş çözüm adaylarının uygunluk değerleri hesaplanmaktadır. Her bir çözüm için kalitesi oranında feromon değeri çözümün geçtiği kenarlara eklenmektedir. Feromon güncelleme aşamasında ayrıca her bir kenardan belirli bir oranda buharlaştırma yapılarak feromon miktarı azaltılmaktadır. Eşitlik 2.5'te feromon güncelleme formülü yer almaktadır. Burada, τ_{ij}, (i,j) kenarındaki feromon miktarını, $\rho \in [0,1]$, buharlaşma katsayısını, m karınca sayısını ve $\Delta_{\tau_{ij}^{k}}$ ise k. karıncanın (i,j) kenarı üzerine bıraktığı feromon miktarını temsil etmektedir.

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta_{\tau_{ij}^{k}} \quad (2.5)$$

2.2.2.2 Yapay arı kolonisi

Yapay Arı Kolonisi (Artificial Bee Colony, ABC) metasezgiseli bal arılarının yiyecek arama davranışlarından esinlenmekte olup Karaboğa (2005) tarafından sürekli eniyileme problemlerinin çözümü kapsamında önerilmiştir. ABC algoritması kapsamında kullanılan önemli terimler aşağıda listelenmektedir.

- Besin kaynağı: Algoritmada tanımlanmış bir besin kaynağı bir aday çözüme karşılık gelmektedir. Algoritmanın amacı iterasyonlar boyunca daha iyi besin kaynaklarını aramaktır.
- İşçi arı: Bir işçi arının görevi var olan bir besin kaynağından faydalanarak besin elde etmektir. Bu işlem arama sürecinde eldeki çözümün komşuluğunda aday çözüm bulunmasıdır. Eğer bulunan aday çözüm eskisinden daha kaliteli ise o besin kaynağından faydalanılmış sayılmaktadır. Algoritmadaki işçi arı sayısı aynı zamanda besin kaynağı sayısına eşittir.

- Gözlemci arı: Gözlemci arılar, işçi arılardan gelen geri bildirime göre seçilen kaliteli besin kaynaklarına tekrar giderek besin elde etmeye çalışmaktadır.
- İzci arı: İzci arılar, tükenmiş besin kaynakları yerine yeni (rastgele) besin kaynağı bulmakla görevlidirler. Bir besin kaynağının tükenmiş sayılabilmesi için o kaynaktan belirli bir iterasyon boyunca hiç besin elde edilememiş olmaması gerekmektedir.

ABC algoritmasına ait sözde kod Algoritma 2.9'da verilmektedir. İlkleme aşamasında besin kaynakları (aday çözümler) rastgele olarak oluşturulmakta ve kontrol parametreleri belirlenmektedir. İlkleme işleminin ardından sonlanma koşuluna kadar yinelemeli olarak eldeki çözümler sırasıyla işçi, gözlemci ve izci arılar aşamalarıyla iyileştirilmeye çalışılmaktadır.

Algoritma 2.9 ABC

```
İlkleme
while sonlanma koşulu sağlanmadı do
   İşçi arılar aşaması
   Gözlemci arılar aşaması
   İzci arılar aşaması
   En iyi çözümün hafızaya alınması
end while
```

İşçi arılar aşamasında her bir aday çözümden yeni bir çözüm elde edilmektedir (Eşitlik 2.6). Burada *X*, şimdiki çözümleri, *V*, üretilen yeni çözümleri, *i*, şimdiki çözüm numarasını, *k*, rastgele seçilmiş olan komşu çözüm numarasını, *j*, boyut numarasını ve φ ise [0,1] aralığında tekdüze dağılıma göre üretilmiş rastgele bir sayıyı ifade etmektedir. Eğer elde edilen yeni çözümün uygunluk değeri eskisinden daha yüksekse, yeni çözüm eski çözümün yerini almaktadır.

$$V_{i,j} = X_{i,j} + \varphi_{i,j}(X_{i,j} - X_{k,j}) \tag{2.6}$$

Gözlemci arılar aşamasında, uygunluk değeri yüksek olan çözümlerden Eşitlik 2.7 yoluyla yeni çözümler elde edilerek iyileştirme sağlanmaya çalışılmaktadır. Bir çözümün bu kapsamda seçilme olasılığı ise Eşitlik 2.7'ye göre hesaplanmaktadır. Burada p_i, i. çözümün seçilme olasılığını, $fit(X_i)$, i. çözümün uygunluk değerini, SN ise toplam çözüm sayısını ifade etmektedir. Gözlemci arılar aşaması algoritmanın faydalanma özelliğini güçlendirici bir rol oynamaktadır.

$$p_i = \frac{fit(X_i)}{\sum_{i=1}^{SN} fit(X_i)} \tag{2.7}$$

İzci arılar aşamasında, belirli bir iterasyon boyunca iyileştirilemeyip limit değerine ulaşan çözümler atılarak yerine rastgele yeni çözümler oluşturulmaktadır. Bu aşama algoritmanın keşif özelliğini güçlendirmektedir.

2.2.2.3 Parçacık sürü eniyilemesi

Parçacık Sürü Eniyilemesi (Particle Swarm Optimization, PSO) metasezgiseli kuşların sürü oluşturma davranışından esinlenmekte olup Kennedy and Eberhart (1995) tarafından literatüre kazandırılmıştır. PSO algoritmasında sürüyü oluşturan bireyler parçacık adı ile anılmaktadır. Her bir parçacık, eniyilenecek olan problemin bir aday çözümüne karşılık gelmektedir. Parçacıkların birleşiminden ise arama uzayında birbirlerine göre konum değiştiren ve evrensel en iyi noktayı bulmayı hedefleyen sürü yapısı oluşmaktadır.

PSO'da parçacıklar hız ve konum vektörlerine sahiptirler. Konum vektörü parçacığın arama uzayındaki yerini belirtmekte olup, doğrudan aday çözümü temsil etmektedir. Hız vektörü ise parçacığın bir sonraki iterasyonda hangi konumda olacağını belirlemektedir. Hız vektörünü yönlendirmekte olan iki parametre ise sırasıyla parçacığın elde ettiği o ana kadar en iyi kişisel konum vektörü ve sürünün o ana kadar elde ettiği evrensel en iyi konum vektörüdür.

PSO metasezgiseline ait sözde kod, Algoritma 2.10'da verilmektedir. İlkleme aşamasında sürüyü oluşturan parçacıkların konum ve hız vektörleri rastgele bir şekilde oluşturulmaktadır. Ardından, sonlanma koşuluna kadar her bir iterasyonda sürünün tüm bireylerinin sırasıyla önce hız ardından konum vektörleri güncellenmektedir. Bu değişim sonucunda parçacığın geçmişteki en iyi konumu iyileştirilmişse, kişisel en iyi konumu güncellenmektedir. Ayrıca evrensel en iyi çözümden daha iyi bir çözüm bulunmuşsa bu değer de güncellenmektedir.

Algoritma 2.10 PSO

```
İlkleme
while sonlanma koşulu sağlanmadı do
  for sürü içerisindeki her bir parçacık do
    Hız vektörünü güncelle
    Konum vektörünü güncelle
    Parçacığın kişisel en iyi konum vektörünü güncelle
    Evrensel en iyi konum vektörünü güncelle
  end for
end while
```

Parçacıkların hız vektörlerinin güncellenmesi formülü Eşitlik 2.8'de verilmektedir. Burada, *V*, hız vektörleri listesini, *X*, konum vektörleri listesini, *i*, parçacığın numarası, *j*, boyut numarası, r_1 ve r_2 [0,1] aralığında tekdüze dağılımlı rastgele sayıları temsil etmektedir. p ve g vektörleri ise sırasıyla parçacığın kişisel en iyi konum ve evrensel en iyi konum vektörleridir. $w \in [0,1]$ parametresi durağanlık katsayısı olup parçacığın bir önceki hız vektörünün ne kadar korunacağını ayarlamaktadır. c_1 ve c_2 ise ölçeklendirme parametreleri olup sırasıyla parçacığın kendi kişisel en iyi konumunun ve evrensel en iyi konumun hız vektöründeki ağırlığını belirlemektedirler.

$$V_{i,j} = wV_{i,j} + c_1 r_1 \left(p_{i,j} - X_{i,j}\right) + c_2 r_2 (g_j - X_{i,j}) \tag{2.8}$$

Güncel hız vektörünün elde edilmesinin ardından parçacığın önceki konum vektörüne eklenerek yeni konum vektörü Eşitlik 2.9'a göre hesaplanmaktadır.

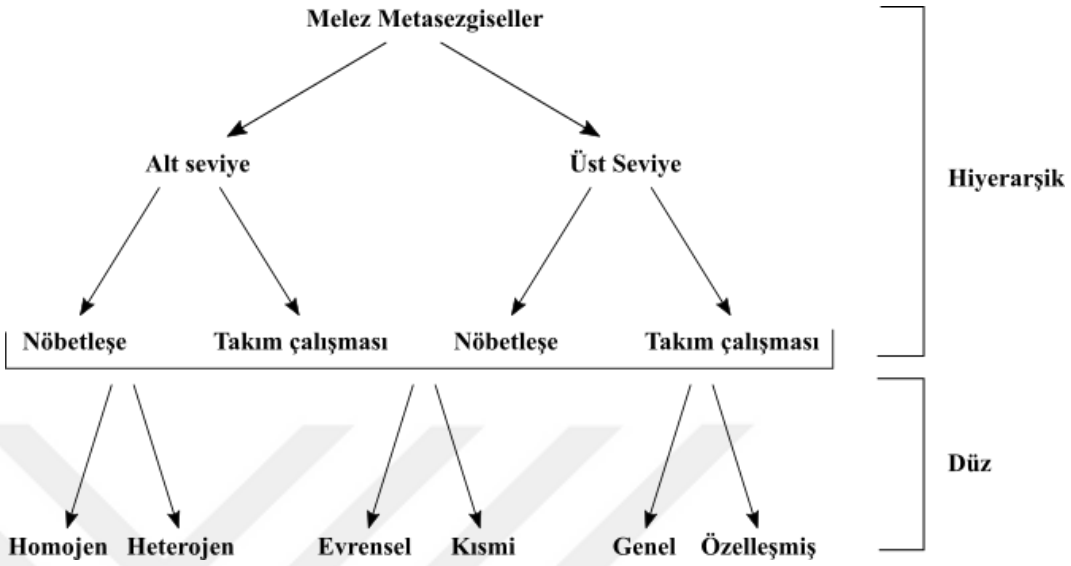
$$X_{i,j} = X_{i,j} + V_{i,j} \tag{2.9}$$

2.3 Melez Metasezgisel Algoritmalar

Melez metasezgisel algoritmalar, bir metasezgisel algoritmanın bir veya daha fazla algoritma ile birleştirilmesi sonucu oluşturulmaktadır. Melezleştirme için diğer metasezgisel algoritmalar seçilebileceği gibi, klasik sezgisel algoritmalar ya da kesin eniyileme algoritmaları da kullanılabilmektedir. Bu geniş tanımlama çerçevesinde literatürde çok çeşitli melez mimari tanımlanmıştır. Melez metasezgisellerin kapsamlı bir sınıflandırması Talbi (2002) tarafından gerçekleştirilmiştir (Şekil 2.3). Bu taksonomiye göre, hiyerarşik açıdan alt ve üst seviye olarak ayrılan melez yöntemler kendi içinde çalışma sırasına göre nöbetleşe ya da takım çalışması olarak gruplanmaktadır. Hiyerarşik bakış açısına göre melez sınıflar şu şekilde özetlenebilmektedir:

- Alt Seviye (low-level) melezleştirmede bir metasezgiselin belirli bir fonksiyonu bir başka metasezgisel tarafından gerçekleştirilmektedir.
- Yüksek Seviye (high-level) melezleştirmede yer alan metasezgiseller kendi başlarına çalışmaktadır. Birbirlerinin iç işleyişleri hakkında doğrudan bilgileri yoktur.
- Nöbetleşe (relay) melezleştirmede bir dizi metasezgisel ardı ardına çalıştırılmaktadır. Bir sonraki sırada bulunan metasezgisel girdi olarak kendinden önceki metasezgiselin çıktısını kullanır.

- Takım Çalışması (teamwork) melezleştirmede metasezgiseller işbirliği içinde çalışır. Çözüm uzayında paralel olarak arama yapan birçok metasezgisel bulunur.



Şekil 2.3. Melez metasezgisellerin taksonomisi (Talbi, 2002).

Şekil 2.3'te yer alan taksonomideki düz sınıflandırmaya göre ise melez yöntemler homojen-heterojen, evrensel-kısmi ve genel-özelleşmiş olarak ayrılmakta ve şu şekilde özetlenebilmektedir:

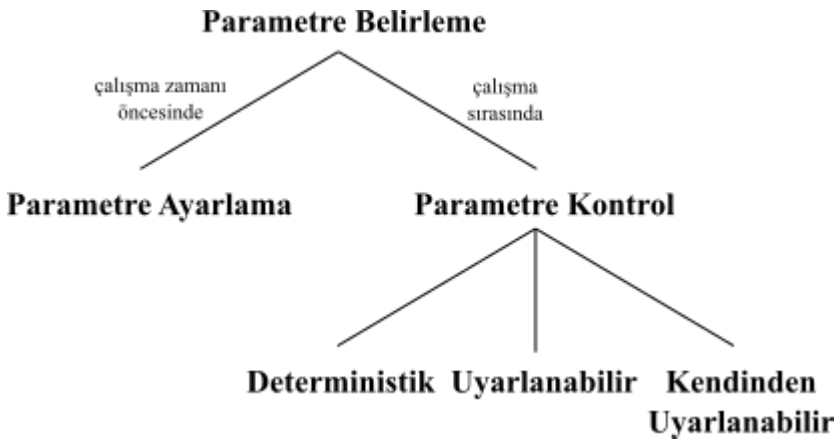
- Homojen (homogeneous) melezleştirmede kullanılan algoritmaların hepsi aynı metasezgiseli kullanmaktadır. Genellikle algoritmalar farklı parametreler ile çalıştırılmaktadır.
- Heterojen (heterogeneous) melezleştirmede farklı metasezgiseller bir arada kullanılmaktadır.
- Evrensel (global) melezleştirmede yer alan tüm algoritmalar arama uzayının tamamı üzerinde çalışmaktadır.
- Kısmi (partial) melezleştirmede problem alt parçalara ayrılır ve her bir algoritma kendi arama uzayında çalışmaktadır.
- Genel (general) melezleştirmede yer alan algoritmaların hepsi aynı eniyileme problemini hedeflemektedir.
- Özelleşmiş (specialist) melezleştirmede yer alan algoritmalar farklı problemleri çözmektedir.

3. PARAMETRE KONTROL YÖNTEMLERİ

Parametreler, eniyileme algoritmalarının performanslarını doğrudan etkilemektedir ve özelliklerine göre 4 ana başlık altında gruplanabilmektedir (Lopez-Ibanez, 2011). Bunlardan ilki olan *reel* parametreler kayan noktalı sayısal değerler alabilirken, ikinci parametre türü olan *tamsayı*'lar sadece tamsayı değerlerini almaktadır. Üçüncü tür olan *kategorik* parametrelere ise önceden belirlenmiş bir küme içerisinde yer alan değerler atanabilmektedir. Son olarak, *sıralı* parametreler kategorik parametreler ile aynı yapıda olup küme içerisindeki değerler belirli bir mantığa göre sıralanmış durumda bulunmaktadır.

Bir eniyileme algoritmasına ait parametrelerin alması gereken en uygun değerlerin belirlenmesi zor ve zaman alıcı bir iştir. Örneğin, belirlenmesi gereken 3 adet parametre ve her bir parametrenin alabileceği 10 çeşit değer olduğu düşünüldüğünde toplam 10^3=1.000 farklı birleşim ortaya çıkmaktadır. Çoğu eniyileme probleminin, tek bir parametre konfigürasyonu ile çözümü bile yoğun işlem gerektirdiğinden, olası tüm parametre birleşimlerinin denenmesi ciddi zaman alacaktır. Parametre belirlemenin bir diğer zorluğu da eniyileme algoritmalarının dinamik doğasına bağlı olarak ortaya çıkmaktadır. Buna göre; eniyileme algoritmasının çalışması boyunca farklı zamanlarda farklı parametre değerleri en uygun olabilmektedir.

Parametre belirleme yöntemleri, parametre değerlerinin belirlenme zamanına ve belirlenme yaklaşımına göre Şekil 3.1'de gösterildiği gibi sınıflandırılabilmektedir (Eiben et al., 2007). Buna göre parametreler belirlenme zamanına göre ikiye ayrılmaktadır. Parametre ayarlama (parameter tuning) yaklaşımında, parametreler algoritmanın çalışma zamanından önce belirlenmekte olup çalışma boyunca değişmemektedir. Parametre kontrol (parameter control) yaklaşımında ise parametreler algoritmanın çalışma sırasında belirlenmekte ve farklı aşamalarda farklı değerler alabilmektedir.



Şekil 3.1. Parametre belirleme yöntemleri taksonomisi.

Parametre kontrol yöntemleri kendi arasında üç alt gruba ayrılmaktadır. Bunların adları ve kısa açıklamaları (Eiben et al., 2007) aşağıda listelenmektedir.

- Determitistik parametre kontrolü: Parametrenin değerleri önceden tanımlanmış bir plana göre değişmektedir. Bu nedenle, parametre kontrol sırasında arama sürecinden toplanan herhangi bir bilgi kullanılmamaktadır.
- Uyarlanabilir parametre kontrolü: Uyarlanabilir (adaptive) parametre kontrolünde arama sürecinden elde edilen geribildirimden faydalanılmaktadır. Geribildirim sonucu elde edilen bilgiye göre parametre değerleri güncellenmektedir.
- Kendinden Uyarlanabilir parametre kontrolü: Kendinden uyarlanabilir (self-adaptive) parametre kontrolünün temel özelliği, parametrelerin kendilerinin de iyileştirilmeye çalışılan aday çözümlerin içerisine dahil edilmesidir. Böylece parametreler, aday çözümlerle eş zamanlı olarak eniyilenmiş olmaktadır.

Parametre kontrol için kullanılacak yöntemler, kullanılan parametrelerin türlerine ve algoritmaların yapılarına göre değişebildiğinden dolayı oldukça çeşitlilik göstermektedir. Bu nedenle, ilerleyen alt bölümlerde açıklanacak olan parametre kontrol yaklaşımlarının sayısı sınırlı tutulmuş olup, daha çok tez kapsamında uygulanmış ya da yapılan benzer araştırmalarda karşılaşılmış olanlara ağırlık verilmektedir.

3.1 Kategorik Parametreler İçin Kullanılan Parametre Kontrol Yaklaşımları

Parametre kontrol işleminin doğrudan melez yapı üzerinde uygulanabildiği mimariler daha çok hiyerarşik sınıflandırmaya göre üst seviyede ve nöbetleşe çalışma sistemine sahiptir (Bkz. Bölüm 2.3). Bu tip melez yapılarda, belirli bir zaman diliminde hangi metasezgisel algoritmanın uygulanacağının belirlenmesi kategorik bir parametre olup, bu parametrenin alacağı değer seçilen algoritmanın kendisi olmaktadır. Bu bölümde, literatürde Uyarlanabilir Operatör Seçimi (Adaptive Operator Selection, AOS) (DaCosta et al., 2008; Fialho, 2010) adıyla anılan ve bir algoritmada yer alan operatörlerin (alt algoritmaların) çevrimiçi (çalışma zamanı) uyarlanabilir şekilde seçilmesini sağlayan çeşitli stratejileri tanımlayan yöntemler ele anılmaktadır.

AOS, temel olarak bir eniyileme probleminin çözümünde görev alan operatörlerin performanslarını izleyerek onlar arasından en uygun olanını seçmektedir. Burada adı geçen operatör, bir algoritma bileşeni, strateji ya da alt algoritma olarak değerlendirilebilmektedir. AOS, kredi atama ve operatör seçimi olmak üzere iki aşamadan oluşmakta olup bunlar hakkında ayrıntılı bilgi takip eden alt bölümlerde yer almaktadır.

3.1.1 Kredi atama

Kredi atama (credit assignment), AOS'nin birinci aşaması olup operatörlerin kalitelerinin nasıl değerlendirileceği ile ilgilenmektedir. Operatörlerin kalitesi "ödül" terimi ile ifade edilmekte olup eniyileme probleminin uygunluk fonksiyonundaki iyileştirme ile doğru orantılı olmaktadır.

3.1.1.1 İyileştirme büyüklüğü tabanlı yöntemler

Kredi atamada kullanılabilecek en temel yaklaşım uygunluk fonksiyonundaki iyileştirme miktarının doğrudan kullanılmasıdır. Eşitlik 3.1'de bu yaklaşım gösterilmekte olup R ödül miktarını, p_f ve c_f ise sırasıyla ebeveyn ve çocuk çözümlerin uygunluk değerini ifade etmektedir.

$$R = p_f - c_f \tag{3.1}$$

Eşitlik 3.1'deki ödül hesaplama yaklaşımının dezavantajı eniyileme sürecinin farklı aşamalarında farklı ölçeklerde iyileştirmelerin gerçekleşebilmesidir. Bu durumun telafi edilebilmesi için Eşitliklik 3.2'de yer alan göreceli uygunluk değeri iyileştirmesi yaklaşımı kullanılabilmektedir (Gong et al., 2011). Bu formülde yer alan η göreceli uygunluk iyileştirmesini ve δ değeri o ana kadar bulunmuş olan en iyi uygunluk değerini ifade etmektedir.

$$\eta = \begin{cases} \frac{\delta}{c_f} . |p_f - c_f|, maksimizasyon\ için \\ \frac{c_f}{\delta} . |p_f - c_f|, minimizasyon\ için \end{cases} \tag{3.2}$$

Eşitlik 3.2'de yer alan göreceli uygunluk iyileştirmesi hesabı operatörlerin her bir uygulamasının ardından gerçekleştirilmektedir. Algoritmanın çalışması boyunca operatörlerin birçok kez uygulanabileceği göz önünde bulundurulduğu zaman, bu bilgilerin birleştirilerek toplam performansın hesaplanması gerekliliği ortaya çıkmaktadır. Gong et al. (2011) tarafından toplam ödül hesaplama konusunda dört farklı yaklaşım önerilmiş olup Eşitlik 3.3, 3.4, 3.5 ve 3.6'da tanımlanmaktadır. Bu eşitliklerde yer alan S_a, α operatörüne ait göreceli uygunluk iyileştirmesi kümesini ifade etmekte olup o ana kadar elde edilen tüm iyileştirme değerleri içermektedir. Eğer α operatörünün uygulanması sonucu iyileştirme elde edilmemişse (ya da kötüleşme olduysa) kümeye "null" anlamında sıfır değeri konulmaktadır ($\eta = 0$). Buna ek olarak, $|S_a|$ gösterimi S_a kümesinde yer alan eleman sayısını ve $r_a(t)$ ise *t.* zamandaki a operatörüne ait ödül miktarını ifade etmektedir. Eğer $|S_a| = 0$ ise ödül değeri olan $r_a(t) = 0$ olarak belirlenmektedir.

$$r_a(t) = \frac{\sum_{i=1}^{|S_a|} S_a(i)}{|S_a|} \tag{3.3}$$

$$r'_a(t) = \frac{\sum_{i=1}^{|S_a|} S_a(i)}{|S_a|};\ r_a(t) = \frac{r'_a(t)}{max_{b=1,\dots,K} r'_b(t)} \tag{3.4}$$

$$r_a(t) = max_{i=1,\dots,|S_a|} S_a(i) \tag{3.5}$$

$$r'_a(t) = max_{i=1,\dots,|S_a|} S_a(i);\ r_a(t) = \frac{r'_a(t)}{max_{b=1,\dots,K} r'_b(t)} \tag{3.6}$$

Eşitlik 3.3'te yer alan yöntem ile "Ortalama Mutlak Ödül" hesaplanmaktadır. Bu yapılırken a operatörüne ait tüm göreceli uygunluk iyileştirmelerinin ortalaması alınmaktadır. Eşitlik 3.4'teki yöntem ise "Ortalama Normalleştirilmiş Ödül" olarak adlandırılmakta ve her bir operatör için ortalama mutlak ödüller [0, 1] aralığında normalleştirilmektedir.

Eşitlik 3.5'teki yaklaşım ile "Uç Mutlak Ödül" hesaplanmaktadır. Bu yöntemde a operatörüne ait tüm göreceli uygunluk iyileştirmeleri arasından en büyük olanı seçilmektedir. Eşitlik 3.5'te ise "Uç Normalleştirilmiş Ödül" her bir operatör için ortalama mutlak ödüllerin [0, 1] aralığında normalleştirilmesi ile hesaplanmaktadır.

Göreceli uygunluk iyileştirmelerinin toplanarak tek bir ödül haline getirilebileceği bir diğer yöntem ise üstel düzeltme (exponential smoothing) yaklaşımıdır. Bu yaklaşımda (Eşitlik 3.7) a operatörünün eski ödül değerine ($r_a(t)$) yeni gelen iyileştirme miktarı ($\eta(t+1)$) düzeltme katsayısı (β) oranında eklenmektedir. Burada β 1'e ne kadar yakınsa eski ödül bilgileri o denli çabuk unutulmakta, 0'a yakın olduğu durumlarda ise yeni gelen iyileştirme bilgisinin ödül üzerindeki ağırlığı azaltılmaktadır.

$$r_a(t+1) = (1-\beta)\, r_a(t) + \beta\, \eta(t+1) \tag{3.7}$$

3.1.1.2 Karşılaştırma tabanlı yöntemler

Uygunluk değeri iyileştirme büyüklüğüne göre atamadaki en önemli problem uygunluk fonksiyonuna doğrudan bağımlılığın ortaya çıkmasıdır. Uygunluk iyileştirme miktarı, eniyileme sürecinin başlarında daha büyük olurken yakınsama noktasına yaklaştıkça daha küçük ölçeklere inmektedir. Bu durum, algoritmanın başlarında üretilen ödül değerleri yönünde bir eğilim oluşturabilmektedir. Her ne kadar normalleştirme yapılsa da bu durum, aykırı değerlerin ortaya çıkıp sisteme egemen olmasına yol açabilmektedir. Bahsedilen bu dezavantajların etkisini azaltabilmek için iyileştirme miktarlarını doğrudan kullanmak yerine onların birbirleriyle karşılaştırmasını temel alan sıralama tabanlı yöntemler önerilmiştir. Bu kapsamda Fialho et al. (2010-a) tarafından sıraların toplamı (sum of ranks, SR) ve eğri altında kalan alan (area under curve, AUC) olmak üzere iki farklı ödül hesaplama stratejisi ortaya konulmuştur. Bu iki stratejinin de ortak noktası, birer kayan çerçeve veri yapısı tutmalarıdır.

Kayan çerçeve, operatörlerin en son W adet uygulaması sonucundaki uygunluk değeri iyileştirmesi verilerini tutmaktadır. Buna göre, çerçevenin her bir elemanı bir adet operatör numarası ve o operatörün uygulanması sonucu elde edilen iyileştirme miktarını içermektedir. Çerçevede sadece en son W adet uygulama bilgisi yer aldığından, her bir adımda eklenme zamanına göre son sırada yer alan eleman silinmekte ve yerine yenisi eklenmektedir.

SR yöntemine göre ödül hesaplanırken, öncelikle kayan çerçevede yer alan operatörler iyileştirme büyüklüklerine göre sıralanmaktadır. Ardından, r sıra numarası olmak üzere, operatörlere $(W - r)$ ilk ödül değeri atanmaktadır. Bu ödül değeri daha sonra $D \in [0,1]$ olmak üzere D^r zayıflatma (decay) çarpanı ile çarpılmaktadır. Son olarak, i. operatörün t. zamanda alacağı ödül olan $SR_{i,t}$'nin değeri Eşitlik 3.8'e göre hesaplanmaktadır.

$$SR_{i,t} = \frac{\sum_{op_r=i} D^r(W-r)}{\sum_{r=1}^{W} D^r(W-r)} \tag{3.8}$$

AUC yöntemine göre ödül hesaplama ise, seçilen bir hedef operatör için Algoritma 3.1'e göre yapılmaktadır. Algoritmanın başlamasından önce W uzunluğundaki kayan çerçeve içerisinde yer alan operatörlerin ödül (kredi) değerlerine göre büyükten küçüğe doğru sıralanmış olması gerekmektedir. Ardından, listenin başından sonuna doğru operatörler sırasıyla gezilirken hedef operatöre rastlanma veya aynı ödül değerine sahip operatörlere rastlanma durumlarına göre alan bilgisi güncellenmektedir. Bu amaçla eş ödül değerine sahip operatörlerin sayısının tutulması için *tiesX* (hedef operatörler) ve *tiesY* (diğer operatörler) değişkenleri tanımlanmıştır.

Algoritma 3.1 AUC hesaplama (Fialho et al.'dan, 2010-a)

```
x ← y ← alan ← 0
for r ← 1 to W do
  Δr ← D^r (W – r)
  tiesX ← hedefOperatörleEşÖdülDeğerlerininSayısı()
  tiesY ← diğerOperatörleEşÖdülDeğerlerininSayısı()
  if tiesX > 0 or tiesY > 0 then
    Δtie ← 0
    for s ← r to (r + tiesX + tiesY) do
      Δtie ← Δtie + (D^s(W – s)) / (tiesX + tiesY)
    x ← x + tiesX * Δtie
    alan ← alan + y * Δtie * tiesX
    y ← y + tiesY * Δtie
    alan ← alan + 0,5 * Δ^2tie * tiesX * tiesY
    r ← r + tiesX + tiesY
  else if op_r = op_hedef then
    y ← y + Δr
  else
    x ← x + Δr
    alan ← alan + y * Δr
  end if
end for
return alan
```

3.1.2 Operatör seçimi

Operatör seçimi AOS'nin ikinci adımı olup, alternatif operatörler arasından seçim yapılmasını sağlamaktadır. Operatör seçimi için kullanılabilecek çeşitli yaklaşımlar bulunmakta olup bunlar literatürde olasılık teorisi kapsamında çok kollu kumar makinesi (multi-armed bandit) adıyla yer alan yöntemlerdir.

3.1.2.1 ε-greedy algoritması

ε-greedy algoritması operatör seçimi için kullanılabilecek en basit yaklaşımlardan birisidir. Sıradaki operatöre karar verileceği zaman (1 - ε) olasılıkla en yüksek kredi değerine sahip olan seçilmektedir. ε olasılıkla ise operatörler arasından rastgele seçim yapılmaktadır. Eşitlik 3.9'da (Kuleshov and Precup, 2014) bu ikili seçim kuralı yer almakta olup, p_i değeri i. operatörün seçilme olasılığını, t değeri zaman ya da iterasyon numarasını, k değeri operatör sayısını ve $\hat{\mu}_j$ ise j. operatörün t zamanına kadar elde ettiği birikimli kredi (ödül) değerini göstermektedir.

$$p_i(t+1) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{k}, & eğer\ i = argmax_{j=1,\dots,k}\ \hat{\mu}_j(t) \\ \frac{\varepsilon}{k}, & değilse. \end{cases} \tag{3.9}$$

3.1.2.2 UCB algoritması

UCB algoritması (Auer et al., 2002) keşif ve faydalanma dengesini gözeterek farklı ödül olasılık dağılımlarına sahip operatörler arasından seçim yapmaktadır. Eşitlik 3.10'da UCB1 seçim kuralı yer almaktadır. Burada $n_{i,t}$, i. operatörün t zamanına kadar kaç kere çekildiğini ve $\widehat{p}_{i,j}$ de i. operatöre ait ortalama ödül miktarını göstermektedir.

$$\widehat{p}_{i,t} + C\sqrt{\frac{2log\sum_k n_{k,t}}{n_{i,t}}} \tag{3.10}$$

Eşitlik 3.10'da yer alan ilk terim olan $\widehat{p}_{i,j}$ faydalanma özelliğini, ikinci terim ise keşif özelliğini desteklemektedir. Çünkü ilk terim o ana kadarki en yüksek ödül ortalamasına sahip kolun seçilme olasılığına katkıda bulunurken, ikinci terim daha az kullanılmış olan kolların tercih edilmesi olasılığını artırmaktadır. C katsayısı ise ikinci terimin ağırlığını ayarlamakta kullanılan bir parametredir.

3.1.2.3 Olasılık Eşleme Algoritması

Olasılık Eşleme (Probability Matching, PM) yönteminde operatör seçimini bir dizi işlem sonucu belirlemiş olduğu olasılıklar üzerinden gerçekleştirmektedir. Algoritma 3.2'de (Thierens, 2005) PM sözdekodları yer almaktadır ve operatör sayısı (K), minimum olasılık değeri (P_{min}) ve uyarlanabilirlik oranı ($\alpha \in [0,1]$) olmak üzere üç adet parametre içermektedir. P ve Q ise sırasıyla operatörlerin seçilme olasılıkları ve ödül kalitelerini tutan dizilerdir. İlkleme aşamasında olasılık değerleri operatörlere eşit olarak dağıtılmakta ve kalite değerleri aynı olacak şekilde sabit bir sayı atanmaktadır.

Algoritma 3.2 PM(P, Q, K, P_{min}, α)
For i ←1 **to** K **do** /*ilkleme*/
  P(i) ← 1/K; Q(i) ← 1
End for
While bitişKoşuluSağlanmadı **do**
  $a^s \leftarrow olasılıkOranındaSeçim(P)$
  $R_{a^s} \leftarrow ödülGetir(a^s)$
  $Q_{a^s}(t+1) \leftarrow Q_{a^s}(t) + \alpha[R_{a^s}(t) - Q_{a^s}(t)]$
  For α = 1 **to** K **do**
    $P_a(t+1) \leftarrow P_{min} + (1 - K.P_{min}) \frac{Q_a(t)}{\sum_{t=1}^{K} Q_i(t)}$
  End for
End while

Sonlanma koşuluna kadar devam eden algoritmanın ilk adımında P olasılık değerleri oranında bir adet operatör (a^s) seçilmektedir. Ardından a^s işletilerek getirdiği ödül miktarı (R_{a^s}) hesaplanmaktadır. R_{a^s} değeri önceki ödül miktarı ile α oranında birleştirilmekte ve güncel Q değerleri elde edilmektedir. Öyle ki, α değeri 0'a yaklaştıkça eski ödül değerlerinin ağırlığı korunmakta, 1'e yaklaştıkça ise yeni ödül değerinin etkisi artmaktadır. Q değerlerinin güncellenmesinin hemen ardından yeni olasılıklar kalite değerleri oranında ve en düşük P_{min} olacak şekilde hesaplanmaktadır.

3.1.2.4 Uyarlanabilir Takip Algoritması

Uyarlanabilir Takip (Adaptive Pursuit, AP) algoritması (Thierens, 2005), PM algoritmasına kıyasla değişen koşullara daha hızlı uyarlanabilmektedir. Algoritma 3.3'te AP sözdekodları yer almaktadır ve operatör sayısı (K), minimum olasılık değeri (P_{min}), kalite uyarlanabilirlik oranı ($\alpha \in [0,1]$) ve olasılık uyarlanabilirlik oranı ($\beta \in [0,1]$) olmak üzere dört adet parametre içermektedir. P ve Q ise sırasıyla operatörlerin seçilme olasılıkları ve ödül kalitelerini tutan dizilerdir. İlkleme

aşamasında olasılık değerleri operatörlere eşit olarak dağıtılmakta ve kalite değerleri aynı olacak şekilde sabit bir sayı atanmaktadır. Ayrıca olasılık güncelleme aşamasında kullanılmak üzere bir operatörün alabileceği maksimum olasılık değeri olan P_{max} hesaplanmaktadır.

Algoritma 3.3 AP(P, Q, K, P_{min}, α, β)

$P_{max} \leftarrow 1 - (K-1)P_{min}$
For i ← 1 **to** K **do** /*ilkleme*/
  P(i) ← 1/K; Q(i) ← 1
End for
While bitişKoşuluSağlanmadı **do**
  $a^s \leftarrow olasılıkOranındaSeçim(P)$
  $R_{a^s} \leftarrow ödülGetir(a^s)$
  $Q_{a^s}(t+1) \leftarrow Q_{a^s}(t) + \alpha[R_{a^s}(t) - Q_{a^s}(t)]$
  $\alpha^* \leftarrow ARGMAX_{a=1\ldots K}(Q_a(t+1))$
  $P_{\alpha^*}(t+1) \leftarrow P_{\alpha^*}(t) + \beta[P_{max} - P_{\alpha^*}(t)]$
  For α ← 1 **to** K **do**
    If $\alpha <> \alpha^*$ **then**
      $P_a(t+1) \leftarrow P_a(t) + \beta[P_{min} - P_a(t)]$
    End if
  End for
End while

AP algoritması belirli bir sonlanma koşuluna kadar operatör seçimi ve elde edilen ödül değerleri doğrultusunda olasılık güncelleme işlemlerini gerçekleştirmektedir. İlk adım olan operatör seçiminde P olasılık değerlerine göre bir adet operatör (a^s) seçilmektedir. Ardından a^s çalıştırılmakta ve getirdiği ödül miktarı olan R_{a^s} hesaplanmaktadır. R_{a^s} değeri önceki ödül miktarı ile α oranında birleştirilmekte ve güncel Q değerleri elde edilmektedir. Öyle ki, α değeri 0'a yaklaştıkça eski ödül değerlerinin ağırlığı korunmakta, 1'e yaklaştıkça ise yeni ödül değerinin etkisi artmaktadır. Olasılık güncelleme aşamasında öncelikle en yüksek kaliteye sahip operatör olan α^* belirlenmektedir. Daha sonra, α^* operatörü β oranında P_{max} değerine yaklaştırılırken diğer operatörler β oranında P_{min} değerine yaklaştırılmaktadır.

3.2 Sayısal Parametreler İçin Kullanılan Parametre Kontrol Yaklaşımları

Sayısal parametreler, metasezgisel algoritmalarda en sık karşılaşılan parametre türleridir. Çaprazlama oranı, uygunluk fonksiyonu kabul olasılığı, topluluk büyüklüğü ve ağırlıklandırma katsayıları bunlara verilebilecek örneklerden bazılarıdır. Sayısal parametreler için parametre kontrol yöntemleri, algoritma

tasarımcısı tarafından tanımlanan özel kurallar ile çok farklı şekillerde oluşturulabilmektedir. Bu nedenle, ilerleyen iki alt bölümde tez kapsamında önemli görülen parametre kontrol yaklaşımlarına odaklanılacaktır.

3.2.1 Başarılı değerlerin ortancası yaklaşımı

Bu yaklaşım, Qin et al. (2009) tarafından önerilen ve SaDE (Self Adaptive Differential Evolution) adıyla anılan algoritmada yer almakta olup DE algoritmasının çaprazlama oranı (CR) parametresinin çalışma zamanında güncellenmesinde kullanılmaktadır. CR parametresi [0, 1) aralığında değer alabilen bir parametre olduğu için bu yöntem, benzer yapıdaki herhangi başka bir parametreye de kolaylıkla uygulanabilecektir.

Algoritma 3.3 UyarlanabilirCR

```
CRHafıza dizisini ilkle
LP değerini ilkle
G ← 0
CROrtanca ← 0,5
While bitişKoşuluSağlanmadı do
  For her bir çözüm adayı için do
    If G >= LP then
      CROrtanca ← CRHafıza içerisindeki ortanca değeri bul
      CR ← NormalDağılım(CROrtanca, 0,1)'e göre rastgele değer üret
    End if
    CR değerini uygulayarak yeni çözüm üret
    If yeni çözüm eskisinden iyiyse then
      CR değerini CRHafıza dizisine ekle
    End if
  End for
  If G >= LP then
    CRHafıza dizisinden (G – LP) zamanında girilmiş değerleri sil
  End if
  G ← G + 1
End while
```

Yöntemin temel çalışma prensibi Algoritma 3.3'te yer almaktadır. Burada *CRHafıza,* güncel ve başarılı *CR* değerlerinin tutulduğu diziyi, *G* ise zaman sayacını göstermektedir. *CROrtanca,* yeni üretilecek *CR* değerleri için bir referans görevi görmektedir. Öyle ki, *CROrtanca*, 0,1 standart sapmalı bir normal dağılımın ortalama parametresi yerine geçmekte ve bu dağılımdan rastgele yeni bir *CR* değeri üretilmektedir. Burada dikkate alınması gereken bir nokta da *LP* değeridir. Bu değer ile *CRHafıza* dizisi içerisinde en son kaç adet iterasyona ait *CR* örneklerinin yer alacağı belirlenmektedir.

3.2.2 Uyarlanabilir kabul fonksiyonu

Bu yaklaşım, Alabas-Uslu and Dengiz (2011) tarafından geliştirilen ve SALS (Self-adaptive Local Search) adıyla anılan algoritmada yer almakta olup yerel arama sırasında üretilen rastgele aday çözümlerin kabul fonksiyonunu düzenlemektedir. Öyle ki; iyi çözümlerin yanı sıra, üretilen kötü çözümler de hesaplanan belirli bir eşik değerine göre kabul edilebilmektedir. Bu eşik değeri arama işleminin gidişatına göre değer alan θ parametresi ile hesaplanmaktadır. θ ne kadar büyük ise kötü çözümlerin kabul edilmesi o oranda kolaylaşmaktadır.

Algoritma 3.4 SALS

$i \leftarrow 1$; $C(L^{(i)}) \leftarrow 1$
$X_z \leftarrow$ rastgele ilk çözüm oluştur
$X \leftarrow X_z$; $X_b^{(i)} \leftarrow X_z$
While bitişKoşuluSağlanmadı **do**
  $\alpha_1 \leftarrow f(X_b^{(i)}) / f(X_z)$
  $\alpha_2 \leftarrow C(L^{(i)}) / i$
  $\theta \leftarrow 1 + \alpha_1\alpha_2$
  $i \leftarrow i + 1$; $r \leftarrow 0$
  Repeat
    $X' \leftarrow X$ komşuluğunda rastgele bir çözüm üret
    $r \leftarrow r + 1$
    If $r > X$ çözümünün toplam komşuluk sayısı **then**
      $\theta \leftarrow \theta + \alpha_1\alpha_2$
    End if
  Until $f(X') <= \theta\, f(X)$
  If $f(X') < f(X_b^{(i)}) /$ **then**
    $C(L^{(i)}) \leftarrow C(L^{(i)}) + 1$; $X_b^{(i)} \leftarrow X'$
  End if
  $X \leftarrow X'$
End while

Algoritma 3.1'de SALS algoritmasının adımları yer almaktadır. Burada *i*, iterasyon numarasını, $C(L^{(i)})$, i. iterasyona kadar en iyi çözümün kaç kere iyileştirildiğini, X_z, başlangıç çözümünü ve $X_b^{(i)}$ ise en iyi çözümü göstermektedir. θ değeri α_1 ve α_2 olmak üzere 0 ve 1 aralığında değer alan iki değişken tarafından belirlenmektedir. Bunlardan α_1 değişkeni en iyi çözüm iyileştirildikçe azalırken α_2 ise artmaktadır. θ değeri algoritma ilerledikçe 0'a yakınsamaktadır. Algoritmanın belirli bir aşamasında, komşu çözüm üretilirken eğer olası tüm denemeler sonuçsuz kaldıysa, θ, $\alpha_1\alpha_2$ kadar artırılarak yerel en iyiden çıkılmaya çalışılmaktadır.

4. ÖNCEKİ ÇALIŞMALAR

Sürekli eniyileme problemlerinin çözümünde sıklıkla kullanılan metasezgisellerin başında DE, PSO ve ABC gelmektedir. Literatürde bu algoritmaların melezleştirilerek kullanımı oldukça yaygındır. PSO ve ABC algoritmalarının melezleştirilmesine yönelik yapılan çalışmalardan birisi Kıran and Gündüz'e (2013) aittir. Bu çalışmada PSO ve ABC algoritmaları iki ayrı topluluk üzerinde paralel olarak çalıştırılmaktadır. Her iki algoritma da sonlandığında, bulmuş oldukları en iyi çözümler çaprazlanarak yeni bir çözüm elde edilmektedir. Çaprazlanmış çözüm bir sonraki tur için her iki algoritmaya da aktarılmaktadır. PSO'nun güncelleme formülünde evrensel en iyi çözüm etki ettiğinden, her iki topluluğun bir ölçüde işbirliği içerisinde evrilmesi sağlanmaktadır. Li et al. (2015) de PSO'nun güçlü faydalanma özelliği ile ABC'nin güçlü keşif özelliğini birleştirmek düşüncesiyle PS-ABC adını verdikleri bir algoritma önermektedirler. Bu algoritmada PSO, gözlemci arılar ve değiştirilmiş izci arılar olmak üzere üç ana aşama bulunmaktadır. Bir diğer örnekte El-Abd (2011), ABC'yi PSO içerisinde yer alan parçacıkların kendi en iyi konumlarını iyileştirmek için kullanan bir melez yöntem geliştirmiştir.

DE ve ABC metasezgisellerinin birlikte kullanıldığı bir çalışmada (Li et al, 2013) Optimal Reaktif Güç Akışı probleminin çözümüne yönelik DE-ABC adında bir melez algoritma geliştirmişlerdir. Bu algoritmada, temel DE işlemlerinin ardından topluluktaki bireylere evrim sürecini hızlandırıcı arı kolonisi yaklaşımı uygulanmaktadır. Yang et al. (2013) ise ABC-DE ismini verdikleri algoritmalarında ABC'nin işçi arılar aşaması için DE'nin çaprazlama ve mutasyon prosedürlerini kullanmaktadırlar. Gözlemci arılar aşaması ise orijinal haliyle bırakılmaktadır. Geliştirdikleri yöntemi hem klasik test fonksiyonları üzerinde hem de Zaman Modüllü Diziler Desen Sentezi üzerinde uygulamışlardır.

Zhang and Xie (2003), DEPSO adını verdikleri algoritmalarında PSO ve DE operatörlerini tek ve çift iterasyon numaralarına göre değişmeli olarak uygulamaktadırlar. PSO ve DE metasezgisellerini melezleştiren bir diğer çalışmada Sayah and Hamouda (2013), klasik DE algoritmasının mutasyon aşamasında her bir birey için iki çözüm üretmektedir. Bunlardan birisi klasik DE operatörlerini kullanırken diğeri PSO güncelleme kurallarını kullanmaktadır. Daha sonra, üretilen iki çözüm adayı arasından seçim yapılmaktadır.

ABC, DE ve PSO metasezgisellerinin tümünün bir arada kullanıldığı bir algoritma ise Gokalp and Ugur (2017) tarafından gerçekleştirilmiştir. Bu çalışmada parametre kontrol tekniği uygulanmamakta, ilgili metasezgisellerin çalışma sırası en başta ayarlanıp çalışma süresince sabit bırakılmaktadır.

Sürekli eniyileme problemlerinin çözümüne yönelik metasezgisel yaklaşımlar bir reel sayı vektörü şeklinde ifade edilen çözüm adaylarının mutasyon ve çaprazlama ile değiştirilmesi üzerine kuruludur. Bu durum, farklı problem tiplerine göre değişen başarı gösterebilen çok sayıda güncelleme stratejisinin ortaya çıkmasına yol açmaktadır. Bu nedenle, literatürde parametre kontrol yöntemlerinin sürekli eniyileme problemleri üzerine uygulandığı alanların önemli bir bölümü uygun strateji seçimine yöneliktir.

DE algoritması için uyarlanabilir strateji seçimini konu alan önemli yaklaşımlardan birisi SaDE (Qin et al., 2009) algoritmasıdır. Bu algoritmada 4 farklı mutasyon stratejisi yer almaktadır. Stratejilerin güncel uygulamalarına ait başarı ve başarısızlık sayıları bu amaçla oluşturulmuş iki adet veri yapısında tutulmaktadır. İlerleyen iterasyonlarda hangi stratejinin kullanılacağına başarı oranlarını temel alan olasılıklı bir seçim ile karar verilmektedir. Çaprazlama oranı (CR) sayısal parametresinin değeri ise başarılı değerlerin ortancası yaklaşımı ile uyarlanabilir olarak belirlenmektedir. Zhang and Sanderson (2009) benzer bir çalışma ile JADE algoritmasını önermişlerdir. Bu çalışmada, SaDE algoritmasından farklı olarak “DE/current-to-pbest/1” adlı tek bir mutasyon stratejisi kullanılmaktadır. Parametre kontrolü işlemi, CR ve diferansiyel ağırlık katsayısı (F) parametreleri üzerinde uygulanmaktadır. Öyle ki, bir önceki iterasyonda başarılı olmuş CR ve F değerleri ile hesaplanan ortalama değerleri, yeni üretilecek olan parametre değerleri için belirleyici olmaktadır. DE için strateji seçiminde farklı bir yaklaşım Gong et al (2010) tarafından getirilmiş olup, daha önce SaDE için kullanılmış olan 4 adet mutasyon stratejisinin olasılık eşleme yöntemiyle seçimi sağlanmıştır. Olasılık eşleme algoritması ile seçim yapılırken, 4 farklı uygunluk değeri tabanlı kredi atama yönteminden birisi ile hesaplanan kredi değerleri kullanılmaktadır. Fialho et al. (2010-b) ise Gong et al. (2010)’dan farklı olarak kredi atama işlemini karşılaştırma tabanlı başarı oranları hesabı yapmakta ve strateji seçiminde UCB algoritmasını kullanmaktadırlar.

Uyarlanabilir strateji seçimi ABC metasezgiseli için de uygulanmaktadır. Örneğin Wang et al. (2014), klasik ABC arama stratejisine ek olarak iki farklı strateji kullanmışlardır. Her bir çözümün son elemanı, bu üç stratejiden hangisinin

kullanılacağını belirleyecek şekilde tasarlanmıştır. Başarılı değişimlere yol açan stratejiler ilgili çözümde korunmakta olup başarısız olanların yerine diğer 2 stratejiden rastgele birisi yerleştirilmektedir. Böylece, topluluk genelinde daha başarılı olan stratejilerin daha çok kullanımı kendinden uyarlanabilir bir şekilde sağlanmaktadır. Uyarlanabilir stratejiler konusundaki bir diğer çalışmada (Yavuz et al., 2016), var olan stratejilerden seçim yapmak yerine onların üretilmesi söz konusudur. Öyle ki, bir ABC stratejisi 4 farklı terimin birleşimi şeklinde tanımlanmakta ve her bir terimin alabileceği değerler önceden belirlenmektedir. Bu değerlerin rastgele birleşimi ile bir strateji havuzu oluşturulmakta ve ilerleyen iterasyonlarda bu havuzdan başarısız olanlar kademeli olarak elenmektedir. Böylece probleme uygun yapıdaki stratejilerin varlığını sürdürmesi, diğerlerinin ise elenmesi sağlanmaktadır.

Literatürde Araç Rotalama Problemi (Vehicle Routing Problem, VRP) için çok başlangıçlı stratejiyi kullanan çeşitli çalışmalar bulunmaktadır. Braysy et al. (2004) zaman kısıtlamalı VRP için üç aşamadan oluşan ve çok başlangıçlı yapıya sahip bir yerel arama sezgiseli önermektedirler. İlk aşamada başlangıç çözümü oluşturulmakta ve içerdiği rotaların sayısı azaltılmaya çalışılmaktadır. İkinci aşamada CROSS-Exchange ile rotaların uzunlukları kısaltılmaya çalışılmaktadır. İleri eniyileme olarak adlandırılan üçüncü aşamada ise SA metasezgiselinin bir türevi olan Eşik Kabulü (Threshold Accepting, TA) (Dueck and Scheuer, 1990) algoritması kullanılarak eniyilenmektedir. Belirli bir iterasyon sayısı boyunca en iyi çözüm iyileştirilememişse, algoritma eldeki en iyi çözümü kullanarak ve gerekli parametreleri ilkleyerek üçüncü aşamayı tekrar etmektedir. Bir diğer çalışmada Duhamel et al. (2011) iki boyutlu müşteri talepleri olan VRP için çok başlangıçlı bir Evrimsel Yerel Arama (Evolutionary Local Search, ELS) algoritması önermektedirler. Çok başlangıçlı yapı için GRASP metasezgiselinden faydalanmaktadırlar. Bu konuda gerçekleştirilen başka bir çalışmada (Michallet et al., 2014) ise zaman kısıtlamalı periyodik VRP için çok başlangıçlı bir ILS algoritması önerilmektedir. ILS algoritmasının yerel en iyi bölgesinde takıldığı durumlarda yeniden başlatma yöntemi uygulanmaktadır.

VRP'nin çözümüne yönelik ILS temelli çeşitli çalışmalar bulunmaktadır. Bu kapsamda Subramanian et al. (2010) eşzamanlı toplama ve dağıtımlı VRP için ILS metasezgiseli tabanlı bir yöntem önermektedirler. Algoritmalarında ILS ve rastgele sıralı VND metasezgiseli melez bir şekilde kullanılmaktadır. Benzer şekilde Penna et al. (2013) tarafından gerçekleştirilen çalışmada ILS ve rastgele sıralı VND yöntemleri birlikte kullanılmaktadır. Ancak bu kez geliştirilen algoritma heterojen

filolu VRP üzerinde uygulanmaktadır. ILS metasezgiseli kullanılarak gerçekleştirilen bir diğer çalışmada (Tang and Wang, 2006) ödül toplamalı VRP için Çok Geniş Ölçekli Komşuluk Yapısı (Very Large-scale Neighborhood Structure, VLNS) (Ahuja et al., 2000) temel alınmaktadır. Bu amaçla döngüsel transfer adı verilen komşuluk yapısı tanımlanmaktadır.

VRP için uygulanan parametre kontrol yaklaşımları daha çok uyarlanabilir yöntemler başlığı altında sunulmaktadır. Örneğin, Ropke and Pisinger (2006) Geniş Komşuluk Arama (Large Neighborhood Search, LNS) (Shaw, 1997) yöntemini zaman kısıtlamalı toplama ve dağıtım problemi için uygulayan ALNS yöntemini geliştirmişlerdir. Bu çalışmada, LNS için bir dizi silme ve ekleme prosedürleri tanımlanmakta her birisi için performanslarına dayalı olarak uyarlanabilir ağırlık değerleri tutulmaktadır. Hangi ekleme/silme prosedürünün seçileceği ise bu ağırlık değerlerine göre rulet tekerleği seçimi ile belirlenmektedir. Benzer şekilde, uyarlanabilir LNS yaklaşımını Hemmelmayr et al. (2012) bu kez iki kademeli araç rotalama problemi üzerine uygulamaktadırlar. Meignan et al. (2010) ise eniyileme etmenlerinden oluşan bir yapı ortaya koymaktadırlar. Buna göre, bir dizi alt sezgisel özelliklerine göre yoğunlaşma ve çeşitlendirme etiketleri altında sınıflandırılmaktadır. İçerisinde barındırdığı öğrenme mekanizması ile bunların hangi sırada uygulanacağına karar verilmektedir. ALNS algoritmasına benzer bir şekilde rulet tekerleği seçimi prensibiyle alt seviye sezgiseller seçilmektedir. ALNS'den farklı olarak etmenler arası bilgi paylaşımı barındırmaktadır.

Parametre kontrol yönteminin VRP için uygulandığı bir başka alan da kabul fonksiyonudur. Alabas-Uslu and Dengiz (2011) gerçekleştirdikleri çalışmada hiçbir parametre ayarlaması gerektirmeyen kendinden uyarlanabilir bir yerel arama algoritması önerilmektedirler. SALS adını verdikleri algoritmada tanımlanmış bir dizi komşuluk yapısına göre üretilen aday çözümler, değeri uyarlanabilir olarak belirlenen bir kabul eşiği ile kabul veya reddedilmektedir. Avci and Topaloglu (2015) ise SALS algoritmasındaki eşik kabul yaklaşımını VND yerel araması ile birleştirerek eşzamanlı ve karışık toplama ve dağıtımlı VRP'nin çözümüne yönelik bir yöntem ortaya koymaktadırlar.

5. YÜKSEK SEVİYELİ VE UYARLANABİLİR METASEZGİSEL SEÇİMİ: HAMS_ABCxDExPSO

Tez kapsamında geliştirilen HAMS_ABCxDExPSO melez metasezgiseli, sınır kısıtlamalı sürekli eniyileme problemlerinin çözümüne yöneliktir. Bu yöntem, ABC, DE ve PSO metasezgisellerinin yüksek seviyeli ve nöbetleşe çalışacak şekilde bir araya getirilmesinden oluşmaktadır. Bu üç metasezgiselin çalışma sıraları bir parametre olarak önceden belirlenmeyip, algoritmanın çalışma zamanında uyarlanabilir parametre kontrolü ile ayarlanmaktadır. Böylece, eniyileme sürecinin gidişatına göre uygun metasezgiseli seçmeyi hedefleyen bir yöntem ortaya konulmaktadır.

5.1 Sınır Kısıtlamalı ve Tek Amaçlı Sürekli Eniyileme Problemleri

Sınır kısıtlamalı ve tek amaçlı sürekli eniyileme problemlerinde, $f: \mathbb{R}^n \rightarrow \mathbb{R}$ şeklinde ifade edilen bir amaç fonksiyonunu minimize edecek $x = (x_1, x_2, \dots, x_n)$ reel sayı dizisi aranmaktadır (Eşitlik 5.1). Burada, *x* dizisindeki her bir elemanın alabileceği değerler birer alt (L) ve üst (U) limit değeri ile sınırlandırılmaktadır (Eşitlik 5.2).

$$Min. f(x), x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n \tag{5.1}$$

$$L_i \leq x_i \leq U_i, i = 1,2, \dots, n \tag{5.2}$$

5.2 Algoritma Tasarımı

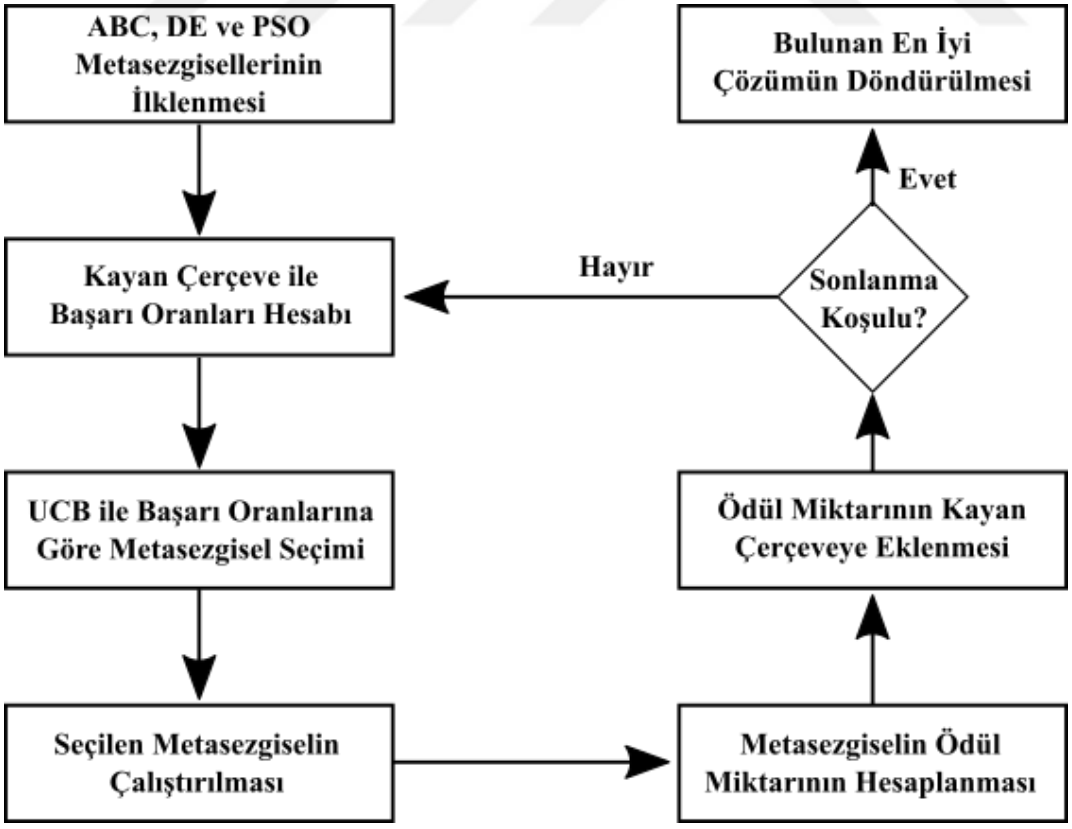
Geliştirilen HAMS_ABCxDExPSO algoritması, sürekli eniyilemede yaygın olarak tercih edilen ABC, DE ve PSO metasezgisellerinden oluşmaktadır. ABC için Karaboğa (2005) tarafından önerilmiş standart algoritma, DE için "DE/rand/1" stratejisi (Storn and Price, 1997) ve PSO için ise Standart PSO 2011 algoritması (SPSO-2011) (Zambrano-Bigiarini et al., 2013) kullanılmıştır.

Önerilen yöntemde metasezgiseller yüksek seviyeli ve nöbetleşe çalışacak biçimde melezleştirilmişlerdir. Yüksek seviyeli melezleştirme yaklaşımı, algoritmaların birbirlerinden bağımsız olarak çalışmasını gerektirmektedir. Nöbetleşe çalışma prensibinde ise algoritmalar sırayla çalıştırılmaktadır. Bir başka deyişle, belirli bir zaman diliminde tek bir metasezgisel çalışmakta ve çalışması

sonlandıktan sonra görevi bir sonraki metasezgisele bırakmaktadır. Sıradaki metasezgiselin seçimi ise algoritmaların yakın zamanda elde ettikleri başarı oranları temel alınarak UCB karar verme yöntemine (Bkz. Bölüm 3.1.2) göre yapılmaktadır. Her bir algoritmanın güncel başarı oranı, kayan çerçeve (Bkz. Bölüm 3.1.1) veri yapısında bulunan ödül değerlerinin sıralamasını dikkate alan SR yöntemine göre hesaplanmaktadır.

Burada açıklandığı şekliyle HAMS_ABCxDExPSO'nun genel işleyişi Şekil 5.1'de sunulmaktadır. Seçilen bir metasezgisel, kendisinden önceki metasezgiselin eniyilemiş olduğu topluluğu (çözüm kümesi) devralarak çalışmaya başlamaktadır. Böylece, başlangıçta rastgele oluşturulmuş olan topluluk farklı aşamalarda farklı metasezgiseller tarafından iyileştirilmeye devam etmektedir. Bir metasezgiselin çalışmasının tamamlanmasının ardından, topluluktaki en iyi bireyin uygunluk değerini ne kadar artırdığına bağlı olarak ödül miktarı hesaplanmakta ve başarı değerlerinin güncellenmesi için kayan çerçeve adlı veri yapısına eklenmektedir.

Algoritmanın sonlanma koşulu olarak maksimum fonksiyon hesaplama sayısı kullanılmaktadır. Öyle ki, her bir metasezgiselin fonksiyon hesaplama sayıları (fhs) toplanmakta ve sınıra ulaştığı zaman algoritmanın çalışması durdurulmaktadır.



Şekil 5.1.Geliştirilen HAMS_ABCxDExPSO mimarisinin genel yapısı.

Algoritma 5.1 HAMS_ABCxDExPSO (L, W, D, C, topluluk, makFhs)

```
ABC ← varsayılan parametreler ile ABC metasezgiselini oluştur
DE ← varsayılan parametreler ile DE metasezgiselini oluştur
PSO ← varsayılan parametreler ile PSO metasezgiselini oluştur
metasezgiseller[] ← ABC, DE, PSO
kç ← kayanÇerçeveyiOluştur(W, K)
S* ← topluluk.enİyiBirey
f* ← uygunlukDeğeri(S*)
fhs ← 0
while kç.elemanSayısı < W do              //kayan çerçevenin doldurulması
  indisKümesi ← {0, 1, …, K}
  while indisKümesi ≠ ∅ do
    opNo ← rastgeleElemanSeç(indisKümesi)
    ms ← metasezgiseller[opNo]
    topluluk ← ms.çalıştır(topluluk, L)
    S ← topluluk.enİyiBirey
    f ← uygunlukDeğeri(S)
    delta ← f*-f
    kç.ekle(delta, k)
    if delta > 0 then
      S* = S
      f* ← f
    end if
    fhs ← fhs + L
    indisKümesi ← indisKümesi \ {opNo}
  end while
end while
while fhs < makFhs do                     //ana döngü
  SR[] ← SRHesapla(kç.sıralıListe)
  opNo ← UCB.operatörNoSeç(SR, C)
  ms ← metasezgiseller[opNo]
  topluluk ← ms.çalıştır(topluluk, L)
  S ← topluluk.enİyiBirey
  f ← uygunlukDeğeri(S)
  delta ← f*-f
  kç.ekle(delta, k)
  if delta > 0 then
    S* = S
    f* ← f
  end if
  fhs ← fhs + L
  L ← L * 1.05
end while
return S*
```

Önerilen HAMS_ABCxDExPSO melez metasezgiseline ait sözde kod Algoritma 5.1'de yer almaktadır. İlkleme işlemlerinin başlangıcında ABC, DE ve PSO metasezgiselleri varsayılan parametreler ile oluşturulmaktadır. Bir sonraki adımda kayan çerçeve (kç) veri yapısı çerçeve büyüklüğü (W) ve metasezgisel

sayısı (K) parametreleri ile oluşturulmaktadır. Ardından, rastgele sırada ve olabildiğince eşit sayıda seçilen metasezgisellerin L fhs ile çalıştırılması sonucu elde edilen değerler ile kç veri yapısı ilklenmektedir. Algoritmanın ana döngüsünde ise maksimum fhs değerine ulaşılana kadar her bir adımda kç'den bir metasezgisel seçilip L fhs boyunca çalıştırılmaktadır. Ancak, L değeri sabit kalmayıp her bir döngüde 1.05 ile çarpılarak büyütülmektedir. Gerçekleştirilen ön deneysel çalışma sonucu elde edilen bu katsayıyı kullanmanın sebebi, algoritmanın sahip olduğu toplam fhs bütçesinin, alt metasezgisel performanslarının daha da belirginleştiği ilerleyen iterasyonlarda daha fazla kullanılmasını sağlamaktır. Arama süreci sonlandıktan sonra, o ana kadar bulunmuş en iyi çözüm (S*) döndürülmektedir.

Kayan çerçeve veri yapısı kullanılarak başarı oranlarının (SR) hesaplanması işleminin alt adımları Algoritma 5.2'de sunulmaktadır. "sıralıListe" adlı dizide, kç içerisinde yer alan operatör numaralarının performanslarına (delta değerleri) göre büyükten küçüğe doğru sıralanmış olduğu varsayılmaktadır. r sıra numarası olmak üzere, bir operatörün SR büyüklüğü, kç'de her bulunduğu konum için hesaplanan $D^r(W-r)$ değerlerinin toplamının genel toplama bölümü şeklinde hesaplanmaktadır ($D \in [0,1]$). Dikkat edilirse, ilk sıralarda yer alan operatör girdileri daha yüksek SR değerleri kazanmaktadır.

Algoritma 5.2 SRHesapla(sıralıListe)

```
for i ← 0 to K do
  SR[i] ← 0
end for
if sıralıListe.Uzunluk < W then
  return SR
end if
toplamSR ← 0
for r ← 1 to W do
  başarıDeğeri ← D^r(W−r)
  opNo ← operatörNoGetir(r)
  SR[opNo] ← SR[opNo] + başarıDeğeri
  toplamSR ← başarıDeğeri
end for
for i ← 0 to K do
  SR[i] ← SR[i] / toplamSR
end for
return SR
```

6. ÇOK BAŞLANGIÇLI VE UYARLANABİLİR KABUL FONKSİYONLU ILS-VND MELEZ METASEZGİSELİ: MA_ILSxVND

Tez projesi kapsamında geliştirilen Çok Başlangıçlı ve Uyarlanabilir Kabul Fonksiyonlu ILS-VND Melez Metasezgiseli (MA_ILSxVND), kombinasyonel eniyileme alanında önemli bir yeri olan Kapasiteli Araç Rotalama Probleminin çözümüne yöneliktir. Bu yöntemde, ILS metasezgiseli ana eniyileme algoritması olarak kullanılmakta olup yerel arama aşamasında VND metasezgiseli görev almaktadır. Önerilen MA_ILSxVND melez algoritmasının kabul fonksiyonu için uyarlanabilir parametre kontrolü uygulanmıştır.

6.1 Kapasiteli Araç Rotalama Problemi

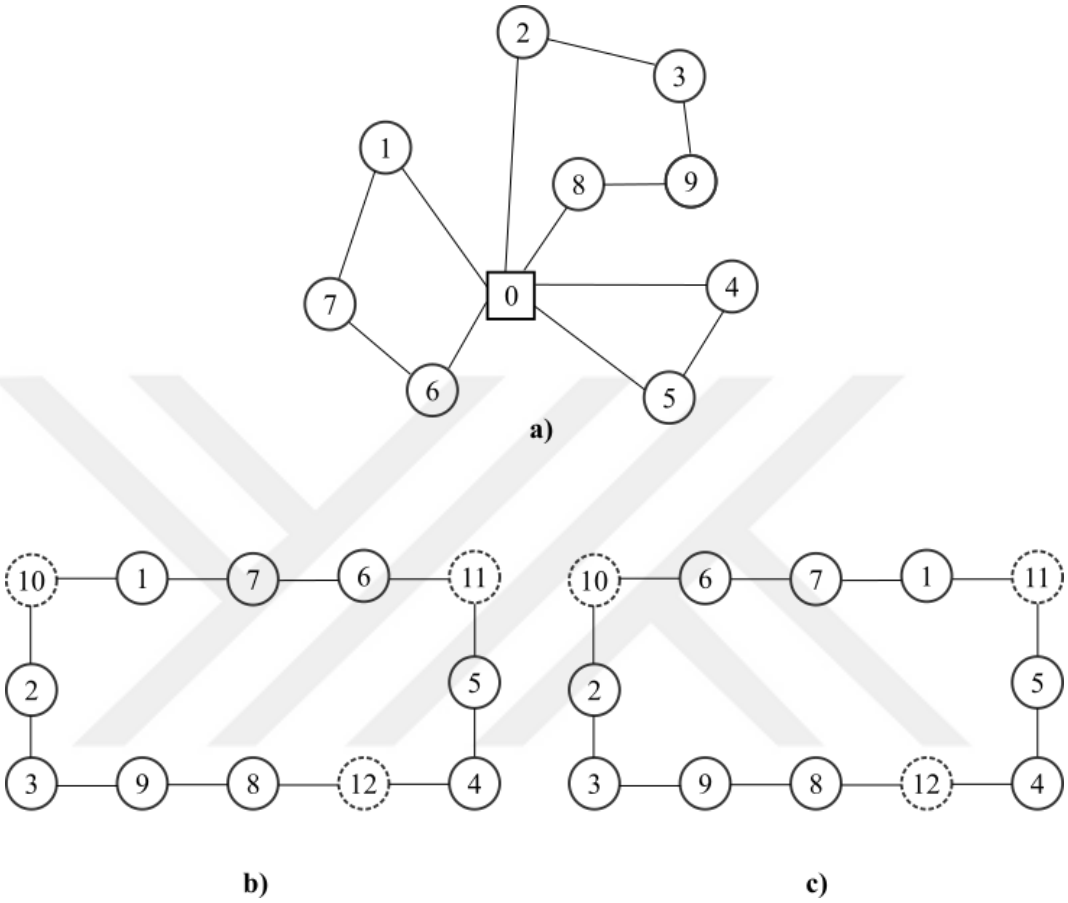
Kapasiteli Araç Rotalama Problemi (Capacitated Vehicle Routing Problem, CVRP) ilk olarak Dantzig and Ramser (1959) tarafından "The truck dispatcing problem" adı altında ortaya konulmuştur. CVRP kısaca şöyle tanımlanabilir (Laporte et al., 2000):

G = (V, E) bir tam yönsüz çizge olmak üzere $V = \{0, \dots, n\}$ köşe (tepe) setini, $E = \{(i,j): i,j \in V, i < j\}$ ise kenar (ayrıt) setini temsil etmektedir. Burada, 0. köşe depoyu gösterirken kalan köşeler müşterileri temsil etmektedir. V\{0} içerisinde yer alan her bir köşe negatif olmayan q_i talebi ile ilişkilendirilirken (i,j) ile gösterilen her bir kenar da c_{ij} kadar negatif olmayan maliyete sahiptir. CVRP'de amaç, toplam maliyeti en az olmak üzere m adet araç rotası (tur) bulmaktır. Bu turlardan her birisi depo noktasından başlayıp müşterilere uğradıktan sonra tekrar depo noktasına dönmek zorundadır. Her bir rotadaki müşterilerin toplam talebi araç kapasitesi olan Q değerini geçememektedir. Tüm rotalar birlikte düşünüldüğünde, her bir müşteriye tam olarak bir kere uğranılması gerekmektedir. Ayrıca, herhangi bir rotanın uzunluğu, maksimum rota uzunluğu olan L'yi geçememektedir.

6.2 CVRP Çözümlerinin Temsili

Tez çalışmasında, CVRP çözümlerinin temsili için rotalar arasına yapay depo noktaları yerleştiren büyük tur temsili (Irnich et al., 2006) kullanılmaktadır. $R^i = (0, v_1^i, v_2^i, \dots, v_{p_i}^i, 0), i \in \{1,2,\dots,F\}, p_i \in \mathbb{N}$ şeklinde tanımlanan ve F adet farklı turdan oluşan bir tur planı (problemin olası bir çözümü) için büyük tur temsili $x = (0_1, v_1^1, \dots, v_{p_1}^1, 0_2, v_1^2, \dots, v_{p_2}^2, 0_3, v_1^3, \dots, \dots, 0_F, v_1^F, \dots, v_{p_F}^1, 0_1)$ biçiminde ifade

edilmektedir. Burada, $0_1, 0_2, \ldots, 0_F$ şeklinde ifade edilen noktaların her biri rotaları birbirinden ayırmak için kullanılmakta olup, aynı depo noktasına işaret etmektedir. Büyük turun bitiş ve başlangıç noktası aynı depo (0_1) olarak tanımlandığından, dairesel bir yapı söz konusu olmaktadır.



Şekil 6.1. CVRP çözümlerinin temsili a) örnek bir CVRP çözümü, b) eşdeğer büyük tur temsili, c) bir diğer eşdeğer büyük tur temsili.

9 müşteri ve 3 rotadan oluşan örnek bir CVRP çözümü Şekil 6.1 (a)'da yer almaktadır. Burada, depo noktası 0 numara ile kare içerisinde gösterilirken, müşteriler 1'den 9'a kadar numaralandırılmışlardır. Şekil 6.1 (b)'de bu örnek çözümün büyük tur ile temsili yer almaktadır. Kesikli çizgili gösterimler depo noktasına karşılık gelmektedir. Bu gösterimde rotaların hangi yönden dolaşıldığının bir önemi olmamaktadır. Örneğin Şekil 6.1 (b) ve (c) birbirlerinden farklı olarak sırasıyla 1-7-6 ve 7-6-1 rotalarını içerseler de temelde aynı çözümü temsil etmektedirler.

6.3 Algoritma Tasarımı

ILS metasezgiseli tek çözüm tabanlı bir metasezgisel algoritma olup, sarsım ve yerel arama olmak üzere iki ardışık prosedür uygulayarak aday çözümler elde etmektedir (Bkz. Bölüm 2.1.5). Aday çözümlerin yeni çözüm olarak kabul edilip edilmeyeceğine ise kabul fonksiyonu tarafından karar verilmektedir. Adı geçen bu üç temel bileşen: **sarsım prosedürü**, **yerel arama prosedürü** ve **kabul fonksiyonu**, ILS algoritma tasarımında dikkate alınması gereken en önemli öğelerdir.

Geliştirilen MA_ILSxVND algoritmasında, ILS'nin yerel arama aşamasında VND metasezgiseli uygulanarak melez bir yapı kurulmuştur. Kabul fonksiyonu aşamasında ise uyarlanabilir parametre kontrolü tabanlı yöntem eklenmiştir. Ayrıca, algoritmanın durağanlığa girdiği durumlara yönelik yeniden başlatma stratejisi eklenerek arama uzayının daha geniş kapsamda keşfi sağlanmıştır.

MA_ILSxVND algoritmasına ait sözde kod Algoritma 6.1'de gösterilmektedir. Algoritmada $pKatsayı, thKatsayı, k$ ve bi olmak üzere 4 adet parametre bulunmaktadır ve bunlar aşağıdaki gibi özetlenmektedir:

- ***pKatsayı***: Sarsım büyüklüğünün (p) hesaplanmasında belirleyici olan parametredir. Deneysel çalışmalardan edinilen tecrübe doğrultusunda sarsım büyüklüğünün müşteri sayısının karekökü civarında değer alması gerektiği çıkarımı yapılmıştır. Bu nedenle p değeri, $pKatsayı\ x\ \sqrt{n}$ çarpımı ile belirlenmektedir. Böylece problem boyutuna bağlı ölçeklenebilirlik sağlanmaktadır.
- ***thKatsayı***: Yeniden başlatma stratejisini yöneten eşik değerinin (th) hesaplanmasında belirleyicidir. Deneysel çalışmalardan edinilen tecrübe doğrultusunda eşik değerinin müşteri sayısının katı şeklinde hesaplanması gerektiği çıkarımı yapılmıştır. Bu nedenle th değeri, $thKatsayı\ x\ n$ çarpımı ile belirlenmektedir. Böylece problem boyutuna bağlı ölçeklenebilirlik sağlanmaktadır.
- ***k***: VND içerisindeki *Or-opt* ve *string-exchange* komşuluklarının parça uzunluğu sınırıdır. Öyle ki, çözüm içerisinden değişim için seçilen müşteri zinciri uzunluğu 1'den k'ye kadar değişebilmektedir.
- ***bi***: VND içerisinde yer alan komşuluklarda yerel arama yapılırken en iyi ya da ilk iyileştiren değişimlerden hangisinin seçileceğini kontrol eden mantıksal parametredir. Algoritma tarafından "bi = true" olması

durumunda en iyi iyileştiren, "*bi* = false" olması durumunda ise ilk iyileştiren strateji benimsenmektedir.

Algoritma 6.1 MA_ILSxVND (pKatsayı, thKatsayı, k, bi)

```
p ← ⌊pKatsayı x √n⌋;
th ←thKatsayı x n;
sayaç ← 0
S_z←Paralel kazanç algoritması ile başlangıç çözümü üret      //Bkz. Bölüm 0
S ← S_z;  S* ← S_z;  S_eniyi← S_z
while sonlanma koşulu sağlanmadı do
  if sayaç = th then                    //Bkz. Bölüm 6.3.5
    S ← S_z
    S* ← S
    i ← 1
    sayaç ← 0
  end if
  S' ← CE(S, p)                         //Bkz. Bölüm 6.3.2
  S''← RVND(S', k, bi)                  //Bkz. Bölüm 6.3.3
  α1 ← f(S*)/f(S_z)
  α2 ← 1/i
  θ ← 1 + α1α2
  i ← i + 1
  if f(S'') < f(S) then
    S ← S''
    if f(S) < f(S*) then
      S* ← S
      sayaç ← 0
      if f(S*) < f(S_eniyi) then
        S_eniyi ← S*
      end if
    else sayaç ← sayaç + 1
    end if
  else
    sayaç ← sayaç + 1
    if f(S'') ≤ θf(S*) then             //Bkz. Bölüm 6.3.4
      S ← S''
    end if
  end if
end while
return S_eniyi
```

Algoritmada n, müşteri sayısına karşılık gelmektedir. Başlangıç çözümü ve şimdiki çözüm ise sırasıyla S_z ve S değişkenleriyle temsil edilmektedir. S^*, son yeniden başlatmadan bu yana elde edilen en iyi çözümü ifade ederken S_{eniyi}, algoritmanın başından bu yana elde edilen en iyi çözümü göstermektedir. S'nin sarsım aşamasında değiştirilmesi sonucu S' adlı çözüm elde edilmektedir. S' ise

daha sonra yerel arama ile iyileştirilerek S'' çözümüne çevrilmektedir. S'' çözümünün uygunluk değerinin S'den iyi olması durumunda doğrudan yeni çözüm olarak kabul edilmektedir. Aksi takdirde uyarlanabilir kabul fonksiyonunun sonucuna göre kabul veya reddedilebilmektedir.

Uyarlanabilir kabul fonksiyonunun davranışı θ değişkeni ile yönetilmektedir. θ ise α_1 ve α_2 olmak üzere iki adet değişkenin çarpımından oluşmaktadır. Burada α_2 değişkeninin değerinin hesaplanmasında kullanılan i, yeniden başlatmadan bu yana geçen iterasyon sayısını ifade etmektedir.

Yeniden başlatma işlemi sayaç değişkeni ile yönetilmektedir. *sayaç*'ın önceden belirlenmiş olan *th* eşiğini geçmesi durumunda algoritmanın yeniden başlatılmasına karar verilmektedir.

6.3.1 Başlangıç çözümü oluşturma aşaması

Başlangıç çözümü oluşturulurken literatürde yaygın olarak kullanılan Clarke and Wright (1964) tarafından geliştirilen paralel kazanç (parallel savings) algoritması kullanılmıştır.

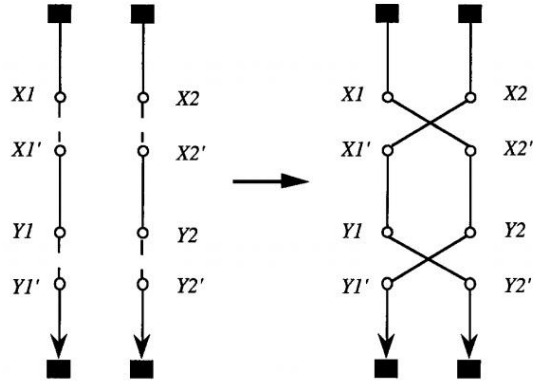
c_{ij} değeri i ve j indeksleri arasındaki maliyeti, 0 numaralı indeks depoyu ve n de müşteri sayısını göstermek üzere, paralel kazanç algoritmasının genel işleyişi şu şekilde özetlenebilir:

- $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ olmak üzere $i, j = 1, \dots, n \; ve \; i \neq j$ için tüm s_{ij} *Kazanç* değerleri hesaplanır.
- *Kazanç* değerleri büyükten küçüğe doğru sıralanır.
- $i = 1, \dots, n$ olmak üzere n adet araç rotası $(0, i, 0)$ şeklinde oluşturulur.
- Sıralanmış *Kazanç* listesinin başından başlanarak şu işlemler gerçekleştirilir: Sıradaki s_{ij} için *(0, j)* ile başlayan ve *(i, 0)* ile biten herhangi iki rotanın bulunup bulunmadığına bakılır. Eğer kapasite sınırları da aşılmıyorsa bu iki rota *(0, j)* ve *(i, 0)* kenarlarının silinip *(i, j)* kenarının oluşturulmasıyla birleştirilir.

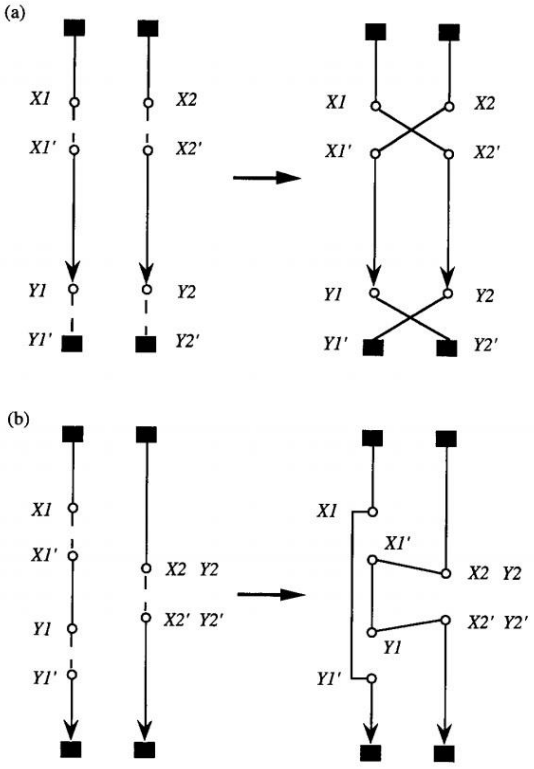
6.3.2 Sarsım aşaması

Önerilen yöntemde, ILS algoritmasının sarsım aşamasında literatürde Çapraz Değişim (CROSS Exchange, CE) (Taillard et al., 1997) adıyla geçen komşuluk

yapısı kullanılmıştır. CE, bir kenar değişim sezgiseli olup, seçilen iki adet rotadan ikişer kenar silinerek farklı şekilde tekrar bağlanmaktadır. Böylece, rotalar arasında farklı şekillerde müşteri değişimleri gerçekleştirilebilmektedir.



Şekil 6.2. CE örneği (Taillard et al.'dan 1997).



Şekil 6.3. CE özel durumları a) 2-opt*, b) Or-opt (Taillard et al.'dan 1997).

Şekil 6.2'de örnek bir CE hareketi yer almaktadır. İçi dolu dikdörtgenler depo noktasını temsil ederken, içi boş yuvarlaklar müşterileri göstermektedir. İlk olarak, $X1, X2, Y1, Y2$ olmak üzere 1. ve 2. rotalardan ikişer adet köşe noktası seçilmektedir. Ardından, seçilen bu 4 nokta ile ortak kenara sahip komşu

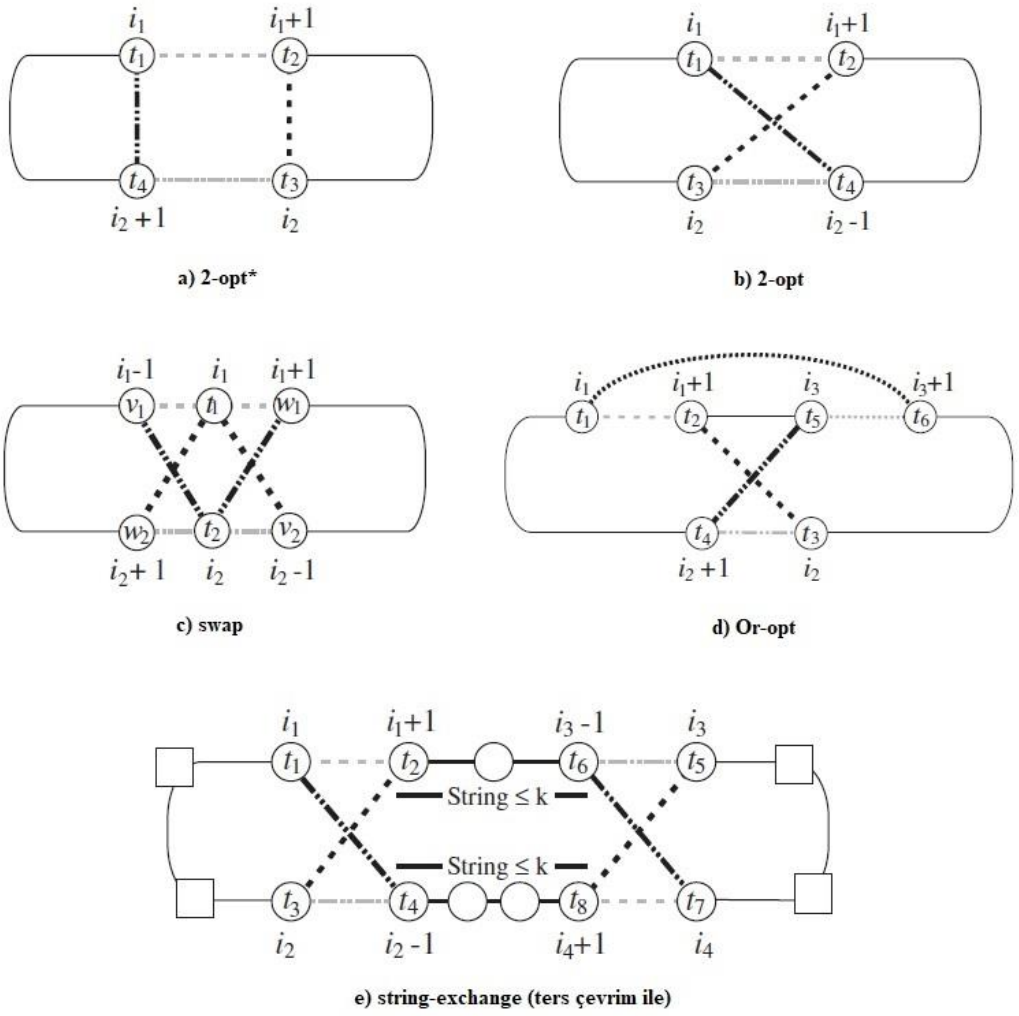
$X1', X2', Y1', Y2'$ noktaları belirlenmektedir. Tüm bu seçimlerden sonra, 1. rotadan $(X1, X1')$ ve $(Y1, Y1')$ kenarları, 2. rotadan ise $(X2, X2')$ ve $(Y2, Y2')$ kenarları kaldırılmakta ve onların yerine $(X1, X2')$, $(Y2, Y1')$, $(X2, X1')$ ve $(Y1, Y2')$ kenarları eklenmektedir. Böylece, birinci rotadan $(X1', Y1)$ parçası ile $(X2', Y2)$ parçası karşılıklı yer değiştirmiş olmaktadır.

Şekil 6.3'de ise CE sırasında karşılaşılabilecek özel durumlar yer almaktadır. Şekil 6.3 (a) 2-opt* (Potvin and Rousseau, 1995) adı verilen özel bir işleme karşılık gelirken Şekil 6.3 (b) literatürde Or-opt (Or, 1976) adıyla bilinen bir hamledir.2-opt*'da değişim yapılacak kenarlardan ikisi doğrudan depo noktasına bağlıdır. Or-opt'ta ise rotalardan birisi sıfır uzunluklu parçaya sahiptir. Bu nedenle karşılıklı değişim yerine, bir rotadan diğerine parça transferi söz konusudur.

CE ile yeni bir komşu üretilirken, seçilebilecek parçanın uzunluğu en fazla L olacak şekilde kısıtlanırsa, bir rotadan O(nL) adet kenar çifti seçilebilmektedir. Diğer rotadan da yine aynı şekilde O(nL) adet seçim yapılabileceği için toplam karmaşıklık $O(n^2L^2)$ düzeyindedir. CE yöntemi, tez kapsamında ILS algoritmasının sarsım aşamasında kullanılırken, sarsım büyüklüğü olan p parametresi L parametresi ile bire bir eşleştirilmekte olup, komşuluğun büyüklüğünü belirlemektedir.

6.3.3 Yerel arama aşaması

Önerilen yöntemde, ILS algoritmasının yerel arama aşaması için VND metasezgiseli (Bkz. Bölüm 2.1.3) kullanılmıştır. VND algoritmasının kullanacağı komşuluk yapıları ise 2-opt, 2-opt*, Or-opt, swap ve string-exchange olarak seçilmiş ve Irnich et al. (2006) tarafından önerilen ardışık arama (sequential search) tekniği kullanılarak gerçekleştirimi yapılmıştır.



Şekil 6.4. VND içerisinde kullanılan komşuluk yapıları için kenar değişim örnekleri a) 2-opt*, b) 2-opt, c) swap, d) Or-opt, e) string-exchange (ters çevrim ile) (Irnich et al.'dan, 2006).

Şekil 6.4'te, kullanılan komşuluk yapılarına ait örnek hareketler yer almaktadır. Bu noktada, çözümlerin temsilinin Bölüm 6.2'de açıklandığı şekliyle dairesel bir yapıda olduğu hatırlanmalıdır. Şekil 6.4 (a) incelendiğinde bir *2-opt** değişiminin nasıl yapıldığı anlaşılabilmektedir. Görüldüğü üzere 2-opt*, (t_1, t_2) ve (t_3, t_4) kenarlarını kaldırarak (t_1, t_4) ve (t_2, t_3) kenarlarını eklemekte ve çözümü iki parçaya ayırmaktadır. Şekilde yer alan i_1 ve i_2 , sırasıyla t_1 ve t_3 'ün konumlarıdır. Böylece, sırasıyla $i_1 + 1$ ve $i_2 + 1$ konumlarında yer alan t_2 ve t_4'ün, çözüm temsilinde t_1 ve t_3'ten birer sonra yer aldıkları anlaşılabilmektedir. Şekil 6.4 (b)'de *2-opt* değişimi yer almaktadır. 2-opt değişikliği için (t_1, t_2) ve (t_3, t_4) kenarları silinmekte ve bağlama işlemi (t_1, t_4) ve (t_2, t_3) kenarları eklenerek çapraz bir şekilde gerçekleştirilmektedir. Bunun sonucunda elde edilen yeni çözümde eski çözümün bir parçası ters yöne çevrilmiş olmaktadır. Şekil 6.4 (c) *swap* adlı komşuluk yapısını ifade etmektedir. Burada, belirlenen i_1 ve i_2

konumlarında yer alan iki nokta karşılıklı yer değişmektedir. Şekil 6.4 (d)'de yer alan *Or-opt* ise çözüm içerisinden seçilen bir parçanın başka bir konuma transfer edilmesini sağlamaktadır. Son olarak, Şekil 6.4 (e)'de gösterilen *string-exchange* değişimi en çok *k* uzunluklu iki parçanın değimini sağlamaktadır. Şekilde bu işlem parçaların yönünün ters çevrilmesi ile yapılmıştır. Tez kapsamında *string-exchange* hareketinin parçaların yönünü koruyan türevi de eklenmiştir.

Dikkat edilirse, yukarıda bahsedilen komşuluk yapılarından Or-opt ve string-exchange, çözüm içerisinde belirli uzunluktaki parçaları değiştirebilmektedirler. Bu da komşuluğun boyutunu belirleyen k parametresini ortaya çıkarmaktadır. k parametresi, Şekil 6.4'ten de anlaşılacağı üzere seçilebilecek en yüksek parça uzunluğunu belirlemektedir.

Tez kapsamında VND temelli yerel arama algoritması oluşturulurken komşuluk yapılarının sırasının her çağrımda rastgele olarak ve tekrar etmeyecek şekilde belirlenmesi sağlanmıştır. Çünkü buradaki komşuluk yapılarının hiyerarşik bir düzeni yoktur ve tek bir sıra belirlenmesi algoritmada yanlılığa yol açabilecektir. Tüm bunlar ışığında, önerilen yöntemde kullanılmak üzere Rastgele VND (RVND) adı verilen algoritma üretilmiştir (Algoritma 6.2). Komşuluklarda yerel en iyi aranırken ilk iyileştiren çözümler kullanılmıştır. Ayrıca, string-exchange komşuluk yapısının parçaları ters çeviren ve çevirmeyen iki sürümü de kullanılmıştır.

Algoritma 6.2 RVND (k, bi)

```
S ← başlangıç çözümünü oluştur
rastgeleSıra[] ← 1'den 6'ya kadar numaraları rastgele yerleştir
komşuluklar[] = {"2-opt*", "2-opt", "Or-opt", "swap", "string-exchange",
"string-exchange(ters çevrim)"}
n ← komşuluklar[].Uzunluk
i ← 1
while i <= n do
  N_i ← komşuluklar[rastgeleSıra[i]]
  S' ← S.İyileştirenKomşuSeç(N_i, k, bi)
  if f(S') < f(S)
      S ← S'
      i ← 1
  else
    i ← i + 1
  end if
end while
return S
```

6.3.4 Uyarlanabilir kabul fonksiyonu

Önerilen yöntemde, ILS algoritmasının kabul fonksiyonu aşaması için Bölüm 3.2.2'de yer alan SALS algoritmasından esinlenilmiştir. SALS algoritmasında iterasyon sayısını, çözüm iyileştirme sayısını ve en iyi çözümün uygunluk değeri değişkenleri kullanılarak bir katsayı (θ) elde edilmektedir. θ değeri, algoritmanın çalışması boyunca uyarlanabilir olarak değişmekte ve kabul karşılaştırması sırasında eski çözümün uygunluk değeri ile çarpılarak yeni çözümün kabul şansını artırmaktadır.

Bu tez çalışmasında, SALS algoritmasındaki yaklaşımın bazı bölümleri değiştirilerek kullanılmıştır. İlk değişiklik, θ hesabı sırasında yapılmıştır. Burada, $\alpha_2 = C(L^{(i)})/i$ hesabının değiştirilerek $\alpha_2 = 1/i$ şekline kullanılmasının başarıyı artırdığı gözlemlenmiştir. Ayrıca, önerilen MA_ILSxVND melez algoritma ve SALS algoritmasının çalışma şeklindeki farklılıktan dolayı komşuluk sayısının aşılması durumunda ekleme yapan $\theta \leftarrow \theta + \alpha_1\alpha_2$ adımına ihtiyaç duyulmamaktadır. Geliştirilen yöntemdeki bir diğer farklılık da θ değerinin hesaplanmasında şimdiki çözüm yerine (S) o anki en iyi çözümün (S^*) kullanılmasıdır. Sonuç olarak, algoritmadaki uyarlanabilir kabul fonksiyonu, S çözümünü $\theta xf(S^*)$ sonucu ile kıyaslamakta ve uygunluk değerinin bu çarpımdan daha küçük olması durumunda onu kabul etmektedir.

6.3.5 Çok başlangıçlı strateji

Geliştirilen yöntemde kullanılan bir diğer teknik de yeniden başlatma stratejisidir. ILS-VND algoritması bazı durumlarda yere en iyiye takılarak durağanlığa girmektedir. Bu durum, kullanılan yerel arama algoritmasının kuvvetli çalışmasından kaynaklanmaktadır. Bu nedenle önerilen algoritmaya bazı durumlarda yeniden başlamasını gerektiren kural eklenmiştir. Yeniden başlatma işlemini yönetmek için *sayaç* değişkeni ve *th* değeri (Bkz. Algoritma 6.1) tanımlanmış olup şu şekilde kullanılmaktadırlar:

- Eğer yeni üretilen çözüm şimdiki en iyi çözümü (S^*) iyileştirememiş ise *sayaç* değişkeni 1 artırılır.
- Eğer herhangi bir anda şimdiki en iyi çözüm iyileştirilmişse *sayaç* değişkeni sıfırlanır.
- Eğer *sayaç* değişkeninin değeri *th* değerine ulaşmışsa ILS-VND algoritması yeniden başlatılır ve *sayaç* sıfırlanır.

7. DENEYSEL ÇALIŞMALAR

7.1 HAMS_ABCxDExPSO İçin Deneysel Çalışma

7.1.1 Kullanılan veri seti

Bu bölümde gerçekleştirilen deneysel çalışmalarda sürekli eniyileme alanındaki algoritmaların performansını ölçme ve karşılaştırmada yaygın olarak kullanılan CEC'17 veri seti (Awad et al., 2016) kullanılmıştır. Bu veri seti içerisinde 4 farklı grupta 30 tane fonksiyon bulunmaktadır (Bkz. Çizelge 7.1 ve Çizelge 7.2).

Tek doruklu fonksiyonlar tek bir yerel en iyi noktaya sahipken çok doruklu fonksiyonlarda çok sayıda yerel en iyi bulunabilmektedir. Melez fonksiyonlar ise N adet basit fonksiyonun birer alt bileşen şeklinde bir araya getirilmesiyle oluşturulmaktadır. Bileşik fonksiyonlar ise N adet fonksiyonun parametrik bir şekilde birleştirilmesiyle üretilmektedir.

Tüm fonksiyonlar ve boyutlar için arama sınırı $[-100,100]^D$ olarak belirlenmiştir. Her bir fonksiyon için elde edilebilecek olan en iyi değerler son kolonda yer almaktadır ve problem boyutundan (D) bağımsız olarak aynı büyüklüktedir. En iyi değerlerin her fonksiyon için farklı olmasına dikkat edilmiştir. Ayrıca tüm test fonksiyonları için kaydırma ve döndürme işlemleri uygulanarak, geliştirilecek olan algoritmaların yanlılığını engelleme yoluna gidilmiştir.

Çizelge 7.1. CEC'17 veri seti içerisinde yer alan fonksiyonlar ve özellikleri (1. Bölüm).

Grup	No	Fonksiyon	En iyi
Tek Doruklu Fonksiyonlar	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Sum of Different Power Function	200
	3	Shifted and Rotated Zakharov Function	300
Basit Çok Doruklu Fonksiyonlar	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Rastrigin's Function	500
	6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	7	Shifted and Rotated Lunacek Bi_Rastrigin Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel's Function	1000

Çizelge 7.2. CEC'17 veri seti içerisinde yer alan fonksiyonlar ve özellikleri (2. Bölüm).

Grup	No	Fonksiyon	En iyi
Melez Fonksiyonlar	11	Hybrid Function 1 (*N*=3)	1100
	12	Hybrid Function 2 (*N*=3)	1200
	13	Hybrid Function 3 (*N*=3)	1300
	14	Hybrid Function 4 (*N*=4)	1400
	15	Hybrid Function 5 (*N*=4)	1500
	16	Hybrid Function 6 (*N*=4)	1600
	17	Hybrid Function 6 (*N*=5)	1700
	18	Hybrid Function 6 (*N*=5)	1800
	19	Hybrid Function 6 (*N*=5)	1900
	20	Hybrid Function 6 (*N*=6)	2000
Bileşik Fonksiyonlar	21	Composition Function 1 (*N*=3)	2100
	22	Composition Function 2 (*N*=3)	2200
	23	Composition Function 3 (*N*=4)	2300
	24	Composition Function 4 (*N*=4)	2400
	25	Composition Function 5 (*N*=5)	2500
	26	Composition Function 6 (*N*=5)	2600
	27	Composition Function 7 (*N*=6)	2700
	28	Composition Function 8 (*N*=6)	2800
	29	Composition Function 9 (*N*=3)	2900
	30	Composition Function 10 (*N*=3)	3000

7.1.2 Deneysel kurulum

Deneysel kurulum CEC'17 veri seti dökümanında belirtildiği şekliyle yapılmış olup aşağıdaki gibi listelenmektedir:

- Her fonksiyon 51 kez farklı rastgele çekirdek ile çalıştırılmaktadır. Elde edilen 51 farklı sonuca ait ortalama, ortanca, standart sapma, en iyi ve en kötü değerleri hesaplanmaktadır.
- Arama sınırları her bir boyut için aynı olacak şekilde $[-100,100]^D$ olarak belirlenmektedir.
- İlkleme aşamasında, belirtilen arama sınırları içerisinde tekdüze dağılıma göre rastgele çözümler oluşturulmaktadır.
- Maksimum fonksiyon hesaplama sayısı Dx10000 olarak belirlenmektedir.

- En iyi noktadan uzaklık olarak tanımlanan hata değeri, 10^{-8} eşik değerine eşit veya ondan daha az ise 0 olarak kabul edilmektedir.
- Kullanılacak olan parametre seti tüm fonksiyon ve problem boyutları için aynı olmalıdır.

7.1.2.1 Parametrelerin belirlenmesi

HAMS_ABCxDExPSO algoritmasının parametrelerinin belirlenmesi iRace (Lopez-Ibanez et al., 2016) yazılım paketi kullanılarak gerçekleştirilmiştir. iRace konfigürasyonu sırasında maksimum bütçe 5000 iterasyon olarak belirlenmiş olup CEC'17 veri setinde yer alan 1'den 30'a kadar fonksiyonların tek numaralı olanları eğitim, çift numaralı olanları ise test için ayrılmıştır. Ayrıca, eğitim ve test için ayrılmış her bir fonksiyonun 10, 30 ve 50 boyutlu halleri birlikte kullanılmıştır. Böylece eğitim ve test için ayrı ayrı 15x3=45 adet örnek kullanılmıştır. Bunun dışındaki konfigürasyon varsayılan değerleri ile bırakılmıştır.

Çizelge 7.3. HAMS_ABCxDExPSO algoritmasının iRace ile ayarlanmış parametre değerleri.

Parametre adı	Parametre türü	Parametre ayarlama değer aralığı	Ayarlanmış değer
L	Tamsayı	[250, 1500]	608
W	Tamsayı	[5, 40]	31
D	Reel	[0.1, 1.0]	0.32
C	Reel	[0.01, 5.0]	2.87

Belirtilen konfigürasyon ile ayarlanmış parametre değerleri Çizelge 7.3'te sunulmaktadır. Bu parametreler hakkında ayrıntılı açıklamaya Bölüm 5.2'den ulaşılabilmektedir.

Alt seviye parametrelerin belirlenmesi

Tüm algoritmalar için eniyilenecek topluluk büyüklüğü (NP) 50 olarak sabitlenmiştir. ABC, DE metasezgisellerinin iç parametreleri Karaboga and Akay'da (2009) olduğu gibi belirlenmiştir. Buna göre ABC'deki limit değeri DxNP olarak alınırken DE'deki CR parametresi 0.9 ve F parametresi 0.5 olarak alınmıştır. PSO (SPSO-2011 sürümü) için ise Zambrano-Bigiarini (2013) çalışmasında yer alan $w = 1/(2 \times \ln(2))$, $c1 = 0.5 + \ln(2)$, $c2 = 0.5 + \ln(2)$ ve $K = 3$ parametre değerleri kullanılmıştır.

7.1.3 Deneysel sonuçlar

Bu bölümde, geliştirilen HAMS_ABCxDExPSO algoritmasının ayrıntılı deneysel sonuçları yer almaktadır. Ayrıca, geliştirilen algoritmanın kendisini oluşturan ABC, DE ve PSO algoritmaları ile karşılaştırılması yapılmıştır. Buna ek olarak, metasezgisel seçimi için kullanılan UCB tabanlı parametre kontrol yaklaşımının performansı, rastgele seçime kıyasla değerlendirilmiştir.

Çizelge 7.4. HAMS_ABCxDExPSO algoritmasının D=10 için CEC'17 veri seti üzerinde çalıştırılması ile elde edilen hata değerleri.

Fonk.	En iyi	En kötü	Ortanca	Ortalama	Std. Sapma
1	**0.00E+00**	2.24E+02	**0.00E+00**	6.67E+00	3.29E+01
2	**0.00E+00**	1.35E-06	**0.00E+00**	4.90E-08	2.07E-07
3	**0.00E+00**	1.45E-03	**0.00E+00**	2.85E-05	2.04E-04
4	4.67E-02	4.46E+00	7.50E-01	1.01E+00	8.32E-01
5	7.59E-03	3.98E+00	1.32E+00	1.62E+00	9.16E-01
6	**0.00E+00**	8.01E-05	5.72E-06	1.60E-05	2.04E-05
7	1.06E+01	1.45E+01	1.27E+01	1.26E+01	7.86E-01
8	1.31E-02	4.13E+00	2.00E+00	1.77E+00	9.31E-01
9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
10	3.62E-01	8.45E+01	4.15E+00	7.99E+00	1.39E+01
11	**0.00E+00**	3.70E+00	1.49E+00	1.36E+00	1.10E+00
12	**0.00E+00**	1.95E+03	1.23E+01	1.20E+02	2.82E+02
13	**0.00E+00**	1.17E+01	5.74E+00	5.39E+00	2.24E+00
14	**0.00E+00**	6.22E+00	5.36E-01	8.26E-01	1.18E+00
15	2.25E-04	1.53E+00	1.35E-01	3.25E-01	4.03E-01
16	1.95E-02	1.17E+01	6.87E-01	9.43E-01	1.66E+00
17	3.12E-01	8.14E+00	1.26E+00	1.51E+00	1.37E+00
18	1.40E-06	1.30E+03	2.45E-01	2.66E+01	1.81E+02
19	**0.00E+00**	1.52E+00	1.94E-02	4.43E-02	2.11E-01
20	**0.00E+00**	2.03E+01	3.12E-01	6.67E-01	2.86E+00
21	1.29E+02	2.06E+02	2.00E+02	1.89E+02	2.00E+01
22	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.57E-01
23	3.00E+02	3.06E+02	3.01E+02	3.02E+02	1.70E+00
24	2.93E+02	3.32E+02	3.28E+02	3.26E+02	8.71E+00
25	3.98E+02	4.46E+02	3.98E+02	4.15E+02	1.96E+01
26	2.00E+02	3.00E+02	3.00E+02	2.98E+02	1.40E+01
27	3.87E+02	3.91E+02	3.90E+02	3.89E+02	5.15E-01
28	3.00E+02	6.12E+02	5.84E+02	4.63E+02	1.47E+02
29	2.28E+02	2.57E+02	2.42E+02	2.42E+02	6.10E+00
30	3.95E+02	8.18E+05	4.61E+02	4.64E+04	1.76E+05

10 boyut için elde edilen sonuçlar (Çizelge 7.4) incelendiğinde, tek doruklu fonksiyonları oluşturan ilk 3 fonksiyonun hepsinde ve basit çok doruklu ve melez fonksiyonları kapsayan 4-20 numara aralığındaki fonksiyonların yaklaşık yarısında en iyi sonuca ulaşıldığı görülmektedir. 21-30 numara aralığını kapsayan bileşik

fonksiyonlar ise en zorlayıcı grup olup, bu fonksiyonlar için en iyi sonuca ulaşılamadığı görülmektedir. 51 kez çalıştırmanın hepsinde en iyi sonuca sadece 9. fonksiyonda ulaşılmıştır. Ortanca değerler dikkate alındığında ise tek doruklu 3 adet fonksiyonda en iyi değerlere ulaşıldığı görülmektedir.

Çizelge 7.5. HAMS_ABCxDExPSO algoritmasının D=30 için CEC'17 veri seti üzerinde çalıştırılması ile elde edilen hata değerleri.

Fonk.	En iyi	En kötü	Ortanca	Ortalama	Std. Sapma
1	6.61E-01	1.09E+04	1.05E+03	2.81E+03	3.54E+03
2	**0.00E+00**	5.84E+09	2.20E+01	1.17E+08	8.18E+08
3	**0.00E+00**	2.01E+03	1.09E-03	1.24E+02	3.72E+02
4	4.08E+00	8.74E+01	8.22E+01	7.26E+01	1.48E+01
5	5.07E+00	1.84E+01	1.22E+01	1.22E+01	3.02E+00
6	**0.00E+00**	6.68E-02	5.47E-07	1.32E-03	9.35E-03
7	3.73E+01	5.30E+01	4.34E+01	4.36E+01	3.31E+00
8	5.06E+00	2.19E+01	1.23E+01	1.27E+01	3.33E+00
9	**0.00E+00**	1.75E+01	4.54E-01	2.27E+00	3.86E+00
10	4.76E+02	2.17E+03	1.27E+03	1.29E+03	3.58E+02
11	2.37E+00	7.92E+01	8.98E+00	1.48E+01	1.77E+01
12	1.75E+03	4.48E+04	1.42E+04	1.67E+04	1.13E+04
13	1.70E+01	5.42E+04	8.00E+02	6.17E+03	1.14E+04
14	5.32E+00	4.31E+01	2.63E+01	2.51E+01	6.78E+00
15	3.79E+00	8.29E+01	1.19E+01	1.74E+01	1.54E+01
16	3.35E+00	4.52E+02	2.29E+01	7.93E+01	1.05E+02
17	1.55E+01	1.53E+02	3.35E+01	4.25E+01	3.00E+01
18	4.83E+01	6.27E+04	1.11E+03	3.45E+03	9.14E+03
19	3.94E+00	6.39E+02	8.62E+00	2.21E+01	8.82E+01
20	3.19E+00	3.48E+02	3.51E+01	9.34E+01	7.89E+01
21	2.07E+02	2.24E+02	2.13E+02	2.14E+02	3.96E+00
22	1.00E+02	1.02E+02	1.00E+02	1.00E+02	4.86E-01
23	3.46E+02	3.77E+02	3.57E+02	3.57E+02	6.69E+00
24	4.26E+02	4.46E+02	4.31E+02	4.32E+02	4.37E+00
25	3.83E+02	3.88E+02	3.87E+02	3.87E+02	5.49E-01
26	8.84E+02	1.25E+03	1.03E+03	1.03E+03	8.65E+01
27	4.85E+02	5.19E+02	5.08E+02	5.05E+02	7.35E+00
28	3.00E+02	4.83E+02	3.00E+02	3.38E+02	5.58E+01
29	3.58E+02	5.47E+02	4.32E+02	4.35E+02	2.72E+01
30	2.01E+03	8.00E+03	2.59E+03	3.22E+03	1.53E+03

30 boyutlu fonksiyonlar için 2., 3., 6. ve 9. olmak üzere 4 fonksiyonda en iyi değere ulaşılabilmiştir (Çizelge 7.5). İlk 3 grupta yer alan 1-20 numara aralığındaki diğer fonksiyonlar için ise genellikle en iyiye yaklaşan sonuçlar elde edilebilmiştir. 10 boyut için alınan sonuçlarda olduğu gibi, 21-30 numara aralığındaki bileşik fonksyionlar yine en zorlayıcı grup olmuştur. Bu fonksiyonlar için ortalamada 10^2 seviyelerinde hata değerlerine rastlanmaktadır.

Geliştirilen algoritmanın 50 ve 100 boyutlu problemlerdeki davranışı hakkında fikir verebilmesi için sadece ortalama ve standart sapma hata değerlerini içeren Çizelge 7.6 oluşturulmuştur. Beklenildiği üzere, artan boyut sayısı ile paralel olarak ortalama hata değerlerinde artış gözlemlenmektedir.

Çizelge 7.6. HAMS_ABCxDExPSO algoritmasının D=50 ve D=100 için CEC'17 veri seti üzerinde çalıştırılması ile elde edilen hata değerleri.

	D=50		**D=100**	
Fonk.	**Ortalama**	**Std. Sapma**	**Ortalama**	**Std. Sapma**
1	4.18E+03	4.51E+03	6.74E+03	7.42E+03
2	1.81E+08	1.07E+09	2.54E+44	1.80E+45
3	9.44E+03	8.54E+03	1.39E+05	4.17E+04
4	6.22E+01	4.32E+01	2.10E+02	3.43E+01
5	3.34E+01	6.22E+00	1.26E+02	2.15E+01
6	4.26E-03	1.33E-02	1.75E+00	1.93E+00
7	8.48E+01	8.09E+00	2.66E+02	3.58E+01
8	3.15E+01	6.96E+00	1.25E+02	2.12E+01
9	2.39E+01	3.14E+01	4.35E+02	3.56E+02
10	2.61E+03	6.41E+02	8.14E+03	8.94E+02
11	6.08E+01	3.98E+01	4.33E+02	3.43E+02
12	1.38E+05	7.30E+04	6.07E+05	2.74E+05
13	3.92E+03	5.29E+03	3.71E+03	3.39E+03
14	5.96E+02	9.88E+02	3.18E+04	1.81E+04
15	4.06E+03	4.40E+03	2.06E+03	2.15E+03
16	3.91E+02	2.56E+02	1.65E+03	4.70E+02
17	3.34E+02	1.75E+02	1.39E+03	5.08E+02
18	1.06E+04	8.75E+03	1.49E+05	6.32E+04
19	6.63E+03	7.18E+03	1.95E+03	2.59E+03
20	1.67E+02	1.51E+02	9.38E+02	4.03E+02
21	2.34E+02	6.26E+00	3.56E+02	2.11E+01
22	4.49E+02	9.76E+02	6.12E+03	4.20E+03
23	4.50E+02	1.07E+01	6.39E+02	3.23E+01
24	5.32E+02	9.41E+00	9.95E+02	3.47E+01
25	5.15E+02	3.58E+01	7.67E+02	6.28E+01
26	1.49E+03	1.36E+02	4.39E+03	4.33E+02
27	5.42E+02	2.17E+01	6.74E+02	2.71E+01
28	4.80E+02	2.28E+01	5.71E+02	3.35E+01
29	4.77E+02	1.58E+02	2.11E+03	4.96E+02
30	6.38E+05	5.21E+04	6.61E+03	3.41E+03

Geliştirilen melez algoritmanın, kendisini oluşturan metasezgiseller ile kıyaslanması için fonksiyon bazında gerçekleştirilen 51 adet çalıştırma sonuçları üzerinde Mann Whitney U testi gerçekleştirilmiştir. p=0.05 anlamlılık düzeyi dikkate alınmış olup, geliştirilen algoritmanın karşılaştırma yapılan algoritmaya üstünlük kurduğu durumlarda "+" sembolü, ondan geride kaldığı durumlarda "-"

sembolü ve onunla eşit performans gösterdiği beraberlik durumlarında ise “=” sembolü kullanılmıştır.

Çizelge 7.7’de HAMS_ABCxDExPSO algoritmasının 10. ve 30. boyutlar için kendisini oluşturan üç metasezgisel ile karşılaştırması yer almaktadır. Tüm algoritmalar HAMS_ABCxDExPSO’da olduğu gibi Dx10000 fonksiyon hesaplama sayısı ile çalıştırılmıştır. Çizelgede yer alan sayısal değerler ilgili metasezgiselin 51 adet çalıştırılması sonucu elde edilen ortalama hata değerleridir. “T” kolonu Mann Whitney U testi sonucu elde edilen karşılaştırma sonucunu vermektedir. En alt satırda yer alan “+/=/-“ ise sırasıyla toplam kazanma, beraberlik ve kaybetme sayılarını vermektedir.

Çizelge 7.7. D=10 ve D=30 için ABC, DE ve PSO algoritmalarının ortalama hata değerleri ve fonksiyon bazında HAMS_ABCxDExPSO algoritması ile kıyaslanması.

	D=10						**D=30**					
Fonk.	**ABC**	**T**	**DE**	**T**	**PSO**	**T**	**ABC**	**T**	**DE**	**T**	**PSO**	**T**
1	2.27E+02	+	0.00E+00	-	8.85E+02	+	1.88E+02	-	3.13E+03	=	2.69E+03	=
2	4.29E-04	+	0.00E+00	=	2.27E-04	+	6.74E+09	+	1.27E+17	+	1.43E+07	+
3	6.26E+03	+	0.00E+00	=	0.00E+00	=	1.24E+05	+	1.60E-05	-	1.06E+04	+
4	1.47E-01	-	4.44E-01	-	2.41E+00	+	2.79E+01	-	5.37E+01	-	6.67E+01	=
5	7.71E+00	+	7.07E+00	+	7.59E+00	+	8.28E+01	+	3.75E+01	+	7.96E+01	+
6	0.00E+00	-	2.79E-08	-	9.21E-01	+	0.00E+00	-	5.79E-04	+	2.38E+01	+
7	1.77E+01	+	1.98E+01	+	1.50E+01	+	9.96E+01	+	1.26E+02	+	1.47E+02	+
8	9.21E+00	+	5.95E+00	+	5.97E+00	+	9.48E+01	+	2.82E+01	+	6.39E+01	+
9	4.30E-02	+	0.00E+00	=	2.71E+00	+	8.30E+02	+	2.95E-01	-	1.03E+03	+
10	2.58E+02	+	3.29E+02	+	6.19E+02	+	2.31E+03	+	4.21E+03	+	3.12E+03	+
11	4.46E+00	+	7.96E-01	-	2.52E+01	+	5.50E+02	+	1.81E+01	+	1.09E+02	+
12	2.70E+04	+	6.75E+01	=	9.27E+03	+	4.12E+05	+	1.59E+04	=	5.08E+04	+
13	3.11E+02	+	4.41E+00	-	6.46E+03	+	6.59E+03	+	1.23E+03	-	9.93E+03	+
14	2.72E+02	+	5.46E-01	-	8.02E+02	+	6.81E+04	+	2.72E+01	+	7.19E+03	+
15	1.49E+02	+	2.87E-01	=	2.30E+03	+	1.56E+03	+	1.08E+01	-	1.90E+03	+
16	5.05E+00	+	5.87E+00	=	1.69E+02	+	6.22E+02	+	4.79E+02	+	9.07E+02	+
17	2.32E+00	+	5.37E+00	=	4.80E+01	+	2.03E+02	+	7.61E+01	+	3.74E+02	+
18	1.15E+03	+	2.94E+00	=	5.78E+03	+	1.68E+05	+	1.52E+03	=	7.91E+04	+
19	8.01E+01	+	1.25E-01	=	4.22E+03	+	1.38E+03	+	8.97E+00	-	3.53E+03	+
20	5.57E-01	+	9.35E-01	=	5.18E+01	+	2.43E+02	+	8.69E+01	=	3.17E+02	+
21	1.13E+02	-	2.03E+02	+	2.08E+02	+	2.63E+02	+	2.34E+02	+	2.58E+02	+
22	6.19E+01	-	9.84E+01	=	1.00E+02	=	6.30E+02	+	2.39E+03	+	1.05E+02	+
23	3.08E+02	+	3.07E+02	+	3.11E+02	+	4.15E+02	+	3.79E+02	+	4.66E+02	+
24	1.01E+02	-	3.32E+02	+	3.31E+02	+	4.29E+02	=	4.56E+02	+	5.36E+02	+
25	1.84E+02	-	4.11E+02	=	4.31E+02	+	3.82E+02	-	3.87E+02	+	4.12E+02	+
26	9.55E+01	-	3.29E+02	-	3.31E+02	=	2.96E+02	-	1.24E+03	+	2.31E+03	+
27	3.95E+02	+	3.90E+02	+	3.99E+02	+	5.14E+02	+	5.07E+02	=	5.72E+02	+
28	2.60E+02	-	4.02E+02	-	5.52E+02	+	4.01E+02	+	3.71E+02	=	3.74E+02	+
29	2.53E+02	+	2.31E+02	-	2.93E+02	+	6.36E+02	+	4.60E+02	=	1.08E+03	+
30	1.96E+04	+	8.19E+04	=	1.83E+05	+	6.01E+03	+	2.27E+03	-	5.11E+03	+
+/=/-	22/0/8		8/13/9		27/3/0		24/1/5		16/7/7		28/2/0	

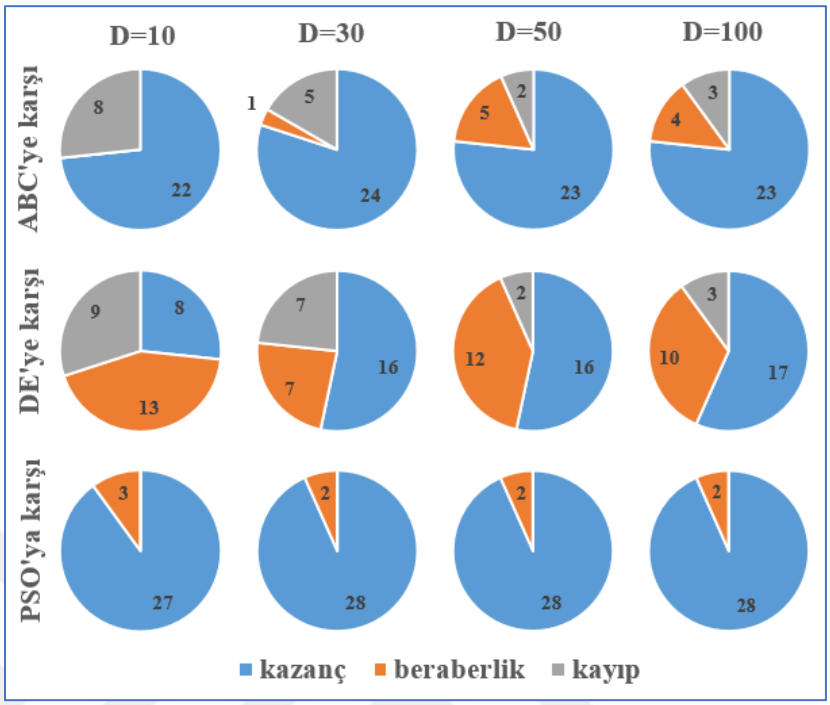
Çizelge 7.7'den görüldüğü üzere 10 boyut için ABC ve PSO metasezgisellerine karşı sırasıyla 22/0/8 ve 27/3/0'lik kazanç/beraberlik/kayıp üstünlüğü kurulmuştur. Ancak DE metasezgiseli için daha dengeli bir sonuç olan 8/13/9 değerleri elde edilmiştir. Boyut sayısı 30 olduğunda ise bu değerler melez yöntemin lehinde gelişerek kazanç yönünde artış göstermiştir. Öyle ki DE için 10 boyutlu durumda 8 olan kazanç sayısı ikiye katlanarak 16'ya yükselmiştir.

Çizelge 7.8. D=50 ve D=100 için ABC, DE ve PSO algoritmalarının ortalama hata değerleri ve fonksiyon bazında HAMS_ABCxDExPSO algoritması ile kıyaslanması.

	D=50						**D=100**					
Fonk.	**ABC**	**T**	**DE**	**T**	**PSO**	**T**	**ABC**	**T**	**DE**	**T**	**PSO**	**T**
1	1.69E+03	-	5.84E+03	=	2.62E+03	=	2.33E+03	-	8.20E+03	=	7.31E+03	=
2	4.11E+18	+	1.49E+38	+	2.44E+21	+	3.16E+62	+	4.88E+93	+	2.82E+79	+
3	2.68E+05	+	5.63E+03	=	5.30E+04	+	6.52E+05	+	2.11E+05	+	1.96E+05	+
4	4.17E+01	=	8.28E+01	=	1.11E+02	+	2.04E+02	=	2.02E+02	=	2.78E+02	+
5	2.00E+02	+	6.84E+01	+	1.71E+02	+	6.75E+02	+	1.64E+02	+	5.28E+02	+
6	0.00E+00	-	1.06E-02	+	3.84E+01	+	0.00E+00	-	9.09E-01	=	4.76E+01	+
7	2.16E+02	+	2.37E+02	+	4.06E+02	+	6.79E+02	+	4.53E+02	+	1.59E+03	+
8	2.01E+02	+	6.86E+01	+	1.78E+02	+	7.06E+02	+	1.68E+02	+	5.82E+02	+
9	4.82E+03	+	1.61E+01	=	5.49E+03	+	3.06E+04	+	6.51E+02	+	2.01E+04	+
10	4.21E+03	+	1.22E+04	+	5.94E+03	+	1.11E+04	+	2.99E+04	+	1.34E+04	+
11	1.65E+03	+	5.20E+01	=	1.76E+02	+	6.60E+04	+	1.75E+02	-	1.17E+03	+
12	2.72E+06	+	8.24E+04	-	7.78E+05	+	1.44E+07	+	5.77E+05	=	2.06E+06	+
13	3.55E+03	=	8.25E+03	+	6.17E+03	+	2.39E+03	=	6.40E+03	=	8.50E+03	+
14	8.63E+05	+	6.40E+02	=	2.94E+04	+	3.80E+06	+	2.08E+04	-	1.11E+05	+
15	8.72E+03	+	3.36E+03	=	4.81E+03	=	9.92E+02	=	4.13E+03	=	1.59E+03	=
16	1.28E+03	+	1.18E+03	+	1.50E+03	+	3.26E+03	+	3.34E+03	+	4.00E+03	+
17	9.62E+02	+	8.99E+02	+	1.45E+03	+	2.66E+03	+	2.70E+03	+	3.35E+03	+
18	8.20E+05	+	9.43E+03	=	9.86E+04	+	3.41E+06	+	1.49E+05	=	3.62E+05	+
19	7.27E+03	+	1.25E+03	-	1.33E+04	+	1.34E+03	=	6.52E+03	+	2.33E+03	+
20	7.63E+02	+	8.54E+02	+	7.75E+02	+	2.54E+03	+	3.15E+03	+	2.52E+03	+
21	4.07E+02	+	2.88E+02	+	3.56E+02	+	9.04E+02	+	3.92E+02	+	8.32E+02	+
22	4.94E+03	+	1.25E+04	+	6.49E+03	+	1.27E+04	+	3.03E+04	+	1.62E+04	+
23	6.48E+02	+	4.79E+02	+	7.62E+02	+	8.83E+02	+	7.21E+02	+	1.49E+03	+
24	1.03E+03	+	5.88E+02	+	8.16E+02	+	1.48E+03	+	1.08E+03	+	2.24E+03	+
25	5.10E+02	=	5.26E+02	=	5.59E+02	+	7.19E+02	-	7.74E+02	=	8.23E+02	+
26	1.63E+03	=	1.71E+03	+	5.57E+03	+	9.12E+03	+	5.24E+03	+	1.98E+04	+
27	6.47E+02	+	6.04E+02	+	1.01E+03	+	7.59E+02	+	7.06E+02	+	1.60E+03	+
28	4.83E+02	=	4.93E+02	=	5.01E+02	+	5.80E+02	+	5.62E+02	=	6.08E+02	+
29	1.11E+03	+	4.98E+02	=	2.01E+03	+	4.05E+03	+	1.69E+03	-	4.76E+03	+
30	7.02E+05	+	6.50E+05	=	8.96E+05	+	1.22E+04	+	6.54E+03	=	1.13E+04	+
+/=/-	23/5/2		16/12/2		28/2/0		23/4/3		17/10/3		28/2/0	

Çizelge 7.8'de ise HAMS_ABCxDExPSO algoritması bu sefer 50. ve 100. boyutlar için ABC, DE ve PSO metasezgiselleri ile kıyaslanmıştır. Problem boyutuna paralel olarak algoritmalar arasındaki performans kıyaslaması HAMS_ABCxDExPSO lehinde gelişmeye devam etmektedir. Öyle ki, geliştirilen yöntemin kayıp sayıları 100 boyuta doğru iyice azalarak 30 fonksiyonda en fazla 3 tane seviyesine gelmektedir. Artan problem boyutlarına göre önerilen algoritmanın

ABC, DE ve PSO karşısındaki kazanç, berabelik ve kayıp sayılarını gösteren bir grafik Şekil 7.1'de sunulmuştur.



Şekil 7.1. HAMS_ABCxDExPSO algoritmasının artan problem boyutuna göre ABC, DE ve PSO karşısındaki kazanç, beraberlik ve kayıp sayıları.

Çizelge 7.7 ve Çizelge 7.8'de fonksiyon bazında yapılmış olan karşılaştırmalara ek olarak 30 fonksiyonun tümünü hesaba katan ikinci bir karşılaştırma yapılmıştır. Burada amaç, bütün veri seti için geliştirilen yöntem ve kendisini oluşturan algoritmalar arasındaki farkı ortaya koymaktır. Bu kapsamda her bir boyut için önce 4 algoritma için Friedman skorları hesaplanmış, ardından Wilcoxon işaret sırası testi ile bu skorların istatistiksel anlamlılığı test edilmiştir.

Çizelge 7.9. Önerilen algoritmanın ABC, DE ve PSO algoritmaları ile veri setinin tümü için Friedman skorları ve Wilcoxon işaret sırası testi kullanılarak karşılaştırılması.

		HAMS_ABCxDExPSO	**ABC**	**DE**	**PSO**
D=10	**Friedman sıra değeri**	**1.88**	2.40	2.03	3.68
	p-değeri	-	0.11	0.55	**1.83E-6**
D=30	**Friedman sıra değeri**	**1.67**	2.87	2.10	3.37
	p-değeri	-	**2.32E-4**	0.15	**1.20E-4**
D=50	**Friedman sıra değeri**	**1.43**	2.77	2.37	3.43
	p-değeri	-	**7.84E-5**	**0.02**	**9.77E-6**
D=100	**Friedman sıra değeri**	**1.53**	2.73	2.40	3.33
	p-değeri	-	**6.39E-4**	**0.006**	**3.69E-6**

Çizelge 7.9'dan görüldüğü üzere tüm boyutlarda en iyi Friedman skorunu, geliştirilen yöntem elde etmiştir. Ancak, 30 adet fonksiyon için ortalama hata değerleri üzerinde gerçekleştirilen Wilcoxon işaret sırası testi, $p = 0.05$ olan istatistiksel anlamlılık seviyesini ABC için 30 ve üstü boyutlarda, DE için ise 50 ve üstü boyutlarda yakalayabilmiştir. PSO için ise her durumda eşik değerinin altında p değerleri elde edilmiştir. Genel değerlendirmeye göre, önerilen melez yapı özellikle artan problem boyutlarında işe yaramaktadır.

Çizelge 7.10. AMS_ ABCxDExPSO algoritmasının rastgele seçim ile çalıştırılması sonucu elde edilen ortalama hata değerleri ve UCB seçimli ilk durumu ile kıyaslanması.

	D=10		**D=30**		**D=50**		**D=100**	
Fonk.	**Rast.**	**T**	**Rast.**	**T**	**Rast.**	**T**	**Rast.**	**T**
1	2.38E+02	+	2.69E+03	=	4.78E+03	=	6.69E+03	=
2	5.94E-09	=	4.71E+05	=	4.18E+07	=	1.81E+40	=
3	1.27E-05	+	1.89E+02	+	1.16E+04	+	1.24E+05	=
4	1.43E+00	+	8.67E+01	+	1.05E+02	+	2.32E+02	+
5	1.38E+00	-	1.02E+01	-	2.85E+01	-	9.91E+01	-
6	1.69E-04	+	4.61E-05	=	7.52E-02	+	2.73E+00	+
7	1.17E+01	-	3.92E+01	-	7.84E+01	-	2.60E+02	=
8	1.12E+00	-	1.12E+01	-	2.60E+01	-	9.51E+01	-
9	4.12E-08	=	6.21E-02	-	7.41E+00	-	4.98E+02	=
10	2.55E+01	-	1.61E+03	+	3.62E+03	+	9.87E+03	+
11	8.82E-01	=	3.42E+01	+	1.05E+02	+	7.15E+02	+
12	1.41E+03	+	2.61E+04	+	3.50E+05	+	9.77E+05	+
13	6.58E+00	+	1.52E+04	+	3.75E+03	=	4.82E+03	=
14	2.87E+00	+	3.93E+01	+	2.95E+03	+	4.67E+04	+
15	6.93E-01	+	7.19E+02	+	6.34E+03	+	2.43E+03	=
16	1.26E+00	=	8.97E+01	=	4.57E+02	+	1.91E+03	+
17	3.22E+00	=	3.97E+01	=	3.91E+02	=	1.78E+03	+
18	4.04E+00	+	8.76E+03	+	3.37E+04	+	1.92E+05	+
19	2.74E-01	+	1.10E+03	+	1.24E+04	+	2.37E+03	=
20	1.61E+00	+	7.42E+01	=	1.27E+02	=	9.71E+02	=
21	1.79E+02	=	2.12E+02	=	2.29E+02	-	3.30E+02	-
22	1.00E+02	-	1.00E+02	+	4.50E+02	=	7.08E+03	+
23	3.02E+02	=	3.59E+02	=	4.52E+02	=	6.65E+02	+
24	3.27E+02	+	4.29E+02	-	5.24E+02	-	1.00E+03	=
25	4.17E+02	=	3.87E+02	=	5.20E+02	=	7.60E+02	=
26	2.98E+02	+	1.00E+03	=	1.38E+03	-	4.35E+03	=
27	3.90E+02	=	5.08E+02	=	5.72E+02	+	7.41E+02	+
28	4.52E+02	=	3.51E+02	+	4.85E+02	+	5.71E+02	=
29	2.36E+02	-	4.67E+02	+	5.68E+02	+	2.71E+03	+
30	3.03E+04	=	4.36E+03	+	7.50E+05	+	6.21E+03	=
+/=/-	13/11/6		14/11/5		15/8/7		13/14/3	
p-değeri	0.410		0.080		**0.039**		**0.048**	

Geliştirilen yöntem ile ilgili bir diğer deneysel çalışma da parametre kontrol aşamasında kullanmış olduğu başarı oranı tabanlı UCB seçim yöntemine yöneliktir. Bunun için HAMS_ABCxDExPSO algoritması, metasezgisel seçiminin herhangi bir yönteme bağlı kalınmadan rastgele yapıldığı yöntem ile kıyaslanmıştır. Adil bir

karşılaştırma olması açısından rastgele seçimli yöntemde HAMS_ABCxDExPSO ile aynı L parametre değeri kullanılmıştır. Diğer parametreler ise başarı oranı hesaplama ve UCB seçimine yönelik olduklarından kullanılmamıştır.

Çizelge 7.10'da HAMS_ABCxDExPSO algoritması ile onun rastgele metasezgisel seçimli türevinin (Rast.) fonksiyon bazında kıyaslaması verilmiştir. Son satırda 30 fonksiyonun ortalama hata değerlerinin hepsini dikkate alan Wilcoxon işaret sırası testi ile elde edilmiş p değerleri yer almaktadır. Genel değerlendirmeye göre, melez algoritmaya eklenen başarı oranı tabanlı UCB seçimi, algoritmanın genel performansını artırmakta ve bu seçim özellikle 30 boyut üstü problemlerde istatistiksel anlamlılık seviyesini yakalamaktadır.

7.1.4 Diğer algoritmalar ile karşılaştırma

Bu bölümde, geliştirilen yöntemin 50 boyutlu problemler üzerinde literatürdeki bazı önemli algoritmalar ile karşılaştırması yer almaktadır. Algoritmalar seçilirken, metasezgisel seviyesinde seçim yapan melez algoritma bulunamadığından mutasyon stratejisi seviyesinde seçim yapan 2 adet algoritma belirlenmiştir. Bunlardan ilki olan SaDE (Qin et al., 2009), DE mutasyon stratejileri seviyesinde uyarlanabilir seçim yapmaktadır. Benzer şekilde MEABC (Wang et al., 2014) algoritması, çeşitli ABC mutasyon stratejilerini kendinden uyarlanabilir olarak seçip uygulamaktadır. Bu iki algoritmanın dışında son olarak, sürekli eniyileme alanında önemli bir yere sahip olan ve yüksek performans gösteren CMA-ES (Hansen et al., 2003) algoritması seçilmiştir. SaDE ve MEABC algoritmalarının gerçekleştirimi JAVA programlama diliyle yapılmış olup ilgili yayınlarda önerilen parametre değerleri kullanılmıştır. CMA-ES için ise "Apache Commons Math" JAVA kütüphanesi içerisinde yer alan gerçekleştirim [3] kullanılmıştır. Tüm algoritmalar için topluluk büyüklüğü önceki deneylerde olduğu gibi 50 olarak verilmiştir.

Çizelge 7.11'de SaDE, MEABC ve CMA-ES algoritmalarından elde edilen ortalama hata değerleri yer almaktadır. Son sütunda, geliştirilen HAMS_ABCxDExPSO algoritmasına ait ortalama hata değerleri bulunmaktadır. Seçilen 3 adet algoritmanın, tek tek fonksiyon bazında HAMS_ABCxDExPSO ile Mann Whitney U testi kullanılarak kıyaslanması sonucu elde edilen sonuçlar ilgili "T" sütunlarında verilmektedir. Son satırda yer alan toplam sonuçlar incelendiğinde, geliştirilen melez algoritmanın performansı SaDE ve MEABC

[3] http://commons.apache.org/proper/commons-math/

algoritmalarının üzerine çıktığı görülmektedir. Ancak, CMA-ES algoritmasının genel performansı tez kapsamında önerilen yöntemi geçmektedir.

Çizelge 7.11. SaDE, MEABC ve CMA-ES algoritmalarının ortalama hata değerleri ve fonksiyon bazında HAMS_ABCxDExPSO algoritması ile kıyaslanması.

Fonk.	SaDE	T	MEABC	T	CMAES	T	HAMS_ABCxDExPSO
1	4.91E+02	-	6.60E+03	+	**0.00E+00**	-	4.18E+03
2	8.19E+18	+	8.29E+23	+	**0.00E+00**	-	1.81E+08
3	6.43E+04	+	2.55E+05	+	**0.00E+00**	-	9.44E+03
4	4.55E+01	-	5.60E+01	=	4.34E+01	-	6.22E+01
5	6.89E+01	+	1.25E+02	+	3.48E+01	=	3.34E+01
6	**0.00E+00**	-	**0.00E+00**	-	4.00E-03	-	4.26E-03
7	1.16E+02	+	1.63E+02	+	8.32E+01	=	8.48E+01
8	6.89E+01	+	1.25E+02	+	3.83E+01	+	3.15E+01
9	8.72E-01	-	1.18E+03	+	1.03E-01	-	2.39E+01
10	3.57E+03	+	3.91E+03	+	2.62E+03	=	2.61E+03
11	6.72E+01	+	2.00E+03	+	1.25E+02	+	6.08E+01
12	1.13E+05	-	3.82E+06	+	1.29E+03	-	1.38E+05
13	1.17E+03	-	7.04E+03	+	8.26E+01	-	3.92E+03
14	2.47E+04	+	9.17E+05	+	5.42E+01	-	5.96E+02
15	9.12E+02	-	8.56E+03	+	8.27E+01	-	4.06E+03
16	9.25E+02	+	1.13E+03	+	5.97E+02	+	3.91E+02
17	6.62E+02	+	7.95E+02	+	5.18E+02	+	3.34E+02
18	2.32E+05	+	1.46E+06	+	3.01E+01	-	1.06E+04
19	3.31E+03	-	1.32E+04	+	6.14E+01	-	6.63E+03
20	5.40E+02	+	6.22E+02	+	2.16E+02	+	1.67E+02
21	2.66E+02	+	3.39E+02	+	2.40E+02	+	2.34E+02
22	3.55E+03	+	4.35E+03	+	9.40E+02	+	4.49E+02
23	4.91E+02	+	5.85E+02	+	4.68E+02	+	4.50E+02
24	5.65E+02	+	8.49E+02	+	5.27E+02	-	5.32E+02
25	5.50E+02	+	5.14E+02	=	5.23E+02	=	5.15E+02
26	1.83E+03	+	1.63E+03	=	1.38E+03	-	1.49E+03
27	5.55E+02	+	6.61E+02	+	5.98E+02	+	5.42E+02
28	4.96E+02	+	4.83E+02	=	4.91E+02	=	4.80E+02
29	4.96E+02	+	9.27E+02	+	6.75E+02	+	4.77E+02
30	6.90E+05	+	9.04E+05	+	6.14E+05	-	6.38E+05
+/=/-	22/0/8		25/4/1		10/5/15		-

Çizelge 7.12. HAMS_ABCxDExPSO algoritmasının SaDE, MEABC ve CMA-ES algoritmaları ile veri setinin tümü için Friedman skorları ve Wilcoxon işaret sırası testi kullanılarak karşılaştırılması.

	HAMS_ABCxDExPSO	SaDE	MEABC	CMA-ES
Friedman sıra değeri	1.9	2.72	3.68	**1.7**
p-değeri	-	**0.03**	**3.34E-6**	0.15

HAMS_ABCxDExPSO algoritmasının SaDE, MEABC ve CMA-ES algoritmaları ile karşılaştırılması ise Çizelge 7.12'de yer almaktadır. Buna göre, en iyi Friedman skorunu CMA-ES elde etmekte, geliştirilen yöntem ise ikinci sırada bulunmaktadır. HAMS_ABCxDExPSO algoritmasının tüm veri için diğer

algoritmalarla Wilcoxon işaret sırası testi kullanılarak karşılaştırılması sonucu elde edilen p değerleri ise SaDE ve MEABC algoritmaları ile farkın istatistiksel anlamlılık seviyesinde olduğunu göstermektedir. CMA-ES ile farkın ise $p = 0.05$ eşik değerinin üzerinde olduğu görülmektedir.

7.1.5 Sonuç ve değerlendirme

Bu bölümde, tez kapsamında sınır kısıtlamalı sürekli eniyileme probleminin çözümü için önerilmiş olan HAMS_ABCxDExPSO algoritmasının CEC'17 veri seti üzerinde gerçekleştirilen deneysel çalışması sunulmuştur. Öncelikle, geliştirilen yöntemin kendisini oluşturan 3 adet metasezgisel ile karşılaştırması yapılmış ve melez yapının değişken problem boyutlarında ne gibi bir katkı verdiğinin anlaşılması sağlanmıştır. Elde edilen bulgular, melez yöntemin avantajının artan problem boyutlarında daha fazla ortaya çıktığına işaret etmektedir. Bir diğer deneysel çalışma da uyarlanabilir seçim yöntemi için yapılmıştır. Bu amaçla, geliştirilen algoritma, kendisinin rastgele seçimli türevi ile kıyaslanmış ve özellikle büyük boyutlu problemlerde olmak üzere anlamlı performans artışı getirdiği gösterilmiştir. Son olarak, geliştirilen yöntem literatürdeki bazı önemli algoritmalar ile kıyaslanmış ve onlarla rekabet edebilir düzeyde olduğu ortaya koyulmuştur.

7.2 MA_ILSxVND İçin Deneysel Çalışma

7.2.1 Kullanılan veri setleri

Bu alt bölümde CVRP kapsamında gerçekleştirilen deneysel çalışmalarda kullanılan veri setleri ve içeriklerine ait bilgiler yer almaktadır. Veri setleri içerisindeki örnekleri tanımlayan bazı değişkenler ise şunlardır: n: müşteri sayısı; L: maksimum rota süresi; Q: araç kapasitesi.

Christofides et al. (1979) CMT veri seti: CMT veri seti müşteri sayıları 50-199 arasında değişebilen 14 farklı problem örneği barındırmaktadır (Çizelge 7.13). Müşteri dağılımları ilk 10 örnekte rastgele bir şekilde iken son 4 örnekte kümeli bir yapıdadır. Ayrıca örneklerin yarısında maksimum rota süresi kısıtı uygulanmaktadır.

Çizelge 7.13. CMT veri seti problem örnekleri ve özellikleri (Uchoa et al.'dan, 2017).

Problem Örneği	n	L	Q	Bilinen En iyi[4]
CMT 1	50	∞	160	524.61*
CMT 2	75	∞	140	835.26*
CMT 3	100	∞	200	826.14*
CMT 4	150	∞	200	1028.42*
CMT 5	199	∞	200	1291.29*
CMT 6	50	200	160	555.43
CMT 7	75	160	140	909.68
CMT 8	100	230	200	865.94
CMT 9	150	200	200	1162.55
CMT 10	199	200	200	1395.85
CMT 11	120	∞	200	1042.11*
CMT 12	100	∞	200	819.56*
CMT 13	120	720	200	1541.14
CMT 14	100	1040	200	866.37

Golden et al. (1998) veri seti: Golden et al. veri seti müşteri sayıları 240-483 arasında değişebilen 20 farklı problem örneği barındırmaktadır (Çizelge 7.14). Müşteri dağılımları kare, yıldız, daire vb. geometrik şekillere benzeyecek şekilde oluşturulmuştur. Ayrıca, 1-8 arasındaki problem örneklerinde maksimum rota uzunluğu kısıtı uygulanmaktadır.

[4] * simgesi, en iyi olduğu kanıtlanmış değerleri göstermektedir.

Çizelge 7.14. Golden et al. veri seti problem örnekleri ve özellikleri (Uchoa et al.'dan, 2017).

Problem Örneği	n	L	Q	Bilinen En iyi [5]
Golden 1	240	650	550	5623.27
Golden 2	320	900	700	8404.61
Golden 3	400	1200	900	11036.22
Golden 4	480	1600	1000	13590
Golden 5	200	1800	900	6460.98
Golden 6	280	1500	900	8400.33
Golden 7	360	1300	900	10102.7
Golden 8	440	1200	900	11635.3
Golden 9	225	∞	1000	579.71
Golden 10	323	∞	1000	735.66
Golden 11	399	∞	1000	912.03
Golden 12	483	∞	1000	1101.5
Golden 13	252	∞	1000	857.19
Golden 14	320	∞	1000	1080.55*
Golden 15	396	∞	1000	1337.87
Golden 16	480	∞	1000	1611.56
Golden 17	240	∞	200	707.76*
Golden 18	300	∞	200	995.13*
Golden 19	360	∞	200	1365.6*
Golden 20	420	∞	200	1817.59

Uchoa et al. (2017) veri seti: Uchoa et al. veri seti müşteri sayıları 100-1000 arasında değişebilen 100 adet problem örneği içermektedir. Bu veri seti eşdeğerlerine göre daha büyük boyutlu ve fazla sayıda örnek içermekte olup, farklı müşteri, depo ve talep dağılımları ile yeterince kapsayıcı niteliktedir.

Problem örnekleri üretilirken [0, 1000] x [0, 1000] sınırları içerisinde yer alan tamsayı koordinat değerleri kullanılmaktadır. Depo konumları orta (500, 500), dış merkezli (0, 0) ve rastgele olmak üzere belirlenmekte olup sırasıyla C, E ve R harfleri ile temsil edilmektedirler. Müşteri konumları rastgele (R), kümelenmiş (C) ve yarı kümelenmiş (RC) olmak üzere üç farklı dağılıma göre üretilmektedir. Müşteri talepleri ise sabit değer, belirli aralıkta rastgele değer ya da standart sapma ile ortalamanın oranı olarak tanımlanan değişim katsayısına (CV) göre rastgele değer olmak üzere 7 farklı kurala göre aşağıdaki gibi oluşturulmaktadır:

- *Birimsel (U)*: Tüm talepler 1 değerini almaktadır.
- *Küçük değerler, büyük CV (1-10)*: Talepler [1,10] tekdüze dağılımına göre üretilmektedir.

[5] * simgesi, en iyi olduğu kanıtlanmış değerleri göstermektedir.

- *Küçük değerler, küçük CV (5*-10): Talepler [5,10] tekdüze dağılımına göre üretilmektedir.
- *Büyük değerler, büyük CV (1*-100): Talepler [1,100] tekdüze dağılımına göre üretilmektedir.
- *Büyük değerler, küçük CV (50*-100): Talepler [50,100] tekdüze dağılımına göre üretilmektedir.
- *Çeyrek daireye göre (Q)*: (500, 500) noktası merkez olmak üzere talepler, eğer müşteri çift numaralı çeyrek dairede yer alıyorsa [1, 50] tekdüze dağılımına göre, değilse, [51, 100] tekdüze dağılımına göre üretilmektedir.
- *Fazla sayıda küçük değerler, az sayıda büyük değerler (SL)*: Taleplerin büyük bölümü (%70-%90 müşteri) [1, 10] tekdüze dağılımına göre, kalanlar ise [50, 100] tekdüze dağılımına göre üretilmektedir.

CVRP çözümlerinin içerdiği minimum rota sayısının (K_{min}) müşteri sayısına (n) bölünmesi ile hesaplanan ortalama rota sayısı (r) özellikle kesin çözüm algoritmalarının başarısını etkilemektedir. Bu nedenle veri setinde farklı r değerlerine sahip problem örneklerinin bulunmasına dikkat edilmektedir. Bu veri seti hakkında değinilmesi gereken bir diğer önemli nokta ise TSPLIB[6] geleneğine uyularak Öklid uzaklıkları hesabında en yakın tamsayıya yuvarlanma yönteminin kullanılmasıdır.

Uchoa et al. veri setinde yer alan problem örneklerinin adları ve sahip olduğu özellikler Çizelge 7.15 ve Çizelge 7.16'da iki parça halinde sunulmaktadır. Müşteri sayıları artan sıradadır. Müşteri yerleşimlerinin kümeli ve yarı kümeli olduğu durumlarda kullanılan küme çekirdekleri sayısı parantez içerisinde verilmektedir. Daha önce ifade edildiği gibi r sütunu bir rotada bulunan ortalama müşteri sayısını ifade etmekte olup, n/K_{min} sütunu gerçekte bu değere ne kadar yaklaşıldığını göstermektedir.

[6] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf

Çizelge 7.15. Uchoa et al. veri seti ve özellikleri (1. Bölüm).

Problem Örneği	n	Depo	Müşteri	Talep	Q	r	n/K_{min}	Bilinen En iyi
X-n101-k25	100	R	RC (7)	1–100	206	4.0	4	27591
X-n106-k14	105	E	C (3)	50–100	600	8.0	7.5	26362
X-n110-k13	109	C	R	5–10	66	8.8	8.4	14971
X-n115-k10	114	C	R	SL	169	12.5	11.4	12747
X-n120-k6	119	E	RC (8)	U	21	21.8	19.8	13332
X-n125-k30	124	R	C (5)	Q	188	4.2	4.1	55539
X-n129-k18	128	E	RC (8)	1–10	39	7.4	7.1	28940
X-n134-k13	133	R	C (4)	Q	643	10.4	10.2	10916
X-n139-k10	138	C	R	5–10	106	14.0	13.8	13590
X-n143-k7	142	E	R	1–100	1190	22.6	20.3	15700
X-n148-k46	147	R	RC (7)	1–10	18	3.2	3.2	43448
X-n153-k22	152	C	C (3)	SL	144	7.1	6.9	21220
X-n157-k13	156	R	C (3)	U	12	12.0	12	16876
X-n162-k11	161	C	RC (8)	50–100	1174	15.5	14.6	14138
X-n167-k10	166	E	R	5–10	133	17.8	16.6	20557
X-n172-k51	171	C	RC (5)	Q	161	3.4	3.4	45607
X-n176-k26	175	E	R	SL	142	6.8	6.7	47812
X-n181-k23	180	R	C (6)	U	8	8.4	7.8	25569
X-n186-k15	185	R	R	50–100	974	13.0	12.3	24145
X-n190-k8	189	E	C (3)	1–10	138	25.0	23.6	16980
X-n195-k51	194	C	RC (5)	1–100	181	3.8	3.8	44225
X-n200-k36	199	R	C (8)	Q	402	5.6	5.5	58578
X-n204-k19	203	C	RC (6)	50–100	836	11.2	10.7	19565
X-n209-k16	208	E	R	5–10	101	13.5	13	30656
X-n214-k11	213	C	C (4)	1–100	944	19.4	19.4	10856
X-n219-k73	218	E	R	U	3	3.6	3	117595
X-n223-k34	222	R	RC (5)	1–10	37	6.5	6.5	40437
X-n228-k23	227	R	C (8)	SL	154	10.0	9.9	25742
X-n233-k16	232	C	RC (7)	Q	631	14.5	14.5	19230
X-n237-k14	236	E	R	U	18	18.6	16.9	27042
X-n242-k48	241	E	R	1–10	28	5.0	5	82751
X-n247-k50	246	C	C (4)	SL	134	5.3	4.9	37274
X-n251-k28	250	R	RC (3)	5–10	69	9.2	8.9	38684
X-n256-k16	255	C	C (8)	50–100	1225	16.0	15.9	18880
X-n261-k13	260	E	R	1–100	1081	21.0	20	26558
X-n266-k58	265	R	RC (6)	5–10	35	4.6	4.6	75478
X-n270-k35	269	C	RC (5)	50–100	585	7.7	7.7	35291
X-n275-k28	274	R	C (3)	U	10	10.8	9.8	21245
X-n280-k17	279	E	R	SL	192	16.5	16.4	33503
X-n284-k15	283	R	C (8)	1–10	109	20.2	18.9	20226
X-n289-k60	288	E	RC (7)	Q	267	4.8	4.8	95151
X-n294-k50	293	C	R	1–100	285	5.9	5.9	47167
X-n298-k31	297	R	R	1–10	55	9.6	9.6	34231
X-n303-k21	302	C	C (8)	1–100	794	15.0	14.4	21744
X-n308-k13	307	E	RC (6)	SL	246	24.2	23.6	25859
X-n313-k71	312	R	RC (3)	Q	248	4.4	4.4	94044
X-n317-k53	316	E	C (4)	U	6	6.2	6	78355
X-n322-k28	321	C	R	50–100	868	11.6	11.5	29866
X-n327-k20	326	R	RC (7)	5–10	128	17.0	16.3	27556
X-n331-k15	330	E	R	U	23	23.4	22	31103

Çizelge 7.16. Uchoa et al. veri seti ve özellikleri (2. bölüm).

Problem Örneği	n	Depo	Müşteri	Talep	Q	r	n/K_{min}	Bilinen En iyi
X-n336-k84	335	E	R	Q	203	4.0	4	139197
X-n344-k43	343	C	RC (7)	5–10	61	8.0	8	42099
X-n351-k40	350	C	C (3)	1–100	436	8.8	8.8	25946
X-n359-k29	358	E	RC (7)	1–10	68	12.5	12.3	51509
X-n367-k17	366	R	C (4)	SL	218	21.8	21.5	22814
X-n376-k94	375	E	R	U	4	4.2	4	147713
X-n384-k52	383	R	R	50–100	564	7.4	7.4	66081
X-n393-k38	392	C	RC (5)	5–10	78	10.4	10.3	38269
X-n401-k29	400	E	C (6)	Q	745	14.0	13.8	66243
X-n411-k19	410	R	C (5)	SL	216	22.6	21.6	19718
X-n420-k130	419	C	RC (3)	1–10	18	3.2	3.2	107798
X-n429-k61	428	R	R	50–100	536	7.1	7	65501
X-n439-k37	438	C	RC (8)	U	12	12.0	11.8	36391
X-n449-k29	448	E	R	1–100	777	15.5	15.4	55358
X-n459-k26	458	C	C (4)	Q	1106	17.8	17.6	24181
X-n469-k138	468	E	R	50–100	256	3.4	3.4	221909
X-n480-k70	479	R	C (8)	5–10	52	6.8	6.8	89535
X-n491-k59	490	R	RC (6)	1–100	428	8.4	8.3	66633
X-n502-k39	501	E	C (3)	U	13	13.0	12.8	69253
X-n513-k21	512	C	RC (4)	1–10	142	25.0	24.4	24201
X-n524-k153	523	R	R	SL	125	3.8	3.4	154594
X-n536-k96	535	C	C (7)	Q	371	5.6	5.6	95122
X-n548-k50	547	E	R	U	11	11.2	10.9	86710
X-n561-k42	560	C	RC (7)	1–10	74	13.5	13.3	42756
X-n573-k30	572	E	C (3)	SL	210	19.4	19.1	50780
X-n586-k159	585	R	RC (4)	5–10	28	3.6	3.7	190543
X-n599-k92	598	R	R	50–100	487	6.5	6.5	108813
X-n613-k62	612	C	R	1–100	523	10.0	9.9	59778
X-n627-k43	626	E	C (5)	5–10	110	14.5	14.6	62366
X-n641-k35	640	E	RC (8)	50–100	1381	18.6	18.3	63839
X-n655-k131	654	C	C (4)	U	5	5.0	5	106780
X-n670-k130	669	R	R	SL	129	5.3	5.1	146705
X-n685-k75	684	C	RC (6)	Q	408	9.2	9.1	68425
X-n701-k44	700	E	RC (7)	1–10	87	16.0	15.9	82292
X-n716-k35	715	R	C (3)	1–100	1007	21.0	20.4	43525
X-n733-k159	732	C	R	1–10	25	4.6	4.6	136366
X-n749-k98	748	R	C (8)	1–100	396	7.7	7.6	77700
X-n766-k71	765	E	RC (7)	SL	166	10.8	10.8	114683
X-n783-k48	782	R	R	Q	832	16.5	16.3	72727
X-n801-k40	800	E	R	U	20	20.2	20	73587
X-n819-k171	818	C	C (6)	50–100	358	4.8	4.8	158611
X-n837-k142	836	R	RC (7)	5–10	44	5.9	5.9	194266
X-n856-k95	855	C	RC (3)	U	9	9.6	9	89060
X-n876-k59	875	E	C (5)	1–100	764	15.0	14.8	99715
X-n895-k37	894	R	R	50–100	1816	24.2	24.2	54172
X-n916-k207	915	E	RC (6)	5–10	33	4.4	4.4	329836
X-n936-k151	935	C	R	SL	138	6.2	6.2	133105
X-n957-k87	956	R	RC (4)	U	11	11.6	11	85672
X-n979-k58	978	E	C (6)	Q	998	17.0	16.9	119194
X-n1001-k43	10 0 0	R	R –	1–10	131	23.4	23.3	72742

7.2.2 Deneysel kurulum

MA_ILSxVND algoritmasının gerçekleştirimi Java programlama dili ile Intel Core i7-6700 3.40 GHz CPU ve 16 GB RAM özelliklerine sahip bilgisayar üzerinde yapılmıştır.

Yapılan deneysel çalışmalarda, geliştirilen algoritmalar her bir problem örneği için 10'ar kez farklı rastgele çekirdek değeri ile çalıştırılmıştır. Algoritmaların sonlanma koşulu ise maksimum iterasyon sayısı ile belirlenmektedir.

Bir algoritmanın verilen bir veri seti için ne kadar başarılı olduğuna ortalama hata yüzdesi (OHY) hesabı ile karar verilmektedir (Eşitlik 7.1). OHY, ilgili algoritmanın bilinen en iyi uygunluk değerlerinden ortalama % kaç oranında sapma gösterdiğini ifade etmektedir. Eşitlikte yer alan n, veri setindeki toplam problem örneği sayısını, i, veri setindeki i. problem örneğini, f_i, algoritma tarafından i. problem için bulunan en iyi uygunluk değerini, f_i^* ise literatürde i. problem için rapor edilen en iyi çözüm değerini ifade etmektedir.

$$\text{OHY} = \sum_{i=1}^{n}\left(\frac{f_i - f_i^*}{f_i^*} x100\right) \tag{7.1}$$

7.2.2.1 Parametrelerin belirlenmesi

MA_ILSxVND algoritmasının parametrelerinin belirlenmesi iRace (Lopez-Ibanez et al., 2016) yazılım paketi kullanılarak gerçekleştirilmiştir. iRace konfigürasyonu sırasında maksimum bütçe 1000 iterasyon olarak belirlenmiş olup ilgili veri setinin tek numaralı örnekleri eğitim, çift numaralı örnekleri ise test için ayrılmıştır. Bunun dışındaki konfigürasyon varsayılan değerleri ile bırakılmıştır. Hedef algoritmanın maksimum iterasyon sayısı ise 20000 olarak belirlenmiştir.

Çizelge 7.17. MA_ILSxVND algoritmasının iRace ile ayarlanmış parametre değerleri.

Parametre adı	**Parametre türü**	**Parametre ayarlama değer aralığı**	**Ayarlanmış değer (CMT)**	**Ayarlanmış değer (Golden et al.)**	**Ayarlanmış değer (Uchoa et al.)**
pKatsayı	Reel	[0.25, 2.0]	0.54	1.54	1.04
thKatsayı	Tamsayı	[1, 16]	8	14	9
k	Tamsayı	[2, 5]	4	5	4
bi	Kategorik	(true, false)	false	false	false

Çizelge 7.17'nin ilk 3 kolonunda sırasıyla algoritmada yer alan parametreler, türleri ve değer aralıkları yer almaktadır. Son üç kolon ise sırasıyla CMT, Golden et al. ve Uchoa et al. veri setleri için ayarlanmış parametre değerlerini göstermektedir.

7.2.3 Deneysel sonuçlar

7.2.3.1 CMT veri seti için deneysel sonuçlar

MA_ILSxVND algoritmasının belirlenen parametre değerleri ile CMT veri seti üzerinde 40000 iterasyon çalıştırılması ile elde edilen sonuçlar Çizelge 7.18'de sunulmaktadır. Koyu renkli değerler, literatürde o problem örneği için bilinen en iyi sonuca ulaşıldığını göstermektedir. En iyi, en kötü ve ortalama sütunları sırasıyla, algoritmanın 10 kez çalıştırılması sonucu elde edilen en iyi, en kötü ve ortalama sonuçları ifade etmektedir. Süre sütunu, ortalama algoritma çalışma süresini saniye cinsinden vermektedir. En alt satırda yer alan ortalama satırı ise ilk üç sütun için OHY değerini gösterirken, süre sütunu için çalışma zamanlarının ortalamasını temsil etmektedir.

Çizelge 7.18. MA_ILSxVND algoritmasının CMT veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar.

Problem	En iyi	En kötü	Ortalama	Süre(sn.)	Bilinen En iyi
CMT 1	**524.61**	**524.61**	**524.61**	35.08	524.61
CMT 2	**835.26**	**835.26**	**835.26**	72.54	835.26
CMT 3	**826.14**	827.39	826.27	148.65	826.14
CMT 4	**1028.42**	1031.07	1029.24	237.02	1028.42
CMT 5	1291.50	1305.30	1299.59	369.57	1291.29
CMT 6	**555.43**	**555.43**	**555.43**	25.85	555.43
CMT 7	912.30	912.89	912.83	53.52	909.68
CMT 8	**865.94**	**865.94**	**865.94**	76.67	865.94
CMT 9	**1162.55**	1166.1	1163.75	144.41	1162.55
CMT 10	1399.48	1408.52	1402.23	204.51	1395.85
CMT 11	**1042.11**	**1042.11**	**1042.11**	437.61	1042.11
CMT 12	**819.56**	**819.56**	**819.56**	179.88	819.56
CMT 13	1542.86	1542.97	1542.88	228.60	1541.14
CMT 14	**866.37**	**866.37**	**866.37**	158.21	866.37
-	**OHY**	**OHY**	**OHY**	**Ort. Süre**	-
	0.05	0.23	0.13	169.44	

Çizelge 7.18'deki sonuçlara bakıldığında, ortalama çözüm değerleri için %0.13'lük hata hesaplandığı görülmektedir. Bu değer, en iyi çözümler dikkate alındığında %0.05'e kadar düşerken, en kötü çözümler için %0.23'e çıkmaktadır.

Bunun dışında, 1., 2., 6., 8., 11., 12. ve 14. problemler için her durumda bilinen en iyi sonuca ulaşıldığı görülmektedir. Sadece en iyi çözümlere bakıldığında ise 3., 4. ve 9. problemler de bu listeye eklenerek 14 problemin 10 tanesinde bilinen en iyi sonuca ulaşılmaktadır. Algoritma çalışma süreleri problemlerin boyutlarına göre değişebilmekte olup, bir problem için ortalama 159.29 sn. süre harcandığı belirlenmiştir.

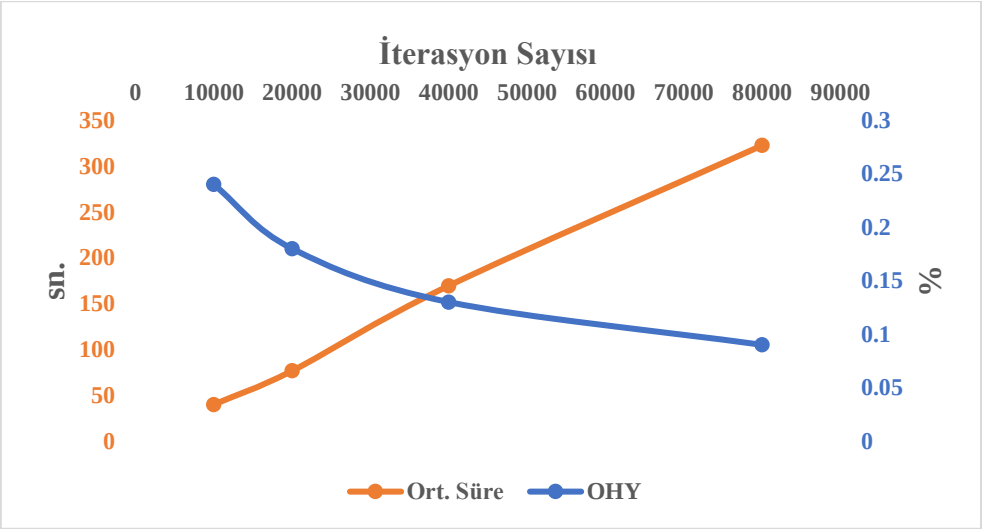
MA_ILSxVND algoritmasının farklı iterasyon sayılarındaki davranışını izleyebilmek için bir deneysel çalışma gerçekleştirilmiştir. Çizelgede yer alan CVRP maliyet değerleri 10 farklı çalıştırma sonucunun ortalamasıdır. Daha önce 40000 iterasyon ile çalıştırılan algoritma, 10000, 20000 ve 80000 iterasyon için tekrar çalıştırılmış ve elde edilen sonuçlar Çizelge 7.19'da sunulmuştur. Karşılaştırmanın daha sağlıklı yapılabilmesi açısından her bir iterasyon sayısı için aynı rastgele sayı çekirdeği seti kullanılmıştır. Değişen iterasyon sayıları için elde edilen ortalama çözüm değerleri ve çalışma süreleri çizelgeden görülebilmektedir. Koyu renkli değerler, literatürde o problem örneği için bilinen en iyi sonuca ulaşıldığını göstermektedir. En alt satırda yer alan ortalama satırı ise çözüm değeri sütunları için OHY değerlerini gösterirken, süre sütunları için çalışma zamanlarının ortalamasını temsil etmektedir.

Çizelge 7.19. MA_ILSxVND algoritmasının CMT veri seti üzerinde artan iterasyon sayısı ile çalıştırılması karşısındaki davranışı.

Prob.	**10000 iterasyon**	**20000 iterasyon**	**40000 iterasyon**	**80000 iterasyon**	**Bilinen En İyi**
CMT 1	**524.61**	**524.61**	**524.61**	**524.61**	524.61
CMT 2	836.34	835.49	**835.26**	**835.26**	835.26
CMT 3	826.71	826.39	826.27	**826.14**	826.14
CMT 4	1030.41	1028.75	1029.24	1029.04	1028.42
CMT 5	1303.43	1301.88	1299.59	1295.38	1291.29
CMT 6	**555.43**	**555.43**	**555.43**	**555.43**	555.43
CMT 7	912.89	912.89	912.83	912.51	909.68
CMT 8	**865.94**	**865.94**	**865.94**	**865.94**	865.94
CMT 9	1169.56	1168.56	1163.75	1163.199	1162.55
CMT 10	1408.26	1404.93	1402.23	1401.892	1395.85
CMT 11	**1042.11**	**1042.11**	**1042.11**	**1042.11**	1042.11
CMT 12	**819.56**	**819.56**	**819.56**	**819.56**	819.56
CMT 13	1543.30	1543.06	1542.88	1542.86	1541,14
CMT 14	**866.37**	**866.37**	**866.37**	**866.37**	866.37
OHY	0.24	0.18	0.13	0.09	0.00
Ort. Süre	39.89	76.62	169.44	322.46	-

İterasyon sayısının 80000'e çıkarılması sonucunda ortalama hata yüzdesinin önemli bir iyileştirmeyle 0.09'a kadar geriletildiği Çizelge 7.19'da gösterilmektedir. 20000'e indirildiğinde ise bu değer %0.18'e çıkmaktadır. 10000

iterasyonda ise ortalama hata daha da artarak %0.24'e ulaşmaktadır. Algoritmaların çalışma süreleri ise artan iterasyon sayılarına göre sırasıyla 39.89, 76.62, 169.44 ve 322.46 sn. olarak ölçülmüştür. Artan iterasyon sayısına paralel olarak sürelerdeki artış ve OHY'deki azalış eğilimi Şekil 7.2'den görülebilmektedir.



Şekil 7.2. MA_ILSxVND algoritmasının CMT veri seti üzerinde farklı iterasyon sayıları ile çalıştırılması sonucu elde edilen ortalama süre ve OHY değişimi.

MA_ILSxVND algoritması ILSxVND melez algoritmasına çok başlangıçlı strateji ve uyarlanabilir kabul fonksiyonu bileşenleri eklenerek geliştirilmiştir. Bu bileşenlerin algoritmanın çözüm kalitesine hangi ölçüde katkı sağladığının görülebilmesi amacıyla bir deneysel çalışma gerçekleştirilmiş olup Çizelge 7.20'de sunulmaktadır. Çizelgede yer alan CVRP maliyet değerleri 10 farklı çalıştırma sonucunun ortalamasıdır. Bu deneysel çalışmada yer alan farklı bileşen kombinasyonları aşağıda listelenmektedir:

- **ILSxVND**: Yalın ILSxVND melez metasezgiseli
- **A_ILSxVND**: ILSxVND melez metasezgiseli ve uyarlanabilir kabul fonksiyonunun birlikte kullanımı
- **M_ILSxVND**: ILSxVND melez metasezgiseli ve çok başlangıçlı stratejinin birlikte kullanımı

Çizelge 7.20'de yer alan sonuçlara göre en yüksek ortalama hata yüzdesini 0.45 ile yalın ILSxVND melez metasezgiseli vermektedir. Bu algoritmaya uyarlanabilir kabul fonksiyonunun eklenmesi ile %0.07'lik bir iyileştirme ile OHY 0.38'e düşmektedir. ILSxVND melez metasezgiseline çok başlangıçlı strateji eklenmesi ise %0.22'lik bir iyileştirme sağlayarak OHY değerini 0.23'e

geriletmiştir. Her iki stratejinin birlikte uygulandığı ve tez kapsamında önerilmiş olan MA_ILSxVND algoritması ise ILSxVND'ye göre %0.32 iyileştirme sağlayarak OHY değerini 0.13'e kadar çekmektedir. Buradan çıkan sonuca göre, melez algoritmaya eklenen her iki strateji de gerekli ve ortalama performansı artırıcı niteliktedir. Algoritmaların çalışma zamanları dikkate alındığında, çok başlangıçlı stratejinin ve uyarlanabilir kabul fonksiyonunun küçük bir maliyet artışı getirdiği görülmektedir. Ancak, tur uzunluğu maliyetlerindeki iyileştirme miktarına kıyasla bu artış ihmal edilebilir düzeydedir.

Çizelge 7.20. MA_ILSxVND algoritmasının CMT veri seti üzerinde farklı bileşen kombinasyonlarına göre karşılaştırılması (40000 iterasyon).

Prob.	**ILSxVND**	**A_ILSxVND**	**M_ILSxVND**	**MA_ILSxVND**	**Bilinen En İyi**
CMT 1	**524.61**	**524.61**	**524.61**	**524.61**	524.61
CMT 2	837.74	839.75	835.61	**835.26**	835.26
CMT 3	827.39	827.63	827.27	826.27	826.14
CMT 4	1031.46	1030.65	1029.30	1029.24	1028.42
CMT 5	1305.12	1306.63	1300.12	1299.59	1291.29
CMT 6	558.59	557.56	**555.43**	**555.43**	555.43
CMT 7	920.80	914.56	912.76	912.83	909.68
CMT 8	869.42	869.61	**865.94**	**865.94**	865.94
CMT 9	1174.67	1170.76	1164.71	1163.75	1162.55
CMT 10	1410.79	1405.83	1407.97	1402.23	1395.85
CMT 11	**1042.11**	**1042.11**	**1042.11**	**1042.11**	1042.11
CMT 12	**819.56**	**819.56**	**819.56**	**819.56**	819.56
CMT 13	1543.92	1547.39	1543.13	1542.88	1541,14
CMT 14	866.37	866.37	866.37	866.37	866.37
OHY	0.45	0.38	0.18	0.13	-
Ort. Süre	151.44	159.03	168.70	169.44	

Çizelge 7.20'deki sonuçların istatistiksel anlamlılığını ölçmek için Wilcoxon işaret sırası testi (signed rank test) uygulanmıştır. İkili karşılaştırmalar sonucu elde edilmiş olan p değerleri Çizelge 7.21'de sunulmaktadır. 0.05 düzeyinin altında yer alan değerler koyu renkle gösterilmiş olup iki algoritma arasında istatistiksel olarak anlamlı bir farka işaret etmektedir. Buna göre, geliştirilen MA_ILSxVND algoritmasının CMT veri seti üzerindeki performansı, diğer üç algoritma ile kıyaslandığında istatistiksel olarak anlamlı çıkmaktadır. Bu durum, yeniden başlatma ve uyarlanabilir kabul fonksiyonu stratejilerinin her ikisinin beraber kullanılmasının, teker teker kullanımlarına göre daha avantajlı olduğunu ortaya koymaktadır. Çizelgeden çıkarılabilecek bir diğer sonuç da yalın ILSxVND algoritmasına sadece uyarlanabilir kabul fonksiyonunun eklenmesinin CMT veri seti için bir fark ortaya koymamasıdır. Bu strateji, yeniden başlatma stratejisi ile birleştiğinde ise fark yaratarak genel performansı artırıcı etki göstermektedir.

Çizelge 7.21. CMT veri seti için MA_ILSxVND, ILSxVND, M_ILSxVND ve A_ILSxVND algoritmaları arasında gerçekleştirilen Wilcoxon işaret sırası testi ile elde edilen p değerleri.

	ILSxVND	**A_ILSxVND**	**M_ILSxVND**	**MA_ILSxVND**
ILSxVND	-	0.557	**0.002**	**0.002**
A_ILSxVND		-	**0.020**	**0.002**
M_ILSxVND			-	**0.023**
MA_ILSxVND				-

7.2.3.2 Golden et al. veri seti için deneysel sonuçlar

MA_ILSxVND algoritmasının belirlenen parametre değerleri ile Golden et al. veri seti üzerinde 40000 iterasyon çalıştırılması ile elde edilen ayrıntılı sonuçlar Çizelge 7.22'de sunulmaktadır.

Çizelge 7.22. MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar.

Problem	**En iyi**	**En kötü**	**Ortalama**	**Süre(sn.)**	**Bilinen En İyi**
Golden 1	5678.94	5699.73	5687.94	244.61	5623.27
Golden 2	8465.72	8493.8	8476.61	255.17	8404.61
Golden 3	11088.62	11138.61	11108.43	390.91	11036.22
Golden 4	13625.72	13732.4	13709.50	603.81	13590
Golden 5	**6460.98**	**6460.98**	**6460.98**	236.52	6460.98
Golden 6	8412.90	8429.56	8414.89	318.47	8400.33
Golden 7	10195.58	10268.37	10245.91	388.80	10102.7
Golden 8	11747.48	11897.66	11818.57	656.61	11635.3
Golden 9	581.40	585.03	583.12	548.12	579.71
Golden 10	740.29	743.22	741.29	827.38	735.66
Golden 11	917.74	922.1	919.30	1150.99	912.03
Golden 12	1108.25	1115.91	1111.53	1588.29	1101.5
Golden 13	858.77	865.72	861.22	313.94	857.19
Golden 14	1081.65	1088.63	1085.80	457.17	1080.55
Golden 15	1348.15	1353.91	1351.35	640.41	1337.87
Golden 16	1621.97	1631.34	1625.80	902.18	1611.56
Golden 17	707.79	708.7	708.20	538.15	707.76
Golden 18	1007.17	1009.03	1007.97	834.72	995.13
Golden 19	1367.53	1374.36	1370.81	1152.27	1365.6
Golden 20	1824.27	1839.43	1831.87	1650.50	1817.59
-	**OHY**	**OHY**	**OHY**	**Ort. Süre**	-
	0.50	1.01	0.75	684.95	

Çizelgede yer alan koyu renkli değerler, literatürde o problem örneği için bilinen en iyi sonuca ulaşıldığını göstermektedir. En iyi, en kötü ve ortalama sütunları sırasıyla, algoritmanın 10 kez çalıştırılması sonucu elde edilen en iyi, en kötü ve ortalama sonuçları ifade etmektedir. Süre sütunu, ortalama algoritma çalışma süresini saniye cinsinden vermektedir. En alt satırda yer alan ortalama satırı

ise ilk üç sütun için OHY değerini gösterirken, süre sütunu için çalışma zamanlarının ortalamasını temsil etmektedir.

Çizelge 7.22'de yer alan sonuçlar incelendiğinde, ortalamada 0.75'lik bir OHY değerinin elde edildiği görülmektedir. Sadece en iyi değerler temel alındığında OHY 0.50'ye gerilerken, en kötü değerler için 1.01'e kadar çıkmaktadır. Algoritmanın çalışma süreleri ise probleme göre değişebilmekte olup ortalama 828.17 sn. olarak ölçülmüştür.

MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde artan iterasyon sayısı ile çalıştırılması karşısındaki davranışı Çizelge 7.23'ten görülebilmektedir. Çizelgede yer alan CVRP maliyet değerleri 10 farklı çalıştırma sonucunun ortalamasıdır. Daha önceden 40000 iterasyon boyunca çalıştırma ile ölçülmüş olan 0.75'lik OHY değeri 80000 iterasyon için 0.65'e gerilemektedir. 10000 ve 20000 iterasyon için çalıştırıldığında ise sırasıyla 1.04 ve 0.88 OHY değerlerini üretmektedir.

Çizelge 7.23. MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde artan iterasyon sayısı ile çalıştırılması karşısındaki davranışı.

Prob.	**10000 iterasyon**	**20000 iterasyon**	**40000 iterasyon**	**80000 iterasyon**	**Bilinen En İyi**
Golden 1	5705.40	5692.74	5687.94	5673.93	5623.27
Golden 2	8501.89	8480.41	8476.61	8475.19	8404.61
Golden 3	11130.92	11114.40	11108.43	11091.87	11036.22
Golden 4	13745.15	13734.25	13709.50	13677.62	13590
Golden 5	**6460.98**	**6460.98**	**6460.98**	**6460.98**	6460.98
Golden 6	8450.43	8450.00	8414.89	8412.99	8400.33
Golden 7	10265.96	10254.16	10245.91	10244.15	10102.7
Golden 8	11904.68	11840.92	11818.57	11783.80	11635.3
Golden 9	585.13	584.56	583.12	582.95	579.71
Golden 10	743.71	742.60	741.29	740.78	735.66
Golden 11	921.29	920.45	919.30	917.82	912.03
Golden 12	1115.51	1113.96	1111.53	1109.82	1101.5
Golden 13	864.19	862.81	861.22	860.52	857.19
Golden 14	1088.60	1086.27	1085.80	1084.32	1080.55
Golden 15	1353.41	1352.44	1351.35	1350.14	1337.87
Golden 16	1629.38	1627.57	1625.80	1624.69	1611.56
Golden 17	711.84	709.24	708.20	707.98	707.76
Golden 18	1009.40	1008.58	1007.97	1007.08	995.13
Golden 19	1374.29	1371.79	1370.81	1370.02	1365.6
Golden 20	1838.71	1835.48	1831.87	1830.10	1817.59
OHY	1.04	0.88	0.75	0.65	-
Ort. Süre	242.61	454.24	684.95	1451.95	

Çizelge 7.23'teki sonuçlardan görüldüğü üzere artan iterasyon sayıları üretilen OHY değerini düşürse de algoritma çalışma sürelerini artırmaktadır. Çalışma

süreleri iterasyon sayılarına göre sırasıyla 242.61, 454.24, 684.95 ve 1451.95 sn. olarak ölçülmüştür. Sürelerdeki artış iterasyon sayısına paralel seyretmektedir.

MA_ILSxVND algoritmasının içerdiği yeniden başlatma stratejisi ve uyarlanabilir kabul fonksiyonu stratejilerinin farklı kombinasyonlarda kullanımlarının karşılaştırması Çizelge 7.24'te verilmektedir. Çizelgede yer alan CVRP maliyet değerleri 10 farklı çalıştırma sonucunun ortalamasıdır. ILSxVND, A_ILSxVND ve M_ILSxVND algoritmaları Bölüm 7.2.3.1'de de tanımlandığı gibidir.

Çizelge 7.24. MA_ILSxVND algoritmasının Golden et al. veri seti üzerinde farklı bileşen kombinasyonlarına göre karşılaştırılması (40000 iterasyon).

Prob.	**ILSxVND**	**A_ILSxVND**	**M_ILSxVND**	**MA_ILSxVND**	**Bilinen En İyi**
Golden 1	5716.33	5704.38	5695.11	5687.94	5623.27
Golden 2	8523.89	8479.93	8494.02	8476.61	8404.61
Golden 3	11142.11	11114.48	11091.70	11108.43	11036.22
Golden 4	13730.00	13730.80	13722.43	13709.50	13590
Golden 5	**6460.98**	**6460.98**	**6460.98**	**6460.98**	6460.98
Golden 6	8486.55	8437.39	8425.19	8414.89	8400.33
Golden 7	10285.33	10245.68	10265.13	10245.91	10102.7
Golden 8	11886.08	11832.16	11872.64	11818.57	11635.3
Golden 9	587.71	584.56	586.91	583.12	579.71
Golden 10	746.18	742.46	746.62	741.29	735.66
Golden 11	924.81	919.24	926.39	919.30	912.03
Golden 12	1121.29	1112.38	1120.85	1111.53	1101.5
Golden 13	863.45	863.55	863.18	861.22	857.19
Golden 14	1090.58	1087.10	1088.59	1085.80	1080.55
Golden 15	1353.83	1351.85	1354.34	1351.35	1337.87
Golden 16	1636.71	1626.21	1636.94	1625.80	1611.56
Golden 17	714.41	714.14	709.36	708.20	707.76
Golden 18	1008.36	1008.82	1008.07	1007.97	995.13
Golden 19	1376.30	1372.84	1374.67	1370.81	1365.6
Golden 20	1838.95	1834.78	1840.26	1831.87	1817.59
OHY	1.22	0.90	1.06	0.75	-
Ort. Süre	677.05	681.10	695.88	684.95	

Çizelge 7.24'ten görüldüğü üzere Golden et al. veri seti için en yüksek OHY değerini yalın ILSxVND algoritması vermekte olup bu algoritmaya sadece uyarlanabilir kabul fonksiyonu eklenmesi OHY'ni 0.90' düşürmektedir. Sadece çok başlangıçlı stratejinin eklenmesi ise daha az bir düşüşe yol açmakta olup 1.06 gibi bir OHY elde edilmektedir. Her iki stratejinin birlikte uygulandığı ve tez kapsamında önerilmiş olan MA_ILSxVND algoritması ise ILSxVND'ye göre %0.47 iyileştirme sağlayarak OHY değerini 0.75'e kadar çekmektedir. Buradan çıkan sonuca göre, melez algoritmaya eklenen her iki strateji de gerekli ve ortalama performansı artırıcı niteliktedir. Algoritmaların çalışma zamanları incelendiğinde

ise yalın ILSxVND algoritması en düşük olmak üzere ölçülen sürelerin birbirine yakın olduğu görülmektedir. Bu durum, eklenen yeniden başlatma ve uyarlanabilir kabul fonksiyonu stratejilerinin getirdiği ek süre maliyetinin ihmal edilebilir seviyede olduğunu göstermektedir.

Çizelge 7.24'te yer alan sonuçların istatistiksel anlamlılığını ölçmek için Wilcoxon işaret sırası testi (signed rank test) uygulanmıştır. İkili karşılaştırmalar sonucu elde edilmiş olan p değerleri Çizelge 7.25'te sunulmaktadır. 0.05 düzeyinin altında yer alan değerler koyu renkle gösterilmiş olup iki algoritma arasında istatistiksel olarak anlamlı bir farka işaret etmektedir. Buna göre, geliştirilen MA_ILSxVND algoritmasının Golden et al. veri seti üzerindeki performansı, diğer üç algoritma ile kıyaslandığında istatistiksel olarak anlamlı çıkmaktadır. Bu durum, yeniden başlatma ve uyarlanabilir kabul fonksiyonu stratejilerinin her ikisinin beraber kullanılmasının, teker teker kullanımlarına göre daha avantajlı olduğunu ortaya koymaktadır. Çizelgeden çıkarılabilecek bir diğer sonuç da yalın ILSxVND algoritmasına sadece çok başlangıçlı eklenmesinin Golden et al. veri seti için bir fark ortaya koymamasıdır. Bu strateji, yeniden başlatma stratejisi ile birleştiğinde ise fark yaratarak genel başarıyı artırıcı etki göstermektedir.

Çizelge 7.25. Golden et al. veri seti için MA_ILSxVND, ILSxVND, M_ILSxVND ve A_ILSxVND algoritmaları arasında gerçekleştirilen Wilcoxon işaret sırası testi ile elde edilen p değerleri.

	ILSxVND	**A_ILSxVND**	**M_ILSxVND**	**MA_ILSxVND**
ILSxVND	-	**4.6339e-04**	**0.0074**	**1.3183e-04**
A_ILSxVND		-	0.2772	**2.1335e-04**
M_ILSxVND			-	**0.0015**
MA_ILSxVND				-

7.2.3.3 Uchoa et al. veri seti için deneysel sonuçlar

MA_ILSxVND algoritmasının belirlenen parametre değerleri ile Uchoa et al. veri seti üzerinde 40000 iterasyon çalıştırılması ile elde edilen ayrıntılı sonuçlar sırasıyla Çizelge 7.26 ve Çizelge 7.27'de iki parça halinde sunulmaktadır. Koyu renkli değerler, literatürde o problem örneği için bilinen en iyi sonuca ulaşıldığını göstermekte iken, hem koyu hem altı çizili değerler literatürdeki en iyi sonucun iyileştirildiğini ifade etmektedir. En iyi, en kötü ve ortalama sütunları sırasıyla, algoritmanın 10 kez çalıştırılması sonucu elde edilen en iyi, en kötü ve ortalama sonuçları ifade etmektedir. Süre sütunu, ortalama algoritma çalışma süresini saniye cinsinden vermektedir. En alt satırda yer alan ortalama satırı ise ilk üç sütun için OHY değerini gösterirken, süre sütunu için çalışma zamanlarının ortalamasını temsil etmektedir.

10 çalıştırma sonucu elde edilen en iyi değerler dikkate alındığında 12 problem örneği için bilinen en iyi sonuca ulaşıldığı, 2 problem örneği için ise bilinen en iyi sonucun iyileştirildiği görülmektedir. En iyi sonuçlara ilişkin OHY değeri 0.34 olarak hesaplanmıştır. En kötü sonuçlar dikkate alındığında OHY 0.80 olurken, algoritmanın genel performansını daha iyi yansıtan ortalama sonuçlar için 0.55 olmaktadır.

MA_ILSxVND algoritmasının ortalama çalışma süresi ise 2359.79 sn. (39.33 dk.) olarak ölçülmüştür. Özellikle büyük boyutlu problem örneklerinin çözümü uzun süreler alabilmektedir. Bu nedenle Uchoa et al. veri seti üzerinde tek bir deneysel çalışma gerçekleştirilmiş olup, CMT ve Golden et al. veri setlerinde yapılan farklı iterasyon sayıları ve problem bileşenlerine ait deneyler tekrar edilmemiştir.

Çizelge 7.26. MA_ILSxVND algoritmasının Uchoa et al. veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar (1. Bölüm).

Problem	En iyi	En kötü	Ortalama	Süre(sn.)	Bilinen En İyi
X-n101-k25	27793	27812	27797.9	48.35	27591
X-n106-k14	26381	26392	26383.3	130.5405	26362
X-n110-k13	**14971**	**14971**	**14971**	44.8061	14971
X-n115-k10	**12747**	**12747**	**12747**	61.5276	12747
X-n120-k6	13332	13340	13332.8	141.2626	13332
X-n125-k30	55967	56027	55987.7	125.0111	55539
X-n129-k18	**28940**	28999	28953.3	125.8666	28940
X-n134-k13	**10916**	10940	10930	129.9198	10916
X-n139-k10	**13590**	13607	13591.7	72.3541	13590
X-n143-k7	15726	15737	15728.2	164.6897	15700
X-n148-k46	**43448**	43566	43503	103.0978	43448
X-n153-k22	21366	21915	21534.5	238.4731	21220
X-n157-k13	**16876**	**16876**	**16876**	244.3694	16876
X-n162-k11	14147	14174	14169.1	107.4069	14138
X-n167-k10	20592	20724	20631.2	222.44	20557
X-n172-k51	**45607**	45734	45653.1	135.6568	45607
X-n176-k26	48407	48861	48768.2	391.318	47812
X-n181-k23	**25569**	25585	25576.3	272.5681	25569
X-n186-k15	24147	24217	24169.1	231.1251	24145
X-n190-k8	16989	17229	17034.5	744.603	16980
X-n195-k51	44264	44420	44342.4	169.9475	44225
X-n200-k36	58729	59478	58950	422.9769	58578
X-n204-k19	19611	19694	19668.7	196.3767	19565
X-n209-k16	30710	30837	30769.9	385.6882	30656
X-n214-k11	10914	11002	10947.7	307.0573	10856
X-n219-k73	**117595**	117613	117601.3	602.7441	117595
X-n223-k34	40580	40695	40647.6	260.4594	40437
X-n228-k23	25964	26296	26108.8	407.2765	25742
X-n233-k16	19276	19387	19335	243.8007	19230
X-n237-k14	27046	27119	27081.2	479.9041	27042
X-n242-k48	82974	83136	83033.1	491.2693	82751
X-n247-k50	37740	37983	37881.4	705.1144	37274
X-n251-k28	38854	38938	38892	428.5219	38684
X-n256-k16	**18880**	18919	18893.4	274.8695	18880
X-n261-k13	26660	26782	26727.3	567.4432	26558
X-n266-k58	75731	75948	75840.3	577.3482	75478
X-n270-k35	35358	35455	35413.3	291.0527	35291
X-n275-k28	21250	21325	21284	408.0024	21245
X-n280-k17	33627	33819	33707.8	478.6689	33503
X-n284-k15	20331	20530	20435.8	471.4462	20226
X-n289-k60	95597	95892	95703	808.6183	95151
X-n294-k50	47274	47450	47351.1	346.816	47167
X-n298-k31	34317	34577	34450.3	385.3478	34231
X-n303-k21	21845	21930	21895.3	491.7789	21744
X-n308-k13	25949	26363	26141.3	914.5245	25859
X-n313-k71	94403	94861	94569.9	784.8316	94044
X-n317-k53	**78355**	78391	78373	1467.7256	78355
X-n322-k28	29971	30110	30067.7	355.896	29866
X-n327-k20	27744	27850	27795.8	646.7566	27556
X-n331-k15	31129	31326	31221.5	892.0977	31103

Çizelge 7.27. MA_ILSxVND algoritmasının Uchoa et al. veri seti üzerinde 40000 iterasyon ile çalıştırılmasına ait ayrıntılı sonuçlar (2. Bölüm).

Problem	**En iyi**	**En kötü**	**Ortalama**	**Süre(sn.)**	**Bilinen En İyi**
X-n336-k84	139683	140052	139864.5	1009.5536	139197
X-n344-k43	42273	42498	42379.4	495.4953	42099
X-n351-k40	26101	26231	26150.2	606.7736	25946
X-n359-k29	51787	52077	51919.9	1220.1528	51509
X-n367-k17	22946	23038	22982.1	814.4443	22814
X-n376-k94	147716	147751	147735	1985.3274	147713
X-n384-k52	66320	66487	66399.9	866.4244	66081
X-n393-k38	38389	38597	38491	657.5398	38269
X-n401-k29	66457	66608	66544.3	1719.0698	66243
X-n411-k19	19807	20063	19986.3	1021.552	19718
X-n420-k130	108115	108194	108143	974.0619	107798
X-n429-k61	65727	66080	65813.9	854.7442	65501
X-n439-k37	36443	36500	36470.4	933.1618	36391
X-n449-k29	55645	56152	55821.5	1356.9663	55358
X-n459-k26	24341	24462	24393.8	1215.5296	24181
X-n469-k138	222722	223335	223127.3	2031.3897	221909
X-n480-k70	89725	90003	89887.8	1626.3361	89535
X-n491-k59	66925	67162	67021.5	1384.3358	66633
X-n502-k39	**69244**	69339	69301.8	3382.2903	69253
X-n513-k21	24386	24449	24414.3	968.6279	24201
X-n524-k153	155445	157121	156104.6	3374.4111	154594
X-n536-k96	95513	95765	95665.1	2256.2801	95122
X-n548-k50	86825	87000	86905.2	4189.7293	86710
X-n561-k42	42952	43221	43051.1	1191.8045	42756
X-n573-k30	50907	51058	51005.4	3835.0932	50780
X-n586-k159	191054	191643	191321.9	2818.2257	190543
X-n599-k92	109064	109400	109235.6	2518.206	108813
X-n613-k62	60054	60314	60182.4	1495.2056	59778
X-n627-k43	62517	62795	62653.8	3430.2732	62366
X-n641-k35	64386	64625	64500.4	3361.6846	63839
X-n655-k131	106810	106860	106831.2	7888.2149	106780
X-n670-k130	148612	151127	149709.9	5835.7654	146705
X-n685-k75	68805	69047	68879.2	1922.0472	68425
X-n701-k44	82558	82932	82746	4560.1209	82292
X-n716-k35	43726	44538	44160.9	4473.5672	43525
X-n733-k159	136637	136972	136845.4	2769.3644	136366
X-n749-k98	77925	78122	78021.8	3514.5327	77700
X-n766-k71	115583	116416	115921.1	5397.932	114683
X-n783-k48	73035	73411	73268.1	4398.8444	72727
X-n801-k40	73621	73876	73750.6	8373.3175	73587
X-n819-k171	159045	159598	159305	4966.7701	158611
X-n837-k142	194418	195045	194718.7	6456.2275	194266
X-n856-k95	**89041**	89319	89194.4	6032.1133	89060
X-n876-k59	99958	100396	100142.7	8253.6032	99715
X-n895-k37	54463	54914	54720.5	4062.1452	54172
X-n916-k207	330214	330759	330527.3	10257.2475	329836
X-n936-k151	135055	136765	135917.6	7259.3704	133105
X-n957-k87	85706	85877	85771.9	7752.2152	85672
X-n979-k58	119929	120704	120415.8	9191.4438	119194
X-n1001-k43	73422	73785	73612.5	6559.6398	72742
	OHY	**OHY**	**OHY**	**Ort. Süre**	
-	0.34	0.80	0.55	2359.79	-

7.2.4 Diğer algoritmalar ile karşılaştırma

Bu bölümde, geliştirilen MA_ILSxVND algoritmasının literatürdeki diğer önemli CVRP algoritmaları ile karşılaştırması yer almaktadır. CVRP literatüründeki genel yaklaşımlardan birisi, eniyileme algoritmasını ilgili veri setindeki her bir problem için 10 farklı çekirdek değeriyle çalıştırarak ortalama ve en iyi maliyet değerlerini bulmaktır. İkinci yaklaşımda ise veri setindeki problem örneklerinin bir kez çalıştırılmasının sonucu rapor edilmektedir.

Deneysel çalışmalarda yer verilmesi gereken bir diğer önemli bilgi de algoritmaların çalışma süreleridir. Süre ölçümleri, üzerinde çalışılan bilgisayarın işlem gücüne bağlı olarak değişebilmektedir. Bu nedenle, ölçülen süre değerlerinin yanında deneylerin gerçekleştirildiği bilgisayarın konfigürasyonunun belirtilmesi gerekmektedir. Bu bölümde yer alan karşılaştırmalarda sürelerin ölçeklendirilebilmesi için Dongarra (2014) çalışmasında yer verilen birim zamandaki kayan noktalı işlem gücü (flop/sn.) değerleri dikkate alınmıştır. Aynı aileye ait işlemciler için farklı hızlardaki modeller arasında frekans hızları ile doğru orantılı dönüşüm gerçekleştirilmiştir. Karşılaştırmada yer alan algoritmaların tahmini işlem gücünü hesaplarken kullanılan ve Dongarra (2014) içerisinde rapor edilen işlemci modellerine ait saniyedeki milyon flop (Mflop) işlem gücünü gösteren değerler aşağıdaki gibidir:

- Intel Core 2, 2.4 Ghz = 2426 MFlop
- Pentium IV, 3.0 GHz = 1573 MFlop
- AMD 854 Opteron, 2.8GHz = 1717 MFlop
- Intel Xeon 64 (dual), 3.6 GHz = 1779 MFlop

Yukarıdaki değerlere göre örnek bir dönüşüm ile bu tez çalışmasının gerçekleştirildiği işlemcinin (Intel Core i7 3.4 GHz) hesaplama gücü ((3.4/2.4)x2426) = 3436.83 MFlop şeklinde hesaplanabilmektedir. Literatür karşılaştırmalarında yer alan algoritmalara ait süreler, hesaplanan bu değere göre ölçeklendirilerek sunulmuş ve daha sağlıklı kıyaslama yapılabilmesi sağlanmıştır.

Kıyaslamalarda kullanılan algoritmalar hakkında gerekli bilgiler Çizelge 7.28'den görülebilmektedir. Çizelgedeki algoritmaların seçilmesinde kullanılan ölçütler şunlardır: (i) literatürde ilgili veri seti üzerindeki karşılaştırmalarda sıklıkla kullanılması, (ii) her bir problem örneği için ayrı ayrı maliyet değeri rapor edilmesi,

(iii) maliyet değerlerinin birçok çalıştırmanın ortalaması veya tek bir çalıştırma sonucu elde edilmiş olması.

Çizelge 7.28. Literatür karşılaştırmalarında yer alan algoritmalara ait bilgiler.

Ad	**Kaynak**	**Sonuçların elde edilmesi**	**İşlemci**	**Dil**	**Tahmini Mflop**
Bone Route	Tarantilis and Kiranoudis (2002)	1 kez çalıştırma	Pentium II 400 MHz	MS Visual C++ 6.0	209.73
GTS	Toth and Vigo (2003)	10 kez çalıştırmanın ortalaması	Pentium 200 MHz	-	104.87
MA	Prins (2004)	1 kez çalıştırma	Pentium III 1.0 GHz	Delphi 5	524.33
SEPAS	Tarantilis (2005)	1 kez çalıştırma	Pentium II 400 MHz	MS Visual C++	209.73
AGES best	Mester and Brysy (2007)	1 kez çalıştırma	Pentium IV 2800 Mhz	Visual Basic 6.0	1468.13
GRELS	Prins (2009)	1 kez çalıştırma	Pentium IV 2800 Mhz	Delphi 7	1468.13
NB	Nagata and Byrsy (2009)	10 kez çalıştırmanın ortalaması	Opteron 2.4 GHz	C++	1471.71
HGSADC	Vidal et al. (2012)	10 kez çalıştırmanın ortalaması	Opteron 2.4 GHz	-	1471.71
ILS-SP	Subramanian et al. (2013)	50 kez çalıştırmanın ortalaması	Xeon 3.07 Ghz (Uchoa et al. için)	-	1517.09
UHGS	Vidal et al. (2014)	50 kez çalıştırmanın ortalaması	Xeon 3.07 Ghz (Uchoa et al. için)	-	1517.09

MA_ILSxVND algoritmasının diğer algoritmalar ile CMT veri seti üzerinde karşılaştırılması Çizelge 7.29'da yer almaktadır. Çizelgede MA_ILSxVND için yer alan CVRP maliyet değerleri 10 farklı çalıştırma sonucunun ortalamasıdır. Algoritmaların rapor edilmiş olan süre değerleri, hesaplanan tahmini MFlop işlem güçleri oranında ölçeklenerek son satırda sunulmaktadır. Önerilen MA_ILSxVND algoritmasının OHY ve ortalama çalışma süresi performansına bakıldığında, literatürdeki rakipleriyle kıyaslanabilir seviyede olduğu görülmektedir.

Çizelge 7.29. MA_ILSxVND algoritmasının literatürde yer alan diğer önemli algoritmalar ile CMT veri seti üzerinde karşılaştırılması.

Problem	En iyi	Bone Route	GTS	MA	SEPAS	AGES Best	GRELS	NB	HGSADC	MA_ILSxVND (10000 iter.)	MA_ILSxVND (20000 iter.)
CMT1	524.61	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**
CMT2	835.26	**835.26**	838.6	**835.26**	**835.26**	**835.26**	**835.26**	835.61	**835.26**	836.34	835.49
CMT3	826.14	**826.14**	828.56	**826.14**	**826.14**	**826.14**	**826.14**	**826.14**	**826.14**	826.71	826.39
CMT4	1028.42	1030.88	1033.21	1031.63	1029.64	**1028.42**	1029.48	**1028.42**	**1028.42**	1030.41	1028.75
CMT5	1291.29	1314.11	1318.25	1300.23	1311.48	**1291.29**	1294.09	1291.84	1294.06	1303.43	1301.88
CMT6	555.43	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**
CMT7	909.68	**909.68**	920.72	912.3	**909.68**	**909.68**	**909.68**	910.41	**909.68**	912.89	912.89
CMT8	865.94	**865.94**	869.48	**865.94**	**865.94**	**865.94**	**865.94**	**865.94**	**865.94**	**865.94**	**865.94**
CMT9	1162.55	1163.19	1173.12	1164.25	1163.19	**1162.55**	**1162.55**	1162.56	**1162.55**	1169.56	1168.56
CMT10	1395.85	1408.82	1435.74	1420.2	1407.21	1401.12	1401.46	1398.3	1400.23	1408.26	1404.93
CMT11	1042.11	**1042.11**	1042.87	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**
CMT12	819.56	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**
CMT13	1541.14	1544.01	1545.51	1542.97	1544.01	**1541.14**	1545.43	1542.99	1543.07	1543.30	1543.06
CMT14	866.37	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**
OHY	-	0.23	0.64	0.24	0.20	0.03	0.07	0.03	0.05	0.24	0.18
Ort. Süre (sn.)	-	313.03	230.31	311.17	337.50	162.66	15.93	83.14	132.60	39.89	76.62
Mflop	-	209.73	104.87	524.33	209.73	1468.13	1468.13	1471.71	1471.71	3436.83	3436.83
Çarpan	-	0.06	0.03	0.15	0.06	0.43	0.43	0.43	0.43	1.00	1.00
Ölçekli	-	18.78	6.91	46.68	20.25	69.94	6.85	35.75	57.02	39.89	76.62

Çizelge 7.30. MA_ILSxVND algoritmasının literatürde yer alan diğer önemli algoritmalar ile Golden et al. veri seti üzerinde karşılaştırılması.

Problem	**En iyi**	**GTS**	**MA**	**SEPAS**	**AGES Best**	**GRELS**	**NB**	**HGSADC**	**MA_ILSxVND (20000 iter.)**	**MA_ILSxVND (40000 iter.)**
Golden1	5623.47	5736.15	5648.04	5676.97	5627.54	5644.52	5632.05	5627	5692.74	5687.94
Golden2	8404.61	8553.03	8459.73	8459.91	8447.92	8447.92	8840.25	8446.65	8480.41	8476.61
Golden3	11036.22	11402.75	**11036.22**	**11036.22**	**11036.22**	**11036.22**	**11036.22**	**11036.22**	11114.40	11108.43
Golden4	13590.00	14910.62	13728.8	13637.53	13624.52	13624.52	13618.55	13624.52	13734.25	13709.50
Golden5	6460.98	6697.53	**6460.98**	**6460.98**	**6460.98**	**6460.98**	**6440.98**	**6460.98**	**6460.98**	**6460.98**
Golden6	8400.33	8963.32	8412.9	8414.28	8412.88	8412.9	8413.41	8412.9	8450.00	8414.89
Golden7	10102.70	10547.44	10267.5	10216.5	10195.56	10195.59	10186.93	10157.63	10254.16	10245.91
Golden8	11635.30	12036.24	11865.4	11936.16	11663.55	11643.9	11691.54	11646.58	11840.92	11818.57
Golden9	579.71	593.35	596.89	585.43	583.39	586.23	581.46	581.79	584.56	583.12
Golden10	735.66	751.66	751.41	746.56	741.56	744.36	739.56	739.86	742.60	741.29
Golden11	912.03	936.04	939.74	923.17	918.45	922.4	916.27	916.44	920.45	919.30
Golden12	1101.50	1147.14	1152.88	1130.4	1107.19	1116.12	1108.21	1106.73	1113.96	1111.53
Golden13	857.19	868.8	877.71	865.01	859.11	862.32	858.42	859.64	862.81	861.22
Golden14	1080.55	1096.18	1089.93	1086.07	1081.31	1089.35	1080.84	1082.41	1086.27	1085.80
Golden15	1337.87	1369.44	1371.61	1353.91	1345.23	1352.39	1344.32	1343.52	1352.44	1351.35
Golden16	1611.56	1652.32	1650.94	1634.74	1622.69	1634.27	1622.26	1621.02	1627.57	1625.80
Golden17	707.76	711.07	717.09	708.74	707.79	708.85	707.78	708.09	709.24	708.20
Golden18	995.13	1016.83	1018.74	1006.9	998.73	1002.15	995.91	998.44	1008.58	1007.97
Golden19	1365.60	1400.96	1385.6	1371.01	1366.86	1371.67	1366.7	1367.83	1371.79	1370.81
Golden20	1817.59	1915.83	1845.55	1837.67	1820.09	1830.98	1821.65	1822.02	1835.48	1831.87
OHY	-	3.23	1.68	0.95	0.35	0.65	0.52	0.29	0.88	0.75
Ort. Süre (sn.)	-	1052.79	4014.24	2728.8	1461.22	436.209	2135.87	1711.5	454.24	684.95
Mflop	-	209.73	524.33	209.73	1468.13	1468.13	1471.71	1471.71	3436.83	3436.83
Çarpan	-	0.06	0.15	0.06	0.43	0.43	0.43	0.43	1.00	1.00
Ölçekli	-	63.17	602.14	163.73	628.32	187.57	918.42	735.95	454.24	684.95

MA_ILSxVND algoritmasının literatürdeki diğer önemli algoritmalar ile Golden et al. veri seti üzerinde karşılaştırılması Çizelge 7.30'da yer almaktadır. Çizelgede MA_ILSxVND için yer alan CVRP maliyet değerleri 10 farklı çalıştırma sonucunun ortalamasıdır. Algoritmaların rapor edilmiş olan süre değerleri, hesaplanan tahmini MFlop işlem güçleri oranında ölçeklenerek son satırda sunulmaktadır. MA_ILSxVND algoritmasının OHY ve ortalama çalışma süresi performansına bakıldığında, rakipleriyle kıyaslanabilir olduğu görülmektedir.

MA_ILSxVND algoritması ile diğer algoritmalar arasındaki son karşılaştırma Uchoa et al. veri seti üzerinde gerçekleştirilmiştir. Uchoa et al. veri seti, CMT ve Golden et al. veri setlerine kıyasla daha yeni olduğu için, karşılaştırmada kullanılabilecek sınırlı sayıda algoritma bulunmaktadır. Bu algoritmalardan olan ILS-SP ve UHGS için rapor edilmiş deneysel sonuçlar Uchoa et al.'dan (2017) alınmıştır.

Çizelge 7.31'de Uchoa et al. veri seti için MA_ILSxVND algoritması ile diğer iki algoritmanın karşılaştırması bulunmaktadır. Veri setinin büyük boyutlu olası nedeniyle, diğer algoritmaların tek tek problem bazında ürettiği sonuçlara yer verilmeyip elde edilen özet sonuçlar sunulmuştur. Çizelgedeki "Ort." ve "En iyi" sütunları her bir algoritma için sırasıyla birden fazla farklı çekirdek ile çalıştırma sonucu elde edilen ortalama ve en iyi hata yüzdelerini gösterirken, "Süre" sütunu ölçülen süreyi dakika cinsinden vermektedir. "Min.", "Maks.", "Ort." Ve "Med." satırları ise sırasıyla 100 adet problem örneğinden elde edilen sonuçlar için en küçük, en büyük, ortalama ve ortanca değerleri ifade etmektedir.

"Ort." sütunları incelendiğinde, tüm algoritmaların hata yüzde değerlerinin en küçüğü için %0.00 elde ettiği görülmektedir. Bu da, problem örneklerinden en az birinin çözümünde her seferinde bilinen en iyi sonucun elde edildiği anlamına gelmektedir. En büyük, ortalama ve ortanca hata yüzdeleri için UHGS algoritmasının daha küçük değerler ürettiği görülürken, ILS-SP ve MA_ILSxVND algoritmalarının görece daha yakın sonuçlar elde ettiği anlaşılmaktadır.

Çizelge 7.31. MA_ILSxVND algoritmasının diğer algoritmalar ile Uchoa et al. veri seti üzerinde karşılaştırılması.

	ILS-SP			**UHGS**			**MA_ILSxVND**		
	Ort.	En iyi	Süre (dk.)	Ort.	En iyi	Süre (dk.)	Ort.	En iyi	Süre (dk.)
Min.	% 0.00	% 0.00	0.06	% 0.00	% 0.00	0.63	% 0.00	% -0.02	2.23
Maks.	% 2.50	% 1.42	349.94	% 0.55	% 0.13	247.55	% 2.07	% 1.44	147.93
Ort.	% 0.52	% 0.25	31.65	% 0.19	% 0.01	43.61	% 0.55	% 0.34	30.41
Med.	% 0.38	% 0.10	7.80	% 0.20	% 0.00	9.88	% 0.48	% 0.29	15.97

"En iyi" sütunlarına bakıldığında, en küçük hata yüzdeleri bazında sadece önerilen MA_ILSxVND algoritmasının negatif değer elde ettiği görülmektedir. Bu durum, daha önce deneysel sonuçlarda bahesedildiği üzere iki adet problem örneği için bilinen en iyi sonucun iyileştirilmesi sonucu ortaya çıkmıştır. En büyük, ortalama ve ortanca hata yüzdeleri için UHGS en iyi sonucu vermekte olup, ILS-SP ve MS_ILSxVND yine birbirine yakın sonuçlar ortaya koymuştur.

Çalışma süreleri incelendiğinde ise MA_ILSxVND ile ILS-SP algoritmaları yaklaşık 30 dk.'lık ortalama çalışma zamanına sahipken UHGS algoritması yaklaşık 45 dk. ile bu algoritmalardan 1.5 kat daha uzun süre harcamıştır.

7.2.5 Sonuç ve değerlendirme

Bu bölümde, tez kapsamında CVRP probleminin çözümü için önerilmiş olan MA_ILSxVND algoritmasının 3 farklı veri seti üzerinde deneysel çalışması gerçekleştirilmiştir. Bu deneysel çalışmalar sonucunda, geliştirilen ILSxVND melez metasezgiselinе (i) kabul fonksiyonu için uyarlanabilir parametre kontrolü ve (ii) yeniden başlatma stratejisinin birlikte eklenmesinin istatistiksel olarak anlamlı düzeyde iyileştirme sağladığı ortaya konulmuştur. Her bir veri seti için, önerilen algoritmanın performansı ortalama hata yüzdesi ve çalışma süresi cinsinden hesaplanarak kaydedilmiştir.

MA_ILSxVND algoritması 40000 iterasyon ile çalıştırıldığında CMT, Golden et al. ve Uchoa et al. veri setleri için sırasıyla % 0.13, % 0.75 ve % 0.55'lik ortalama hata yüzdeleri elde edilmiştir. Bu sonuçlar, geliştirilen yöntemin önemli bir performans gösterdiği ve CVRP problemlerinin çözümünde başarı ile kullanılabileceğini göstermektedir.

Elde edilen sonuçlar veri seti bazında diğer başarılı algoritmalar ile karşılaştırılmıştır. Hem ortalama hata yüzdesi hem de çalışma süresi bakımından geliştirilen algoritmanın literatürdeki başarılı rakipleri ile kıyaslanabilir seviyede olduğu gösterilmiştir. Buna ek olarak, MA_ILSxVND algoritması iyi tanımlanmış ve sade algoritmik yapısı ile gerçekleştirim kolaylığı sağlamaktadır ve bu yönüyle birçok rakip algoritmanın önüne geçmektedir.

8. SONUÇ

Bu tezde, melez metasezgisel algoritmalara yönelik parametre kontrol uygulamaları konusunda biri sürekli eniyileme diğeri ise kombinasyonel eniyileme alanlarında olmak üzere iki adet yöntem önerilmiştir. Geliştirilen iki yöntemin, hem melezleştirme hem de parametre kontrol konularının farklı noktalarını olabildiğince kapsaması gözetilmiştir. Öyle ki, sürekli eniyileme alanında önerilen HAMS_ABCxDExPSO algoritması, melezleştirme bakımından yüksek seviyeli ve nöbetleşe çalışma kuralına bağlıdır. Buna karşılık, kombinasyonel eniyileme alanında önerilen MA_ILSxVND algoritması, alt seviyeli ve takım çalışmasını uygulayan bir melez yapıdadır. Parametre kontrol açısından yöntemler incelendiğinde ise HAMS_ABCxDExPSO yönteminde, kendisini oluşturan algoritmaların çalışma sırasını ifade eden kategorik yapıdaki bir parametre ile uğraşılmaktadır. Diğer yandan, MA_ILSxVND algoritmasında, nümerik bir parametre olan kabul fonksiyonu eşik değeri belirlenmeye çalışılmaktadır.

HAMS_ABCxDExPSO algoritması, sınır kısıtlamalı ve tek amaçlı sürekli eniyileme problemlerinin çözümüne yönelik tasarlanmıştır ve bünyesinde ABC, DE ve PSO olmak üzere üç farklı metasezgiseli barındırmaktadır. Bu melez yapıda tek seferde sadece seçilen metasezgisel çalışmaktadır. Seçim işlemi ise metasezgisellerin önceki seçimleri ve çalışmaları sonucu elde edilen başarı oranlarını kullanan UCB algoritması ile yapılmaktadır. Geliştirilen algoritma ile CEC veri seti üzerinde deneysel çalışmalar gerçekleştirilmiştir. Hem teker teker fonksiyon bazında, hem de genel olarak veri seti üzerinde istatistiksel testlere dayalı karşılaştırmalar gerçekleştirilmiştir. Elde edilen sonuçlar geliştirilen melez yöntemin kendisini oluşturan metasezgisellerin bireysel performanslarını geçtiğini göstermektedir. Literatürdeki diğer bazı önemli uyarlanabilir algoritmalar ile yapılan karşılaştırmalar da HAMS_ABCxDExPSO algoritmasının onlarla rekabet edebilir düzeyde olduğunu göstermektedir.

MA_ILSxVND algoritması CVRP'nin çözümüne yönelik tasarlanmıştır. ILS metasezgiselinin yerel arama görevi VND algoritmasına verilerek ILSxVND melez metasezgiseli oluşturulmuştur. VND ise kendi içerisinde 6 farklı alt yerel arama algoritmasını yönetmektedir. Bu melez yapıda kabul fonksiyonuna eşik parametresi konularak üretilen yeni çözümlerin hangi oranda kabul edileceği kontrol edilmektedir. Eşik değerinin büyüklüğü ise sabit olmayıp uyarlanabilir parametre kontrol yöntemi ile algoritmanın gidişatına göre belirlenmektedir. Tüm bunlardan başka, algoritmanın yerel eniyiye takıldığı durumlarda o noktadan kaçınması için

yeniden başlatma stratejisi getirilmiştir. Böylece, melez algoritma çok başlangıçlı bir özelliğe kavuşmuş olmaktadır. MA_ILSxVND'nin performansı CMT, Golden et al. ve Uchoa et al. olmak üzere 3 farklı veri seti üzerinde test edilmiştir. Bu amaçla, CVRP literatüründe performans kriteri olarak sıkça tercih edilen ortalama hata yüzdesi ve algoritma çalışma süreleri ölçülmüştür. Elde edilen sonuçlar, yalın ILSxVND melez algoritmasına uyarlanabilir parametre kontrolü ve çok başlangıçlı stratejinin eklenmesinin ayrı ayrı performans artışı sağladığını istatistiksel olarak göstermektedir. Ayrıca, CVRP alanında yer alan önemli algoritmalar ile yapılan kıyaslama MA_ILSxVND'nin onlarla rekabet edebilir düzeyde olduğunu ortaya koymaktadır.

Tez çalışmasında gerçekleştirimi yapılan metasezgiseller ve onların melez tasarımının, üzerinde çalışılan eniyileme problemi türüne göre değişebildiği görülmüştür. Bir kombinasyonel eniyileme problemi olan CVRP açısından bakıldığında, MA_ILSxVND algoritmasının tasarımında ILS metasezgiselinin yanında birçok alt yerel arama prosedürü ve komşuluk yapılarının bir araya getirilmesine ihtiyaç duyulmuştur. Alt yerel arama algortimalarının organizasyonu ise VND metasezgiseli ile sağlanmıştır. Bu nedenle oluşturulan melez algoritma daha alt seviye ve takım çalışması şeklindedir. Buna bağlı olarak, parametre kontrol uygulamaları da alt seviye parametreler üzerine yoğunlaşmak durumundadır.

Tez kapsamında çalışılan bir diğer problem türü olan sürekli eniyileme açısından bakıldığında ise, geliştirilen HAMS_ABCxDExPSO algoritmasının tasarımında doğrudan metasezgisellerin kullanıldığı, alt prosedürlerin yer almadığı görülmektedir. Bu problemde çözümlerin temsili ve onların üzerinde yapılan işlemler benzer olduğundan, her bir metasezgisel bağımsız olarak çalışıp elde ettiği çözümleri diğerine kolayca aktarabilmektedir. Bu durum, yüksek seviyeli ve nöbetleşe çalışan bir melez algoritmanın geliştirilmesine olanak sağlamaktadır. Ayrıca, parametre kontrol yönteminin doğrudan melez yapının işleyişini kontrol edecek şekilde, daha üst seviyeden uygulanabilmesini sağlanmış olmaktadır.

Elde edilen genel deneysel bulgular, metasezgisel algoritmaların melezleştirilmesinin önemini bir kez daha ortaya koymuştur. Buna ek olarak, parametre kontrol yöntemlerinin, melez metasezgisellerin performanslarını daha da artırabildiğini göstermektedir. Bu nedenle, gelecekteki çalışmalarda her iki yöntemin de kullanımının artarak devam edeceği öngörülmektedir.

KAYNAKLAR DİZİNİ

Ahuja, R. K., Orlin, J. B. and Sharma, D., 2000, Very large-scale neighborhood search, *International Transactions in Operational Research*, 7(4-5):301-317 pp.

Alabas-Uslu, C. and Dengiz, B., 2011, A self-adaptive local search algorithm for the classical vehicle routing problem, *Expert Systems with Applications*, *38*(7):8990-8998 pp.

Auer, P., Cesa-Bianchi, N. and Fischer, P., 2002, Finite-time analysis of the multiarmed bandit problem, *Machine learning*, 47(2-3): 235-256 pp.

Avci, M. and Topaloglu, S., 2015, An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries, *Computers & Industrial Engineering*, 83:15-29 pp.

Awad, N. H., Ali, M. Z., Liang, J. J., Qu, B. Y. and Suganthan, P. N., 2016, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization, Technical Report, NTU, Singapore.

Boussaid, I., Lepagnot, J. and Siarry, P., 2013, A survey on optimization metaheuristics, *Information Sciences*, 237:82-117 pp.

Braysy, O., Hasle, G. and Dullaert, W., 2004, A multi-start local search algorithm for the vehicle routing problem with time Windows, *European Journal of Operational Research*, 159(3):586-605 pp.

Christofides, N., Mingozzi, A., and Toth, P, 1979, The vehicle routing problem, N. Christofides and al., editors, In: Combinatorial Optimization, 339-369 pp.

Clarke, G.U. and Wright, J.W., 1964, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, 12(4):568-581 pp.

DaCosta, L., Fialho, A., Schoenauer, M. and Sebag, M., 2008, Adaptive operator selection with dynamic multi-armed bandits, *In Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 913-920 pp.

Dantzig, G.B. and Ramser, J.H., 1959, The truck dispatching problem, *International transactions in operational research*, 7(4-5):285-300 pp.

Dongarra, J., 2014, Performance of Various Computers Using Standard Linear Equations Software, University of Manchester, UK.

KAYNAKLAR DİZİNİ (devam)

Dorigo, M., 1992, Optimization, learning and natural algorithms, Ph. D. Thesis, Politecnico di Milano, Italy.

Dorigo, M., Di Caro, G. and Gambardella, L. M., 1999, Ant algorithms for discrete optimization, *Artificial life*, 5(2):137-172 pp.

Dueck, G. and Scheuer, T., 1990, Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing, *Journal of computational physics*, 90(1):161-175 pp.

Duhamel, C., Lacomme, P., Quilliot, A. and Toussaint, H., 2011, A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem, *Computers & Operations Research*, 38(3):617-640 pp.

Eiben, A., Michalewicz, Z., Schoenauer, M. and Smith, J., 2007, Parameter control in evolutionary algorithms, *Studies in Computational Intelligence*, 54:19-46 pp.

El-Abd, M., 2011, A hybrid ABC-SPSO algorithm for continuous function optimization, *2011 IEEE Symposium on Swarm Intelligence (SIS)*, 1-6 pp.

Feo, T. A. and Resende, M. G., 1989, A probabilistic heuristic for a computationally difficult set covering problem, *Operations research letters*, 8(2):67-71 pp.

Feo, T. A., & Resende, M. G., 1995, Greedy randomized adaptive search procedures, *Journal of global optimization*, 6(2):109-133 pp.

Fialho, A., 2010, Adaptive operator selection for optimization, Doctoral dissertation, Université Paris Sud-Paris XI.

Fialho, A., Schoenauer, M. and Sebag, M., 2010-a, Toward comparison-based adaptive operator selection, *In Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 767-774 pp.

Fialho, A., Ros, R., Schoenauer, M. and Sebag, M., 2010-b, Comparison-based adaptive strategy selection with bandits in differential evolution, *International Conference on Parallel Problem Solving from Nature*, 194-203 pp.

Glover, F., 1986, Future paths for integer programming and links to artificial intelligence, *Computers & operations research*, 13(5):533-549 pp.

KAYNAKLAR DİZİNİ (devam)

Gokalp, O. and Ugur, A., 2017, An order based hybrid metaheuristic algorithm for solving optimization problems, International Conference on Computer Science and Engineering (UBMK), IEEE, 604-609 pp.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I. M., 1998, The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In Fleet management and logistics, Springer US , 33-56 pp.

Gong, W., Fialho, Á., Cai, Z. and Li, H., 2011, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, *Information Sciences*, 181(24):5364-5386 pp.

Hansen, P. and Mladenovic, N. 2001, Variable neighborhood search: Principles and applications, *European journal of operational research*, 130(3):449-467 pp.

Hansen, N., Müller, S. D. and Koumoutsakos, P., 2003, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evolutionary computation*, 11(1):1-18 pp.

He, Q. and Wang, L., 2007, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, *Engineering applications of artificial intelligence*, 20(1):89-99 pp.

Hemmelmayr, V. C., Cordeau, J. F. and Crainic, T. G., 2012, An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics, *Computers & operations research*, 39(12):3215-3228 pp.

Holland, J. H., 1975, Adaptation In Natural And Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press, 183p.

Irnich, S., Funke, B. & Grünert, T, 2006, Sequential search and its application to vehicle-routing problems, *Computers & Operations Research*, 33(8):2405-2429.

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Karaboga, D. and Akay, B., 2009, A comparative study of artificial bee colony algorithm, *Applied mathematics and computation*, 214(1):108-132 pp.

KAYNAKLAR DİZİNİ (devam)

Kennedy, J. and Eberhart, R., 1995, Particle swarm optimization, *IEEE International Conference on Neural Networks 4*, 1942-1948 pp.

Kıran, M. S. and Gündüz, M., 2013, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, *Applied Soft Computing*, 13(4):2188-2203 pp.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P, 1983, Optimization by simulated annealing, *science*, 220(4598):671-680 pp.

Koza, C. R., 1992, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 819p.

Kuleshov, V. and Precup, D., 2014, Algorithms for multi-armed bandit problems, arXiv preprint arXiv:1402.6028.

Laporte G., Gendreau M., Potvin, J.Y. and Semet, F., 2000, Classical and modern heuristics for the vehicle routing problem, *Management science*, 6(1):80-91 pp.

Li, Y., Wang, Y. and Li, B., 2013, A hybrid artificial bee colony assisted differential evolution algorithm for optimal reactive power flow, *International Journal of Electrical Power & Energy Systems*, 52:25-33 pp.

Li, Z., Wang, W., Yan, Y. and Li, Z., 2015, PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems, *Expert Systems with Applications*, 42(22):8881-8895 pp.

Lopez-Ibanez, M., Dubois-Lacoste, J., Stützle, T. and Birattari, M, 2011, The irace package, iterated race for automatic algorithm configuration, Technical Report TR/IRIDIA/2011-004, IRIDIA, Universite Libre de Bruxelles, Belgium.

Lopez-Ibanez, M., Dubois-Lacoste, J., Caceres, L. P., Birattari, M. and Stützle, T, 2016, The irace package: Iterated racing for automatic algorithm configuration, *Operations Research Perspectives*, 3:43-58 pp.

Meignan, D., Koukam, A. and Creput, J. C., 2010, Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism, *Journal of Heuristics*, 16(6):859-879 pp.

KAYNAKLAR DİZİNİ (devam)

Mester, D. and Braysy, O., 2007, Active-guided evolution strategies for large-scale capacitated vehicle routing problems, Computers & Operations Research, 34(10):2964-2975 pp.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E., 1953, Equation of state calculations by fast computing machines, *The journal of chemical physics*, 21(6):1087-1092 pp.

Michallet, J., Prins, C., Amodeo, L., Yalaoui, F. and Vitry, G., 2014, Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services, *Computers & operations research*, 41:196-207 pp.

Mladenovic, N. and Hansen, P., 1997, Variable neighborhood search, *Computers & operations research*, 24(11):1097-1100 pp.

Nagata, Y. and Braysy, O., 2009, Edge assembly-based memetic algorithm for the capacitated vehicle routing problem, *Networks*, 54(4):205-215 pp.

Or, I., 1976, Traveling Salesman-type Combinatorial Problems And Their Relation To The Logistics Of Blood Banking, Ph.D. dissertation, Department of Indsutrial Engineering and Management Science, Northwestern University.

Penna, P. H. V., Subramanian, A. and Ochi, L. S., 2013, An iterated local search heuristic for the heterogeneous fleet vehicle routing problem, *Journal of Heuristics*, 19(2):201-232 pp.

Potvin, J. Y. and Rousseau, J. M., 1995, An exchange heuristic for routeing problems with time Windows, *Journal of the Operational Research Society*, 46(12):1433-1446 pp.

Prins, C., 2004, A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers & Operations Research*, 31(12):1985-2002 pp.

Prins, C., 2009, A GRASP x evolutionary local search hybrid for the vehicle routing problem, In Bio-inspired algorithms for the vehicle routing problem, Springer, Berlin, Heidelberg, 35-53 pp.

Qin, A. K., Huang, V. L. and Suganthan, P. N., 2009, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE transactions on Evolutionary Computation*, *13*(2):398-417 pp.

KAYNAKLAR DİZİNİ (devam)

Rechenberg, I., 1971, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, PhD thesis, TU Berlin.

Ropke, S. and Pisinger, D., 2006, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation science*, 40(4):455-472 pp.

Sayah, S. and Hamouda, A., 2013, A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems, *Applied Soft Computing*, 13(4):1608-1619 pp.

Schwefel, H. P., 1974, Numerische Optimierung von Computer-Modellen, PhD thesis.

Shaw, P., 1997, A new local search algorithm providing high quality solutions to vehicle routing problems, APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.

Storn, R. and Price, K., 1997, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, 11(4):341-359 pp.

Stützle, T., 1998, Local search algorithms for combinatorial problems, PhD Thesis, Darmstadt University of Technology, 204p.

Subramanian, A., Drummond, L. M. D. A., Bentes, C., Ochi, L. S. and Farias, R., 2010, A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery, *Computers & Operations Research*, 37(11):1899-1911 pp.

Subramanian, A., Uchoa, E. and Ochi, L. S., 2013, A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519-2531 pp.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J. Y., 1997, A tabu search heuristic for the vehicle routing problem with soft time Windows, Transportation science, 31(2):170-186 pp.

Talbi, E. G., 2002, A taxonomy of hybrid metaheuristics, *Journal of heuristics*, 8:(5):541-564 pp.

Tang, L. and Wang, X., 2006, Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem, *The International Journal of Advanced Manufacturing Technology*, 29(11-12):1246-1258 pp.

KAYNAKLAR DİZİNİ (devam)

Tarantilis, C. D. and Kiranoudis, C. T., 2002, BoneRoute: An adaptive memory-based method for effective fleet management, Annals of operations Research, 115(1-4):227-241 pp.

Tarantilis, C. D., 2005, Solving the vehicle routing problem with adaptive memory programming methodology, Computers & Operations Research, 32(9):2309-2327 pp.

Thierens, D., 2005, An adaptive pursuit strategy for allocating operator probabilities, In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 1539-1546 pp.

Toth, P. and Vigo, D., 2003, The granular tabu search and its application to the vehicle-routing problem, Informs Journal on computing, 15(4):333-346 pp.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T. and Subramanian, A., 2017, New benchmark instances for the capacitated vehicle routing problem, *European Journal of Operational Research*, 257(3):845-858 pp.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N. and Rei, W., 2012, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, *Operations Research*, 60(3):611-624 pp.

Vidal, T., Crainic, T. G., Gendreau, M. and Prins, C., 2014, A unified solution framework for multi-attribute vehicle routing problems, *European Journal of Operational Research*, 234(3):658-673 pp.

Wang, H., Wu, Z., Rahnamayan, S., Sun, H., Liu, Y. and Pan, J. S., 2014, Multi-strategy ensemble artificial bee colony algorithm, *Information Sciences*, 279:587-603 pp.

Wolpert, D. H. and Macready, W. G., 1997, No free lunch theorems for optimization, *IEEE transactions on evolutionary computation*, *1*(1):67-82 pp.

Yang, J., Li, W. T., Shi, X. W., Xin, L. and Yu, J. F., 2013, A hybrid ABC-DE algorithm and its application for time-modulated arrays pattern synthesis, *IEEE Transactions on Antennas and Propagation*, 61(11):5485-5495 pp.

Yavuz, G., Aydin, D. and Stützle, T., 2016, Self-adaptive search equation-based artificial bee colony algorithm on the CEC 2014 benchmark functions, *2016 IEEE Congress on Evolutionary Computation (CEC)*, 1173-1180 pp.

KAYNAKLAR DİZİNİ (devam)

Zambrano-Bigiarini, M., Clerc, M. and Rojas, R., 2013, Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements, In Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, 2337-2344 pp.

Zhang, J. and Sanderson, A. C., 2009, JADE: adaptive differential evolution with optional external archive, *IEEE Transactions on evolutionary computation*, 13(5):945-958 pp.

Zhang, W. J. and Xie, X. F., 2003, DEPSO: hybrid particle swarm with differential evolution operator, IEEE International Conference on Systems, Man and Cybernetics, 3816-3821 pp.

ÖZGEÇMİŞ

OSMAN GÖKALP (osman.gokalp@ege.edu.tr)

Kişisel Bilgiler:

Uyruğu: T.C. Doğum yeri / tarihi: Konya / 19.10.1987

Eğitim:

Doktora (2013 -): Ege Üniversitesi, Fen Bil. Ens., Bilgisayar Mühendisliği A.B.D.
Yüksek Lisans (2010-2012): Ege Üniversitesi, Fen Bil. Ens., Bilgisayar Mühendisliği A.B.D.
Lisans (2005–2010) : Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü.

Akademik Görevler

Araştırma Görevlisi (2014 –), Ege Üniv., Bilgisayar Mühendisliği Bölümü
Araştırma Görevlisi (2010 – 2012), Yaşar Üniv., Yazılım Mühendisliği Bölümü

Uluslararası bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler:

Gökalp Osman, Uğur Aybars (2017). An order based hybrid metaheuristic algorithm for solving optimization problems. 2017 International Conference on Computer Science and Engineering (UBMK).

Gokalp Osman, Ugur Aybars (2012). Improving performance of ACO algorithms using crossover mechanism based on mean of pheromone tables. 2012 International Symposium on Innovations in Intelligent Systems and Applications.

Ulusal bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler:

Gökalp Osman, Uğur Aybars, Bodur Sema (2017). CONTOPT-JS: Sürekli Eniyileme Problemleri için Metasezgisel Algoritmalar tabanlı bir JavaScript YazılımKütüphanesi. Akıllı Sistemlerde Yenilikler ve Uygulamalar Konferansı (ASYU).

Caner Ulutürk, Osman Gökalp, Aybars Uğur (2011). Android işletim sisteminde opengl ile grafik programları geliştirme. XVI. Türkiye'de İnternet Konferansı.

Projelerde yaptığı görevler:

Sürekli Eniyileme Problemlerinin Çözümü İçin Metasezgisel Yöntemler Tabanlı Bir Java Script Kütüphanesinin Geliştirilmesi, BAP projesi, Görevi: Araştırmacı, 13/05/2016 - 08/12/2017.

Uydu Görüntülerinde Alan Önceliği Tahminleme ve Sensör Yerleştirme Eniyileme Yöntemlerinin Geliştirilmesi, TÜBİTAK PROJESİ, Görevi: Bursiyer, 15/04/2014 - 15/04/2015.

Yaprakları Yapay Sinir Ağları ve Görüntü İşleme Yöntemleri İle Kalitelerine Göre Sınıflandıran Bir Yazılım Geliştirilmesi, BAP Projesi, Görevi: Araştırmacı, 15/04/2011 - 11/10/2012.