T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÇOK AMAÇLI OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN KURBAĞA SIÇRAMA VE GRİ KURT OPTİMİZASYONU ALGORİTMALARI TABANLI HİBRİT BİR YÖNTEMİN GELİŞTİRİLMESİ

Murat KARAKOYUN

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Nisan-2021
KONYA

TEZ KABUL VE ONAYI

Murat KARAKOYUN tarafından hazırlanan "Çok Amaçlı Optimizasyon Problemlerinin Çözümü için Kurbağa Sıçrama ve Gri Kurt Optimizasyonu Algoritmaları Tabanlı Hibrit Bir Yöntemin Geliştirilmesi" adlı tez çalışması 15/04/2021 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda DOKTORA TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Harun UĞUZ	……..
Danışman Prof. Dr. Halife KODAZ	……..
Üye Dr. Öğr. Üyesi Şaban GÜLCÜ	……..
Üye Dr. Öğr. Üyesi Sait Ali UYMAZ	……..
Üye Dr. Öğr. Üyesi Ahmet ÖZKIŞ	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Murat KARAKOYUN

Tarih:

ÖZET

DOKTORA TEZİ

ÇOK AMAÇLI OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN KURBAĞA SIÇRAMA VE GRİ KURT OPTİMİZASYONU ALGORİTMALARI TABANLI HİBRİT BİR YÖNTEMİN GELİŞTİRİLMESİ

Murat KARAKOYUN

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Halife KODAZ

2021, 126 Sayfa

Jüri
Prof. Dr. Harun UĞUZ
Prof. Dr. Halife KODAZ
Dr. Öğr. Üyesi Şaban GÜLCÜ
Dr. Öğr. Üyesi Sait Ali UYMAZ
Dr. Öğr. Üyesi Ahmet ÖZKIŞ

Gerçek dünya problemlerine bakıldığında çoğunun birden fazla hedefi gerçekleştirmeye yönelik olduğu görülmektedir. Gerçekleştirilmek istenen bu hedefler kimi zaman birbiri ile uyum içinde iken kimi zaman da birbiri ile çatışma halinde olabilmektedir. Amaçların birbiri ile olan bu ilişkilerine bağlı olarak çok amaçlı problemlerin çözülme zorlukları farklı olabilmektedir. Birbiri ile uyum içinde olan amaçlara sahip bir problem tek amaçlı bir probleme dönüştürülerek çözülebilmesine rağmen amaçları çatışan bir problem için bu durum söz konusu değildir. Etrafımıza baktığımızda karşılaştığımız problemlerin birçoğunun, amaçları birbiri ile çatışan çok amaçlı problemler olduğunu görebiliriz. Bu problemlerin çözümü için kullanılan birçok klasik yöntem mevcuttur. Klasik yöntemlerin çözüm geliştirme noktasında farklı sebeplerden dolayı eksik kalması araştırmacıları farklı yaklaşımlar geliştirmeye yöneltmiştir. Genellikle doğada sürü halinde yaşayan hayvanların veya farklı yaşam alanlarına sahip bitkilerin davranışlarından esinlenilerek geliştirilen doğa esinli algoritmalar bu yaklaşımlardan bir tanesi olmuştur. Doğa esinli algoritmalar, klasik yöntemler ile kıyaslandığında farklı problemlere uyarlanabilmeleri açısından daha avantajlı bir durumdadırlar.

Tez çalışması kapsamında, tek amaçlı problemlerin çözümü için geliştirilmiş olan kurbağa sıçrama (SFLA) ve gri kurt optimizasyonu (GWO) algoritmaları hibrit bir şekilde kullanılarak çok amaçlı optimizasyon problemlerine uygulanmıştır. Önerilen algoritmanın değerlendirilmesi üç aşamada farklı problem setleri üzerinde yapılmıştır. İlk aşamada özellikleri birbirinden farklı 36 kısıtsız çok amaçlı optimizasyon problemi kullanılmıştır. Önerilen algoritmanın performansı çok amaçlı optimizasyon algoritmalarından NSGA-II, IBEA, MOCell, MOEA/D, MOAAA ve MOVS algoritmalarının performansı ile kıyaslanmıştır. İkinci aşamada mühendislik tasarım problemleri ve kısıtlı problemlerden oluşan farklı özellikteki 10 adet çok amaçlı optimizasyon problemi kullanılmıştır. Önerilen algoritmanın performansı NSGA-II, IBEA, MOCell ve PAES algoritmaları ile kıyaslanmıştır. İlk iki aşamada performans karşılaştırma metriği olarak hiperküp (HV), terslenmiş nesilsel mesafe (IGD), yayılım (Spread) ve Epsilon metrikleri kullanılmıştır. Bu dört farklı metrik ile elde edilen sonuçlar Friedman ve Wilcoxon istatistiksel testleri ile analiz edilmiştir. Ayrıca algoritmalar tarafından elde edilen sonuçların görsel olarak sunulması için grafiksel çizimler ve kutu grafiği kullanılmıştır. Üçüncü ve son aşamada ise

önerilen algoritma, görüntü işleme çalışmalarında sıklıkla kullanılan 10 adet gri seviye görüntünün segmentasyonunda test edilmiştir. Üçüncü aşamada önerilen algoritmanın, SFLA ve GWO algoritmaları ile performans karşılaştırması yapılmıştır. Bu bölümde tek amaçlı bir problem olan eşikleme, önerilen algoritma kullanılarak çok amaçlı bir problem olarak ele alınmıştır. Görüntü segmentasyonu yapılan bu aşamada performans metriği olarak tepe sinyali gürültü oranı (PSNR) kullanılmıştır. Elde edilen deneysel sonuçlar Friedman ve Wilcoxon istatistik testleri ile analiz edilmiştir. Ayrıca elde edilen sonuç görüntüleri görsel olarak sunulmuştur. Üç aşamada elde edilen deneysel sonuçlara bakıldığında önerilen algoritmanın genel olarak karşılaştırma yapılan algoritmalardan daha başarılı olduğu görülmektedir.

Anahtar Kelimeler: Çok amaçlı optimizasyon, eşikleme, görüntü segmentasyonu, gri kurt algoritması, kurbağa sıçrama algoritması.

ABSTRACT

PhD THESIS

DEVELOPING A HYBRID METHOD BASED ON SHUFFLED FROG LEAPING AND GRAY WOLF OPTIMIZATION ALGORITHMS TO SOLVE MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

Murat KARAKOYUN

Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering

Advisor: Prof. Dr. Halife KODAZ

2021, 126 Pages

Jury
Prof. Dr. Harun UĞUZ
Prof. Dr. Halife KODAZ
Asst. Prof. Dr. Şaban GÜLCÜ
Asst. Prof. Dr. Sait Ali UYMAZ
Asst. Prof. Dr. Ahmet ÖZKIŞ

When looking to the real world problems, it is seen that many of them are aimed at achieving more than one goal. While these goals are sometimes in accordance with each other, they may be in conflict with each other. Depending on these relations of objectives with each other, the difficulties of solving multi-objective problems can be different. Although a problem with goals that are in accordance with each other can be solved by transforming it to a single-objective problem, this is not possible for a problem whose goals are in conflict. When looking around, it can be seen that many of the problems we encounter are multi-objective problems whose goals are in conflict with each other. There are many classical methods used to solve these problems. The deficiency of the classical methods at the point of developing solutions due to different reasons has led researchers to develop different approaches. Nature-inspired algorithms developed by taking inspiration from the behavior of animals that generally live with a swarm in nature or plants with different habitats have been one of these approaches. Nature-inspired algorithms are more advantageous than classical methods in terms of being adaptable to different problems.

In this thesis, shuffled frog leaping (SFLA) and gray wolf optimizer (GWO) algorithms developed for the solution of single-objective problems were used in a hybrid method and applied to multi-objective optimization problems. The proposed algorithm was evaluated in three stages on different problem sets. In the first stage, 36 unconstrained multi-objective optimization problems with different properties were used. The performance of the proposed algorithm was compared with the performance of the six multi-objective optimization algorithms (NSGA-II, IBEA, MOCell, MOEA/D, MOAAA and MOVS) in the first stage. In the second stage, 10 multi-objective optimization problems with different features, consisting of engineering design problems and constrained problems, were used. In this stage, the performance of the proposed algorithm was compared with NSGA-II, IBEA, MOCell and PAES algorithms. In the first two stages, hypervolume (HV), inverted generational distance (IGD), Spread and Epsilon metrics were used as performance comparison metrics. The results obtained with these four different metrics were analyzed by Friedman and Wilcoxon statistical tests. In addition, graphical drawings and box plots were used to visually present the results obtained by the algorithms. In the third and last stage, the proposed algorithm was tested in the segmentation of 10 gray level images, which are

frequently used in image processing studies. In the third stage, performance comparison was made with the SFLA and GWO algorithms, which are the components of the proposed algorithm. In this section, thresholding, which is a single-objective problem, is handled as a multi-objective problem by the proposed algorithm. Peak signal to noise ratio (PSNR) was used as a performance metric at this stage where image segmentation was made. Experimental results obtained were analyzed by Friedman and Wilcoxon statistical tests. In addition, the segmented images, which were generated by the algorithms, were presented visually. Considering the experimental results obtained in three stages, it is seen that the proposed algorithm is generally more successful than the compared algorithms.

Keywords: Gray wolf optimizer, image segmentation, multi-objective optimization, shuffled frog leaping algorithm, thresholding.

ÖNSÖZ

Tez süresi boyunca değerli bilgi ve fikirleri ile bana yön veren danışmanım Prof. Dr. Halife KODAZ'a ve yapıcı eleştirileri ile yol gösteren değerli tez izleme komitesi üyeleri Dr. Öğr. Üyesi Sait Ali UYMAZ'a ve Dr. Öğr. Üyesi Şaban GÜLCÜ'ye, tezin oluşması ve gelişmesi noktasında her türlü tecrübe ve bilgisi ile yardımcı olan Dr. Öğr. Üyesi Ahmet ÖZKIŞ'a teşekkürlerimi sunarım.

Rahmetli babamın eksikliğini hissettirmemek adına hayatını bize adayan değerli annem Raide KARAKOYUN'a başta olmak üzere; eğitim hayatım boyunca yanımda olan kardeşlerime teşekkürlerimi sunarım.

Tez çalışmam boyunca bana motivasyon kaynağı olan kızım Yağmur ve oğlum Yusuf'a; gösterdiği sabır ve hoşgörü için sevgili eşim Lütfiye ENGİN KARAKOYUN'a en kalbi şükranlarımı sunarım.

İlkokuldan sonra eğitim hayatımın sonlanmasından bir yıl sonra beni eğitim hayatına tekrar kazandıran ve bugünlere gelmeme vesile olan ustam Hacı Beşir BULUT'a özel bir teşekkürü borç bilirim.

Murat KARAKOYUN
KONYA-2021

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

Ω	:	Uygun çözümler kümesini kapsayan arama uzayı
$\prec$	:	Baskınlık (*dominance*)
N	:	Popülasyon boyutu
P	:	Popülasyon
Q	:	Yavru popülasyon
ε	:	Epsilon
α	:	GWO algoritması en iyi çözüme sahip birey (*alpha*)
β	:	GWO algoritması ikinci en iyi çözüme sahip birey (*beta*)
δ	:	GWO algoritması üçüncü en iyi çözüme sahip birey (*delta*)
ω	:	GWO algoritması lider takım dışındaki çözümler (*omega*)
$w_\alpha\ w_\beta, w_\delta$	:	GWO algoritması lider takımın etki katsayıları
D	:	Problem boyutu
$\boldsymbol{\Gamma}$	:	Standart Gama fonksiyonu
L	:	Görüntünün sahip olduğu gri seviye değeri
σ	:	Varyans
H	:	Entropi

Kısaltmalar

SFLA	:	Kurbağa Sıçrama Algoritması (*Shuffled Frog Leaping Algorithm*)
GWO	:	Gri Kurt Optimizasyonu (*Grey Wolf Optimizer*)
NSGA-II	:	Bastırılamayan Sıralamalı Genetik Algoritma-II (*Non-dominated Sorting Genetic Algorithm II*)
IBEA	:	İndikatör Temelli Evrimsel Algoritma (*Indicator-Based Evolutionary Algorithm*)
MOCell	:	Çok Amaçlı Hücresel Genetik Algoritma (*Multi Objective Cellular Genetic Algorithm*)
MOEA/D	:	Ayrıştırma Tabanlı Çok Amaçlı Evrimsel Algoritma (*Multi Objective Evolutionary Algorithm Based on Decomposition*)

MOAAA	:	Çok Amaçlı Yapay Alg Algoritması (*Multi Objective Artificial Algae Algorithm*)
MOVS	:	Çok Amaçlı Girdap Arama (*Multi Objective Vortex Search*)
PAES	:	Pareto Arşivlenmiş Evrim Stratejisi (*Pareto Archived Evolution Strategy*)
HV	:	Hiperküp (*Hypervolume*)
IGD	:	Terslenmiş nesilsel mesafe (*Inverted generational distance*)
PSNR	:	Tepe sinyali gürültü oranı (*Peak Signal to Noise Ratio*)
MOSG	:	Çok amaçlı karıştırılmış gri kurt algoritması (*MO: Multi-objective, S: SFLA, G: GWO*)
D-MOSG	:	Kesikli çok amaçlı karıştırılmış gri kurt algoritması (*Discrete-MOSG*)
PS	:	Pareto-optimal set (*Pareto set*)
PF	:	Pareto-optimal yüzey (*Pareto Front*)
PF_t	:	Gerçek Pareto-optimal yüzey (*Pareto front true*)
CD	:	Kalabalık mesafesi (*Crowding distance*)
QR	:	Kalite sıralaması (*Quality rank*)
LF	:	Levy uçuşu (*Levy flight*)

1. GİRİŞ

Optimizasyon, ele alınan problem için belirtilen kısıtlamaları sağlayacak şekilde en iyi çözümü üretme sürecidir. Bir problemin çözüme kavuşabilmesi öncelikli olarak matematiksel modelinin çıkarılmasına bağlıdır. Matematiksel model, bir problemin değişken, kısıt, amaç fonksiyonu gibi unsurlarının matematiksel olarak ifade edilmesidir. Bu matematiksel modelde hedef, amaç fonksiyonunu minimize veya maksimize eden değişken değerlerini, belirtilen kısıtlar çerçevesinde elde etmektir. Matematiksel modeli basit, değişken sayısı ve kısıtları az olan problemler için uygun çözümlerin elde edilmesi zor olmamaktadır. Ancak matematiksel model karmaşıklaştıkça ve karar değişken sayısı arttıkça problemin çözüme kavuşması farklı açılardan daha zor bir hale gelmektedir. Bu tür karmaşık problemlerde çözüm süresinin çok uzun olması ve uygun çözüme ulaşılamaması en önemli sorunların başında gelmektedir. Karşılaşılan bu sorunlar araştırmacıları farklı çözümler üretmeye teşvik etmiştir. Tarih boyunca insanoğlu için esin kaynağı olan doğa ve doğadaki canlılar bu noktada da ilham vermiştir. Birçok araştırmacı, doğa ve canlıların yaşamlarını model alarak farklı özellikteki algoritmalar geliştirmiştir. Literatüre bakıldığında Yapay Alg Algoritması (*Artificial Algae Algorithm, AAA*) (Uymaz ve ark., 2015a), GWO (Mirjalili ve ark., 2014), SFLA (Eusuff ve ark., 2006), Yapay Arı Kolonisi (*Artificial Bee Colony, ABC*) (Karaboga, 2005), Parçacık Sürü Optimizasyonu (*Particle Swarm Optimization, PSO*) (Kennedy ve Eberhart, 1995), Genetik Algoritma (*Genetic Algorithm, GA*) (Holland, 1992) gibi birçok doğa esinli algoritmanın geliştirildiği görülmektedir. Bununla beraber araştırmacılar, performansı arttırmak amacıyla bu algoritmaların çoğunu modifiye ederek farklı türevlerini elde etmişlerdir. Çalışmalar sonucunda doğa esinli algoritmaların optimizasyon problemleri üzerinde kısa sürelerde kabul edilebilir çözümler ürettiği görülmüştür.

Optimizasyon problemleri amaç fonksiyonları sayısı bakımından iki ana başlıkta toplanabilir. Şayet optimize edilecek problem bir tane amaç fonksiyonuna sahip ise tek amaçlı, birden fazla amaç fonksiyonuna sahip ise çok amaçlı optimizasyon problemi olarak adlandırılmaktadır. Tek amaçlı optimizasyon problemlerinde, problemin sahip olduğu tek amaç fonksiyonunun en uygun değerini veren karar değişkenlerine sahip tek çözüm elde edilmeye çalışılmaktadır. Çok amaçlı optimizasyon problemlerinde ise birden fazla amaç için uygun olan bir çözüm kümesi sunulmaktadır. Birden fazla amaç fonksiyonu olan çok amaçlı optimizasyon problemlerinde amaç fonksiyonları birbiri ile

uyum içinde olabilir veya çatışma durumunda olabilmektedir. Amaç fonksiyonları arasındaki bu ilişkiye göre problemin ele alınışı ve işlenişi farklılık gösterebilmektedir. Amaç fonksiyonlarının uyum içinde olduğu problemlerde, bir amaç için uygun bir çözüm sunan karar değişkenleri diğer amaç için de uygunluk göstermektedir. Bu tür problemler uygun yöntemler kullanılarak tek amaçlı bir problem haline getirilebilir ve tek amaçlı optimizasyon için geliştirilen algoritmalar ile çözüme kavuşturulabilir. Problem tek amaçlı olarak ele alındığından sonuç, tüm amaç fonksiyonları için uygun değere karşılık gelen karar değişkenlerine sahip bir çözüm olacaktır. Ancak gerçek dünya problemlerinin çoğu amaçları birbiri ile çakışan çok amaçlı problemlerdir ve bu problemlerin her bir amaç fonksiyonu için uygun değer veren tek bir çözüm söz konusu değildir. Bu tür problemlerde tek çözüm yerine bir çözüm kümesi sunulmaktadır. Bu kümedeki çözümler amaç fonksiyonları arasında bir denge sağlamayı hedeflemektedir. Uygun çözümlerden oluşması hedeflenen bu kümeyi elde edebilmek adına birçok yaklaşım ve algoritma önerilmiştir. Bunlardan bazıları direk olarak çok amaçlı problemlerin çözülmesi için geliştirilmişken bir kısmı da tek amaçlı optimizasyon yaklaşımlarının modifiye edilmesiyle elde edilmiştir. Her problem için başarılı tek bir algoritma olamaması gerçeği düşünüldüğünde, ne kadar çok sayıda geliştirilmiş algoritma olursa olsun, günümüzde halen yeni yaklaşımların önerilmesine ve geliştirilmesine şaşmamak gerekir.

Tez çalışması kapsamında, tek amaçlı problemlerin çözümü için geliştirilmiş olan SFLA ve GWO algoritmaları hibrit bir şekilde kullanılarak çok amaçlı optimizasyon problemlerine uygulanmıştır. Önerilen algoritma; *Multi-objective: MO, SFLA: S* ve *GWO: G* olmak üzere MOSG olarak adlandırılmıştır. Önerilen algoritmanın değerlendirilmesi üç aşamada farklı problem setleri üzerinde yapılmıştır. İlk aşamada özellikleri birbirinden farklı 36 kısıtsız çok amaçlı optimizasyon problemi kullanılmıştır. İkinci aşamada mühendislik tasarım problemleri ve kısıtlı problemlerden oluşan farklı özellikteki 10 adet çok amaçlı optimizasyon problemi kullanılmıştır. Üçüncü ve son aşamada ise önerilen algoritma, görüntü işleme çalışmalarında sıklıkla kullanılan 10 adet gri seviye görüntünün segmentasyonunda test edilmiştir. Önerilen algoritmanın performansı, ilk iki aşamada çok amaçlı optimizasyon algoritmalarından NSGA-II, IBEA, MOCell, MOEA/D, PAES, MOAAA ve MOVS algoritmalarının performansı ile kıyaslanmıştır. Bu iki aşamada performans karşılaştırma metriği olarak *hiperküp (HV)*, *ters nesilsel mesafe (IGD)*, *yayılım (Spread)* ve *Epsilon* metrikleri kullanılmıştır. Üçüncü aşamada ise kendi bileşenleri olan SFLA ve GWO algoritmaları

ile performans karşılaştırması yapılmıştır. Görüntü segmentasyonu yapılan bu aşamada performans metriği olarak *tepe sinyali gürültü oranı (PSNR)* kullanılmıştır. Elde edilen deneysel sonuçlar Friedman ve Wilcoxon istatistik testleri ile analiz edilmiştir. Ayrıca sonuçlar görsel grafikler ile sunulmuştur.

Bu tez çalışması 6 ana bölüm ve ekler kısmından oluşmaktadır ve aşağıda belirtildiği gibi organize edilmiştir:

Birinci bölüm olan Giriş başlığı altında optimizasyon tanımı ve çok amaçlı optimizasyon hakkında genel bilgilendirme yapılmıştır. Ayrıca tez çalışması kapsamında yapılanlar genel hatları ile sunulmuştur. İkinci bölümde çok amaçlı optimizasyon konusunun kronolojik olarak gelişim süreci ele alınmış ve bu konuda yapılan çalışmalar ile ilgili kaynak araştırması yapılmıştır. Üçüncü bölümde çok amaçlı optimizasyon ile ilgili detaylı bilgi verilmiş, tez kapsamında kullanılan çok amaçlı algoritmalar özetlenmiş, kullanılan problem setleri hakkında bilgi verilmiş ve algoritmaların performans karşılaştırmasında kullanılan metrikler detaylıca açıklanmıştır. Dördüncü bölümde önerilen algoritma ile ilgili detaylı bir bilgilendirme yapılmıştır. Beşinci bölümde problem setleri üzerinde elde edilen deneysel sonuçlar analiz edilmiştir. Altıncı ve son bölümde sonuçların özet olarak değerlendirilmesi ve gelecekte yapılabilecek çalışmaların önerisi yapılmıştır.

2. KAYNAK ARAŞTIRMASI

Gerçek dünya problemlerinin büyük bir çoğunluğunun çok amaçlı ve çözülmesi zor olan problemler olması sebebiyle geçmişten günümüze bu alanda çalışmalar süre gelmektedir. Tek amaçlı optimizasyon problemlerinde amaç uzayı tek boyutludur ve mevcut çözümler içerisinde sayısal bir karşılaştırma ile en uygun çözüm elde edilebilir. Ancak çok amaçlı optimizasyon problemlerinde, daha önce belirtildiği gibi tek çözüm yerine amaç fonksiyonları arasında denge kurmaya çalışan bir çözümler kümesi elde edilmektedir. Çok amaçlı optimizasyon alanında Edgeworth (1881) ve Pareto (1896) optimal çözüm için *Pareto optimum* olarak isimlendirilen ilk tanımı yapmışlardır (Edgeworth, 1881; Pareto, 1896). Pareto çözümleri kapsayan kümeye *Pareto Kümesi* ve bu çözümlerin amaç uzayında oluşturduğu şekle *Pareto yüzeyi* denmiştir (Altınöz, 2015). Bu kavramlar ile ilgili detaylı açıklama üçüncü bölümde verilmiştir.

Geçmiş yıllarda, çok amaçlı optimizasyon problemlerinin çözümü için ağırlıklı toplam, doğrusal programlama, hedef programlama gibi bazı matematiksel yaklaşımlar kullanılmıştır. Ancak bu yöntemlerin kullanılması farklı sebeplerden dolayı sorunlar oluşturmuştur. Bu sorunlar, klasik yöntemlerin kendi yapısından kaynaklandığı gibi uygulanan problemin sahip olduğu Pareto yüzeyinin şeklinden de oluşabilmekteydi. Bu şeklin iç bükey, dış bükey veya ayrık olması uygulanan algoritmanın performansını önemli ölçüde etkilemekteydi. Bu sorunların giderilmesi ve problem yapısına bağımlılığın ortadan kaldırılması amacıyla çalışmalar yapılsa da hesaplama maliyetinin oldukça artması sebebiyle yeni sorunlar ile karşılaşılmıştır (Guzmán ve ark., 2010; Yu ve ark., 2012; Altınöz, 2015; Erdoğmuş, 2016).

Bilim tarihi boyunca olduğu gibi bu alanda da araştırmacılar - klasik matematiksel yaklaşımların kullanılmasından kaynaklanan sorunlar nedeniyle - farklı arayışlara girmişlerdir. Rosenberg, 1967 yılında yayımladığı tezinde evrimsel algoritmaların çok amaçlı optimizasyon problemlerinde kullanılabileceğine değinmiştir (Rosenberg, 1967). Bu açıdan bakıldığında evrimsel algoritmalar ile doğa esinli yaklaşımların çok amaçlı optimizasyon problemlerine uygulanmasının önü açılmıştır. Rosenberg her ne kadar düşünceyi ortaya atsa da çalışmasında, problemleri çok amaçlı ele almak yerine tek amaçlı probleme dönüştürerek çözmeye çalışmıştır. Genetik algoritma tabanlı Vektörel Değerlendirmeli Genetik Algoritma (Vector Evaluated Genetic Algorithm, VEGA), problemleri ilk defa çok amaçlı olarak ele almıştır (Schaffer, 1984; 1985). Algoritma, amaç fonksiyonlarını optimize etme şekli

bakımından çok başarılı sonuçlar elde edememiştir. Popülasyon, alt gruplara ayrılarak her bir amaç fonksiyonu farklı grup tarafından optimize edilir. Bu aşamanın sonucunda elde edilen çözümler bir araya getirilir ve genetik algoritma stratejileri ile yeni çözümler üretilir. Bu yaklaşımla genellikle uç noktalardaki çözümler elde edilerek arada kalanlar gözden kaçmış olur. Sonuç olarak VEGA ürettiği çözümler açısından çok başarılı sayılmasa da bu konuda öncü olması sebebiyle önemli bir algoritmadır. Goldberg, VEGA'nın eksikliklerine vurgu yaparak bastırılamayan/baskılanamayan sıralama (*non-dominated ranking*), seçim (*selection*) ve yerleştirme (*niching*) gibi yaklaşımların çok amaçlı optimizasyon problemlerinin çözümünde kullanılmasına değinmiştir. Bu öneri ile popülasyonu oluşturan bireyler tarafından elde edilen aday çözümlerin birbirlerini bastırma durumuna göre derecelendirilmesi amaçlanmıştır. Bu yaklaşım Pareto optimal kavramının evrimsel algoritmalar aracılığıyla çok amaçlı optimizasyonda kullanılmasına katkı sağlamıştır (Golberg, 1989; Coello, 2006). Goldberg bu önerileri ile araştırmacılara esin kaynağı olmuş ve sonraki yıllarda yeni algoritmaların geliştirilmesine vesile olmuştur. Çok Amaçlı GA (*Multi-objective GA, MOGA*) (Fonseca ve Fleming, 1993), İçe Yerleştirilmiş Pareto GA (*Niched Pareto GA, NPGA*) (Horn ve ark., 1994) ve Bastırılamayan Sıralamalı GA (*Non-dominated Sorting GA, NSGA*) (Srinivas ve Deb, 1994) bu süreçte geliştirilmiş en bilinen algoritmalardandır. Birincil nesil algoritmalar olarak anılan bu algoritmalarda popülasyondaki iyi çözümlere sahip bireylerin tespit edilip muhafaza edilmesi sağlanmıştır. *Elitizm* denilen bu durumda bastırılamayan çözümler ayrı bir yapıda tutularak sonraki nesillere aktarımı garantilenmiş olur. *Elitizm* kavramının çok amaçlı optimizasyon algoritmalarında kullanılması yakınsama probleminin çözümüne katkı da sağlamıştır (Rudolph ve Agapie, 2000; Altınöz, 2015). Takip eden yıllarda *elitizm* kavramı, ikinci nesil olarak adlandırılan algoritmalarda da kullanılmaya devam etmiştir. İkinci nesil algoritmalar içinde yaygın olarak kullanılan algoritmalara; Kuvvet Pareto Evrimsel Algoritması (*Strength Pareto Evolutionary Algorithm, SPEA*) (Zitzler ve Thiele, 1999), Pareto Arşivlenmiş Evrimsel Stratejisi (*Pareto Archived Evolution Strategy, PAES*) (Knowles ve Corne, 1999), NSGA-II (Deb ve ark., 2000) ve SPEA-II (Zitzler ve ark., 2001) örnek olarak gösterilebilir.

Çok amaçlı optimizasyon alanında yaşanan bu gelişmeler araştırmacılara yeni ilhamlar vermiştir. Bu tez çalışmasında yapıldığı gibi son yıllarda araştırmacılar, tek amaçlı optimizasyon için önerilen doğa esinli algoritmaları çok amaçlı optimizasyon problemlerinin çözümüne uygulanacak şekilde düzenlemişlerdir. Bu düzenlemeler kimi

zaman modifikasyon şeklinde olurken kimi zaman da algoritmaların hibrit bir şekilde kullanılması ile olmuştur. Kaynak araştırmasının son bölümü doğa esinli algoritmaların çok amaçlı problemlere uygulanması ile elde edilen çalışmaların bir literatür taraması şeklinde verilmiştir.

Kennedy ve Eberhart, sürü halinde yaşayan kuş ve balıkların yiyecek arama davranışlarından yola çıkarak PSO algoritmasını geliştirmişlerdir (Kennedy ve Eberhart, 1995). PSO sürü zekâsına dayanan popüler ve başarılı bir algoritmadır. Algoritma tek amaçlı optimizasyon problemlerinin çözümü için önerilmiş olmasına rağmen çok amaçlı optimizasyon problemlerinin çözümünde kullanılmak üzere bir çok farklı özelliklerdeki çalışmalarda modifiye edilerek literatüre katkı sağlamıştır. PSO algoritmasının çok amaçlı optimizasyon problemleri için önerilen versiyonları arasında; MOPSO (Coello ve Lechuga, 2002; Coello ve ark., 2004), OMOPSO (Sierra ve Coello, 2005), CLPSO (Liang ve ark., 2006), TV-MOPSO (Tripathi ve ark., 2007), NSPSO (Liu, 2008), MOPSO/D (Peng ve Zhang, 2008), ClustMPSO (Janson ve ark., 2008), SMPSO (Nebro ve ark., 2009a), MPSO/D (Dai ve ark., 2015), MOVPSO (Meza ve ark., 2017), SMPSO-MM (Liang ve ark., 2018) çalışmaları örnek olarak gösterilebilir.

Karınca Kolonisi Optimizasyonu (*Ant Colony Optimization, ACO*) (Dorigo ve ark., 1999) algoritması Dorigo tarafından karınca kolonilerinin yiyecek kaynaklarını arama davranışları modellenerek önerilmiş başarılı bir doğa esinli algoritmadır. Kolonideki karınca sürüsü, yiyecek ile yaşam alanları arasında *feromon* denilen bir madde salgılayarak hem en iyi yolu bulma hem de kolonideki diğer karıncalara bilgi verme amacını gütmektedir. ACO algoritması karıncaların bu özelliğinden esinlenilerek matematiksel bir model ile önerilmiştir (Dikmen ve ark., 2014). Literatüre bakıldığında ACO algoritmasının farklı çalışmalarda çok amaçlı optimizasyon problemlerine uyarlandığı görülmektedir. Chaharsooghi ve Kermani yaptıkları çalışmalarında ACO algoritmasını modifiye ederek çok amaçlı kaynak tahsis probleminin çözümü için kullanmışlardır (Chaharsooghi ve Kermani, 2008). Yagmahan ve Yenisey, *NP-hard* tip bir problem olan akış çizelgeleme problemine çözüm getirmek amacıyla ACO algoritması ve bir yerel arama stratejisini birlikte kullanarak MOACSA olarak isimlendirdikleri yaklaşımlarını önermişlerdir (Yagmahan ve Yenisey, 2010). Liu ve Liu çalışmalarında yeni bir feromon güncelleme stratejisi önererek çok amaçlı ACO algoritmasını tesis yerleştirme problemine uygulamışlardır (Liu ve Liu, 2019).

Karaboğa tarafından 2005 yılında önerilen Yapay Arı Kolonisi (*Artificial Bee Colony, ABC*) algoritması sürü şeklinde yaşayan arıların hareketlerini matematiksel

olarak modellemiştir (Karaboga, 2005). ABC, sürüdeki arıların hiyerarşik-sosyal yaşantısı ve besin ararken takip ettikleri stratejileri üzerine kurulmuş başarılı bir algoritmadır. Algoritma önerildikten sonra kısa bir süre içerisinde araştırmacıların dikkatlerini çekmiş ve farklı problemlere uyarlanmıştır. Algoritma tek amaçlı optimizasyon problemlerinde genel olarak başarılı sonuçlar elde etmektedir. Bu nedenle araştırmacılar algoritmayı çok amaçlı optimizasyon problemlerinin çözümünde kullanmak üzere farklı versiyonlar ile geliştirmeye devam etmişlerdir. Wang ve arkadaşları, iş çizelgeleme problemine çözüm bulmak için geliştirilmiş Pareto tabanlı çok amaçlı EPABC algoritmasını önermişlerdir (Wang ve ark., 2012). Çalışmalarında işin tamamlanma süresini, kritik noktalardaki makine ve toplam makine iş yükünü minimize etmeyi hedeflemişlerdir. Simülasyon ve deneysel sonuçlarına dayanarak önerdikleri algoritmanın etkili sonuçlar elde ettiğini ve karşılaştırma yaptıkları algoritmalardan daha iyi olduklarını belirtmişlerdir. Akbari ve arkadaşları, yaptıkları çalışmalarında ızgara tabanlı bir yaklaşım kullanarak MOABC algoritmasını geliştirmişlerdir (Akbari ve ark., 2012). Benchmark problemleri üzerinde yaptıkları çalışmalar neticesinde elde ettikleri sonuçlara göre önerdikleri algoritmanın karşılaştırma yaptıkları algoritmalardan daha rekabetçi olduğunu belirtmişlerdir. Khorsandi ve arkadaşları, optimal güç akışı (*optimal power flow, OPF*) problemini çözmek amacıyla bulanık mantık tabanlı MABC algoritmasını geliştirmişlerdir (Khorsandi ve ark., 2013). Önerdikleri algoritmalarını IEEE 30 ve IEEE 118 veri yolu test sistemlerinde uygulamışlardır. Elde edilen sonuçların tatminkâr düzeyde olduğunu belirtmişlerdir. Bunlara ek olarak; DABC (Li ve ark., 2014), dMOABC (Zhong ve ark., 2014), EMOABC (Huo ve ark., 2015), MOABC (Hancer ve ark., 2015), DMCMOABC (Xiang ve Zhou, 2015), NSABC (Kishor ve ark., 2016), MOHABC (Zhou ve Yao, 2017), ABC-DP (Ding ve ark., 2017), AMOABC (Ning ve ark., 2018), MHABCP (Arslan ve Ozturk, 2019) çalışmaları ABC algoritmasının farklı versiyonları olarak literatüre katkı sağlamış algoritmalar arasında gösterilebilir.

Sürü halinde yaşayan kurbağaların besine ulaşma ve tehlikeden kaçma gibi davranışlarını matematiksel olarak modellemek isteyen Eusuff ve arkadaşları, 2006 yılında SFLA'yı geliştirmişlerdir (Eusuff ve ark., 2006). SFLA algoritması sürüdeki kurbağaların gruplar şeklinde avlanması ve besin ile ilgili bilgilerini birbiri ile paylaşması esasına dayanmaktadır. Algoritmada temel amaç az minimum hareketle maksimum besine ulaşmaktır (Karakoyun, 2015). Esasında tek amaçlı optimizasyon problemleri için geliştirilmiş olan algoritma araştırmacılar tarafından çok amaçlı

optimizasyon alanındaki farklı problemlere de uyarlanmış ve uygulanmıştır. Li ve arkadaşları, esnek iş çizelgeleme problemine çözüm getirmek amacıyla SFLA'nın çok amaçlı hibrit bir versiyonunu önermişlerdir (Li ve ark., 2012). HSFLA olarak adlandırdıkları yaklaşımlarında ilk popülasyonun üretilmesi aşaması, memetik evrim ve yerel arama süreçlerinin her birinde farklı stratejilerden faydalanarak etkin ve verimli sonuçlara ulaştıklarını belirtmişlerdir. Luo ve arkadaşları, çok amaçlı hibrit bir SFLA yaklaşımı ile araç rotalama problemi üzerinde çalışmış ve etkili sonuçlar elde ettiklerini belirtmişlerdir (Luo ve ark., 2015). Ali ve Hasanien yaptıkları çalışmalarında SFLA'yı kullanarak bir enine akı doğrusal motorun çok amaçlı tasarımını gerçekleştirmeyi hedeflemişlerdir (Ali ve Hasanien, 2016). Amaç fonksiyonlarını; motor ağırlığının ve kilit kuvvetinin minimize edilmesi ve itme kuvvetinin maksimize edilmesi olarak belirlemişlerdir. GA ve PSO algoritmalarının performansları ile yaptıkları karşılaştırmalarda tatminkâr sonuçlar elde ettiklerini belirtmişlerdir. SFLA'nın çok amaçlı optimizasyon alanında yapılan diğer literatür çalışmalarına; MMSFLA (Niknam ve ark., 2011), MOSFLA-MRPP (Hidalgo-Paniagua ve ark., 2015), MODE-CSFLA (Fang ve ark., 2018), ε-MaOSFLA (Na ve ark., 2020) örnek olarak gösterilebilir.

2014 yılında Mirjalili ve arkadaşları tarafından geliştirilen GWO algoritması, gri kurtların sosyal hiyerarşi ve avlanma davranışlarını model almaktadır (Mirjalili ve ark., 2014). Algoritma, sürüdeki en güçlü üç bireyin (alfa, beta ve delta) sürünün geri kalanını yönetmesi ve yönlendirmesi üzerine tasarlanmıştır. Tek amaçlı optimizasyon problemlerinde genel olarak başarılı sonuçlar elde eden algoritmanın çok amaçlı optimizasyon problemlerine uyarlanması hızlı bir süreçle gerçekleşmiştir. Mirjalili ve arkadaşları, GWO algoritmasını ilk defa çok amaçlı optimizasyon problemlerinin çözümü için uyarlayarak benchmark problemlerine uygulamışlardır (Mirjalili ve ark., 2016). MOEA/D ve MOPSO algoritmalarının performansı ile kıyasladıkları sonuçlarının oldukça etkin ve başarılı olduğunu belirtmişlerdir. Dilip ve arkadaşları, güç sistemlerinde ele alınan oldukça zor bir problem olan güç akışını optimize etmek amacıyla çok amaçlı GWO algoritmasını kullanmışlardır (Dilip ve ark., 2018). Aktif güç kaybı ve yakıt maliyetinin minimizasyonunu hedefledikleri çalışmalarını *IEEE-30* veri yolu test sistemi üzerinde gerçekleştirmiş ve başarılı sonuçlar elde ettiklerini belirtmişlerdir. Zapotecas-Martinez ve arkadaşları, ayrıştırma tabanlı çok amaçlı bir GWO yaklaşımı önererek bazı çok amaçlı benchmark problemleri ve mühendislik problemleri üzerine çalışmışlardır (Zapotecas-Martinez ve ark., 2019). Karşılaştırma yaptıkları diğer algoritmalara kıyasla oldukça başarılı sonuçlar elde ettiklerini

belirtmişlerdir. GWO algoritmasının çok amaçlı optimizasyon alanında yapılan çalışmalarına ek olarak; MODGWO (Lu ve ark., 2016), HMOGWO (Lu ve ark., 2017), 2ArchMGWO (Nuaekaew ve ark., 2017), MOCGWO (Lu ve ark., 2019), EMOGWO (Zhu ve Zhou, 2020), BMOGWO-S (Al-Tashi ve ark., 2020) gibi çalışmalar örnek olarak gösterilebilir.

Belirtilen çalışmalar dışında literatürde, tek amaçlı optimizasyon için geliştirilmiş olup sonradan çok amaçlı optimizasyon problemlerine uyarlanmış doğa esinli algoritmaların sayısız örneğini görmek mümkündür. Bu çalışmaların tek tek ele alınıp incelenmesi çok zor ve uzun bir süreç gerektirmektedir. Bu nedenle yukarıda belirtilen çalışmalar dışında son yıllarda literatüre kazandırılmış çalışmalardan bazı örnekler eklenerek kaynak araştırması kısmı sonlandırılacaktır. Yapay alg algoritmasının çok amaçlı optimizasyon alanındaki çalışmalarına MOAAA (Babalik ve ark., 2018) ve MO-AAA (Tawhid ve Savsani, 2018) örnek olarak verilebilir. Karınca Aslanı Optimizasyonu (*Ant Lion Optimizer, ALO*) algoritmasının çok amaçlı versiyonu (Mirjalili ve ark., 2017) çalışmasında sunulmuştur. Kumawat ve arkadaşları, Balina Optimizasyonu Algoritmasının (*Whale Optimization Algorithm, WOA*) çok amaçlı versiyonunu (Kumawat ve ark., 2017), Du ve arkadaşları, çok amaçlı Şahin Optimizasyonu (*Harris Hawks Optimization, HHO*) algoritmasını (Du ve ark., 2020) literatüre kazandırmışlardır.

3. MATERYAL VE YÖNTEM

3.1. Çok Amaçlı Optimizasyon

Bu bölümde çok amaçlı optimizasyonun genel hatları ve matematiksel modellemesi ele alınmıştır. Bununla beraber çok amaçlı optimizasyonda kullanılan temel kavramlardan bahsedilmiştir.

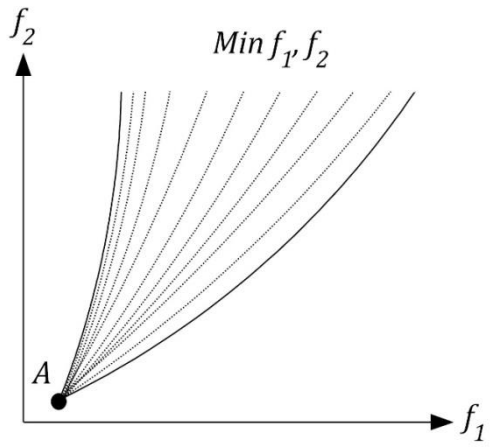
3.1.1. Çok amaçlı optimizasyon problemlerine genel bakış

Çok amaçlı optimizasyon problemlerinde birden fazla amacı birlikte optimize eden karar değişkenlerine ulaşılmak istenmektedir (Osyczka, 1985). Çok amaçlı bir problemin matematiksel olarak ifade edilmesi Denklem 3.1 ile gösterilmiştir.

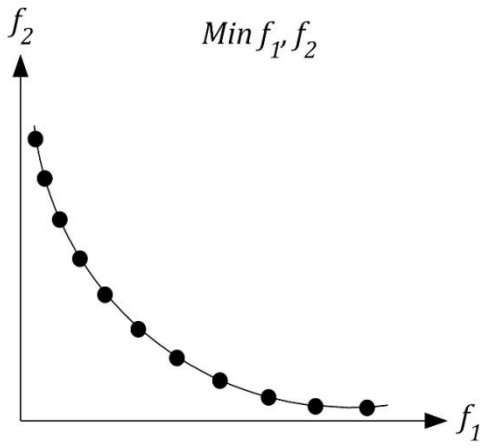
$$
\begin{aligned}
&(min/max)\ y = f(x) = \{f_1(x), f_2(x),..., f_M(x)\} \\
&g(x) \leq \{g_1(x), g_2(x),..., g_i(x)\} \leq 0 \\
&h(x) = \{h_1(x), h_2(x),..., h_j(x)\} = 0 \\
&X = [x_1, x_2, ..., x_B]^T \\
&A_b \leq x_b \leq U_b,\ b = 1,2, ..., B
\end{aligned}
\tag{3.1}
$$

Denklem 3.1'de verilen y=*f(x)* dizisi *M*-amaçlı bir vektör, *m*=1,2, ..., *M* olmak üzere $f_m(x)$ minimize/maksimize edilmek istenen *m.* amaç fonksiyonu, *g(x)* ve *h(x)* sırasıyla sağlanması gereken eşitsizlik ve eşitlik kısıtları dizisi, *X* karar değişkeni vektörü ve A_b ile U_b sırasıyla (*b.*karar değişkeni için) alt ve üst sınırları temsil etmektedir.

Tez çalışmasının önceki bölümlerinde belirtildiği üzere çok amaçlı bir optimizasyon probleminde amaçların birbirine göre iki farklı durumda olduğu bilinmektedir. Amaç fonksiyonlarının birbiri ile çatışma durumunda olmaması problemin tek amaçlı optimizasyon problemine dönüştürülerek ele alınmasına imkan sağlar (Şekil 3.1). Öte yandan amaçlar birbiri ile çekişme halinde ise tüm amaçların optimum değerini veren tek çözümden bahsedilemez (Şekil 3.2). Bu tür problemlerde birbirine üstünlük sağlayamayan çözümlerden oluşan bir çözüm kümesi elde edilir.



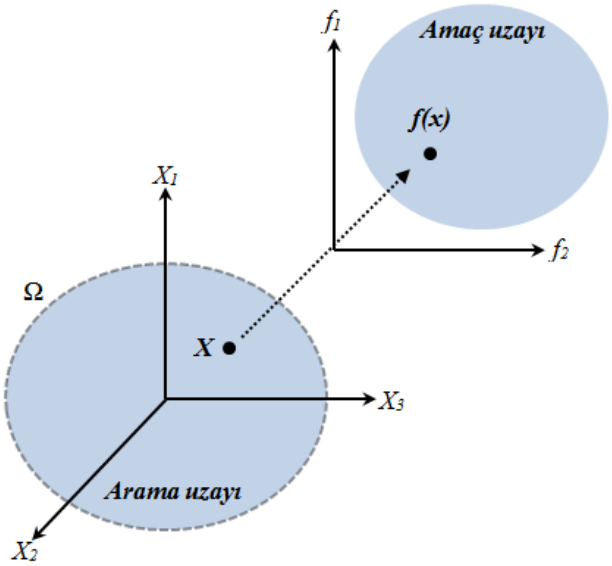
Şekil 3.1. Amaç fonksiyonları çatışma halinde değil



Şekil 3.2. Amaç fonksiyonları çatışma halinde

3.1.2. Karar değişkenleri, amaç fonksiyonları ve kısıtlar

Tek amaçlı optimizasyon problemlerinde olduğu gibi çok amaçlı problemlerde de üzerinde çalışılan problemin bir boyutu mevcuttur. Denklem 3.1'de gösterildiği gibi *X*, üzerinde çalışılan problemin alternatif bir çözümünü temsil etmektedir. Çok amaçlı optimizasyon problemlerinde karar değişkenlerinin uygun çözümler aradığı alana karar uzayı veya arama uzayı denmektedir. Söz konusu problem için arama uzayındaki uygun çözümler kümesine Ω dersek $X \epsilon \Omega$ olmak zorundadır. X_b, alternatif *X* çözümünün *b.*karar değişkeni olmak üzere A_b ve U_b değerleri bu değişkenin alabileceği alt ve üst sınır değerlerini temsil etmektedir. Üzerinde çalışılan problemin yapısı, karar değişkenlerinin veri tipini belirlemektedir. Karar değişkenlerinin arama uzayında elde ettiği değerler amaç uzayındaki amaç fonksiyonlarının kalitesini belirlemektedir. Çok amaçlı optimizasyon problemlerinin çoğunda amaçlar gerçekleştirilirken bazı kısıtlara dikkat edilmesi istenmektedir. Bu tür problemlerde elde edilen çözümün uygun çözümler arasında yer alması belirlenen kısıtlara uyulması ile mümkündür. Şekil 3.3'te karar değişkenlerinin arama uzayında elde ettiği uygun bir çözüm ve bu çözümün amaç uzayındaki karşılığı verilmiştir.



Şekil 3.3. Arama uzayında bir çözüm ve amaç uzayındaki karşılığı

3.1.3. Pareto teoremi ve baskınlık kavramı

Tek amaçlı optimizasyon problemlerinde bilindiği üzere algoritmalar bir amacın en iyi sağlandığı durumu elde etmeye çalışmaktadır. Elde edilen iki aday çözümün kalite kıyaslanması karar değişkenlerine bağlı olarak elde edilen amaç fonksiyonlarının değerleri karşılaştırıldığında rahatlıkla yapılabilmektedir. Bu şekilde minimizasyon problemlerinde amaç fonksiyonu değeri daha düşük olan aday çözümün daha iyi olduğu veya maksimizasyon problemleri için tam tersi söylenebilmektedir. Ancak durum çok amaçlı optimizasyon problemleri için daha karışık bir hal almaktadır. Birden fazla amacı olan ve genellikle amaçları birbiri ile çatışan çok amaçlı optimizasyon problemlerinde elde edilen aday çözümlerin kalite açısından kıyaslanması tek amaçlı problemlere göre daha zor olmaktadır. Çok amaçlı optimizasyondaki bu karmaşıklığı gidermek adına Goldberg, Pareto teoreminin bu problemlere uygulanabilirliğini dile getirmiştir (Golberg, 1989).

Kökeni ve çıkış noktası ekonomik refah üzerine olan Pareto teoremine göre: Toplumdaki bazı bireylerin yaşam kalitesini daha iyi hale getirip bazı bireylerin durumunu daha kötü hale getiriyorsa yapılan değişim/değişimlerin refahı ne yönde (olumlu veya olumsuz) etkilediği net bir şekilde söylenemez. Refahın artması, hiçbir bireyin durumunu kötü yapmadan bazı bireylerin durumunu iyi hale getiren değişimlerin yapılması ile söz konusu olabilir. Ayrıca hiçbir bireyin durumu

kötüleşmeden bir bireyin dahi durumunu iyileştirme olanağı yok ise bu toplumda maksimum refah seviyesine ulaşılmıştır (Sağ, 2008; Özkış, 2017).

Bu aşamadan sonra bu teoremin temel alındığı yeni yaklaşımlar geliştirilmiştir. Ayrıca çoğu tek amaçlı optimizasyon algoritmasının çok amaçlı problemlere uyarlanması da bu teoreme dayanmaktadır. Çok amaçlı algoritmalar elde ettikleri aday çözümlerin kalite kıyasını yapmak için Pareto teoremine başvurmaktadır. Pareto teoreminin çok amaçlı optimizasyon algoritmalarına uygulanışı dört temel kurala dayandırılmıştır (Coello ve ark., 2007; Deb, 2011; Özkış ve Babalık, 2017). Ω arama uzayı, $X\ (x_1, x_2, ..., x_B)$ ve $Y\ (y_1, y_2, ..., y_B)$ arama uzayında birer uygun aday çözüm ($X, Y \epsilon \Omega$) olmak üzere:

i) *Baskınlık (dominance)*: X aday çözümü amaç fonksiyonlarının hiçbirinde Y aday çözümünden daha kötü olmadan eğer en az bir amaç fonksiyonu için Y aday çözümünden daha iyi bir değere sahip ise X aday çözümü Y aday çözümünü baskılar (bastırır veya domine eder) denilmektedir. Bu durum $X \prec Y$ şeklinde gösterilmektedir. M amaç fonksiyonu sayısı olmak üzere Denklem 3.2 baskınlık durumunun matematiksel ifadesini vermektedir.

$$Eğer \begin{matrix} \forall i & \in & \{1,2,\dots,M\} f_i(X) \leq f_i(Y) \\ \wedge \exists j & \in & \{1,2,\dots,M\} f_j(X) < f_j(Y)' \end{matrix} \quad X \prec Y \tag{3.2}$$

Aday çözümlerin her zaman birbirini bastırması beklenilemez. X ve Y aday çözümleri amaç fonksiyonlarının en az birer tanesi için birbirlerine üstünlük sağlamışlarsa birbirini bastıramama (*non-dominated*) durumu söz konusudur. Bu durumda bu iki aday çözüm birbirini bastıramamış denmektedir.

ii) *Pareto optimal*: X aday çözümü arama uzayındaki hiçbir çözüm tarafından bastırılamıyorsa bu çözüme Pareto optimal çözüm denmektedir. Bu durum matematiksel olarak: $Eğer \neg \exists Z \in \Omega: Z \prec X$ gösterilmekte ve bu durum sağlandığında X aday çözümü Pareto optimal bir çözüm olarak değerlendirilmektedir.

iii) *Pareto optimal set (PS)*: Arama uzayındaki bastırılamayan (Pareto optimal) çözümlerden oluşan kümeye denmektedir. *PS* matematiksel olarak Denklem 3.3 ile gösterilmiştir.

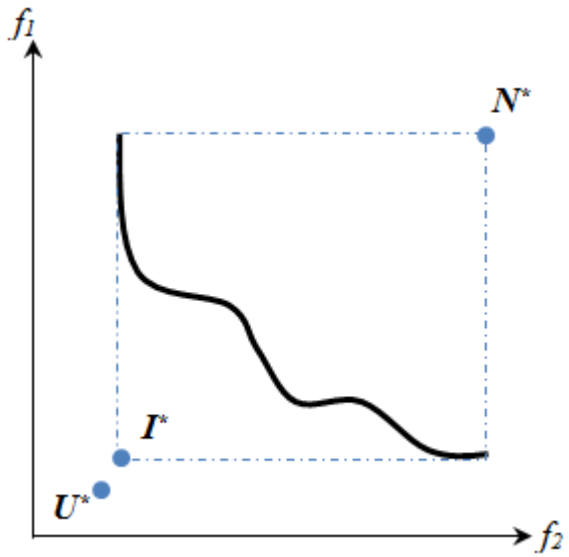
$$PS = \{X \in \Omega | \neg \exists Z \in \Omega: Z \prec X\} \tag{3.3}$$

iv) *Pareto optimal yüzey (Pareto front, PF)*: Pareto optimal çözümlerin karar değişkenlerine karşılık gelen amaç uzayındaki konumların oluşturduğu yüzeye denmektedir. *PF* matematiksel olarak Denklem 3.4 ile gösterilmiştir.

$$PF = \{F(X)|X \in PS\} \tag{3.4}$$

3.1.4. Özel çözüm vektörleri

Çok amaçlı optimizasyon problemlerinde bazı özel durumları niteleyen özel çözüm vektörleri bulunmaktadır. Bu çözüm vektörlerinin görsel olarak sunumu Şekil 3.4 ile verilmiştir. Bu gösterimde iki amaçlı bir problemin referans alındığı unutulmamalıdır.



Şekil 3.4. Özel çözüm vektörleri

Çok amaçlı bir optimizasyon probleminde tüm amaçları birlikte optimize eden bir çözümün olmadığı daha önce belirtilmişti. Ancak amaçlar ayrı ayrı değerlendirildiğinde her biri için optimal birer çözümün olduğu görülmektedir. İdeal amaç vektörü (I^*) her amaç için ayrı ayrı elde edilen optimal çözüm değerlerinin bir araya getirilmesi ile oluşturulur. Amaçlarının çatışma halinde olduğu çok amaçlı optimizasyon problemleri için ideal amaç vektörü gerçek olmayan hayali bir vektördür. Her bir amaç için en uygun çözümün tespit edilmesi Denklem 3.5 ile temsil edilirken bu çözümlerin bir araya getirilerek ideal amaç vektörünün oluşturulması Denklem 3.6 ile gösterilmektedir (Deb, 2011; Altınöz, 2015).

$$f_m^* = min\ f_m(X_m^*) \tag{3.5}$$

$$I^* = [f_1^*, f_2^*, \dots, f_M^*]^T \tag{3.6}$$

Denklem 3.5 ve 3.6'ya bakıldığında; f_m^* *m.*amaç fonksiyonu için optimum amaç fonksiyon değerini, X_m^*, f_m^* değerini sağlayan karar değişkenlerini, I^*ideal amaç vektörünü ifade etmektedir.

Ütopik amaç vektörü (U^*) bazı kaynaklarda ideal amaç vektörü ile aynı anlamda kullanılmasına rağmen bazı kaynaklarda ise ideal vektörden farklı ve daha küçük olan vektör olarak ele alınmaktadır. Ütopik amaç vektörünün farklı olarak değerlendirilmesi ve matematiksel olarak ifade edilmesi Denklem 3.7 ile gösterilmiştir.

$$\begin{aligned} &\epsilon_m > 0, m = 1,2, \dots, M \text{ olmak üzere} \\ &U_m^* = I_m^* - \epsilon_m \end{aligned} \tag{3.7}$$

Nadir amaç vektörü (N^*) Pareto optimal kümedeki çözümlerin her bir amaç fonksiyonu için üretilen maksimum değerlerin seçilmesi ile oluşturulan vektördür.

3.2. Çok Amaçlı Optimizasyon Algoritmaları

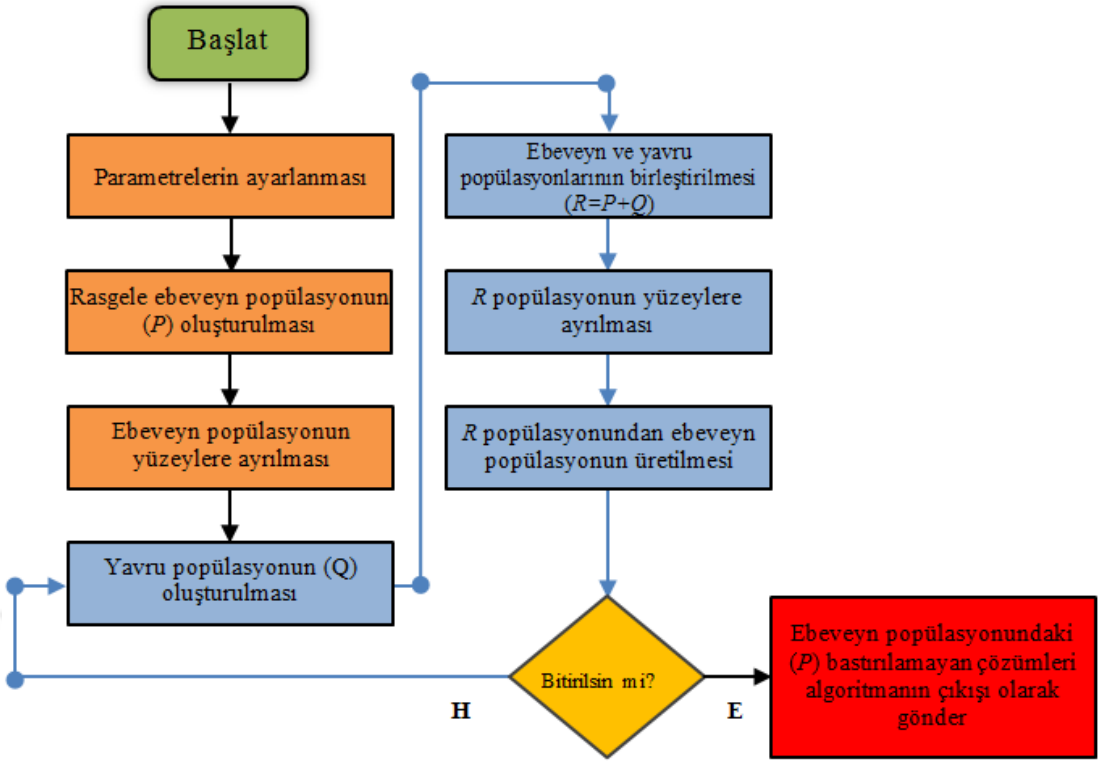
Tezde önerilen MOSG algoritması üç farklı problem setine uygulanmıştır. Her problem setinde önerilen algoritmanın performansı farklı algoritmalar ile karşılaştırılmıştır. Bu bölümde karşılaştırma yapılan çok amaçlı optimizasyon algoritmaları anlatılmıştır. Önerilen algoritmada NSGA-II algoritmasının bazı stratejileri kullanıldığından bu algoritma diğer algoritmalara göre daha detaylı bir şekilde incelenmiştir.

3.2.1. Bastırılamayan sıralamalı genetik algoritma (Non-dominated sorting genetic algorithm, NSGA-II)

Srinivas ve Deb tarafından önerilen NSGA'nın (Srinivas ve Deb, 1994) eksikliklerini gidermek ve algoritmayı daha başarılı bir hale getirmek amacıyla Deb ve arkadaşları tarafından NSGA-II (Deb ve ark., 2000) geliştirilmiştir.

3.2.1.1. NSGA-II algoritmasına genel bakış

NSGA-II diğer doğa esinli algoritmalar gibi bir takım parametrelere ihtiyaç duyar. Algoritmanın bu parametrelerin değerlerinin verilmesi ile başlar. Özel bir durum belirtilmediği sürece algoritma N adet çözümden oluşan ilk popülasyonunu (P) arama uzayında her karar değişkeni için verilen sınırlar çerçevesinde rastgele oluşturur. Oluşturulan çözümler için karar değişkenlerine karşılık gelen amaç fonksiyonları değerleri hesaplanır. Daha sonra popülasyon yüzeylere ayrıştırılır. Yüzeylere ayrıştırma işleminde sonraki başlıkta detaylıca anlatılan hızlı bastırılamayan sıralama stratejisi kullanılır. Ebeveyn olarak adlandırılan popülasyondan turnuva (ikili) yöntemi ile seçilen çözümler mutasyon ve çaprazlamaya tabi tutularak yavru bir popülasyon (Q) elde edilir. Yavru popülasyon da ebeveyn popülasyon gibi N adet çözümden oluşmaktadır. Sonraki aşamada ebeveyn ve yavru popülasyonlar bir araya getirilerek toplam bir popülasyon ($R=2N$ büyüklüğünde) elde edilir. Hızlı bastırılamayan sıralama stratejisi kullanılarak R popülasyonu yüzeylere ayrılır. Bu noktada *elitizm* devreye girerek R popülasyondaki en iyi çözümlerin sonraki nesle aktarılmasını sağlar. N adet en kaliteli çözümün seçilip sonraki neslin ebeveyn popülasyonunu (P^{t+1}) oluşturmasıyla elitizm uygulanmış olur. Sonlandırma kriteri sağlanana kadar ebeveyn popülasyondan çözümlerin seçilmesi ve bu çözümlerden yavru popülasyonun oluşturulması ve bunu takip eden diğer adımlardan oluşan çalışma döngüsü tekrar ettirilir. Çalışma döngüsü sonlandığında N adet çözümden oluşan ebeveyn popülasyonundaki bastırılamayan çözümler (Pareto optimal çözüm kümesi) elde edilir ve algoritma çıkışı olarak sunulur (Ergul ve Eminoglu, 2014; Chan ve ark., 2016; Özkış, 2017). Şekil 3.5 ile NSGA-II'nin akış diyagramı verilmiştir.



Şekil 3.5. NSGA-II akış diyagramı

3.2.1.2. Hızlı bastırılamayan sıralama stratejisi

Hızlı bastırılamayan sıralama stratejisi temel olarak popülasyondaki çözümlerin yüzeylerini belirlemeyi amaçlamaktadır. Pareto teoremine dayanarak gerçekleşen bir dizi işlem sonucunda her çözüm baskılama/baskılanma durumuna göre ait olduğu yüzeye atanır. N adet çözümün bulunduğu popülasyonda ilk yüzeyi bulmak için her çözümün popülasyondaki diğer tüm çözümler (N-1) ile karşılaştırılması gerekir. Popülasyonun tamamında bastırılamayan çözümler bulunduğunda ilk yüzey oluşturulmuş olur. İlk yüzeyi oluşturan çözümlerin popülasyondan çıkarılması ile geriye kalan çözümler tekrar karşılaştırılır ve bu alt popülasyonda bastırılamayan çözümler bulunarak ikinciyi yüzeyi oluşturur. Bu süreç popülasyondaki tüm çözümler bir yüzeye atanana kadar devam ettirilir. Her yüzeyin sadece bir çözümden oluşması en kötü durum olarak düşünülür ve karmaşıklığı $O(N^2)$ olur (Sağ, 2015; Özkış, 2017).

p : P popülasyonunda herhangi bir çözüm,

n_i : p çözümünü bastıran çözümlerin (ya da p çözümünün bastırılma) sayısı,

S_p : p çözümünün bastırdığı çözümlerin kümesi olmak üzere.

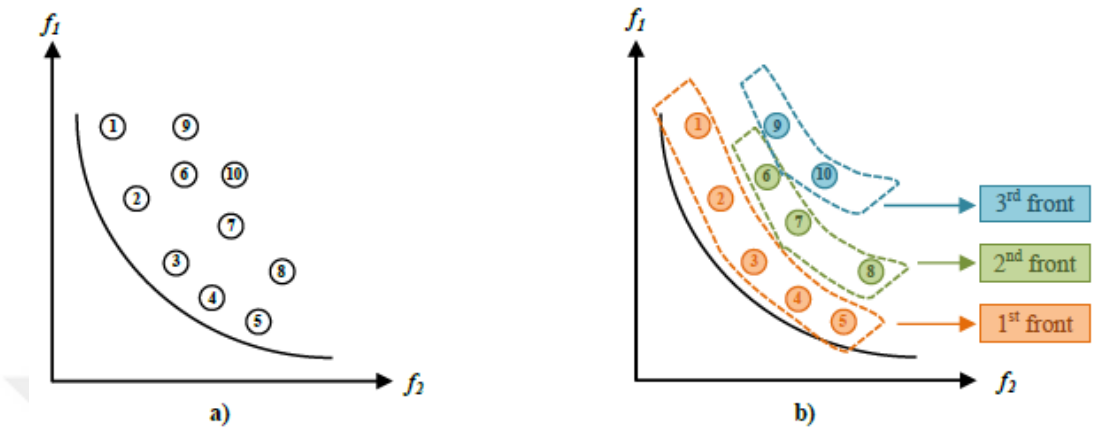
Popülasyondaki çözümlerin yüzeylerini belirlemek için öncelikle her p çözümüne ait n_i ve S_p değerlerinin bulunması gerekmektedir. Şekil 3.6 hızlı bastırılamayan sıralama stratejisinin sözde kodunu göstermektedir.

```
for each p in P
    n_p = 0,  S_p = ∅
    for each q in P
        if (p ≺ q)  // p çözümü q çözümünü bastırıyorsa
            S_p = S_p ∪ {q} // q çözümü p tarafından bastırılan çözümler kümesine eklenir
        elseif (q ≺ p) // q çözümü p çözümünü bastırıyorsa
            n_p = n_p + 1 // p çözümünün bastırılma sayısı arttırılır
        end if
    end // for each q in P
    if (n_p = 0)  // p çözümü hiç bastırılmamışsa (ilk yüzeye mi ait?)
        p_rank = 1
        F_1 = F_1 ∪ {p}
    end if
    i = 1  // yüzey sayacı başlangıç durumuna getirilir
    while F_i ≠ ∅
        Q = ∅  // sonraki yüzeylerin çözümleri için boşaltılır
        for each p ∈ F_i
            for each q ∈ S_p
                n_q = n_q - 1
                if (n_q = 0)  // q çözümünün bastırılma sayacı 0 ise sonraki yüzeye aittir
                    q_rank = i + 1
                    Q = Q ∪ {q}
                end if
            end //for each q ∈ S_p
        end //for each p ∈ F_i
        i = i + 1
        F_i = Q
    end //while
end //for each p in P
```

Şekil 3.6. Hızlı bastırılamayan sıralama stratejisi sözde kodu

Hızlı bastırılamayan sıralama stratejisine göre yapılan yüzeylere ayırma işleminde ilk yüzeye ait olan çözümlerin bastırılma sayısı (n_i) sıfır olur. İlk yüzey belirlendikten sonra ilk yüzeydeki her çözümün bastırdığı çözümlerin (S_p) bastırılma sayısı bir azaltılır ve bu kümede bastırılma sayısı 0 olan çözümler belirlenerek (Q kümesine eklenir) sonraki yüzeye (ikinci yüzey) atanır. Üçüncü yüzeydeki çözümlerin bulunması için Q kümesindeki çözümler tarafından bastırılan çözümlerin bastırılma sayısı bir azaltılır ve bastırılma sayısı sıfır olanlar üçüncü yüzeye aktarılır. Bu süreç tüm

çözümler ait olduğu yüzeye atanana kadar devam ettirilir. Şekil 3.7'de örnek bazı çözümlerin hızlı bastırılamayan sıralama stratejisine göre yüzeylere ayrılması gösterilmektedir.



Şekil 3.7. Hızlı bastırılamayan sıralama stratejisi ile yüzeylerin belirlenmesi

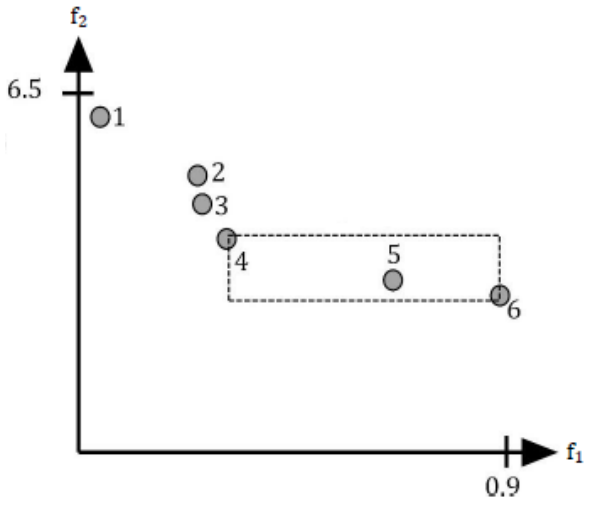
3.2.1.3. Kalabalık mesafesi (Crowding distance)

Çok amaçlı optimizasyon algoritmalarında elde edilen çözümlerin PF_t'ye yakın olmasının istenildiği kadar çözüm çeşitliliğinin de sağlanması istenmektedir. Bu nedenle elitizm aşamasında ve ebeveyn seçiminde çözümler içerisinde çevre yoğunluğu az olan çözümler tercih edilmektedir. Çevre yoğunluğu az olan çözümlerin etrafında keşfedilmemiş çözümlerin olması bu çözümlerin tercih edilmesinde etken olmaktadır. Çevre yoğunluğuna bakılarak çözümlerin belli bir sıralamaya alınması ve değerlendirilmesi adına kalabalık mesafe (*crowding distance, CD*) ölçütü kullanılmaktadır. Şekil 3.8 kalabalık mesafe hesaplama sözde kodunu vermektedir (Özkış, 2017; Roshanian ve ark., 2017).

```
l = | I |                    // I ilgili yüzey olmak üzere I bu yüzeydeki çözüm sayısı
for i=1 to l, setI[i]_CD = 0
for each m in M              // M amaç fonksiyonları olmak üzere
    I = S(I, m)              // m. amaç fonksiyonuna göre yüzeyi sırala
    I[1]_CD = I[l]_CD = inf // uç nokta çözümlerinin CD değerleri sonsuz olarak atanır.
    for i=2 to (l-1)         // ara çözümlerin CD değerleri hesaplanır
        I[i]_CD = I[i]_CD + (I[i+1]_m - I[i-1]_m)/(f_m^max - f_m^min)
    end
end
```

Şekil 3.8. Kalabalık mesafe hesaplama sözde kodu (Deb ve ark., 2002)

Popülasyondaki her bir çözümün kalabalık mesafesi değeri hesaplanırken her yüzey ayrı ayrı değerlendirilir. İlk aşamada çözümler seçilen her hangi bir amaç fonksiyonun değerine göre sıralanır. Kalabalık mesafe değeri bir çözümün her iki tarafındaki en yakın çözümler kullanılarak bulunduğu için uç noktalardaki çözümlere istisnai bir kural uygulanır. Uç noktalardaki çözümlerin bir tarafı boş olduğundan bu çözümlere kalabalık mesafe olarak sonsuz (*inf*) değeri verilir. Uç noktalar arasında kalan çözümlerin her biri için iki taraflarındaki komşu çözümler ile aralarındaki farkın normalize edilerek toplanması ile hesaplanır. Kalabalık mesafe değeri her amaç fonksiyonu için elde edilen normalize değerlerin toplamıdır. Normalize işleminde kullanılan maksimum ve minimum değer her yüzey için farklıdır ve uç noktaların amaç fonksiyonu değerlerinden oluşmaktadır. Şekil 3.9 örnek bir çözümler kümesi (iki amaçlı bir problem için) ve bu çözümlerin her iki amacı için sahip olduğu değerleri göstermektedir (Deb ve ark., 2000; Özkış, 2017).



	f_1	f_2
1.çözüm	0.004	6.293
2.çözüm	0.258	4.423
3.çözüm	0.263	4.412
4.çözüm	0.312	3.981
5.çözüm	0.674	2.957
6.çözüm	0.873	2.899

Şekil 3.9. Kalabalık mesafe için örnek bir çözümler kümesi (Özkış, 2017)

Şekil 3.9 ile verilen örnek çözümler kümesinde 5 numaralı çözüme ait *CD* değerini bulmak için komşu çözümleri olan 4 ve 6 numaralı çözümlerden faydalanılır. 5 numaralı çözüme ait kalabalık mesafe değeri CD_5 olsun. f_1 ve f_2 amaç fonksiyonları için ayrı ayrı kalabalık mesafe değeri hesaplanıp toplandığında CD_5 değeri elde edilmiş olur.

CD_5 = (0.873 – 0.312)/(0.873 – 0.004) + (3.981-2.899)/(6.293-2.899) = 0.964 şeklinde hesaplanır. Ara çözüm olan diğer çözümler (2, 3 ve 4 numaralı çözümler) için de aynı şekilde *CD* değerleri hesaplanır. Uç çözümler olan 1 ve 6 numaralı çözümlerin *CD* değeri sonsuz olarak verilir.

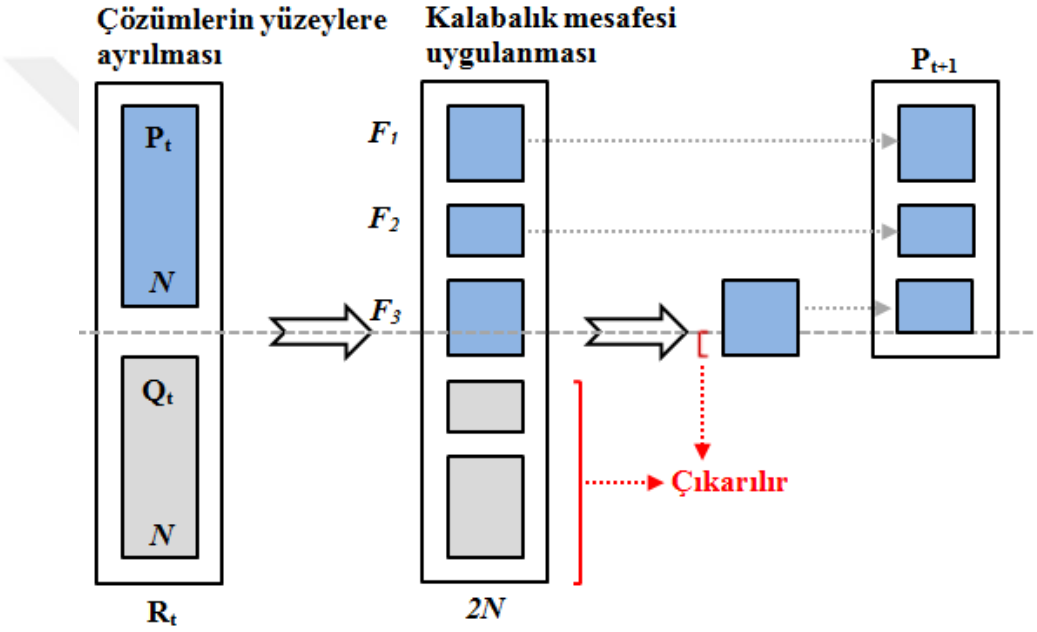
Kalabalık mesafe değeri popülasyondaki çözümler arasında kalite kıyaslaması yapmak için kullanılmaktadır. Eğer üzerinde çalışılan problem kısıtsız bir problem ise kıyaslama ilk etapta yüzeyler arasında yapılır. Popülasyondaki iki çözüm arasında kaliteli olma bakımından bir seçim yapılacaksa yüzey numarası küçük olan çözüm tercih edilir. Ancak çözümlerin ikisi de aynı yüzeyde ise yani yüzey numaraları eşit ise o zaman kalabalık mesafe değerleri ile karşılaştırma yapılır. Bu durumda kalabalık mesafe değeri büyük olan çözüm tercih edilir. Problem kısıtlı ise ve iki çözüm de kısıt ihlali yapmamışsa bu durumda kısıtsız problemlerdeki kalabalık mesafe karşılaştırma prosedürü uygulanır. Şayet çözümlerden sadece biri kısıt ihlali yaptı ise kısıt ihlali yapmayan çözüm seçilir. Her iki çözümün de kısıt ihlali yaptığı durumda daha az kısıt ihlali yapan çözüm tercih edilir (Deb ve ark., 2002).

3.2.1.4. Elitizm, çaprazlama ve mutasyon işlemleri

NSGA-II'de *N* çözümden oluşan popülasyon ilk aşamada belirlenen sınırlar çerçevesinde rastgele oluşturulur. Ancak her iterasyon basamağında ebeveyn popülasyonundan seçilen çözümler kullanılarak *N* büyüklüğünde bir yavru popülasyon elde edilir. Ebeveyn ve yavru popülasyondaki çözümlerin bir araya getirilmesi ile toplamda *2N* büyüklüğünde bir çözümler kümesine ulaşılır. *2N* büyüklüğündeki bu yeni popülasyondan *N* adet çözümün seçilip sonraki nesle aktarılması gerekmektedir. İşte bu noktada elitizm uygulanarak çözümler içerisinde en kaliteli *N* adet çözüm sonraki nesle aktarılır. *2N* büyüklüğündeki popülasyonu oluşturan çözümler arasındaki kalite kıyaslaması ilk etapta hızlı bastırılamayan sıralama ve ardından kalabalık mesafe karşılaştırması uygulanarak yapılmaktadır. Bu şekilde kaliteli çözümlerin saklanarak sonraki nesle aktarımı sağlanmaktadır. Şekil 3.10'da *2N* büyüklüğündeki çözümlere

elitizm uygulanarak sonraki neslin ebeveyn popülasyonunun elde edilmesi gösterilmiştir.

Ebeveyn popülasyondan seçilen çözümlerin kullanılarak yavru popülasyonu oluşturma aşamasında Deb ve Agrawal tarafından önerilen çaprazlama ve mutasyon yaklaşımları kullanılmaktadır (Deb ve Agrawal, 1995). Farklı türleri olan çaprazlama, iki çözümün karar değişkenlerinin belirlenen yönteme göre karşılıklı değiştirilmesi ile gerçekleşir. Mutasyon ise bir çözümün karar değişkenleri üzerinde yerel en iyiye takılma sorununu aşmak için uygulanan bireysel bir değişikliktir (Özsağlam ve Çunkaş, 2008; Özkış, 2017).



Şekil 3.10. Elitizmin uygulanması

3.2.2. İndikatör tabanlı evrimsel algoritma (Indicator based evolutionary algorithm, IBEA)

Metrik tabanlı bir çok amaçlı optimizasyon algoritması olan IBEA (Zitzler ve Künzli, 2004) uyarlanabilir bir yaklaşım sergilemektedir. Algoritmada temel idea popülasyondaki çözümler içerisinde sonraki nesle aktarılacak çözümlerin belirlenmesidir. Çözümler arasında uygunluk kontrolü bir ikili metrik – çoğu zaman hypervolume – ile yapılmaktadır. Belirlenen ikili performans metriğine dayalı olarak (ikili) turnuva seçimi uygulanır ve kötü olan çözüm popülasyondan çıkarılarak sonraki nesle daha uygun çözüm aktarılır (Li ve ark., 2017; Özkış, 2017; Zapotecas-Martínez ve

ark., 2019). Denklem 3.8 popülasyondaki bir çözümün uygunluk değerini hesaplarken Denklem 3.9 ikili turnuva sonucunda kötü çözümün elenmesinden sonra popülasyonda kalan çözümlerin uygunluğunu güncellemek için kullanılır (Özkış, 2017).

$$f(A) = \sum_{B \in P \setminus \{A\}} - e^{-I(\{B\},\{A\})/k} \tag{3.8}$$

$$f(A) = f(A) + e^{-I(\{B\},\{A\})/k} \tag{3.9}$$

P anlık popülasyonu, A ve B popülasyon içerisindeki herhangi iki çözümü, $I(\{B\},\{A\})$ performans metriğini, k ise kullanıcı tarafından probleme göre belirlenen skaler değeri göstermektedir. Şekil 3.11 IBEA'nın çalışma adımlarını göstermektedir.

Parametrelerin ayarlanması
N: popülasyon büyüklüğü
T: maksimum nesil sayısı
k: ölçekleme faktörü
Başlama: Başlangıç popülasyonu (*P*) oluştur ve iterasyon sayacı (*t*) sıfırla
Uygunluk hesaplama: Popülasyondaki çözümlerin uygunluk değerleri hesapla
Çevresel seçim: Çözüm sayısı *N*'yi aşmadığı sürece tekrarla
En küçük uygunluk değerine sahip A∈ P çözümünü seç
A'yı popülasyondan çıkar
Kalan çözümlerin uygunluk değerlerini güncelle
Bitirme: Belirlenen bitirme kriteri sağlandığında bitir ($t \geq T$ veya başka kıstas), *P*'deki bastırılamayan çözümleri çıktı olarak ver.
(Bitirme kriteri sağlanmadıysa)
Seçim: İkili turnuva ile *P*'den iki çözüm seç
Çeşitlilik sağlama: Çaprazlama ve mutasyon ile üretilen yavru çözümü popülasyona dâhil et. İterasyon sayacını arttır ve **Uygunluk hesaplama** adımına git.

Şekil 3.11. IBEA sözde kodu (Zitzler ve Künzli, 2004)

3.2.3. Çok amaçlı hücresel genetik algoritma (Multi-objective cellular genetic algorithm, MOCell)

Hücresel genetik algoritma yaklaşımını kullanan MOCell algoritması literatürde sıklıkla karşılaşılan başarılı bir algoritmadır. Algoritmadaki çözümler çevresindeki yakın komşuları ile etkileşim halinde olmaktadır. Küçük adımlar ile yavaş bir dağılımın söz konusu olduğu algoritmada yerel aramalar genetik operasyonlar ile sağlanmaktadır. İterasyon bazlı olan algoritmada harici olarak saklanan arşiv çözümleri, her iterasyon

adımından sonra popülasyondan rassal olarak belirlenen aday çözümler yerine popülasyona dâhil olmaktadır (Nebro ve ark., 2007; 2009b; Özkış, 2017). Şekil 3.12'de MOCell algoritmasının sözde kodu verilmiştir.

```
Başlat
P_Yüzey = YüzeyOluştur()  // Boş bir pareto yüzey oluştur
while !(Bitirme Şartı)
        for birey = 1 to N    // N: popülasyon büyüklüğü
          n_list = komşuluk_al(MOCell.pozisyon(birey))
          ebeveynler = Seçim(n_list)
          yavru = Çaprazlama (MOCell.P_CR, MOCell.ebeveynler)
          yavru = Mutasyon (MOCell.P_m, yavru)
          Uygunluk_Hesapla(yavru)
          Ekle(pozisyon(birey), yavru, MOCell, yardımcı_pop)
          P_Yüzeye_Ekle (yavru)
        end for
        MOCell.pop = yardımcı _pop
        MOCell.pop = GeriBesleme(MOCell, ParetoFront)
end while
Bitir
```

Şekil 3.12. MOCell algoritması sözde kodu (Nebro ve ark., 2009b)

3.2.4. Ayrıştırma temelli çok amaçlı evrimsel algoritma (Multi-objective evolutionary algorithm based on decomposition, MOEA/D)

Üzerinde çalıştığı problemi alt problem parçalarına ayırarak çözmeye çalıştığı için ayrıştırma tabanlı yaklaşım olarak değerlendirilmektedir. Alt parçalara ayırdığı problemin her bir bölümünü eş zamanlı olarak ele alır ve alt problemler için elde edilen en iyi çözümleri bir araya getirir. Yeni çözümlerin elde edilmesinde komşu alt problemlerdeki çözümlerden de faydalanan algoritma iyiye giden çözümleri popülasyona dâhil eder. Ayrıştırma tabanlı bu algoritmada problemlerin optimize edilmesi Tchebycheff, ağırlıklandırılmış toplam ve sınır kesişme yaklaşımları kullanılmaktadır (Zhang ve Li, 2007; Li ve Zhang, 2008; Gong ve ark., 2011). MOEA/D'ın sözde kodu Şekil 3.12 ile gösterilmiştir.

Girdiler: Çok amaçlı optimizasyon problemleri
MOEA/D'da tanımlanan N adet alt problem
N: Ağırlık vektörünün tek düze yayılımı ($\lambda^1, \lambda^2, \ldots, \lambda^N$.)
T: Her bir ağırlık vektörünün komşuluğundaki ağırlık vektörü sayısı
Çıktı: *EP*: Harici popülasyon
Başlama: **1.** $EP = \emptyset$
2. Herhangi iki ağırlık vektörü arasındaki Öklit mesafesini hesapla ve her bir vektöre
en yakın olan T adet ağırlık vektörünü keşfet. Her $i = 1, \ldots, N$ için $B(i) = \{i_1, \ldots, i_T\}$'yi belirle. $\lambda^{i_1}, \ldots, \lambda^{i_T}$ ağırlık vektörü λ^i'ye en yakın olan T adet ağırlık vektörünü ifade eder.
3. Başlangıç popülasyonunu oluştur. $FV^i = F(x^i)$
4. Probleme özgü metot ile $z = (z_1, \ldots, z_m)^T$'yi başlat.
Güncelleme: for $i = 1, \ldots, N$
1. **Üretim:** B(i)'den rastgele k ve l indisleri alınır, x^k ve x^l çözümlerinden genetik operatörler kullanılarak y çözümü üretilir.
2. **Gelişim:** Probleme özgü sezgisel bir onarım/gelişim uygulanarak y'den y' elde edilir
3. **z'yi güncelle:** Her $j = 1, \ldots, m$ için eğer $z_j < f_j(y')$ ise $z_j = f_j(y')$
4. **Komşu çözümlerin güncellenmesi:** Her $j \in B(i)$ indeksi için, eğer $g^{te}(y' | \lambda^j, z) \leq g^{te}(x^j | \lambda^j, z)$ ise $x^j = y'$ ve $FV^j = F(y')$
5. **EP'yi güncelle:** F(y') tarafından bastırılan tüm çözümleri EP'den çıkar. Eğer EP'de F(y')'nü domine eden çözüm yoksa EP'ye F(y')'nü ekle.

Bitirme: Kriter sağlanmışsa EP'yi çıkış olarak ver ve bitir, değilse **Güncelle** adımına git.

Şekil 3.13. MOEA/D sözde kodu (Zhang ve Li, 2007; Özkış, 2017)

3.2.5. Pareto arşivlenmiş evrim stratejisi (Pareto archived evolution strategy, PAES)

Knowles ve Corne tarafından geliştirilen algoritmanın ana iki hedefi mevcuttur. İlk hedefinde mutasyon operatörünü kullanarak yerel aramayı gerçekleştirme bulunmaktadır. Bunu yaparken küçük ilerlemeler ile mevcut konumundan komşu konumlara geçmeyi hedeflemektedir. Algoritmanın bir diğer hedefi ise popülasyondaki bastırılamayan çözümleri eşit kalitede değerlendirmesi ve en uygun Pareto optimal kümesini elde etmesidir (Knowles ve Corne, 1999; Oltean ve ark., 2005). Algoritmanın sözde kodu Şekil 3.14 ile verilmiştir.

```
Başlangıç parametrelerini belirle
Başlangıç popülasyonunu rastgele oluştur
Çözümlerin uygunluk değerlerini hesapla ve arşive ekle
while !(bitirme kriteri)
    Eldeki çözümden mutasyon ile yeni bir çözüm üret
    Yeni çözümün uygunluk değerini hesapla
    Eğer yeni çözüm eski çözümden daha iyi ise
        Yeni çözümü arşiv çözümleri ile karşılaştır
        Arşivi güncelle
        Sonraki iterasyon adımı için mevcut çözümlerden birini seç
    Yeni çözüm daha kötü ise
        Hiçbir şey yapma
end
Arşivi çıktı olarak ver
```

Şekil 3.14. PAES algoritması sözde kodu

3.2.6. Çok amaçlı yapay alg algoritması (Multi-objective artificial algae algorithm, MOAAA)

Uymaz ve arkadaşları tarafından geliştirilen yapay alg algoritması (Uymaz ve ark., 2015a) esasında tek amaçlı optimizasyon problemlerinin çözümü için önerilmiş bir doğa esinli algoritmadır. Algoritma, mikro alglerin sosyal yaşamlarının ve davranışlarının modellenmesi üzerine kurgulanmıştır. Helisel hareket, evrimsel süreç ve adaptasyon şeklinde üç temel aşamadan oluşmaktadır. Helisel hareket popülasyondaki çözümlerin konum güncellemeleri için kullanılırken evrimsel süreç en başarılı çözümün en başarısız çözüm ile reaksiyona girerek mitoz bölünme sonucu kopyalanmasını sağlamaktadır. Adaptasyon ise aç durumdaki çözümlerin en başarılı çözümlere benzemesine katkı sağlama aşamasıdır (Uymaz ve ark., 2015b; 2015a).

Babalik ve arkadaşları tek amaçlı optimizasyon algoritması olan yapay alg algoritmasını çok amaçlı problemlerin çözümüne uygulanacak şekilde çok amaçlı hale getirmişlerdir. MOAAA olarak isimlendirdikleri algoritma pareto tabanlı bir yaklaşım izlemiştir. Algoritma ayrıca çeşitlilik ve yakınsama yeteneğini geliştirmek üzere mutasyon ve çaprazlama gibi bazı operatörler kullanmaktadır (Babalik ve ark., 2018). MOAAA'nın sözde kodu Şekil 3.15 ile verilmiştir.

```
Başlangıç parametrelerini belirle
Başlangıç popülasyonunu rastgele oluştur
Çözümlerin uygunluk değerlerini hesapla
while !(bitirme kriteri)
    TempPop = Kopyala(Popülasyon)
    for n=1 to N  //N popülasyon büyüklüğü
        TempPop'tan ikili turnuva ile bir çözüm seç (PX)
        TempPop'tan ikili turnuva ile bir komşu çözüm seç (PY)
        PX' = PX
        PX' çözümünden rastgele seçilen üç niteliği (hücre) Helisel hareket ile güncelle
        PX' yavru çözümüne mutasyon uygula
        P' yavru çözümünün uygunluk değerini hesapla
        PX ve PX' çözümleri için enerji kontrolü yap
    end
    Ebeveyn ve yavru çözümleri bir araya getir (N+N=2N)
    Çözümlere hızlı bastırılamayan sıralama uygula
    Elitizm ile en iyi N çözümü sakla
    Algoritmanın parametrelerini güncelle
end
Arşivi çıktı olarak ver
```

Şekil 3.15. MOAAA sözde kodu

3.2.7. Çok amaçlı girdap arama (Multi-objective vortex search, MOVS)

Doğan ve Ölmez tarafından 2015 yılında bir sıvının sahip olduğu girdap akışının oluşturduğu etkiden ilham alarak geliştirilmiş bir algoritmadır (Doğan ve Ölmez, 2015a). Algoritma tek amaçlı sürekli optimizasyon problemlerinin çözümü için önerilmiştir. Algoritma ilk aşamada genel arama oluşturmak amacıyla arama uzayına geniş bir şekilde dağılım sağlar. İlerleyen aşamalarda ise yerel aramayı daha detaylı sağlamak amacıyla arama çemberini daraltarak ilerler. Bu şekilde yerel ve genel arama arasında dengeli bir düzen sağlamak eğilimindedir (Doğan ve Ölmez, 2015b; 2015a).

VS algoritmasının çok amaçlı bir versiyonu olan MOVS algoritması Özkış ve Babalık tarafından önerilmiştir (Özkış ve Babalık, 2017). NSGA-II'nin kalabalık mesafe ve hızlı bastırılamayan sıralama stratejilerini kullanarak geliştirdikleri algoritmada Pareto tabanlı bir yaklaşım ile çok amaçlı optimizasyon problemlerine çözüm getirmeyi hedeflemişlerdir. Şekil 3.16 MOVS algoritmasının sözde kodunu göstermektedir.

```
Başlangıç parametrelerini belirle
Başlangıç popülasyonunu rastgele oluştur
Çözümlerin uygunluk değerlerini hesapla
Hızlı bastırılamayan sıralama ve kalabalık mesafe uygula
while !(bitirme kriteri)
    for i=1 to N  //N popülasyon büyüklüğü
        İkili turnuva ile i.aday (yavru) çözüm için merkez seç
        Aday çözümü üret
        Aday çözüme çaprazlama operatörü uygula
        Aday çözüm için sınırları kontrol et
    end
    Ebeveyn ve yavru çözümleri bir araya getir (N+N=2N)
    Çözümlere hızlı bastırılamayan sıralama uygula
    Elitizm ile en iyi N çözümü sakla
end
Arşivi çıktı olarak ver
```

Şekil 3.16. MOVS algoritmasının sözde kodu

Not: Kıyaslama yapılan algoritmalar ile ilgili daha detaylı bilgi edinmek istenirse referans verilen çalışmalar incelenebilir.

3.3. Kullanılan Problem Setleri

Tez çalışması kapsamında önerilen algoritma 3 farklı problem seti üzerinde test edilmiştir. Bu bölümde önerilen algoritmanın uygulandığı bu problem setlerinden bahsedilmiştir.

3.3.1. Problem seti-1: Kısıtsız problemler

Tamamı kısıtsız olan bu set toplam 36 benchmark probleminden oluşmaktadır. Problem setinde 8 adet klasik problem ve 4 farklı problem ailesi bulunmaktadır. Bu setteki problemlerin ait olduğu problem ailesi ve amaç sayısı Çizelge 3.1 ile gösterilmiştir. Çizelge 3.1'e bakıldığında problemlerin amaç fonksiyonu sayısı olarak 2 ve 3 görülmektedir. Problemlerin daha detaylı bir gösterimi ise Çizelge 3.2 ile verilmiştir. Çizelge 3.2 problemlerin yapısal olarak birbirinden farklı olduğu göstermektedir (Coello ve ark., 2007; Coello ve ark., 2009; Al-Dujaili ve Suresh, 2016). [*DV: Decision Variables (karar değişkenleri), OF: Objective Functions (amaç fonksiyonları)*]

Çizelge 3.1. Kısıtlı problemler aileleri ve amaç sayıları

Problem ailesi	**Problem ismi**	**#Amaç**
Klasik problemler (Schaffer, 1985; Kursawe, 1990; Fonseca ve Fleming, 1995; Vlennet ve ark., 1996; Okabe ve ark., 2004)	Fonseca, Kursawe, OKA1, OKA2, Poloni, Schaffer	2
	Viennet2, Viennet3	3
DTLZ (Deb ve ark., 2005)	DTLZ1, DTLZ2, DTLZ4, DTLZ5, DTLZ6, DTLZ7	3
LZ09F (Li ve Zhang, 2008)	LZ09F1, LZ09F2, LZ09F3, LZ09F4, LZ09F5, LZ09F7, LZ09F8, LZ09F9	2
	LZ09F6	3
WFG (Huband ve ark., 2006)	WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG9	2
ZDT (Zitzler ve ark., 2000)	ZDT1, ZDT2, ZDT3, ZDT4, ZDT6	2

Çizelge 3.2. Kısıtlı problemlerin yapıları

Problem	**Separability**	**Modality**	**Scalability**	**Geometry**
ZDT1	f1 separable, f2 non-separable	uni	DV	convex
ZDT2	f1 separable, f2 non-separable	uni	DV	concave
ZDT3	f1 separable, f2 non-separable	x	DV	concave, disconnected
ZDT4	f1 separable, f2 non-separable	f1 uni, f2 multi	DV	convex
ZDT6	f1 separable, f2 non-separable	multi	DV	concave
WFG1	separable	uni	DV&OF	convex, mixed
WFG2	non-separable	f1 uni, f2 multi	DV&OF	convex, disconnected
WFG3	non-separable	uni	DV&OF	linear, degenerate
WFG4	non-separable	multi	DV&OF	concave
WFG5	separable	deceptive	DV&OF	concave
WFG6	non-separable	uni	DV&OF	concave
WFG7	separable	uni	DV&OF	concave
WFG9	non-separable	multi, deceptive	DV&OF	concave
DTLZ1	separable	multi	DV&OF	linear
DTLZ2	separable	x	DV&OF	concave
DTLZ4	separable	uni	DV&OF	concave
DTLZ5	f1 to fm-1 non-separable	uni	DV&OF	degenerate
DTLZ6	f1 to fm-1 non-separable	uni	DV&OF	degenerate
DTLZ7	separable	x	DV&OF	disconnected
LZ09_F1	x	x	No	convex
LZ09_F2				
LZ09_F3	x	x	No	convex
LZ09_F4	x	x	No	convex
LZ09_F5				
LZ09_F6				
LZ09_F7	x	x	No	convex
LZ09_F8				
LZ09_F9	x	x	No	concave
Fonseca	x	uni	No	convex
Kursawe	x	x	DV	concave
Schaffer	x	uni	No	convex
Viennet2	x	x	No	concave
Viennet3	x	x	No	concave
Poloni	x	x	No	concave
OKA1	x	x	No	convex
OKA2	x	x	No	concave

3.3.2. Problem seti-2: Mühendislik tasarımı ve kısıtlı problemler

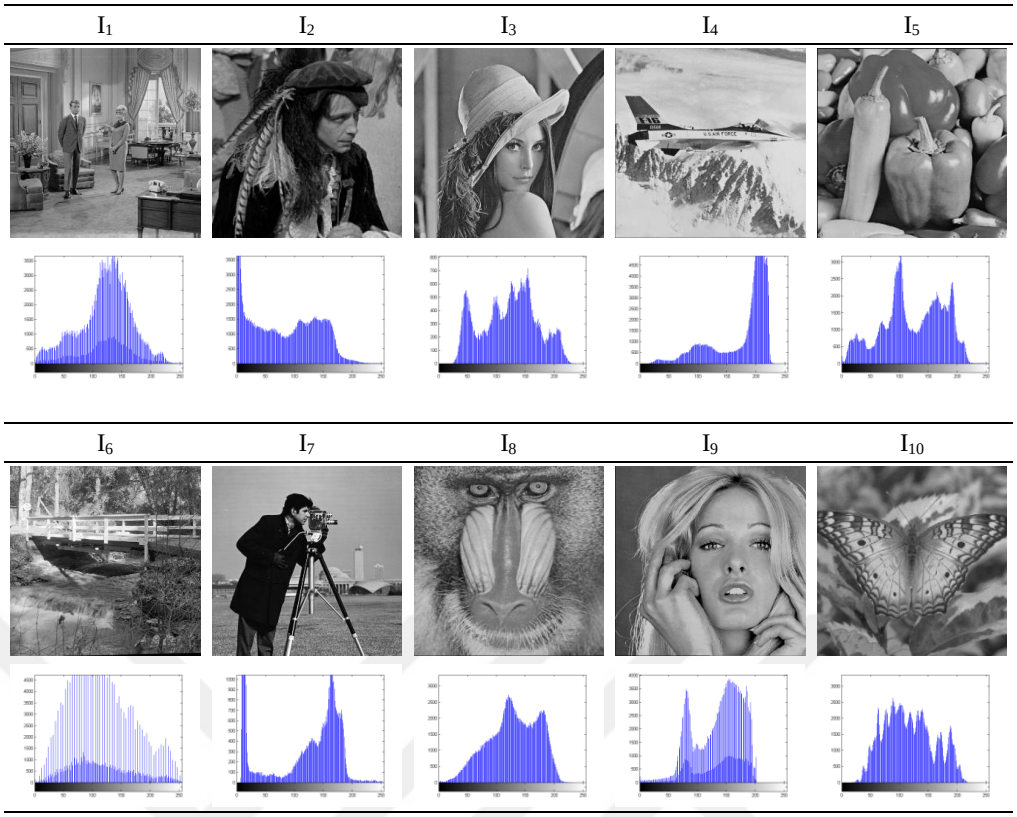
Bu problem seti farklı karar değişkeni sayılarına sahip on adet mühendislik tasarım ve kısıtlı problemlerden oluşmaktadır. Bu problem setinde bulunan problemlerin özellikleri Çizelge 3.3 ile verilmiştir. Water problemi hariç geriye kalan problemlerin amaç sayısı ikidir. Four-bar truss ve gear train problemleri kısıtsız iken diğer tüm problemlerin kısıtları mevcuttur (Durillo ve Nebro, 2011; Özkış, 2017).

Çizelge 3.3. Mühendislik tasarımı ve kısıtlı problemler

Problem	Boyut	#Amaç	Kısıt
Binh2	2	2	Var
ConstrEx	2	2	Var
Srinivas	2	2	Var
KITA	2	2	Var
Water	3	5	Var
Four-bar truss tasarım	4	2	Yok
Disk brake tasarım	4	2	Var
Centilever beam tasarım	2	2	Var
Spring	3	2	Var
Gear train	4	2	Yok

3.3.3. Problem seti-3: Segmentasyon görüntüleri

Tez kapsamında kullanılan son problem seti ise gri seviyeden oluşan on adet segmentasyon (bölütleme) görüntüsüdür. Bu görüntüler literatürdeki segmentasyon çalışmalarının çoğunda kullanılmaktadır. Kullanılan görüntülerden I_3 ve I_7 görüntülerinin büyüklüğü 256x256 piksel geriye kalan tüm görüntülerin büyüklüğü ise 512x512 pikseldir. Segmentasyonda kullanılan görüntüler ve bu görüntülere ait histogram grafikleri Şekil 3.17'de verilmiştir (Images, 2021).

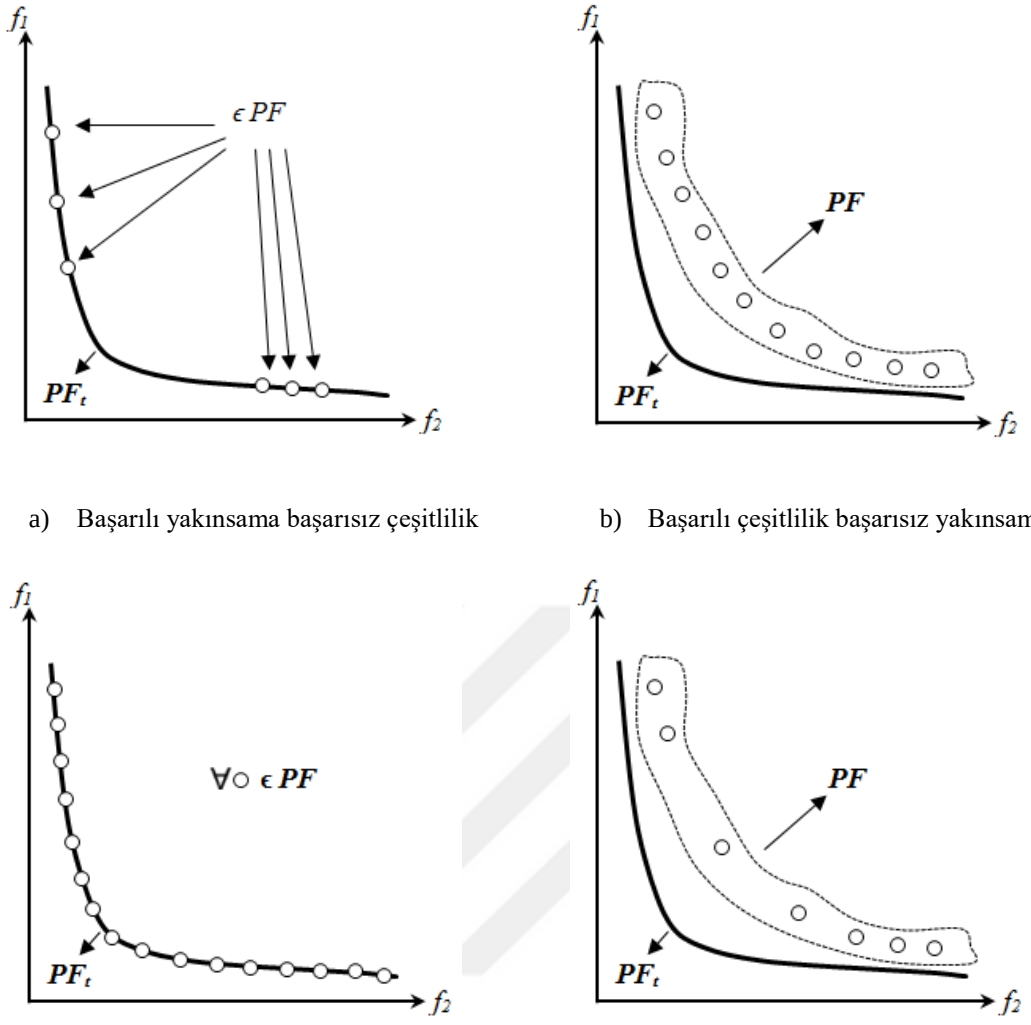


Şekil 3.17. Segmentasyonda kullanılan görüntüler ve histogram grafikleri

3.4. Performans Metrikleri

Daha önceki bölümlerde belirtildiği üzere çok amaçlı optimizasyon algoritmaları problemin çıktısı olarak en iyi tek çözüm yerine birbirini bastıramayan bir çözümler kümesi sunmaktadır. Söz konusu algoritma için Pareto optimal yüzeyi (*Pareto front*, *PF*) oluşturan bu çözümlerin gerçek Pareto optimal (*Pareto front true*, PF_t) yüzeyi en başarılı şekilde tahmin etmesi istenmektedir. Tek amaçlı bir optimizasyon problemi için iki algoritma tarafından elde edilen sonuçların kullanılan metriğe göre kıyaslanması oldukça basittir. Ancak çok amaçlı optimizasyon problemlerinde bu durum biraz daha farklı ve karmaşıktır.

Çok amaçlı bir algoritmanın elde ettiği çözümler kümesinin başarısını ölçmek için kullanılan temel iki ölçüt vardır: Çeşitlilik (*diversity*) ve yakınsama (*convergence*). *PF*'yi oluşturan çözümlerin PF_t çözümleri boyunca geniş bir dağılıma sahip olması çeşitlilik açısından başarı anlamına gelmektedir. Yakınsama başarısında ise *PF* çözümlerinin PF_t çözümlerine yakın olması beklenmektedir. Şekil 3.18 çok amaçlı bir algoritmanın elde ettiği *PF* çözümlerinin olası durumlarını göstermektedir.



a) Başarılı yakınsama başarısız çeşitlilik

b) Başarılı çeşitlilik başarısız yakınsama

c) İdeal yakınsama ve çeşitlilik

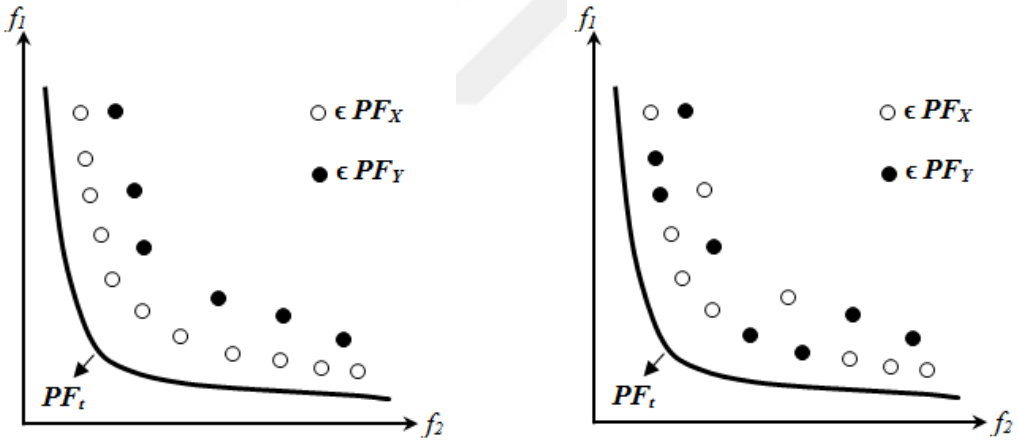
d) Başarısız yakınsama ve çeşitlilik

Şekil 3.18. *PF* için yakınsama ve çeşitlilik durumları

Şekil 3.18 (a)'ya bakıldığında algoritmanın elde ettiği *PF* çözümleri PF_t çözümlerine yakın değerler ürettiği ancak dağılım konusunda yeterli uygunluğu sağlayamadığı görülmektedir. Bu nedenle elde edilen *PF* çözümlerinin yakınsama açısından daha yüksek başarı sağladığı çeşitlilik açısından ise daha düşük başarı sağladığı söylenebilir. Şekil 3.18 (b) ise çeşitlilik açısında daha başarılı yakınsama açısından daha az başarılı bir *PF* çözümler kümesi sunmaktadır. Şekil 3.18 (c) her iki ölçüt için de başarılı ideal bir *PF* sunarken Şekil 3.18 (d) ise hem yakınsama hem de çeşitlilik açısından daha başarısız bir *PF* çözümler kümesi sunmaktadır (Kaya ve Güngör, 2007; Ergül, 2015; Özkış, 2017).

Çok amaçlı optimizasyon problemlerine uygulanan algoritmaların temel amacı Şekil 3.18 (c)'deki gibi bir *PF* çözümler kümesi elde etmektir. Ancak bu her zaman

mümkün olmayabilir. İdeal bir *PF* olmasa da her algoritmanın elde ettiği çözümlerin kalitesi bu küme ile belirlenmektedir. Çok amaçlı optimizasyon algoritmalarının performans karşılaştırması da elde edilen *PF* çözümleri ile yapılmaktadır. PF_X ve PF_Y sırasıyla *X* ve *Y* algoritmalarının bir çok amaçlı problemde elde ettiği optimal çözümler kümesi olmak üzere Şekil 3.19 bu çözümlerin PF_t çözümlerine yakınsama ve çeşitlilik durumlarını göstermektedir. Şekil 3.19 (a)'da PF_X'in açlık bir şekilde yakınsama açısından PF_Y'den daha başarılı olduğu görülmektedir. Çeşitlilik bakımından elde edilen çözümler denk olsa da yakınsama açısından *X* algoritmasının daha başarılı çözümler elde ettiği şekle bakılarak söylenebilir. Ancak performans kıyası yapılan algoritmaların elde ettiği çözümler her zaman bu şekilde görsel olarak bir fikir vermeyebilir. Şekil 3.19 (b)'de gösterilen sonuçlar için hangi algoritmanın daha başarılı olduğu gözlem yoluyla belirlenemez. Bu nedenle algoritmaların başarılarını sayısal olarak hesaplamak ve değerlendirmek adına yakınsama ve çeşitlilik değerlerini hesaplamak üzere bazı metrikler kullanılmaktadır (Deb ve Jain, 2002; Coello ve ark., 2007; Riquelme ve ark., 2015; Łapa, 2019).



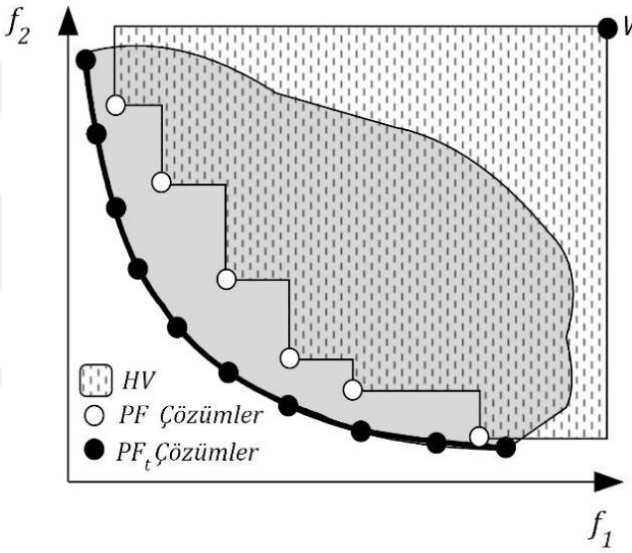
a) *X* algoritması *Y*'den daha başarılı b) Başarılı algoritma belirsiz

Şekil 3.19. İki algoritmanın *PF* açısından birbirine göre olası durumları

Literatüre bakıldığında performans karşılaştırma için önerilen birçok metriğin olduğu görülmektedir. Ancak bu tez kapsamında yakınsama ve çeşitlilik açısından algoritmaların başarısını karşılaştırmak üzere dört farklı metrik kullanılmıştır.

3.4.1. Hiperküp (Hypervolume, HV)

Zitzler ve Thiele tarafından önerilen HV (Zitzler ve Thiele, 1999) yaklaşımı çok amaçlı optimizasyon algoritmalarının yakınsama ve çeşitlilik başarılarını birlikte ölçen bir metriktir. Belirlenen bir *W* referans noktası ile algoritmanın elde ettiği *PF* çözümleri arasında kalan alanın ölçülmesi ile elde edilen bir değere sahiptir. Buradaki *W* referans noktası amaç fonksiyonlarının her biri için elde edilen en başarısız değerlerden meydana gelen vektördür. Şekil 3.20 çok amaçlı bir algoritmanın elde ettiği *PF* çözümlerine göre sahip olduğu *HV* değerinin temsilini göstermektedir (Zitzler ve Thiele, 1998; Van Veldhuizen, 1999; Zitzler, 1999; Tan ve ark., 2002).



Şekil 3.20. *HV* değerinin elde edilmesi (Özkış, 2017)

HV değerinin büyük olması istenen durumdur ve daha büyük *HV* değerine sahip algoritmanın kıyasla daha başarılı olduğu belirtilmektedir. Çok amaçlı bir algoritmanın elde ettiği *PF* çözümlerine göre sahip olduğu *HV* değeri ($i \in PF$ olmak üzere) Denklem 3.10 ile hesaplanmaktadır (Zitzler ve Thiele, 1999; Özkış, 2017).

$$HV = volume\left(\bigcup_{i=1}^{|PF|} v_i\right) \tag{3.10}$$

HV değerinin hesaplanmasında PF_t çözümlerine ihtiyaç duyulmamaktadır. Ancak bazı çalışmalarda bu metriğin normalize edilmiş değeri kullanılmaktadır. Normalize *HV* değerinin hesaplanacağı durumlarda ise PF_t çözümlerine ve bu

çözümlerden elde edilecek olan HV_t değerine ihtiyaç duyulmaktadır. HV_t Denklem 3.10 ile hesaplanır ve bu aşamada *PF* çözümleri yerine PF_t çözümleri kullanılır. Normalize edilmiş *HV* (HV_N) değeri Denklem 3.11 ile hesaplanmaktadır. Elde edilen HV_N değeri 1'e ne kadar yakınsa o kadar başarılı bir *PF* çözümler kümesi elde edilmiş demektir (Zitzler ve Thiele, 1999; Xiang ve ark., 2015; Özkış, 2017).

$$HV_N = \frac{HV}{HV_t} \tag{3.11}$$

3.4.2. Terslenmiş nesilsel mesafe (Inverted generational distance, IGD)

GD (Van Veldhuizen ve Lamont, 1998) metriğinin eksiklerini gidermek amacıyla bu metriğin terslenmiş bir versiyonu olan *IGD* de hem yakınsama hem de çeşitlilik başarısını ölçen bir metriktir. PF_t çözümler kümesindeki her bir noktanın en yakınında bulunan *PF* çözümüne olan mesafelerin ortalaması sonucu bulunan değerdir. Bu metrikte değerin küçük olması elde edilen *PF* çözümlerinin PF_t çözümlerine yakın olduğunu dolayısıyla başarılı bir sonuç alındığını göstermektedir. Hesaplanması için PF_t çözümlerine ihtiyaç duyulan metriğin matematiksel karşılığı Denklem 3.12 ile verilmiştir (Ishibuchi ve ark., 1997; Coello ve Cortés, 2005; Durillo ve Nebro, 2011).

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{3.12}$$

Denklem 3.12'deki *n* değeri PF_t çözümlerinin sayısını ve d_i PF_t 'deki *i*.çözüm ile kendisine en yakın olan *PF* çözümü arasındaki Öklit (Xu ve Xia, 2011) mesafesini göstermektedir.

3.4.3. Yayılım (Spread)

Algoritma tarafından elde edilen bastırılmamış çözümlerin (*PF*), PF_t çözümleri boyunca dağılım kalitesini dolayısıyla çeşitlilik kalitesini ölçmek için kullanılır. Birbirine en yakın iki ardışık çözümün, normalleştirilmiş amaç uzayındaki mesafelerinin hesaplanmasına dayanan bu metrikte elde edilen değerin küçük olması yani sıfıra yaklaşması istenen durumdur. Spread metriğinin hesaplanması Denklem 3.13 ile gösterilmiştir (Li, 2003; Riquelme ve ark., 2015; Sağ, 2015).

$$Spread = \frac{\sum_{i=1}^{m} d(e_i, PF) + \sum_{x \in PF} \left| d(x, PF) - \overline{d} \right|}{\sum_{i=1}^{m} d(e_i, PF) + |PF| * \overline{d}} \quad (3.13)$$

$$d(x, PF) = \min_{y \in PF, y \neq x} \|f(x) - f(y)\|^2 \quad (3.14)$$

$$\overline{d} = \frac{1}{|PF_t|} \sum_{x \in PF_t} d(x, PF) \quad (3.15)$$

Denklem 3.13'te (Özkış, 2017):

x: *PF* çözümler kümesinde bir çözümü

d(x, PF): *x* çözümü ile en yakın çözümler arasındaki en düşük mesafe

$\overline{d}$: *d(x, PF)* mesafelerinin ortalaması

e_i: i.amaç fonksiyonu için PF_t çözümler kümesine ait uç çözüm

$d(e_i, PF)$: *PF* ve PF_t kümelerinin uç çözümleri arasındaki mesafedir.

3.4.4. Epsilon

Epsilon (ε), algoritmaların elde ettiği bastırılmamış çözümlerin yakınsama kalitesini ölçen bir metriktir. Bu metrik değerinin elde edilmesinde PF_t çözümlerine ihtiyaç duyulmaktadır. Metriğin değeri, *PF* çözümlerinin her birinin tek tek ele alınarak PF_t çözümlerinden en az birini bastırması için gereken minimum değişiminin hesaplanması ile elde edilir. Anlaşılacağı üzere elde edilen *PF* çözümleri PF_t çözümlerine ne kadar yakınsa o oranda düşük bir değer elde edilir ve bu metrikte küçük değer elde etmek istenen durumdur. Denklem 3.16 *PF* ve PF_t çözümleri arasındaki Epsilon metrik değerinin hesaplanmasını matematiksel olarak göstermektedir (Durillo ve Nebro, 2011; Xiang ve ark., 2015; Asadzadeh, 2016; Babalik ve ark., 2018).

$$\varepsilon = inf_{\varepsilon \in R^+} \{\forall \vec{p} \in PF_t, \exists \vec{\alpha} \in PF : \vec{\alpha} < \varepsilon \vec{p}\} \quad (3.16)$$

4. ÖNERİLEN ALGORİTMA

Tez çalışması kapsamında tek amaçlı optimizasyon algoritmalarından SFLA ve GWO algoritmaları hibrit bir şekilde kullanılarak çok amaçlı optimizasyon problemlerine uygulanacak bir algoritma önerilmiştir. Bu hibrit yaklaşımda; arama uzayında etkin bir tarama sağlamak için SFLA'nın mempleks yapısı, yerel aramada başarılı bir performans için ise GWO algoritmasının arama stratejisi kullanılmıştır. Önerilen algoritmanın performansını çeşitlilik ve yakınsama açısından arttırmak amacıyla bazı modifikasyon ve stratejiler uygulanmıştır. Bu bölümde, ilk aşamada önerilen algoritmanın bileşenleri anlatılacak daha sonra önerilen algoritma detaylı bir şekilde ele alınacaktır.

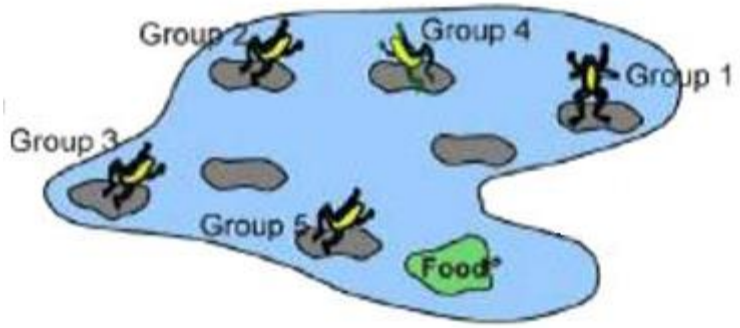
4.1. Algoritma Bileşenleri

Önerilen algoritma doğa esinli SFLA ve GWO algoritmalarının hibrit bir şekilde kullanılmasına dayanmaktadır. Bununla beraber algoritmanın başarısını farklı yönlerden arttırmak adına bazı yaklaşımlar eklenmiştir. Bu başlık altında SFLA ve GWO algoritmaları ile beraber uygulanan yaklaşımlar hakkında bilgi verilmiştir.

4.1.1. Kurbağa sıçrama algoritması (Shuffled frog leaping algorithm, SFLA)

Eusuff ve Lansey tarafından 2003 yılında su dağıtım şebekesine uygulanarak literatüre kazandırılan SFLA, sürü halinde yaşayan kurbağaların sosyal yaşamda sergiledikleri hareketlerini model almaktadır (Eusuff ve Lansey, 2003).

Göl ve benzeri sulak alanlarda toplu şekilde yaşayan kurbağa sürüsünün temel amacı minimum hareket ile maksimum besin sağlamaktır. Topluluktaki her bir kurbağa sahip olduğu bilgileri mem denilen yapıda saklayıp bu bilgilerini topluluktaki diğer bireyler ile paylaşır. Bu açıdan bakıldığında SFLA'nın memetik bir algoritma olduğu söylenebilir. Genetik algoritmalarda iyi bilgilerin gen ile sonraki nesillere aktarıldığı gibi bu algoritmada da bilgiler mem yapısı ile sonraki nesillere aktarılır (Eusuff ve Lansey, 2003; Eusuff ve ark., 2006). Şekil 4.1 gruplara ayrılmış bir kurbağa sürüsünün yaşam alanındaki temsilini göstermektedir.



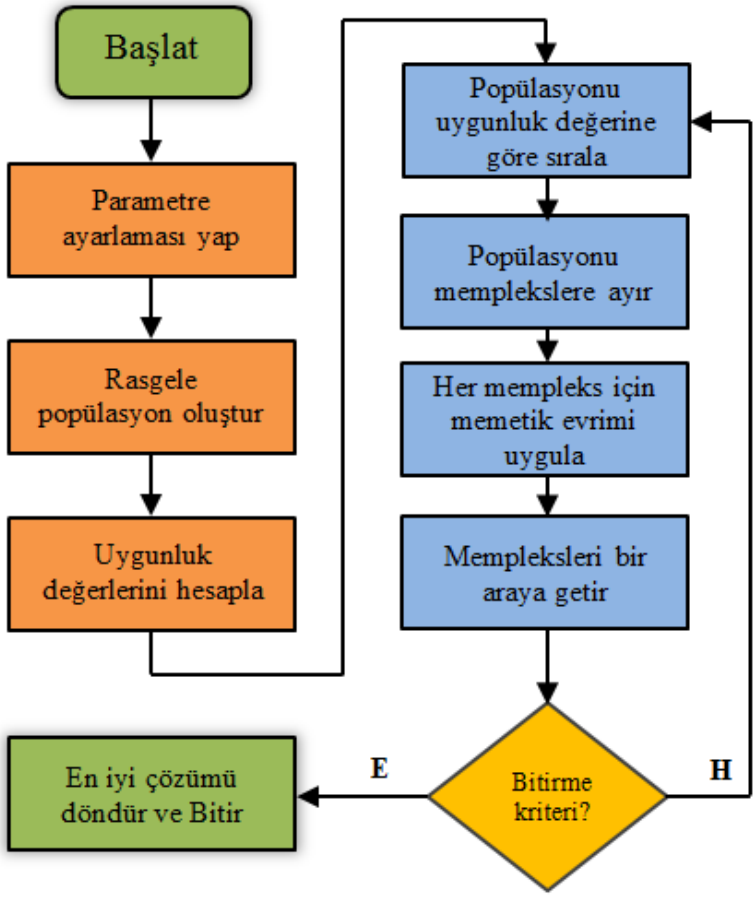
Şekil 4.1. Bir kurbağa sürüsünün yaşam alanı (Elbeltagi ve ark., 2005)

SFLA popülasyondaki bireylerin bağımsız ancak paylaşıma dayalı çözümler üretmesine imkân vermektedir. Her birey ilk etapta bağımsız çözümler üretirken bu çözümler popülasyon genelinde bir araya getirilerek paylaşılır ve bilgi paylaşımı yapılır. Modellenmiş algoritmada sürüdeki her bir kurbağa popülasyondaki bir çözümü temsil eder. Problemin yapısına bağlı olarak belirlenen kısıt ve karar değişkenlerine göre her bireyin bir konumu ve bu konuma bağlı olarak bir uygunluk değeri mevcuttur. SFLA modelinde bu uygunluk değeri bireyin besine olan yakınlık derecesini temsil eder. İlk etapta arama uzayına rastgele dağılan bireyler için uygunluk değerleri hesaplanır. Daha sonra popülasyon en iyi konuma sahip birey ilk sırada olacak şekilde sıralanır. Sıralanan popülasyon Denklem 4.1 kullanılarak mempleklere (gruplara) ayrılır. Bu aşamadan sonra tekrar bir araya getirilene kadar her bir mempleks birbirinden bağımsız olarak değerlendirilir. Her bir mempleks önceden belirlenen *memetik iterasyon sayısı* kadar memetik bir evrime tabi tutulur. Memetik evrim aşamasında amaç; kötü konuma sahip bireylerin nispeten daha iyi konumlara yönlenmesidir. Kötü konumların iyileştirilmesi ise mempleks bazında en iyi konum (*local best*) veya popülasyon bazında en iyi konumun (*global best*) yardımıyla olmaktadır. Tüm mempleksler memetik evrime uğradıktan sonra popülasyon bazında bilgi paylaşımı yapılması adına tekrar bir araya getirilir. Bu sayede bireylerin kişisel olarak sahip olduğu bilgi global düzeyde paylaşılıp popülasyonun daha kaliteli çözümlere doğru ilerlenmesi sağlanır (Eusuff ve ark., 2006; Zhen ve ark., 2009; Tang ve ark., 2016; Karakoyun, 2019).

$$Y_k = \left[(X_i)_k \middle| X_i = X\left(k + m * (i-1)\right)\right], i = 1, \dots, n \text{ ve } k = 1, \dots, m \quad (4.1)$$

Denklem 4.1'de *X* popülasyondaki bireyleri, *m* mempleks sayısını ve *n* her bir memplekse dâhil olacak birey sayısını göstermektedir. Mempleksl ere ayırma ile ilgili detaylı bilgi *Mempleks stratejisi* başlığı altında verilmiştir.

Şekil 4.2 algoritmanın genel adımlarını göstermektedir.



Şekil 4.2. SFLA'nın akış diyagramı

SFLA'nın detaylı bir sözde kodu ise Şekil 4.3 ile verilmiştir.

```
Başlangıç parametrelerini belirle
Başlangıç popülasyonunu rastgele oluştur
Çözümlerin uygunluk değerlerini hesapla
Popülasyonu en iyi uygunluk değerinden en kötüye doğru sırala
İlk sıradaki çözümü global en iyi (g_b) olarak işaretle
while !(bitirme kriteri)
    Popülasyonu mempleklere ayır
    for i=1 to m  //m mempleks sayısı, her mempleks için memetik döngü başlat
        for im=1 to M  //M: Her mempleksin uğrayacağı memetik evrim sayısı
            Rulet seçimi ile i.mempkleksten q adet birey seç // Seçilme olasılıkları uygunluk değerine göre hesaplanır.
            Alt-memplekste (q), en kötü (I_w) ve en iyi (I_b) konuma sahip bireyleri belirle
            I_w' yi I_b kullanarak güncelle ve aday konumu (I_w') elde et
            if f(I_w') < f(I_w)  // minimizasyon problemi için
                I_w = I_w'       // En kötü çözümü aday çözüm ile güncelle
            else
                I_w' yi g_b kullanarak güncelle ve yeni aday konumu (I_w') elde et
                if f(I_w') < f(I_w)
                    I_w = I_w'       // En kötü çözümü aday çözüm ile güncelle
                else  //Hem I_b hem de g_b kullanılarak daha iyi bir çözüm elde edilmezse
                    I_w = RasgeleKonumÜret() // En kötü çözüm için rastgele bir konum üret
        end // im=1 to M bitti
        Mempleksi kendi içinde uygunluk değerine göre sırala
    end  // i=1 to m bitti
    Mempleksleri bir araya getir ve karıştır
    Popülasyonu en iyi uygunluk değerinden en kötüye doğru sırala
    g_b çözümünü kontrol et ve gerekirse güncelle
end  //Bitirme kriteri sağlanınca döngüyü bitir
g_b çözümünü çıktı olarak ver ve bitir.
```

Şekil 4.3. SFLA'nın sözde kodu

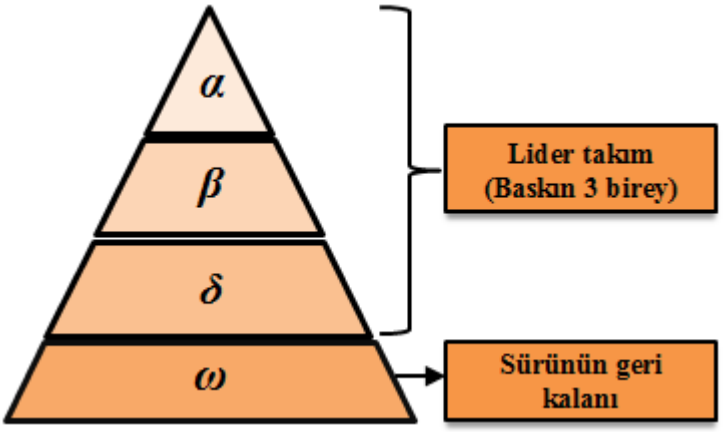
Not: Önerilen hibrit algoritmada SFLA'nın mempleks yapısı kullanıldığından tez çalışması kapsamında algoritma ile ilgili bu kısım ayrı bir başlık açılarak vurgulanmıştır. Algoritmanın detaylı analizini yapmak isteyen araştırmacılar (Eusuff ve ark., 2006; Karakoyun, 2015) çalışmalarına göz atabilir.

4.1.2. Gri kurt algoritması (Grey wolf optimizer, GWO)

Mirjalili ve arkadaşları tarafından önerilen GWO (Mirjalili ve ark., 2014) algoritması sürü halinde yaşayan gri kurtların hiyerarşik yapısı ve avlanma hareketlerini model almıştır. Diğer metasezgisel algoritmalar gibi bu algoritma da popülasyon tabanlı, iteratif bir algoritmadır. GWO, sürüyü oluşturan kurtların sürüdeki sosyal

rollerinin ve bu rollerinin avlanmaya etkilerinin matematiksel olarak modellenmesi esasına dayanmaktadır. GWO algoritmasının modellenmesi sürünün dört temel davranışından oluşmaktadır: Sosyal hiyerarşi, avı çevreleme, avlanma ve ava saldırma (Rodríguez ve ark., 2017; Singh ve Singh, 2017).

Sosyal hiyerarşi: GWO algoritmasında popülasyonu oluşturan bireyler (kurtlar) dört farklı rol ile temsil edilmektedir: alfa (*alpha*, α), beta (β), delta (δ) ve omega (ω). Popülasyondaki her birey bu rollerden yalnızca birini üstlenebilir. Algoritmanın sosyal hiyerarşi düzenine göre en güçlü (en iyi uygunluk değerine sahip) üç birey lider takım olarak adlandırılır. Lider takımındaki en iyi uygunluk değerine sahip birey *alfa*, ikinci sıradaki *beta* ve üçüncü sıradaki birey *delta* olarak adlandırılır. Popülasyonun lider takımı dışında geri kalan bireylerin tamamı ise *omega* olarak isimlendirilir (Saremi ve ark., 2015; Heidari ve Pahlavani, 2017). Şekil 4.4 popülasyondaki bireylerin hiyerarşik rollerinin gösterimini temsil etmektedir.



Şekil 4.4. GWO algoritmasında sosyal hiyerarşi

Bu hiyerarşi düzeninde *alfa* sürünün baskın bireyidir ve uyku, avlanma, dinlenme gibi önemli kararların alınmasında belirleyici rol oynar. Lider takımındaki diğer iki birey (*beta* ve *delta*), hiyerarşi olarak *omega* bireylerin üstündedir ve yönetme konusunda *alfa*'ya yardımcı olurlar (Komaki ve Kayvanfar, 2015; Karakoyun ve ark., 2019).

Avı çevreleme: Sürü halindeki gri kurtların avlanma prosedürünün önemli bir aşaması avı çevrelemedir. GWO algoritmasının modellenmesinde bu aşamada dikkatli bir şekilde ele alınmıştır. Avı çevrelemenin matematiksel modeli Denklem 4.2'de verilmiştir (Mirjalili ve ark., 2014).

$$\vec{X}'(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \quad (4.2)$$

$$\vec{D} = |\vec{C}.\vec{X}_p(t) - \vec{X}(t)| \quad (4.3)$$

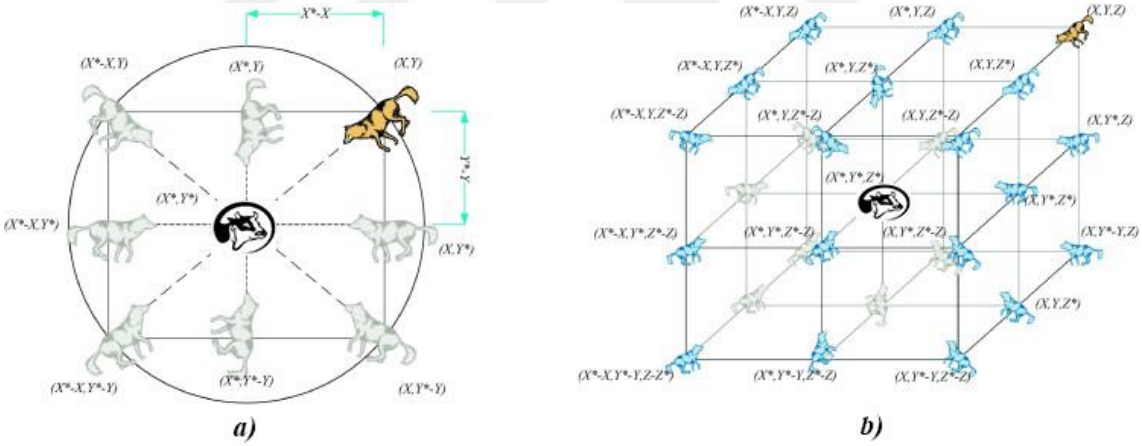
Burada *t* iterasyon numarası, $\vec{A}$ ve $\vec{C}$ sırasıyla Denklem 4.4 ve 4.5 ile elde edilen katsayı vektörleri, $\vec{X}_p$ avın konumu ve $\vec{X}$ popülasyondaki herhangi bir bireyin mevcut konumudur.

$$\vec{A} = 2.\alpha\,.\vec{r}_1 - \alpha \quad (4.4)$$

$$\vec{C} = 2.\vec{r}_2 \quad (4.5)$$

Buradaki a değeri - iterasyona bağlı olarak – 2'den 0' a doğru doğrusal olarak düşen bir değişken, r_1 ile r_2 [0 – 1] aralığında rastgele üretilen sayılardır.

Denklem 4.2'nin etkisi sonucu bir bireyin sonraki adımda olası konumları, iki ve üç boyutlu olarak Şekil 4.5'te gösterilmiştir.



Şekil 4.5. Bir bireyin avı çevreleme sonucu sahip olabileceği olası konumlar

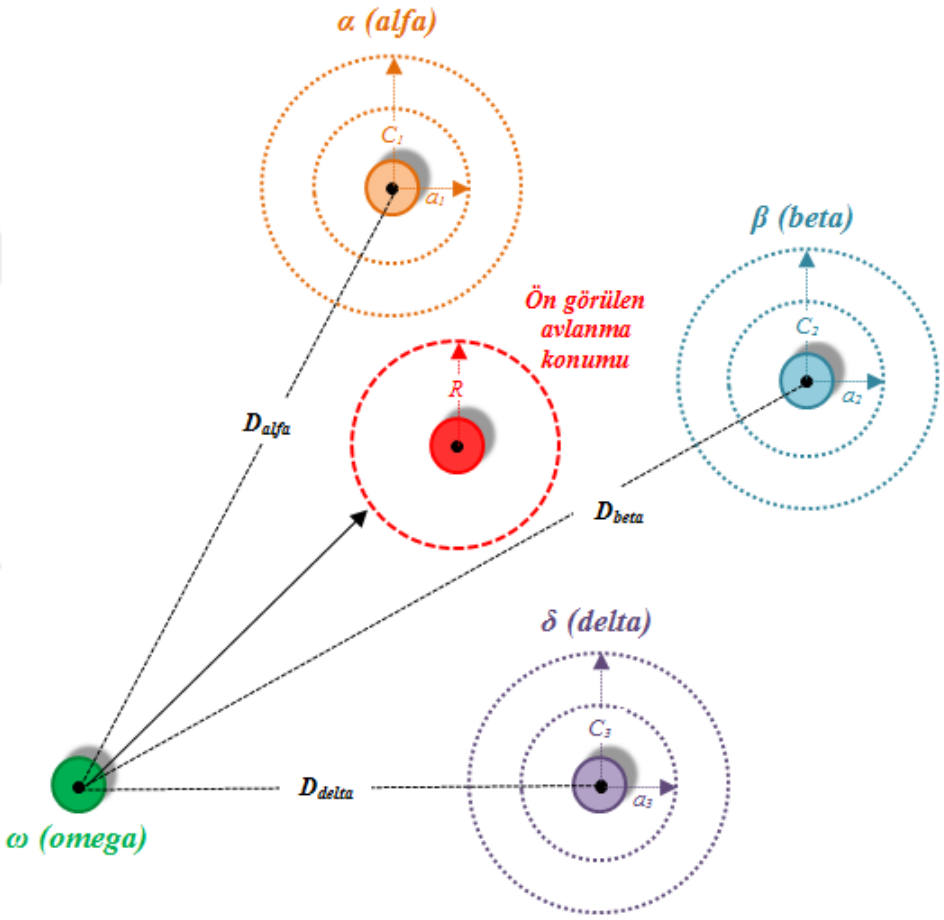
Avlanma: Daha önce belirtildiği üzere sürüye liderlik eden üç baskın birey (*alfa, beta* ve *delta*) mevcuttur. Bu üç birey uygunluk değeri açısından en iyi değere sahiptir ve popülasyondaki bireylerin ilerlemesi bu lider takımın denetiminde olmaktadır. Popülasyondaki bireyler avlanma sırasında lider takımın konumlarını gözeterek ava doğru ilerlemektedir. GWO algoritmasında sürünün avlanma davranışlarının matematiksel modellenmesi Denklem 4.6, 4.7 ve 4.8 ile gösterilmiştir (Mirjalili ve ark., 2014; Tripathi ve ark., 2018).

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}|, \quad \vec{D}_\beta = |\vec{C}_2.\vec{X}_\beta - \vec{X}|, \quad \vec{D}_\delta = |\vec{C}_3.\vec{X}_\delta - \vec{X}| \tag{4.6}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.\vec{D}_\alpha, \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\vec{D}_\beta, \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.\vec{D}_\delta \tag{4.7}$$

$$\vec{X}^{(t+1)} = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{4.8}$$

Verilen denklemler ile matematiksel olarak modellenen avlanma aşamasında lider takımın etkisi Şekil 4.6'da temsil edilmiştir.



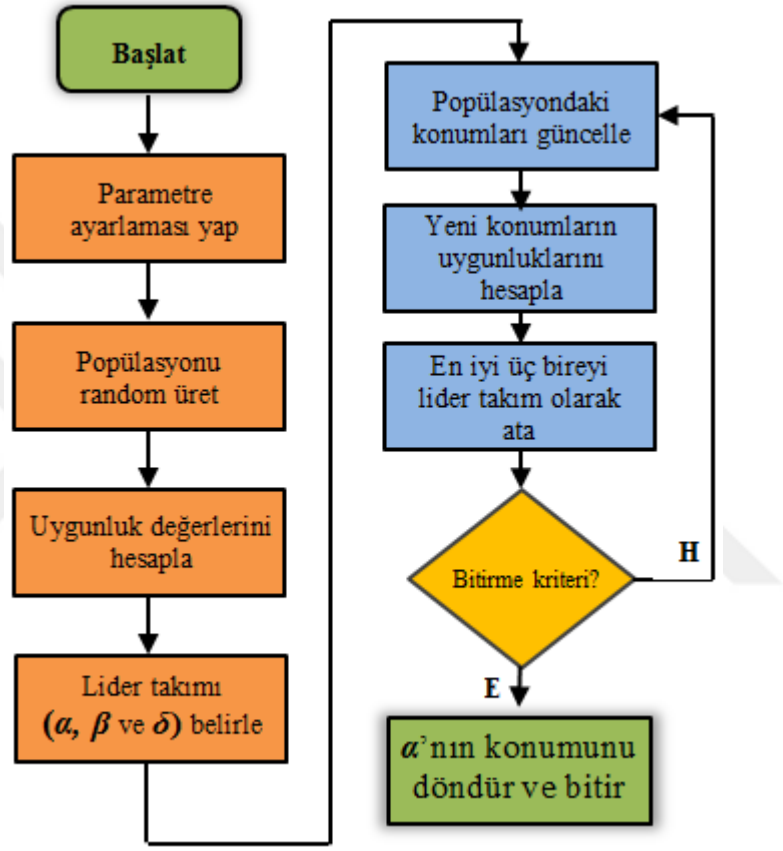
Şekil 4.6. Lider takımın konum güncellemeye etkisi

Şekil 4.6'da görüldüğü üzere popülasyondaki konum güncellemeleri lider takım denetiminde yapılmaktadır. Popülasyondaki bireylerin olası sonraki konumları lider takımın konumlarının referans alınmasına bağlı bir rassallık ile gerçekleşmektedir.

Ava saldırma: Avlanma sürecinin bu son aşamasında sürüdeki kurtların mevcut ava saldırması veya yeni av arayışına girmesi durumunun modellenmesi mevcuttur. Daha önce belirtildiği üzere a iterasyona bağlı olarak 2'den 0'a doğru doğrusal şekilde değişen bir değerdir. Denklem 4.4'e bakıldığında $\vec{A}$'nın değer değişiminin a'ya bağlı

olduğu görülmektedir. Bu durum ele alındığında *a*'nın düşüşüne bağlı olarak, $\vec{A}$'nın -2*a* ile 2*a* arasında rastgele değerler ürettiği gözlemlenmektedir. Bu durumun avlanma modelindeki etkisi "*|A|<1 ise ava saldır, |A|>1 ise yeni av arayışına gir*" şeklinde olmaktadır (Mirjalili ve ark., 2014; Heidari ve Pahlavani, 2017).

GWO algoritmasının tüm aşamaları birlikte ele alındığında Şekil 4.7'deki gibi bir akış diyagramı elde edilmektedir.



Şekil 4.7. GWO algoritmasının akış diyagramı

Popülasyon tabanlı algoritmaların genelinde olduğu gibi GWO algoritmasında da ilk popülasyon rastgele oluşturulur. Algoritmanın parametre ayarlamaları yapıldıktan sonra arama uzayı sınırları içerisinde ilk popülasyon rassal bir şekilde elde edilir. Rastgele oluşturulan popülasyondaki bireylerin uygunluk değeri hesaplanır. En iyi uygunluk değerine sahip üç konum sırasıyla *alfa, beta* ve *delta* olarak işaretlenir. İlk popülasyon ve ilk lider takım bu şekilde elde edildikten sonra konum güncellemelerinin yapıldığı ana döngü başlatılır. Popülasyondaki tüm bireyler – lider takım dâhil – *alfa, beta* ve *delta* bireylerini referans alarak konum güncellemesi yapar. Konum güncellemesinde lider takımın etkisinin yanı sıra rastgele üretilen r_1 ve r_2 rastgele

sayılarının ve iteratif olarak doğrusal bir şekilde azalan *a* değerinin de etkisi mevcuttur. Popülasyondaki tüm bireyler için konum güncellemesi yapıldıktan ve uygunluk değerleri hesaplandıktan sonra bulunan en iyi konum *alfa, beta* ve *delta* bireylerinin konumları olarak işaretlenir. Ana döngüyü bitirecek olan şart sağlandığında *alfa* olarak işaretlenen konum algoritmanın çıktısı olarak verilir ve algoritma sonlandırılır. Çalışma mantığı olarak oldukça basit fakat etkili olan GWO algoritmasının sözde kodu Şekil 4.8 ile verilmiştir.

```
Başlangıç parametrelerini belirle
Başlangıç popülasyonunu rastgele oluştur
Çözümlerin uygunluk değerlerini hesapla
Popülasyondaki en iyi üç konumu alfa, beta ve delta olarak işaretle
while !(bitirme kriteri)
    α değerini güncelle //Bu değer doğrusal olarak 2'den 0'a doğru azalır
    for i=1 to N        //N popülasyondaki birey sayısı
        i.bireyin konumunu alfa, beta ve delta bireylerin konumuna göre güncelle
        Yeni konumun uygunluk değerini hesapla
    end                 // i=1 to N bitti
    Bu aşamaya kadar bulunan en iyi üç konumu lider takım olarak işaretle
end                     //Bitirme kriteri sağlanınca döngüyü bitir
alfa çözümünü çıktı olarak ver ve bitir.
```

Şekil 4.8. GWO algoritmasının sözde kodu

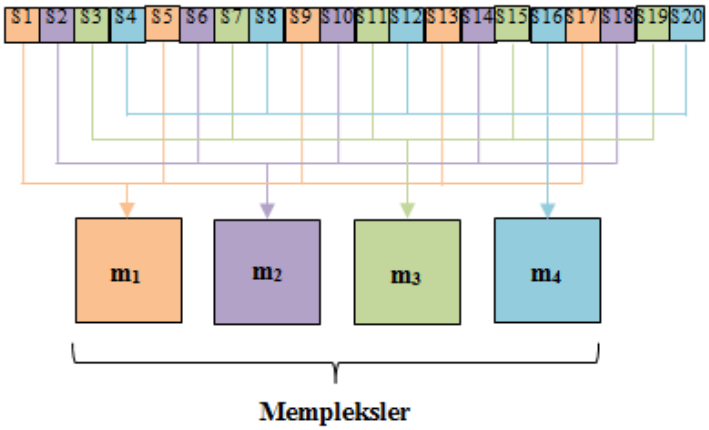
4.1.3. Uygulanan yaklaşım ve modifikasyonlar

Önerilen algoritmanın yakınsama ve çeşitlilik başarısını arttırmak ve algoritmanın çok amaçlı optimizasyon problemlerine etkin ve uyumlu bir şekilde uygulanmasını sağlamak amacıyla bazı modifikasyon ve yaklaşımlar kullanılmıştır.

4.1.3.1. Mempleks stratejisi

GWO algoritması gri kurtların hareketlerini model alarak önerilen doğa esinli bir algoritmadır. Modellenen algoritmada popülasyondaki bireyleri yöneten ve yönlendiren bir lider takımın (*alfa, beta* ve *delta*) olduğu bilinmektedir. Popülasyondaki bireyler lider takımın konumlarını referans alarak kendi konumlarını güncellemektedir. Bu durum popülasyondaki tüm bireylerin bir süre sonra aynı noktada arama yapmasına sebebiyet verebilmektedir. Bu durumu engellemek ve arama uzayını daha etkin bir şekilde araştırmak adına SFLA'nın mempleks stratejisi GWO algoritmasına uyarlanarak

kullanılmıştır. Denklem 4.1 kullanılarak mempleklere ayrılan kurt popülasyonunda her mempleksin kendi lider takımına sahip olması sağlanmıştır. Uyarlanan mempleks stratejisinde her bir mempleks için *alfa* birey arşivden (bastırılamayan çözümler kümesi) ikili turnuva yöntemi ile seçilmiştir. Mempleksin *beta* ve *delta* bireyleri olarak mempleksteki en iyi konuma sahip ilk iki birey alınmıştır. Bu şekilde popülasyondaki bireylerin belirli bir yöne topluca yönelmesinden ziyade arama uzayının daha detaylı aranması amaçlanmıştır. Şekil 4.9 ile Denklem 4.1'in uygulanması görsel olarak sunulmuştur.



Şekil 4.9. Mempleklere ayrılma stratejisi

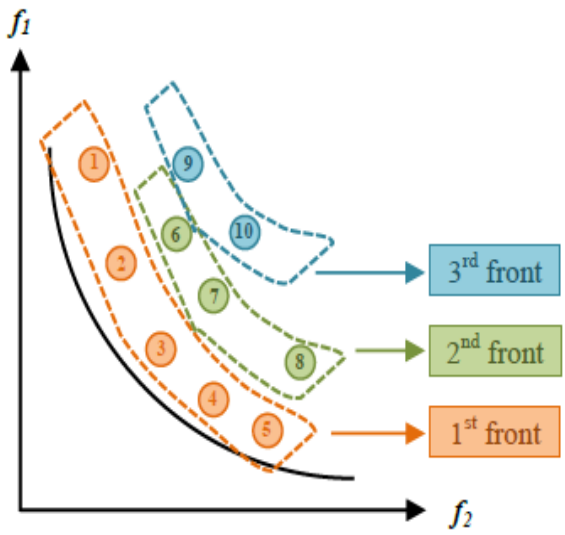
Elimizde büyüklüğü (*N*) 20 olan bir popülasyon olduğunu düşünelim. Mempleks sayısı *m*=4 olsun. İlk dört birey [S1-S4] sırasıyla ilk dört memplekse [m1-m4] atanır. Her memplekse sırasıyla birer birey atandıktan sonra mempleks numarası başa alınarak bireyler kaldığı yerden atanmaya devam eder. Bu durumda ikinci adım olarak sıradaki dört birey [S5-S8] sırasıyla mempleklere atanır. Bu süreç tüm bireyler mempleklere atanana kadar devam etmektedir. Mempleks stratejisinde görüldüğü üzere bireyler çözüm kalitesine göre dengeli bir şekilde gruplara dağılmaktadır. Mempleklere ayırma işleminin sorunsuz yapılabilmesi için mempleks sayısının popülasyon büyüklüğünü tam bölebilen bir olmasına dikkat edilmelidir.

4.1.3.2. Kalite sıralaması (Quality rank, QR)

Mempleks stratejisinin kullanılabilmesi öncelikli olarak popülasyondaki bireylerin uygunluk değerlerine göre sıralanmış olmasına bağlıdır. Tek amaçlı optimizasyon problemlerinde popülasyonu uygunluk değerine göre sıralamak kolay bir

işlemdir. Ancak çok amaçlı optimizasyon problemlerinde bu durum biraz daha karmaşık ve zor olabilmektedir.

Önceki bölümlerde çok amaçlı bir optimizasyon algoritmasının elde ettiği çözümlerin kalitesini ölçmek amacıyla hızlı bastırılamayan sıralama ve kalabalık mesafe yaklaşımlarının kullanıldığı belirtilmişti. Bu yaklaşımların kullanılması sonucu bir popülasyondaki çözümlerin kalite sıralamasına tabi tutulması mümkün olmaktadır. Popülasyondaki çözümler ilk aşamada hızlı bastırılamayan yaklaşım ile yüzeylere ayrılır. Daha sonra her yüzey kendi içerisinde değerlendirilerek yüzeydeki çözümlerin kalabalık mesafe değerleri hesaplanır. Bu aşamadan sonra ilk yüzeyden başlanarak çözümlere kalite sıralaması (*QR*) değerleri atanır. İlk yüzeyin uç noktalarındaki çözümlerin (kalabalık mesafe değeri sonsuz (*inf*) olan çözümler) *QR* değeri 1 olarak verilir. Daha sonra birinci yüzeydeki diğer çözümlere, kalabalık mesafesi değeri yüksek olandan başlanarak sayaç 1 arttırılarak *QR* değeri verilir. İlk yüzey bittikten sonra ikinci yüzeydeki çözümler için *QR* değerleri hesaplanır. İlk önce ikinci yüzeyin uç çözümlerine değer verilir. Bu çözümlere önceki (ilk) yüzeydeki son çözüme verilen *QR* değerin üstüne 1 eklenerek değer verilir. Geriye kalan çözümlere yine kalabalık mesafe değerine göre *QR* değeri atanır. Bu işlem tüm yüzeylerdeki çözümlere *QR* değeri verilene kadar devam ettirilir. Şekil 4.10 örnek bir popülasyondaki çözümler için yüzey, *CD* ve *QR* değerlerinin bir temsilini sunmaktadır (Deb ve ark., 2000; Coello ve Lechuga, 2002; Babalik ve ark., 2018).



#Çözüm	#Yüzey	*CD*	*QR*
1	1	Inf	1
5	1	Inf	1
2	1	0.85	2
3	1	0.80	3
4	1	0.72	4
6	2	Inf	5
8	2	Inf	5
7	2	0.75	6
9	3	Inf	7
10	3	Inf	7

Şekil 4.10. QR değerinin hesaplanması

4.1.3.3. Dinamik katsayı

Önceki bölümlerde GWO algoritmasında popülasyona hükmeden lider takımın üç bireyden (*alfa, beta* ve *delta*) oluştuğu belirtilmişti. Popülasyondaki tüm çözümler konum güncellemelerini yaparken lider takımın konumlarını referans olarak almaktadır. Bu üç bireyden en iyi çözüme sahip olanın *alfa*, ikinci en iyi çözüme sahip olanın *beta* ve üçüncü en iyi çözüme sahip olanın *delta* olduğu bilinmektedir. Buna rağmen Eşitlik 4.8'e bakıldığında lider takımdaki her bir bireyin konum güncellemesine olan katkı katsayısı (1/3) eşittir. Lider takım içerisindeki bireylerin konuma etkilerini çözüm kalitelerine göre belirleyip uygulamak etkilenen çözümü (*gama*) daha iyi bir konuma götürür mü sorusuna cevap bulmak adına *dinamik katsayısı* yaklaşımı önerilmiştir. Bu yaklaşıma göre en iyi çözüm kalitesine (*QR*) sahip *alfa*'nın katsayısı en yüksek, *beta*'nın da etki katsayısı *delta*'dan daha yüksek olmalıdır. Denklem 4.9 lider takımın konum güncellemesine olan katsayılarının dinamik olarak hesaplanması matematiksel olarak göstermektedir.

$$\begin{aligned} w_\alpha &= 0.5 \\ w_\beta &= (1 - w_\alpha) * \frac{1}{QR_\beta + QR_\delta} * QR_\delta \\ w_\delta &= (1 - w_\alpha) * \frac{1}{QR_\beta + QR_\delta} * QR_\beta \end{aligned} \tag{4.9}$$

Burada QR_β ve QR_δ sırasıyla *beta* ve *delta* çözümlerinin kalite değerini göstermektedir. Denklem 4.9 ile elde edilen katsayıların konum güncellemesine uygulanmasını matematiksel olarak göstermek adına Denklem 4.8 revize edilerek Denklem 4.10 elde edilmiştir.

$$\vec{X}^{(t+1)} = w_\alpha * \vec{X}_1 + w_\beta * \vec{X}_2 + w_\delta * \vec{X}_3 \tag{4.10}$$

Denklem 4.10 ile yapılan güncelleme sonucunda arşivden seçilen *alfa* çözümünün etkisinin lider takımdaki diğer iki çözümden daha fazla olması garanti edilmiştir. Ayrıca *beta* ve *delta* çözümlerinin de sahip oldukları *QR* değeri oranında konum güncellemeye olan etkileri revize edilmiştir. Bu yaklaşımın algoritmanın başarısına olan etkisi deneysel sonuçlar kısmında ele alınacaktır.

4.1.3.4. Çaprazlama operatörü

Yakınsama, çoğu optimizasyon algoritmasının sık sık karşılaştığı ve üstesinden gelinmesi gereken bir problemdir. Tez çalışmasında önerilen algoritmanın bu problemin üstesinden gelmesi için (Chen ve ark., 2014) çalışmasında kullanılan çaprazlama operatörü uygulanmıştır. Çaprazlama operatörünün uygulanması için *D* boyutlu bir referans (*R*) ve hedef (*T*) çözüme ihtiyaç duyulmaktadır. Önerilen algoritmada referans olarak, arşivden rastgele seçilen bir çözüm kullanılmaktadır. Denklem 4.11 çaprazlama operatörünün uygulanmasını matematiksel olarak göstermektedir (Chen ve ark., 2014; Özkış ve Babalık, 2017).

$$T_{i+1} = \begin{cases} T_i & , \quad if(rand \leq CR \; or \; i == s) \\ R_i & , \quad otherwise \end{cases} \tag{4.11}$$

Burada *i*=1, 2, ..., *D* olmak üzere; *rand* [0-1] aralığında rastgele üretilen bir sayıyı, *CR* çaprazlama oranı olarak isimlendirilen [0-1] aralığındaki sabit bir sayıyı ve *s* ise [1-*D*] arasında rastgele bir tam sayıyı temsil etmektedir.

4.1.3.5. Levy uçuşu (Levy flight, LF)

Çoğu optimizasyon algoritmasının karşılaştığı en genel ve büyük sorunlardan biri yerel en iyi çözüme ulaşıp orada takılı kalmaktır. Bu durumda söz konusu bireyin konumunu güncellemesi için bir müdahale gerçekleşmezse daha iyi bir çözümün yakalanması ihtimali oldukça düşmektedir. Literatüre bakıldığında bu konuda farklı yöntem ve yaklaşımların kullanıldığı görülebilir. Levy uçuşu, rassal bir adım dağılımı ile söz konusu probleme ışık tutabilecek alternatif bir yaklaşımdır. Levy uçuş stratejisinde, Levy dağılımından gelen sonsuz çeşitlilik içerisindeki rassal adımlar çözüme yeni bir yön kazandırabilmektedir (Yang ve Deb, 2009). Denklem 4.12, Levy uçuşunun uygulanması sonucu bir çözümün konum güncellemesini matematiksel olarak göstermektedir (Yang ve Deb, 2009; 2013; Haklı ve Uğuz, 2014).

$$X_i^{t+1} = X_i^t + \alpha \oplus \text{Levy}(\beta) \tag{4.12}$$

$$\text{Levy}(\beta) = step\ size = 0.01 * s * (X_i^t - Y^t) \tag{4.13}$$

Burada $0 < \beta < 2$ olmak üzere; α (> 0) problemin ölçekleriyle ilişkilendirilmesi gereken adım boyutunu, *0.01* değeri yürüyüş büyüklüğü faktöründen gelen değeri (Jensi ve Jiji, 2016), *Y* ise referans çözümü – bu çalışmada arşivden rastgele seçilen bastırılamayan bir çözüm – göstermektedir. Adım uzunluğu s ise Mantegna algoritmasından elde edilen Denklem 4.14 ile elde edilebilir (Jensi ve Jiji, 2016).

$$s = \frac{u}{|v|^{1/\beta}} \tag{4.14}$$

Burada *u* ve *v* değerleri Denklem 4.15 ile gösterilen normal dağılımlardan gelmektedir.

$$u \sim N(0, \sigma_u^2), \qquad v \sim N(0, \sigma_v^2) \tag{4.15}$$

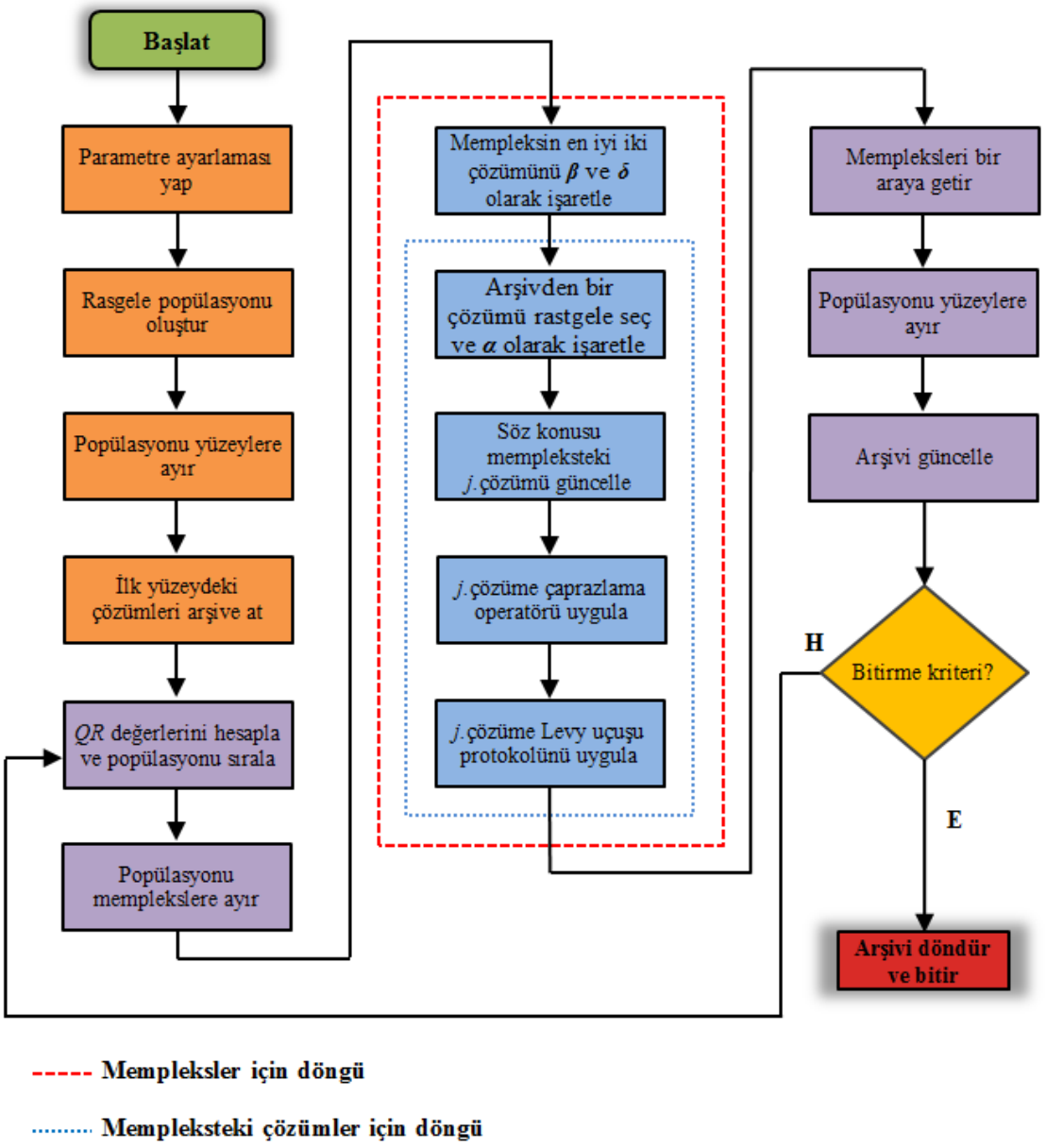
$\boldsymbol{\Gamma}$ standart Gama fonksiyonu olmak üzere σ_u ve σ_v değerleri Denklem 4.16 ile hesaplanmaktadır.

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma(1+\beta/2)\,\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \ \sigma_v = 1 \tag{4.16}$$

4.2. Önerilen MOSG Algoritması

Mirjalili ve arkadaşları tarafından önerilen gri kurt optimizasyonu tek amaçlı optimizasyon problemlerini çözmek amacıyla önerilmiş, popülasyon bazlı iteratif bir algoritmadır. Tez çalışması kapsamında, kurbağa sıçrama algoritmasının mempleks yapısı gri kurt optimizasyonu algoritmasına uygulanarak arama uzayının etkili bir şekilde taranması amaçlanmıştır. Mempleks yapısının, ancak sıralanmış bir popülasyona uygulanabildiği düşünüldüğünde çözümler arasındaki kalite sıralamasını elde etmek adına hızlı bastırılamayan sıralama ve kalabalık mesafe stratejileri uygulanmıştır. Bu iki strateji sonucunda elde edilen yüzey numaraları ve kalabalık mesafeleri, çözümlerin kalite sıralaması için kullanılmıştır. Bununla beraber algoritmanın performansını arttırmak amacıyla çaprazlama operatörü, Levy uçuşu ve dinamik katsayı modifikasyonları yapılmıştır.

Şekil 4.11'de önerilen algoritmanın akış diyagramı verilmiştir.



Şekil 4.11. Önerilen algoritmanın (MOSG) akış diyagramı

Önerilen algoritma ilk olarak kullanılan parametrelerin ayarlanması ile başlar. Bu aşamada algoritmada kullanılan popülasyon büyüklüğü, iterasyon sayısı, mempleks sayısı vb. gibi değişkenlerin değerleri atanır. Daha sonra arama uzayı sınırları içerisinde belirtilen popülasyon büyüklüğü kadar rastgele çözüm üretilerek ilk popülasyon oluşturulur. Arama uzayında elde edilen çözümlerin amaç uzayındaki karşılıkları, yani uygunluk değerleri hesaplanır. Popülasyonun yüzeylere ayrılıp bastırılamayan çözümler ile ilk arşivin oluşturulması için hızlı bastırılamayan yaklaşım uygulanır. Çözümler yüzeylere ayrılıp ilk arşiv elde edildikten sonra popülasyonun mempleklere ayrılması için sırasıyla kalabalık mesafeler ve ardından kalite sıralaması yaklaşımı uygulanır.

Popülasyondaki tüm çözümlerin *QR* değeri hesaplandıktan sonra sıralama işlemi gerçekleştirilir. Sıralama işleminden sonra popülasyon mempleklere ayrılır. Her bir mempleksteki çözümlerin güncellenmesi için gerekli döngüler başlatılır. İlk döngüde her mempleksin ele alınması sağlanırken, içerideki ikinci döngüde ise söz konusu mempleksteki çözümlerin işlenmesi yapılır. GWO algoritmasında konum güncelleme aşamasında lider takımın referans olarak alındığı bilinmektedir. Tez çalışmasında her mempleksin kendi lider takımına sahip olacağı belirtilmişti. Bu lider takımın oluşturulması *alfa* çözümün arşivden seçilmesi ve mempleksteki en iyi *QR* değerlerine sahip ilk iki çözümün sırasıyla *beta* ve *delta* olarak işaretlenmesi şeklinde olmaktadır. Bundan sonra çaprazlama operatörü ve ardından Levy uçuşu protokolü uygulanır. Bu şekilde her mempleks ve içerisindeki her çözüm için konum güncellemeleri yapıldıktan sonra mempleksler bir araya getirilerek yeni popülasyon elde edilir. Bu aşamadan sonra arşiv güncellenir. Arşiv güncellemesi, elde edilen yeni çözümler ile arşivin hızlı bastırılamayan sıralama prosedürüne sokularak bastırılamayan çözümlerin elde edilmesi ve arşive aktarılması şeklinde olmaktadır. Tez çalışmasında arşiv büyüklüğü popülasyon büyüklüğüne eşit olarak belirlenmiştir. Dolayısıyla arşiv güncellemesi yapıldığında arşivdeki çözüm sayısı maksimum sınırı aştığında *QR* değerlerine göre en iyi çözümlerin arşivde kalması sağlanmaktadır. Arşiv güncellemesi yapıldıktan sonra bitirme kriterinin sağlanıp sağlanmadığı kontrol edilir. Eğer bitirme kriteri henüz sağlanmadıysa popülasyon tekrar sıralanır ve mempleklere ayrılır. Bu adımlar bitirme kriteri sağlanana kadar devam ettirilir. Bitirme kriteri sağlandığında ise elde edilen arşiv (*PF*) algoritmanın çıktısı olarak verilir.

Önerilen algoritmanın sözde kodu Şekil 4.12'de verilmiştir.

```
1: Algoritma parametrelerini ayarla
2: Arama uzayı içerisinde ilk popülasyonu rastgele oluştur
3: Üretilen çözümlerin uygunluk değerlerini hesapla
4: Popülasyonu yüzeylere ayır //Hızlı bastırılamayan sıralama ile
5: İlk arşivi 1. yüzeydeki çözümler ile oluştur
6: t= N // N: popülasyon büyüklüğü, t değeri uygunluk fonksiyonu her çağrıldığında arttırılır
7: while t<MAXFEs //MAXFEs: Maximum fitness evaluation size
8:       Çözümlerin QR değerlerini hesapla //Kalabalık mesafe değerleri kullanılarak
9:       Popülasyonu sırala ve mempleklere ayır
10:      i=1;
11:      while i <= m // m: mempleks sayısı
12:            i.mempleksin en iyi iki çözümünü β ve δ olarak işaretle
13:            j = 1;
14:            while j <= n // n: her bir memplekseteki çözüm sayısı
15:                  Arşivden rastgele bir çözümü seç ve α (alfa) olarak işaretle
16:                  Sıradaki çözümün konumunu α, β ve δ kullanarak güncelle
17:                  Çaprazlama operatörünü uygula
18:                  Levy uçuşu protokolünü uygula
19:            end // j<=n
20:      end //i <= m
21:      Mempleksleri bir araya getir
22:      Popülasyonu yüzeylere ayır
23:      Arşivi güncelle
24: end // t<MAXFEs
25: return arşiv //Arşivdeki çözümler PF olarak çıktı verilir
```

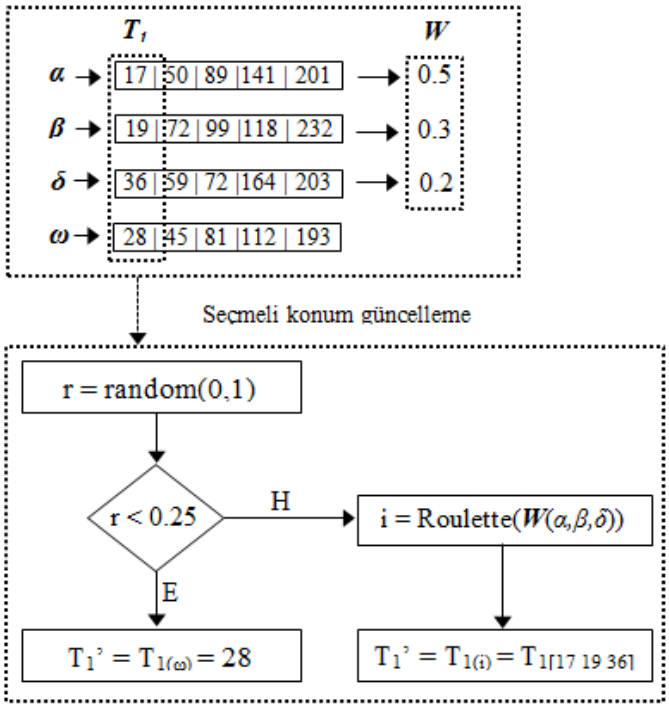
Şekil 4.12. Önerilen MOSG algoritmasının sözde kodu

4.3. Önerilen D-MOSG Algoritması

Önerilen MOSG algoritması çok amaçlı sürekli optimizasyon problemlerinin çözümüne uygun şekilde geliştirilmiştir. Çalışmanın bu aşamasında önerilen algoritmanın eşikleme ile görüntü segmentasyonuna uyarlanması amaçlanmıştır. Bu nedenle algoritmanın, kesikli (*discrete*) problemlere uygun bir hale getirilmesi gerekmektedir. Algoritmanın kesikli bir problem olan eşikleme ile segmentasyona uygulanabilmesi için yeni bir konum güncelleme stratejisi uygulanmıştır. Discrete-MOSG (D-MOSG) olarak adlandırılan algoritmada MOSG algoritmasının genel yapısına sadık kalınmış konum güncelleme kısmı *alfa*, *beta* ve *delta* bireylerin denetiminde olacak şekilde seçmeli bir bakış açısı getirilmiştir.

Yeni konum güncelleme stratejisi temelde iki aşamadan oluşmaktadır. İlk aşamada söz konusu bireyin (*gama*) konumunun söz konusu karar değişkeni için güncellenip güncellenmeyeceğine karar verilir. Konum güncellemenin her zaman iyi sonuç vermediği göz önünde bulundurularak, yeni stratejide *gama* bireyinin eski konumuna da bir şans verilmiştir. İlk aşamada konumun güncellenip güncellenmeyeceği

belirlenir. Eğer konum güncellenmeyecekse sonraki karar değişkenine geçilir ve ilk aşama bu karar değişkeni için çalışır. Ancak söz konusu karar değişkeni için konum güncelleme durumu mevcut ise ikinci aşamaya geçilir. İkinci aşamada konumu güncellenecek olan karar değişkenin hangi bireyi (*alfa, beta* veya *delta*) referans alarak değişime uğrayacağı belirlenir. Bu seçme işlemi Rulet tekerleği seçimi ile yapılmaktadır. D-MOSG algoritmasında da *alfa, beta* ve *delta* bireylerinin seçimi MOSG ile aynı şekilde olmaktadır. Lider takımdaki bu üç bireyin Rulet tekerleği için seçilme olasılıkları değerleri ise Denklem 4.9 ile hesaplanan dinamik katsayılarıdır. Rulet tekerleği ile yapılan seçim sonucunda seçilen bireyin söz konusu karar değişkeni değeri ile *gama* bireyin karar değişkeni güncellenir. Yeni güncelleme stratejisinin bu iki aşaması *gama* bireyinin her bir karar değişkeni için uygulanarak yeni konum elde edilir. Şekil 4.13 ile yeni konum güncelleme stratejisinin görsel sunumu verilmiştir.



Şekil 4.13. Yeni konum güncelleme stratejisi

4.3.1. Eşikleme ile görüntü segmentasyonu

Görüntü segmentasyonu bir görüntü işleme uygulamasının en önemli aşamalarından biridir. Segmentasyon, görüntü işleme başarısını doğrudan etkileyen bir aşama olduğundan bu konu üzerinde dikkatle durulması gerekmektedir. Konunun önem düzeyi yüksek olduğundan araştırmacılar segmentasyon aşamasında uygulanmak üzere

birçok yöntem ve yaklaşım geliştirmiştir. Eşikleme, görüntü segmentasyonu için kullanılan yöntemler arasında, basit ve başarılı olması açısından, oldukça rağbet gören bir tekniktir (Bhandari ve ark., 2015; Karakoyun ve ark., 2017).

Eşikleme metodu, görüntünün histogram değerlerini kullanarak, görüntüyü birbiri ile çakışmayan bölgelere ayırmayı hedeflemektedir. Eşikleme metodunu, eşik sayısına göre iki ana kategoriye ayırmak mümkündür: tek-seviye (*bi-level*) ve çok-seviye (*multi-level*) eşikleme. Tek-seviye eşiklemede görüntüyü iki gruba ayıran tek eşik değerine ihtiyaç duyulur ve sonuçta siyah-beyaz bir görüntü elde edilir. Çok-seviye eşiklemede ise görüntüyü bağımsız bölgelere ayıran birden fazla eşik değerine ihtiyaç vardır (Karakoyun ve ark., 2017; Bhandari ve ark., 2019).

L (*0<L<255*) görüntünün sahip olduğu gri seviye değeri ve *g(x, y)* görüntünün *x* ve *y* koordinatlarında bulunan pikselin gri seviye değeri olmak üzere Denklem 4.17 bir görüntünün eşikleme ile segmente edilmesini matematiksel olarak sunmaktadır.

$$\begin{aligned} C_0 &= \{\, 0 \;\le g(x,y) < T_1 \;\} \\ C_1 &= \{\, T_1 \le g(x,y) < T_2 \;\} \\ C_i &= \{\, T_i \le g(x,y) < T_{i+1} \,\} \\ C_n &= \{\, T_n \le g(x,y) < L \;\} \end{aligned} \tag{4.17}$$

Burada $C=C_0, C_1, ..., C_n$ eşikleme sonucunda piksellerin dahil olacağı sınıfları, *n* eşik sayısını ve $T=T_1, T_2, ..., T_n$ görüntüyü segmente edecek eşik değerlerini temsil etmektedir.

Eşikleme metodu görüntü segmentasyonunun basit ve kullanışlı bir yaklaşımı olsa da eşik sayısı arttıkça görüntüyü segmente edecek en iyi eşik değerlerinin belirlenmesi zorlaşmaktadır. Bu problemin üstesinden gelmek amacıyla araştırmacılar tarafından birçok yöntem önerilmiştir. Bu yöntemler içerisinde Otsu ve Kapur basit ve etkili olmaları açısından yoğun bir şekilde başvurulan yöntemlerin başında gelmektedir. Tez çalışması kapsamında eşikleme için bu iki yöntem tercih edilmiştir.

Otsu Metodu: Eşikleme yöntemleri içerisinde en popüler olan yaklaşımdır. Otsu yaklaşımı *bimodal* (iki tepeli) histogram dağılımına sahip görüntülerde başarılı sonuçlar üretirken kesişen bölgelere sahip görüntülerde zorlanmaktadır. Otsu yaklaşımında temel amaç seçilen eşik değerlerine göre elde edilen piksel sınıflarının içerisindeki çeşitliliğin minimum (veya sınıflar arasındaki çeşitliliğin maksimum) olmasıdır (Otsu, 1979; Tuba ve ark., 2017; Satapathy ve ark., 2018). *I* segmente edilecek görüntü, *L* gri seviye

değeri, *T* eşik değerleri vektörü ve *n* eşik sayısı olmak üzere; Otsu metodu, Denklem 4.18 ile matematiksel olarak verilen sınıflar arası çeşitliliği (*f(T)*) maksimize etmeyi hedeflemektedir.

$$f(T) = \sum_{i=0}^{n} \sigma_i \tag{4.18}$$

Her bir sınıf için varyans (σ) değeri Denklem 4.19 ile hesaplanmaktadır.

$$\begin{aligned}
\sigma_0 &= \omega_0 \left(\sum_{i=0}^{T_0-1} \frac{ip_i}{\omega_0} - \sum_{i=0}^{L-1} ip_i \right)^2 \\
\sigma_1 &= \omega_1 \left(\sum_{i=T_0}^{T_1-1} \frac{ip_i}{\omega_1} - \sum_{i=0}^{L-1} ip_i \right)^2 \\
&\vdots \\
\sigma_n &= \omega_n \left(\sum_{i=T_n-1}^{L-1} \frac{ip_i}{\omega_n} - \sum_{i=0}^{L-1} ip_i \right)^2
\end{aligned} \tag{4.19}$$

Burada ω sınıfın kümülatif olasılığı ve p_i, *i.*pikselin görüntünün tamamındaki bulunma dağılımını göstermektedir. Bu değerler Denklem 4.20 ve 4.21 ile elde edilmektedir.

$$\omega_0 = \sum_{i=0}^{T_0-1} p_i, \quad \omega_1 = \sum_{i=T_0}^{T_1-1} p_i \ \ldots \ \omega_n = \sum_{i=T_n-1}^{L-1} p_i \tag{4.20}$$

$$p_i = \frac{n_i}{N} \tag{4.21}$$

Verilen eşitliklerde n_i, *i.*pikselin görüntüde geçme sayısını ve *N* görüntüdeki toplam piksel sayısını göstermektedir.

Kapur Entropisi: Entropi tabanlı ve oldukça başarılı bir eşikleme yöntemidir. Kapur metodunda temel hedef, benzer ve karakteristik piksel grupları elde etmektir. Bu amaçla segmente edilmiş bölgeler arasında entropiyi maksimize etmeye çalışmaktadır. *I* segmente edilecek *N* piksele sahip bir görüntüyü, *L* görüntünün gri seviye değerini, n_i *i*

değerindeki piksellerin görüntüdeki âdetini, $p_i = n_i/N$ *i.* pikselin görüntüdeki dağılım oranını ve $T=T_1, T_2, ..., T_n$ görüntüyü segmente edecek eşik değerlerini göstermek üzere; Kapur metodu, Denklem 4.22 ile verilen entropi değerini maksimize etmeyi hedeflemektedir (Bhandari ve ark., 2015; Tuba ve ark., 2017).

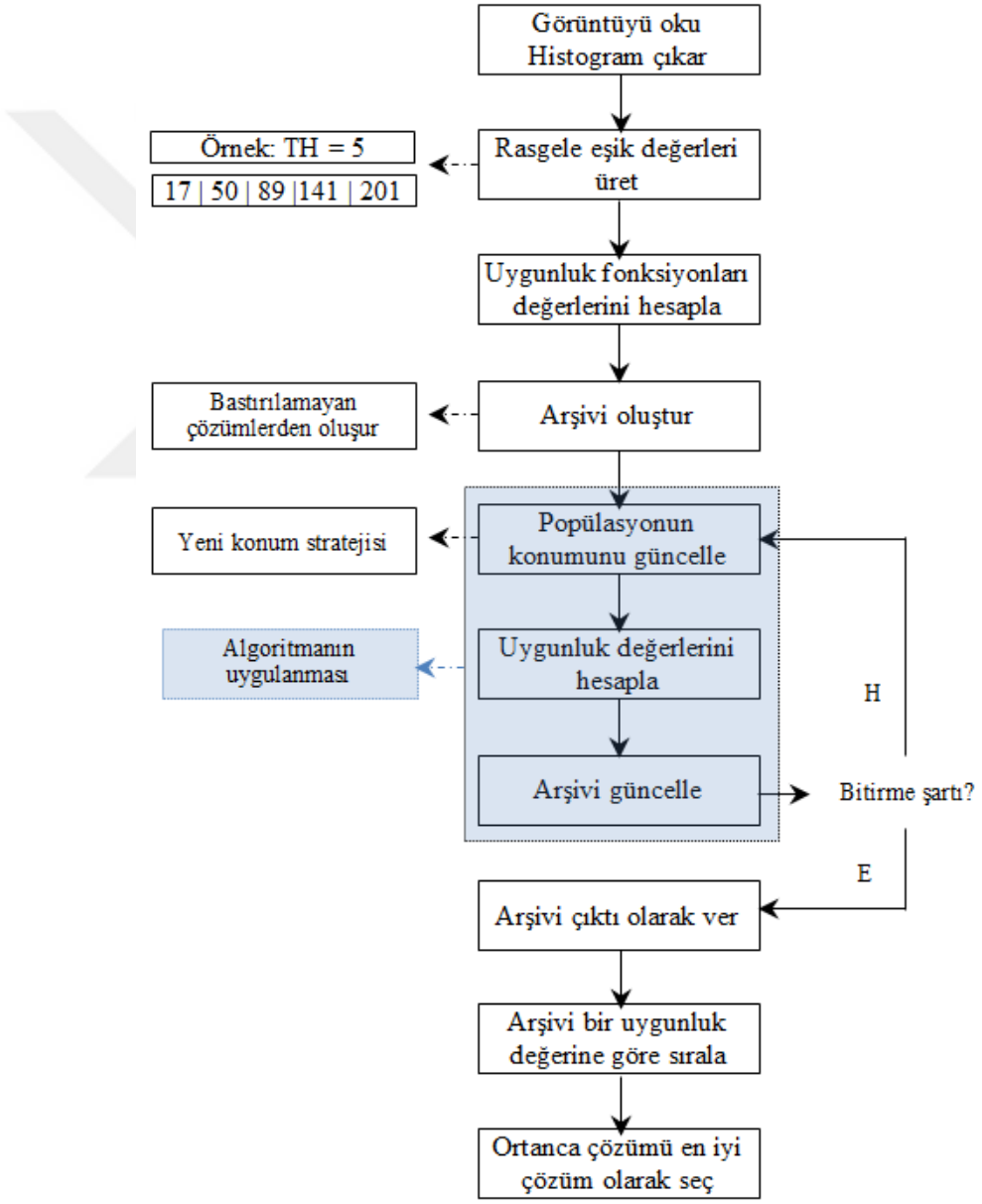
$$f(T) = \sum_{i=0}^{n} H_i \tag{4.22}$$

$$\begin{aligned} H_0 &= -\sum_{i=0}^{T_0-1} \frac{p_i}{\omega_0} \, ln \frac{p_i}{\omega_0}, \quad \omega_0 = \sum_{i=0}^{T_0-1} p_i \\ H_1 &= -\sum_{i=T_0}^{T_1-1} \frac{p_i}{\omega_1} \, ln \frac{p_i}{\omega_1}, \quad \omega_1 = \sum_{i=T_0}^{T_1-1} p_i \\ &\vdots \\ H_n &= -\sum_{i=T_n-1}^{L-1} \frac{p_i}{\omega_n} \, ln \frac{p_i}{\omega_n}, \quad \omega_n = \sum_{i=T_n-1}^{L-1} p_i \end{aligned} \tag{4.23}$$

4.3.2. Çok amaçlı eşikleme ve D-MOSG algoritması

Görüntü işleme uygulamalarının önemli bir aşaması olan segmentasyon için kullanılan eşikleme, görüntüyü homojen bölgelere ayırarak sonraki aşamalar için bir ön hazırlık sağlamaktadır. Eşiklemede temel amaç görüntüyü en iyi segmente edecek eşik değerlerinin tespit edilmesidir. Tek amaçlı bir optimizasyon algoritması seçilen uygunluk fonksiyonunu (Otsu, Kapur, Renyi, Tsalli vb.) maksimize veya minimize ederek en iyi eşik değerlerini bulmaya çalışmaktadır. Seçilen her uygunluk fonksiyonun kendine özgü bir karakteristiği vardır. Bu nedenle ne kadar başarılı olursa olsun bir metodun tüm görüntü ve eşik seviyeleri için iyi sonuç üretmesi beklenemez. Tez çalışmasının bu kısmında eşikleme yöntemi çok amaçlı olarak ele alınmış ve Otsu ile Kapur metotları birlikte optimize edilmeye çalışılmıştır. Bunun sonucunda uygunluk fonksiyonu $F = [F_{Otsu}\ F_{Kapur}]$ olarak güncellenmiştir. Önerilen D-MOSG algoritması, Otsu ve Kapur uygunluk fonksiyonları arasında bir dengeleme sağlayarak görüntüyü en iyi şekilde segmente edecek eşik değerlerini bulmayı hedeflemiştir. Çok amaçlı optimizasyon problemlerinde algoritmalar tek çözüm yerine bir çözüm kümesi

sunmaktadır. Çıktı olarak sunulan bu çözüm kümesi içerisinde bastırılamayan, baskın çözümler bulunmaktadır. D-MOSG algoritmasının performansı tek amaçlı optimizasyon problemlerinin performansı ile kıyaslandığından, bu çözümler içerisinden bir çözümün seçilerek sonuç karşılaştırmasına tabi tutulması gerekmektedir. Tez kapsamında bu seçim işlemi arşiv elde edildikten sonra arşivin herhangi bir uygunluk değerine göre sıralanması ve ortanca çözümün alınması şeklinde olmuştur. Şekil 4.14'te önerilen D-MOSG algoritmasının eşikleme problemine uygulanmasının genel adımları sunulmuştur.



Şekil 4.14. D-MOSG algoritmasının eşiklemeye uygulanması

5. DENEYSEL SONUÇLAR

Bu bölümde önerilen algoritmanın kullanılan problem setleri üzerindeki performansı ele alınmıştır. Önerilen algoritma her bir problem seti için farklı algoritmalar ve metrikler ile kıyaslanmıştır. Bu nedenle bu sonuçlar ayrı başlıklar altında verilmiştir.

5.1. Parametre Analizi

Önerilen algoritmanın ilk test ortamı olan problem seti-1 fonksiyonları kullanılarak parametre analizi yapılmıştır. Tüm algoritmalar için ortak olan popülasyon büyüklüğü ve iterasyon sayısı parametreleri sabit seçilmiş ve tüm algoritmalarda aynı değerler ile kullanılmıştır. Ancak algoritmanın kendine özgü parametreleri için detaylı bir parametre analizi gerçekleştirilmiştir. Önerilen MOSG algoritmasının kendine özgü parametreleri Çizelge 5.1'de verilmiştir.

Çizelge 5.1. Önerilen algoritmanın parametreleri

Parametre	**Simge**
Arşiv büyüklüğü	*ArcMax*
Mempleks sayısı	*m*
Çaprazlama oranı	*CR*
Beta (Levy uçuşu için)	β
Maksimum tekrar (Levy uçuşu için)	*LFMax*

Arşiv büyüklüğü parametresi algoritmanın çalışma esnasında bastırılamayan çözümlerin tutulması için kullanılan *arşiv* dizisinin boyutunu tutmaktadır. Çok amaçlı optimizasyon algoritmalarının genelinde bu değer popülasyon büyüklüğü ile aynı olacak şekilde ayarlanmaktadır. Tez çalışması kapsamında önerilen algoritma için de aynı kabul yapılmış ve popülasyon büyüklüğüne eşit olarak seçilmiştir.

Mempleks sayısı algoritmanın çalışması esnasında popülasyon çözüm uzayına kaliteli bir dağılım sağlaması açısından önemli bir parametredir. Bu parametrenin seçilmesinde dikkat edilmesi gereken husus, seçilen değerin popülasyon büyüklüğünü tam bölen bir değer olması gerektiğidir. Mempleks sayısı için analizi yapılacak değerler popülasyon büyüklüğünün 100 olacağı düşünülerek 5, 10, 20, 25 ve 50 olarak seçilmiştir. Önerilen algoritma belirlenen her bir mempleks sayısı değeri için problem seti-1'deki 36 fonksiyon üzerinde 50 tekrar ile çalıştırılmıştır. Bu tekrarların sonuçları dört farklı (*HV, IGD, Spread, Epsilon*) metrik ile incelenmiştir. Metrikler üzerinde elde

edilen Friedman (Friedman, 1937) istatistiksel test sonuçları Çizelge 5.2'de sunulmuştur. En iyi sonuçlar **kalın (bold)** olarak işaretlenmiştir. Çizelge 5.2'deki sonuçlara bakıldığında mempleks sayısı 5 olduğunda önerilen algoritmanın üç metrik (IGD, Spread ve Epsilon) üzerinde daha iyi sonuçlar ürettiği görülmektedir. Elde edilen sonuçlar dikkate alınarak mempleks sayısı parametresinin değeri 5 olarak belirlenmiştir.

Çizelge 5.2. Mempleks sayısı parametresi analiz sonuçları

Metrik/#Mem.	**5**	**10**	**20**	**25**	**50**
HV	3.21	2.91	**3.53**	3.24	2.12
IGD	**2.38**	3.15	2.53	3.38	3.56
Spread	**2.44**	3.12	2.97	3.27	3.21
Epsilon	**2.21**	3.35	2.77	3.24	3.44

Analizi yapılması gereken bir diğer parametre, çaprazlama operatöründe (Denklem 4.11) kullanılan çaprazlama oranı parametresidir. Parametre için analiz edilecek değerler [0.4, 0.5, 0.6, 0.7, 0.8] olarak seçilmiştir. Bu analiz aşamasında da önerilen algoritma belirlenen her bir değer için 50 kere çalışmıştır. Algoritmanın problem seti-1 üzerinde 50 tekrar sonucu elde ettiği Friedman istatistiksel sonuçları Çizelge 5.3 ile verilmiştir. Çaprazlama oranı için yapılan analiz sonuçlarında her dört metrik için de en iyi değeri elde etmesi sebebiyle bu parametre için 0.4 değeri seçilmiştir.

Çizelge 5.3. Çaprazlama oranı parametresi analiz sonuçları

Metrik/CR	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**
HV	**4.41**	3.82	3.29	2.07	1.4
IGD	**1.94**	2.03	2.91	3.79	4.32
Spread	**2.18**	2.62	3.03	3.35	3.82
Epsilon	**2.03**	2.18	2.56	3.85	4.38

Beta parametresi Levy uçuşu stratejisinin önemli bir parametresidir. Literatürdeki çalışmalara bakıldığında araştırmacıların – üzerinde çalışılan problem türüne ve/veya uygulanan algoritmaya göre – bu parametre için [0, 2] aralığında farklı değerler kullandığı görülmektedir. Önerilen algoritma için bu parametrenin optimum değerini bulmak adına yapılan analizde [0.1, 2.0] aralığında (0.1 adım büyüklüğünde) değerler için analiz yapılmıştır. Bu parametre için belirlenen her bir değer için algoritma 50 tekrar ile problem seti-1'deki fonksiyonlar üzerinde çalıştırılmıştır. Bu çalıştırmalar sonucunda elde edilen Friedman istatistiksel test sonuçları Çizelge 5.4 ile verilmiştir. Bu analiz için elde edilen sonuçlara bakıldığında beta parametresinin 1.7 ve 1.8 değerleri

birer metrikte daha iyi sonuçlar gösterirken 1.4 değerinin iki metrik üzerinde daha başarılı sonuçlar ürettiği görülmektedir. Bu nedenle beta parametresi için 1.4 değeri seçilmiştir.

Çizelge 5.4. Beta parametresi analiz sonuçları

Metrik/*β*	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**1.7**	**1.8**	**1.9**	**2**
HV	11.5	11.1	10.8	9.2	10.9	9.5	10.9	10.9	9.2	11.2	8.8	10.1	11.7	**12.1**	10.7	9.9	10.9	11.8	9.0	9.9
IGD	12.2	9.7	10.6	11.3	11.3	11.0	9.2	9.4	10.9	10.3	10.5	9.5	11.5	10.8	11.1	9.7	**8.9**	9.9	11.9	10.2
Spread	11.4	9.9	8.7	12.1	10.7	10.6	11.3	10.2	10.9	11.9	9.6	12.4	10.5	**8.0**	11.3	8.7	11.0	10.0	10.4	10.6
Epsilon	10.8	9.6	10.2	10.4	10.9	11.9	10.5	10.2	10.8	10.4	10.3	10.7	11.9	9.8	10.7	9.7	9.4	**8.8**	13.2	10.2

Levy uçuşu stratejisinin maksimum tekrar parametresi, önerilen algoritmanın analiz edilmesi gereken son parametresidir. Bu parametre söz konusu çözüme Levy uçuşu uygulanmadan önce (konumunu daha iyi bir noktaya taşımak adına) kaç defa şans verileceğini belirleyen parametredir. Söz konusu çözüm için, bu parametre adedince iterasyona rağmen konumunda herhangi bir iyileşme sağlayamazsa Levy uçuşu uygulanarak daha iyi bir konuma ulaşması amaçlanmaktadır. Literatür üzerinde yapılan incelemelerde araştırmacıların bu parametre için genellikle 10 ve 15 değerini kullandıkları gözlemlenmiştir. Bu parametre için yapılan analizde 10, 15, 20 ve 25 değerleri ele alınmıştır. Algoritmanın problem seti-1 üzerindeki 50 tekrar ile çalışması sonucu elde edilen Friedman istatistiksel test sonuçları Çizelge 5.5 ile verilmiştir. Elde edilen sonuçlara göre parametrenin 10 ve 25 değerleri birer metrikte daha iyi sonuçlar üretirken 15 değeri iki metrik için daha başarılı sonuçlar ürettiği görülmüştür. Deneysel sonuçlara göre bu parametre için 15 değerinin kullanılmasına karar verilmiştir.

Çizelge 5.5. Levy uçuşu maksimum tekrar parametresi analiz sonuçları

Metrik/#LFMax	**10**	**15**	**20**	**25**
HV	**2.71**	2.68	2.06	2.56
IGD	2.38	**2.27**	2.94	2.41
Spread	2.65	2.53	2.44	**2.38**
Epsilon	2.44	**2.38**	2.65	2.53

Önerilen algoritmanın parametre analizi sonucunda elde edilen sonuçlara göre kullanılan parametrelerin değerleri Çizelge 5.6'da verilmiştir.

Çizelge 5.6. Önerilen algoritmanın parametreleri

Parametre	**Simge**	**Değer**
Popülasyon büyüklüğü	*P*	100
Arşiv büyüklüğü	*ArcMax*	100
Uygunluk fonksiyonu çalıştırma sayısı	*MAXFes*	25000
Mempleks sayısı	*m*	5
Çaprazlama oranı	*CR*	0.4
Beta (Levy uçuşu için)	β	1.4
Maksimum tekrar (Levy uçuşu için)	*LFMax*	15

Bu bölümde önerilen algoritmanın parametreleri için en uygun değerlerin bulunması amaçlanmış ve bu doğrultuda farklı analizler yapılmıştır. Bununla beraber konum güncelleme aşamasında popülasyondaki bireylere önderlik eden lider takımın (*alfa, beta, delta*) etkileri de incelenmiştir. Önceki bölümlerde anlatıldığı üzere, GWO algoritmasının saf halinde (Denklem 4.8) lider takımdaki her bireyin etkisinin eşit olduğu bilinmektedir. Tez çalışması kapsamında lider takımdaki bireylerin konum güncellemesine olan etkileri sahip oldukları çözüm kalitelerine göre belirlenerek revize edilmiştir. Yapılan bu değişikliğin algoritma üzerindeki etkisini görmek adına lider takımın konum güncellemesine olan etkileri farklı durumlarda test edilmiştir. İlk durum olarak algoritmanın saf halinde olduğu gibi lider takımın eşit etkiye sahip olması, ikinci durum önerilen algoritmadaki dinamik katsayı ve üçüncü durum olarak lider takımdaki her bir bireyin etkisinin rastgele bir sayı ile belirlenmesi şeklinde yeni bir yaklaşım ele alınmıştır. Üçüncü durumda lider takımdaki bireylerin etki katsayı olarak [0-1] aralığında rastgele üretilen sayı kullanılmaktadır. Lider takımdaki bireylerin konum güncellemesine ne kadar oranda etki edeceği amacıyla yapılan bu analizin sonuçları Çizelge 5.7'de verilmiştir. Algoritma her üç durum için problem seti-1'deki fonksiyonlar üzerinde 50 tekrar ile çalışmıştır. Çizelge 5.7'de 50 tekrarın HV metriği üzerinde elde edilen ortalama sonuçları verilmiştir. Çizelge 5.7'nin son satırı algoritmanın her bir durum için uygulanan fonksiyonlarda kaç defa ortalama 1. ve 2. olduğunu göstermektedir. Elde edilen sonuçlara bakıldığında önerilen dinamik katsayı yaklaşımının 36 fonksiyon üzerinde 11 defa en iyi ve 4 defa ikinci en iyi ortalama değerler ile diğer yaklaşımlardan daha iyi olduğu görülmektedir.

Çizelge 5.7. HV metriği için lider takım katsayı analizi sonuçları

Problem	Dinamik	Eşit(1/3)	Rastgele
ZDT1	6.62E-01	6.62E-01	6.62E-01
ZDT2	3.29E-01	3.29E-01	3.29E-01
ZDT3	5.16E-01	5.16E-01	5.16E-01
ZDT4	6.56E-01	6.54E-01	**6.62E-01**
ZDT6	4.01E-01	4.01E-01	4.01E-01
WFG1	1.42E-01	1.41E-01	**1.46E-01**
WFG2	**5.53E-01**	5.52E-01	5.52E-01
WFG3	4.91E-01	4.91E-01	4.91E-01
WFG4	2.08E-01	2.08E-01	2.08E-01
WFG5	**2.21E-01**	**2.21E-01**	2.20E-01
WFG6	2.06E-01	2.06E-01	2.06E-01
WFG7	2.08E-01	2.08E-01	2.08E-01
WFG9	2.24E-01	2.24E-01	2.24E-01
DTLZ1	**8.81E-02**	8.23E-02	7.82E-02
DTLZ2	3.81E-01	3.81E-01	3.81E-01
DTLZ4	3.83E-01	3.82E-01	**3.85E-01**
DTLZ5	**9.32E-02**	9.31E-02	9.31E-02
DTLZ6	9.50E-02	9.50E-02	9.50E-02
DTLZ7	2.88E-01	**2.92E-01**	2.88E-01
LZ09_F1	**6.37E-01**	6.34E-01	6.36E-01
LZ09_F2	5.14E-01	**5.17E-01**	**5.17E-01**
LZ09_F3	**5.62E-01**	5.58E-01	5.60E-01
LZ09_F4	**5.73E-01**	5.72E-01	**5.73E-01**
LZ09_F5	**5.82E-01**	5.80E-01	5.81E-01
LZ09_F6	**2.13E-01**	2.06E-01	2.03E-01
LZ09_F7	**2.27E-01**	2.26E-01	1.91E-01
LZ09_F8	1.45E-01	**1.83E-01**	1.70E-01
LZ09_F9	1.95E-01	**2.00E-01**	1.96E-01
Fonseca	3.11E-01	3.11E-01	3.11E-01
Kursawe	4.01E-01	4.01E-01	4.01E-01
Schaffer	8.30E-01	8.30E-01	8.30E-01
Viennet2	**9.23E-01**	**9.23E-01**	9.22E-01
Viennet3	8.35E-01	8.35E-01	8.35E-01
Poloni	9.13E-01	9.13E-01	9.13E-01
OKA1	5.46E-01	5.46E-01	5.46E-01
OKA2	4.17E-02	4.17E-02	4.17E-02
# $1^{st}/2^{nd}$	**11/4**	**6/5**	**5/8**

5.2. Deneyler

Bu bölümde önerilen algoritmanın üç farklı problem seti üzerindeki sonuçları incelenmiştir. Önerilen algoritmanın performansı her problem seti için farklı algoritmaların performansı ile kıyaslanmıştır.

5.2.1. Problem seti -1 deney sonuçları

36 farklı fonksiyondan oluşan bu problem setinin özellikleri önceki bölümlerde (Çizelge 3.1 ve Çizelge 3.2) verilmişti. Bu problem seti üzerinde önerilen MOSG algoritmasının performansı altı farklı algoritmanın performansı ile kıyaslanmıştır. Bu

algoritmalardan dört tanesi (NSGA-II, IBEA, MOCell, MOEA/D) çok amaçlı optimizasyon problemleri için geliştirilmiş geleneksel algoritmalar iken diğer iki algoritma (MOAAA, MOVS) orijinalinde tek amaçlı olup sonradan çok amaçlı optimizasyon problemlerine uygulanacak şekilde düzenlenen doğa esinli algoritmalardır. Tüm algoritmalar için ortak olan popülasyon büyüklüğü ve uygunluk fonksiyonu çağrılma sayıları sırasıyla 100 ve 25000 olarak belirlenmiştir. MOSG algoritmasının kendine özgü parametre değerleri Çizelge 5.6'da verilmiştir. Bu problem seti üzerinde gerçekleştirilen deneyler *jMetal* platformu üzerinden yapılmış olup karşılaştırma yapılan algoritmaların (NSGA-II, IBEA, MOCell ve MOEA/D algoritmaları için (Durillo ve Nebro, 2011)) parametre değerleri platformu hazırlayan araştırmacılar tarafından belirlenen değerler olarak kullanılmıştır. Diğer iki algoritma (MOAAA (Babalik ve ark., 2018) ve MOVS (Özkış ve Babalık, 2017)) için kullanılan parametre değerleri ise algoritmaların önerildiği çalışmalardan alınmıştır.

Algoritmaların performans karşılaştırılması farklı özellikler açısından kıyas yapan, dört metrik (HV, IGD, Spread, Epsilon) üzerinden yapılmıştır. Daha önceki bölümlerde belirtildiği üzere bu metrikler algoritmaların yakınsama ve çözüm çeşitlilikleri açısından başarılarını değerlendirmektedir. HV metriğinde daha büyük değerin başarılı sayıldığı ve diğer üç metrik için daha küçük değerlerin başarılı sayıldığı önceki bölümlerde belirtilmişti. Algoritmalar problem setindeki her bir fonksiyon için 100 bağımsız tekrar ile çalıştırılmış ve problem seti-1 için yapılan tüm analizler bu tekrarlar üzerinden yapılmıştır. Algoritmaların ürettiği sonuçların; HV metriği için karşılaştırması Çizelge 5.8'de, IGD metriği için karşılaştırması Çizelge 5.9'da, Spread metriği için karşılaştırması Çizelge 5.10'da ve Epsilon metriği için karşılaştırması Çizelge 5.11'de verilmiştir. Bu çizelgelerde algoritmaların 100 tekrar sonucu elde ettiği sonuçların ortalama ve standart sapma değerleri verilmiştir. Çizelgelerde en iyi ortalama değer koyu gri ile işaretlenirken ikinci en iyi ortalama değer ise açık gri ile işaretlenmiştir.

Çizelge 5.8. Problem Seti-1: HV metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II	**IBEA**	**MOCell**	**MOEA/D**	**MOAAA**	**MOVS**	**MOSG**
	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.
ZDT1	6.59E-01 2.8E-04	6.62E-01 9.0E-05	6.61E-01 2.3E-04	6.40E-01 7.6E-03	6.58E-01 3.5E-04	6.59E-01 2.4E-04	6.62E-01 3.1E-05
ZDT2	3.26E-01 3.1E-04	3.27E-01 1.4E-04	3.28E-01 4.1E-04	3.10E-01 6.8E-03	3.25E-01 3.7E-04	3.26E-01 3.2E-04	3.29E-01 2.2E-05
ZDT3	5.15E-01 1.6E-04	5.09E-01 6.2E-03	5.12E-01 8.8E-03	4.43E-01 2.5E-02	5.15E-01 1.5E-04	5.15E-01 2.9E-04	5.16E-01 3.6E-05
ZDT4	6.55E-01 3.4E-03	2.36E-01 9.3E-02	6.57E-01 2.7E-03	3.01E-01 1.8E-01	5.69E-01 1.0E-01	5.72E-02 9.2E-02	6.62E-01 2.1E-05
ZDT6	3.88E-01 1.5E-03	3.96E-01 5.5E-04	3.95E-01 7.4E-04	4.01E-01 9.4E-04	3.98E-01 3.5E-04	3.99E-01 3.3E-04	4.01E-01 1.2E-05
WFG1	5.20E-01 9.3E-02	4.73E-01 1.1E-01	3.71E-01 9.2E-02	3.22E-01 8.7E-02	6.17E-01 2.0E-02	4.95E-01 6.2E-02	1.41E-01 8.4E-03
WFG2	5.54E-01 2.4E-03	5.50E-01 8.3E-04	5.52E-01 7.0E-03	5.55E-01 3.4E-04	5.57E-01 1.1E-04	5.57E-01 1.1E-04	5.53E-01 9.0E-04
WFG3	4.92E-01 6.6E-04	4.94E-01 3.7E-04	4.94E-01 5.5E-04	4.93E-01 1.3E-04	4.92E-01 3.4E-04	4.93E-01 3.1E-04	4.91E-01 6.7E-04
WFG4	2.09E-01 4.0E-04	2.09E-01 1.8E-04	2.10E-01 2.1E-04	2.04E-01 1.8E-03	2.09E-01 4.4E-04	2.09E-01 2.7E-04	2.08E-01 6.2E-04
WFG5	2.18E-01 2.5E-04	2.17E-01 8.0E-04	2.19E-01 3.0E-05	2.18E-01 4.7E-05	2.18E-01 1.7E-03	2.19E-01 3.8E-03	2.20E-01 2.9E-03
WFG6	2.00E-01 1.1E-02	1.97E-01 1.1E-02	1.83E-01 2.9E-02	2.09E-01 1.5E-04	2.08E-01 7.7E-04	2.06E-01 3.2E-03	2.06E-01 1.2E-03
WFG7	2.09E-01 3.0E-04	2.08E-01 1.8E-04	2.10E-01 1.0E-04	2.09E-01 8.4E-05	2.09E-01 2.8E-04	2.09E-01 2.5E-04	2.08E-01 3.8E-04
WFG9	2.24E-01 1.4E-03	2.24E-01 1.3E-03	2.24E-01 2.7E-03	2.22E-01 6.9E-04	2.24E-01 1.1E-03	2.23E-01 1.1E-03	2.24E-01 1.0E-03
DTLZ1	6.22E-01 2.4E-01	1.74E-01 8.1E-02	3.40E-01 3.0E-01	2.01E-01 1.4E-01	4.57E-01 3.3E-01	5.21E-01 3.3E-01	5.56E-02 1.2E-01
DTLZ2	3.75E-01 5.4E-03	4.12E-01 6.0E-04	3.75E-01 5.5E-03	7.55E-02 3.1E-06	3.71E-01 6.2E-03	3.73E-01 5.0E-03	3.82E-01 3.8E-03
DTLZ4	3.75E-01 4.8E-03	2.42E-01 1.3E-01	2.95E-01 1.6E-01	8.15E-02 2.7E-02	3.71E-01 4.7E-03	3.82E-01 4.5E-03	3.83E-01 4.9E-03
DTLZ5	9.28E-02 2.0E-04	9.17E-02 1.8E-04	9.40E-02 3.5E-05	1.43E-02 1.6E-07	9.28E-02 2.0E-04	9.28E-02 2.4E-04	9.32E-02 1.3E-04
DTLZ6	0.00E+00 0.0E+00	7.22E-02 1.4E-02	0.00E+00 0.0E+00	1.45E-02 3.8E-09	9.36E-02 1.7E-04	9.41E-02 1.7E-04	9.50E-02 2.8E-05
DTLZ7	2.79E-01 3.9E-03	2.34E-01 4.6E-02	2.56E-01 2.0E-02	6.38E-02 4.9E-02	2.86E-01 4.1E-03	2.69E-01 5.1E-03	2.88E-01 3.5E-02
LZ09_F1	6.52E-01 7.3E-04	6.55E-01 1.4E-03	6.50E-01 3.6E-03	6.61E-01 1.1E-04	6.51E-01 7.3E-04	6.50E-01 8.5E-04	6.37E-01 2.2E-03
LZ09_F2	5.08E-01 4.6E-02	4.93E-01 4.4E-02	4.50E-01 7.3E-02	5.27E-01 3.8E-02	5.64E-01 1.9E-02	5.27E-01 1.4E-02	5.19E-01 1.4E-02
LZ09_F3	5.97E-01 9.2E-03	5.85E-01 1.7E-02	5.64E-01 2.4E-02	5.96E-01 2.7E-02	5.97E-01 7.2E-03	5.78E-01 6.4E-03	5.63E-01 4.7E-03
LZ09_F4	6.09E-01 4.6E-03	6.02E-01 5.9E-03	5.93E-01 1.1E-02	6.20E-01 5.2E-03	6.06E-01 5.8E-03	5.93E-01 3.2E-03	5.74E-01 5.5E-03
LZ09_F5	6.09E-01 8.1E-03	6.03E-01 1.2E-02	5.94E-01 1.6E-02	6.09E-01 1.7E-02	6.11E-01 6.0E-03	5.97E-01 4.3E-03	5.83E-01 4.2E-03
LZ09_F6	1.63E-01 3.9E-02	6.43E-02 7.5E-02	1.51E-01 4.3E-02	7.26E-02 1.2E-02	2.05E-01 2.7E-02	2.26E-01 2.6E-02	2.08E-01 2.7E-02
LZ09_F7	4.47E-01 6.0E-02	3.93E-01 6.1E-02	3.48E-01 7.4E-02	4.43E-01 1.5E-01	3.27E-01 1.3E-01	3.75E-01 5.7E-02	2.06E-01 1.4E-01
LZ09_F8	4.07E-01 4.0E-02	3.85E-01 5.5E-02	3.30E-01 7.5E-02	3.50E-01 9.6E-02	2.35E-01 1.2E-01	3.38E-01 5.6E-02	1.81E-01 1.0E-01
LZ09_F9	1.65E-01 4.9E-02	1.63E-01 4.9E-02	1.43E-01 5.5E-02	1.67E-01 6.4E-02	2.26E-01 2.1E-02	2.00E-01 1.2E-02	1.94E-01 1.0E-02
Fonseca	3.08E-01 3.9E-04	3.11E-01 1.1E-04	3.12E-01 9.7E-05	3.12E-01 1.2E-05	3.09E-01 3.2E-04	3.08E-01 4.0E-04	3.11E-01 1.4E-04
Kursawe	4.00E-01 2.6E-04	3.94E-01 9.2E-04	4.01E-01 9.7E-05	4.00E-01 1.1E-04	4.00E-01 2.0E-04	4.00E-01 2.3E-04	4.01E-01 2.4E-04
Schaffer	5.24E-01 1.9E-01	5.30E-01 2.1E-01	5.09E-01 2.1E-01	5.03E-01 1.9E-01	5.28E-01 1.9E-01	8.27E-01 2.8E-04	8.30E-01 2.8E-05
Viennet2	9.20E-01 1.4E-03	9.07E-01 7.4E-03	9.22E-01 6.8E-04	7.25E-01 4.2E-04	9.21E-01 1.3E-03	9.20E-01 1.4E-03	9.23E-01 7.1E-04
Viennet3	8.33E-01 6.4E-04	8.29E-01 4.4E-03	8.35E-01 5.2E-04	5.66E-01 5.7E-05	8.33E-01 5.8E-04	8.32E-01 6.6E-04	8.35E-01 4.5E-04
Poloni	9.13E-01 9.1E-05	9.11E-01 1.3E-03	9.13E-01 1.9E-05	9.11E-01 1.3E-03	9.13E-01 7.4E-05	9.13E-01 9.1E-03	9.13E-01 1.1E-04
OKA1	5.93E-01 6.1E-03	5.67E-01 1.5E-02	5.71E-01 2.0E-02	6.00E-01 7.6E-03	5.55E-01 1.1E-02	6.04E-01 8.0E-03	5.46E-01 9.8E-03
OKA2	1.48E-01 2.5E-02	6.29E-02 5.0E-02	6.07E-02 6.3E-02	5.25E-02 3.5E-02	6.26E-02 2.0E-02	1.18E-01 3.3E-02	4.17E-02 1.3E-02

Algoritmaların HV metriği ile elde edilen Çizelge 5.8'deki ortalama sonuçlara bakıldığında MOSG algoritmasının 15 fonksiyon üzerinde en iyi ve 3 fonksiyon üzerinde ikinci en iyi sonuçlar ürettiği görülmektedir. Diğer algoritmalar için bu sayılar NSGA-II için [7/6], IBEA için [4/3], MOCell için [9/4], MOEA/D için [5/5], MOAAA için [8/5] ve MOVS için [4/10] şeklindedir. HV metriği için algoritmaların ortalama sonuçlarına bakıldığında MOSG algoritmasının diğer algoritmalara karşı sayısal bir üstünlük yakaladığı görülmektedir.

Çizelge 5.9. Problem Seti-1: IGD metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II	**IBEA**	**MOCell**	**MOEA/D**	**MOAAA**	**MOVS**	**MOSG**
	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.
ZDT1	1.87E-04 7.4E-06	1.64E-04 5.1E-06	1.40E-04 1.6E-06	5.31E-04 1.7E-04	2.31E-04 1.0E-05	1.96E-04 7.0E-06	1.34E-04 8.2E-07
ZDT2	1.94E-04 9.0E-06	5.42E-04 3.7E-05	1.41E-04 2.2E-06	4.40E-04 1.6E-04	2.39E-04 1.3E-05	4.58E-04 1.0E-04	1.41E-04 1.7E-06
ZDT3	1.33E-04 6.2E-06	1.51E-03 7.2E-04	6.80E-04 1.3E-03	1.70E-03 7.6E-04	1.58E-04 7.6E-06	1.34E-04 7.4E-06	1.00E-04 1.2E-06
ZDT4	2.84E-04 6.2E-05	2.23E-02 3.1E-03	2.00E-04 1.4E-04	1.10E-02 7.4E-03	2.12E-03 2.4E-03	2.83E-02 1.6E-02	1.36E-04 1.3E-06
ZDT6	3.27E-04 2.9E-05	2.55E-04 1.0E-05	2.05E-04 1.3E-05	1.42E-04 1.1E-05	2.52E-04 1.3E-05	2.20E-04 1.5E-05	1.34E-04 8.5E-07
WFG1	3.16E-03 2.1E-03	3.67E-03 2.2E-03	7.22E-03 1.8E-03	6.70E-03 1.8E-03	4.79E-04 3.7E-04	2.76E-03 1.2E-03	1.21E-02 3.5E-04
WFG2	1.98E-03 2.0E-03	3.76E-03 9.7E-04	3.29E-03 1.7E-03	4.38E-04 1.9E-05	1.24E-04 4.4E-06	1.31E-04 5.6E-06	3.45E-04 1.2E-04
WFG3	1.55E-04 8.4E-06	1.30E-04 2.9E-06	1.11E-04 2.1E-06	1.36E-04 4.8E-07	1.47E-04 7.4E-06	1.45E-04 6.1E-06	1.36E-04 7.4E-06
WFG4	1.65E-04 1.1E-05	5.22E-04 3.4E-05	1.21E-04 1.7E-06	2.28E-04 3.6E-05	1.56E-04 7.2E-06	1.56E-04 6.0E-06	1.78E-04 3.3E-05
WFG5	1.29E-04 6.5E-06	3.71E-04 3.3E-05	9.61E-05 5.7E-06	1.24E-04 9.0E-07	1.31E-04 1.0E-05	1.33E-04 2.4E-05	1.05E-04 1.7E-05
WFG6	3.91E-04 2.8E-04	8.54E-04 2.0E-04	9.26E-04 9.5E-04	2.28E-04 1.8E-06	2.28E-04 1.4E-05	2.50E-04 6.1E-05	2.33E-04 2.6E-05
WFG7	1.57E-04 8.4E-06	5.14E-04 3.2E-05	1.14E-04 1.8E-06	1.48E-04 7.4E-07	1.51E-04 7.0E-06	1.52E-04 6.6E-06	1.62E-04 1.9E-05
WFG9	2.34E-04 1.3E-05	4.83E-04 4.6E-05	1.85E-04 3.9E-05	2.41E-04 5.8E-06	2.29E-04 1.2E-05	2.33E-04 1.3E-05	1.87E-04 9.6E-06
DTLZ1	1.76E-03 2.3E-03	4.17E-03 3.4E-04	6.22E-03 6.6E-03	7.94E-03 6.7E-03	3.95E-03 4.6E-03	2.87E-03 3.5E-03	1.20E-02 1.5E-02
DTLZ2	7.71E-04 3.8E-05	1.40E-03 2.9E-05	7.53E-04 3.5E-05	5.72E-03 1.4E-07	7.54E-04 3.1E-05	7.60E-04 3.0E-05	6.96E-04 2.3E-05
DTLZ4	1.22E-03 1.2E-04	5.11E-03 2.9E-03	3.00E-03 3.4E-03	1.16E-02 1.1E-03	1.23E-03 1.1E-04	1.19E-03 1.2E-04	1.13E-03 1.1E-04
DTLZ5	2.03E-05 1.4E-06	1.02E-04 5.2E-06	1.42E-05 3.9E-07	1.48E-03 2.9E-09	1.94E-05 9.9E-07	2.10E-05 1.1E-06	2.74E-05 6.1E-06
DTLZ6	7.15E-03 6.3E-04	4.49E-04 1.1E-04	1.47E-02 1.1E-03	3.82E-03 3.3E-10	5.66E-05 3.1E-06	5.02E-05 3.5E-06	3.39E-05 1.1E-06
DTLZ7	2.28E-03 1.7E-04	1.59E-02 8.7E-03	1.04E-02 8.0E-03	2.84E-02 1.3E-03	2.29E-03 1.7E-04	2.24E-03 1.3E-04	2.93E-03 4.6E-03
LZ09_F1	4.34E-04 2.2E-05	6.84E-04 3.3E-04	7.03E-04 3.9E-04	2.36E-04 1.5E-05	4.35E-04 2.1E-05	5.02E-04 3.6E-05	8.95E-04 1.5E-04
LZ09_F2	6.38E-03 1.8E-03	8.26E-03 2.7E-03	1.07E-02 4.1E-03	4.89E-03 2.1E-03	3.22E-03 8.0E-04	5.39E-03 5.6E-04	5.69E-03 5.5E-04
LZ09_F3	3.22E-03 9.9E-04	6.65E-03 3.2E-03	7.26E-03 3.3E-03	5.16E-03 3.1E-03	2.46E-03 3.6E-04	3.99E-03 8.5E-04	4.09E-03 3.2E-04
LZ09_F4	3.19E-03 6.8E-04	6.10E-03 1.1E-03	4.06E-03 6.4E-04	2.98E-03 6.4E-04	2.53E-03 5.2E-04	3.76E-03 4.9E-04	4.38E-03 4.8E-04
LZ09_F5	2.74E-03 9.1E-04	4.58E-03 2.2E-03	4.76E-03 2.4E-03	4.03E-03 2.5E-03	1.92E-03 3.4E-04	3.04E-03 4.3E-04	3.49E-03 2.8E-04
LZ09_F6	7.32E-03 1.4E-03	1.62E-02 2.4E-03	8.45E-03 2.6E-03	1.92E-02 9.8E-04	5.79E-03 8.7E-04	5.73E-03 8.2E-04	7.35E-03 2.2E-03
LZ09_F7	1.09E-02 4.4E-03	1.92E-02 5.5E-03	1.90E-02 6.4E-03	9.18E-03 6.3E-03	1.19E-02 4.3E-03	1.34E-02 4.9E-03	1.93E-02 7.3E-03
LZ09_F8	1.03E-02 3.5E-03	1.69E-02 5.7E-03	1.91E-02 6.3E-03	1.06E-02 3.1E-03	1.33E-02 3.7E-03	1.21E-02 2.7E-03	1.76E-02 5.6E-03
LZ09_F9	8.19E-03 2.7E-03	9.02E-03 2.6E-03	1.12E-02 4.0E-03	4.80E-03 1.8E-03	3.87E-03 1.1E-03	5.91E-03 5.3E-04	6.15E-03 4.8E-04
Fonseca	3.21E-04 1.4E-05	2.59E-04 7.5E-06	2.17E-04 3.0E-06	2.04E-04 4.8E-07	2.83E-04 1.1E-05	3.18E-04 1.2E-05	2.42E-04 2.0E-05
Kursawe	1.80E-04 1.1E-05	1.28E-03 1.7E-04	1.24E-04 1.7E-06	1.74E-04 1.5E-06	1.61E-04 6.7E-06	1.80E-04 1.0E-05	1.32E-04 4.3E-06
Schaffer	2.49E-02 1.4E-02	2.61E-02 2.5E-02	2.56E-02 1.9E-02	2.65E-02 1.5E-02	2.47E-02 1.4E-02	5.81E-04 2.4E-05	3.39E-04 6.3E-06
Viennet2	3.37E-04 4.2E-05	1.47E-03 3.6E-04	2.95E-04 2.2E-05	2.91E-03 1.3E-06	3.26E-04 4.0E-05	3.42E-04 3.4E-05	3.07E-04 3.0E-05
Viennet3	1.72E-04 2.5E-05	4.37E-03 2.6E-04	2.02E-04 3.8E-04	7.50E-03 1.5E-06	1.68E-04 1.6E-05	1.82E-04 2.0E-05	1.63E-04 1.6E-05
Poloni	1.00E-04 5.8E-06	1.07E-03 6.3E-04	7.05E-05 9.9E-07	7.87E-04 1.1E-03	9.44E-05 5.6E-06	1.04E-04 5.7E-06	7.84E-05 7.4E-06
OKA1	2.69E-03 6.1E-04	4.94E-03 2.4E-03	5.41E-03 3.2E-03	2.73E-03 9.2E-04	3.59E-03 6.8E-04	2.95E-03 1.4E-03	3.73E-03 6.9E-04
OKA2	9.01E-03 1.9E-03	1.95E-02 9.1E-03	2.17E-02 1.1E-02	1.71E-02 9.1E-03	1.16E-02 8.6E-04	9.42E-03 1.8E-03	1.24E-02 7.1E-04

Algoritmaların IGD metriği ile elde edilen Çizelge 5.9'daki ortalama sonuçlara bakıldığında MOSG algoritmasının 10 fonksiyon üzerinde en iyi ve 5 fonksiyon üzerinde ikinci en iyi sonuçlar ürettiği görülmektedir. Bununla beraber MOCell algoritması 10 fonksiyonda en iyi ve 4 fonksiyonda ikinci en iyi ortalama değerler ile MOSG algoritmasına yakın bir başarı elde etmiştir. Diğer algoritmalar için bu sayılar NSGA-II için [4/6], IBEA için [0/1], MOEA/D için [3/8], MOAAA için [8/4] ve MOVS için [2/8] şeklindedir. IGD metriği için algoritmaların ortalama sonuçlarına bakıldığında MOSG ve MOCell algoritmalarının diğer algoritmalara karşı sayısal bir üstünlük

yakaladığı görülmektedir. Bu iki algoritma IGD metriği üzerinde kendi aralarında rekabetçi düzeyde sonuçlar elde etmiştir.

Çizelge 5.10. Problem Seti-1: Spread metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II	**IBEA**	**MOCell**	**MOEA/D**	**MOAAA**	**MOVS**	**MOSG**
	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.	Ortalama Std. Sap.
ZDT1	3.69E-01 3.0E-02	2.97E-01 1.8E-02	8.18E-02 1.3E-02	3.66E-01 4.8E-02	6.51E-01 5.3E-02	4.67E-01 3.8E-02	7.92E-02 1.0E-02
ZDT2	3.82E-01 3.3E-02	3.37E+01 2.3E-02	8.46E-02 1.2E-02	3.25E-01 9.4E-02	6.79E-01 6.0E-02	5.23E-01 3.4E-02	7.50E-02 1.3E-02
ZDT3	7.47E-01 1.4E-02	1.19E+00 6.9E-02	7.08E-01 1.3E-02	9.93E-01 3.2E-02	8.04E-01 1.9E-02	7.54E-01 1.2E-02	7.05E-01 3.2E-03
ZDT4	3.89E-01 3.5E-02	1.11E+00 5.6E-02	1.27E-01 3.8E-02	9.69E-01 1.7E-01	5.61E-01 1.8E-01	1.24E+00 1.3E-01	8.79E-02 1.6E-02
ZDT6	3.64E-01 2.8E-02	4.15E-01 4.8E-02	1.09E-01 6.1E-02	1.54E-01 8.3E-03	7.97E-01 4.8E-02	5.76E-01 4.7E-02	6.55E-02 1.3E-02
WFG1	7.21E-01 4.9E-02	8.74E-01 6.9E-02	6.80E-01 1.1E-01	1.07E+00 1.5E-01	6.70E-01 1.7E-01	6.40E-01 1.8E-01	8.85E-01 8.6E-02
WFG2	7.91E-01 1.1E-02	1.25E+00 7.9E-02	7.58E-01 3.8E-03	1.11E+00 5.6E-03	7.80E-01 8.1E-03	7.89E-01 9.5E-03	9.57E-01 4.2E-02
WFG3	3.74E-01 2.9E-02	2.55E-01 2.8E-02	5.60E-02 7.7E-03	3.45E-01 8.6E-04	3.15E-01 2.8E-02	3.29E-01 3.2E-02	1.19E-01 1.2E-02
WFG4	3.76E-01 2.8E-02	5.13E-01 3.4E-02	1.22E-01 1.5E-02	5.10E-01 5.2E-02	3.27E-01 3.1E-02	3.57E-01 3.0E-02	2.39E-01 2.4E-02
WFG5	4.10E-01 3.1E-02	5.85E-01 7.3E-02	1.25E-01 1.6E-02	4.53E-01 5.0E-03	4.24E-01 3.2E-02	3.96E-01 3.2E-02	1.36E-01 1.8E-02
WFG6	3.87E-01 3.1E-02	5.26E-01 2.8E-02	1.46E-01 3.3E-02	4.13E-01 6.4E-03	3.29E-01 2.7E-02	3.47E-01 3.2E-02	2.24E-01 2.9E-02
WFG7	3.79E-01 3.1E-02	5.15E-01 2.8E-02	1.24E-01 1.4E-02	4.14E-01 6.6E-03	3.38E-01 3.0E-02	3.66E-01 3.2E-02	2.42E-01 2.4E-02
WFG9	3.67E-01 2.7E-02	5.00E-01 4.3E-02	1.14E-01 1.5E-02	4.48E-01 1.4E-02	3.35E-01 2.5E-02	3.33E-01 3.0E-02	1.59E-01 1.6E-02
DTLZ1	9.09E-01 2.4E-01	1.64E+00 1.1E-01	1.19E+00 2.5E-01	1.09E+00 8.9E-02	7.34E-01 6.4E-02	7.68E-01 6.3E-02	6.80E-01 7.8E-02
DTLZ2	6.94E-01 5.2E-02	5.83E-01 5.0E-02	6.92E-01 4.6E-02	1.00E+00 2.1E-05	6.60E-01 4.3E-02	6.55E-01 4.0E-02	6.27E-01 3.6E-02
DTLZ4	6.74E-01 4.5E-02	7.04E-01 1.4E-01	7.27E-01 1.5E-01	1.02E+00 8.4E-02	6.51E-01 4.2E-02	6.44E-01 3.7E-02	6.40E-01 3.7E-02
DTLZ5	4.48E-01 4.6E-02	6.74E-01 3.9E-02	1.39E-01 4.5E-02	1.00E+00 2.9E-06	4.39E-01 5.4E-02	5.09E-01 5.3E-02	3.04E-01 3.2E-02
DTLZ6	8.15E-01 4.9E-02	9.83E-01 1.7E-01	7.50E-01 4.7E-02	1.00E+00 2.6E-08	6.78E-01 3.9E-02	5.24E-01 3.6E-02	1.03E-01 1.5E-02
DTLZ7	7.56E-01 4.7E-02	8.26E-01 1.1E-01	7.00E-01 6.2E-02	9.90E-01 2.4E-02	7.84E-01 4.6E-02	7.09E-01 5.3E-02	6.73E-01 5.8E-02
LZ09_F1	5.08E-01 1.1E-01	7.66E-01 5.1E-02	4.94E-01 1.7E-01	3.13E-01 4.2E-02	3.69E-01 6.3E-02	4.21E-01 1.3E-01	6.46E-01 8.3E-02
LZ09_F2	1.47E+00 1.2E-01	1.47E+00 1.3E-01	1.37E+00 1.8E-01	9.92E-01 1.4E-01	1.12E+00 1.1E-01	1.52E+00 1.1E-01	1.41E+00 1.0E-01
LZ09_F3	7.29E-01 8.1E-02	1.12E+00 1.0E-01	6.62E-01 1.1E-01	7.01E-01 8.8E-02	6.10E-01 6.6E-02	8.00E-01 9.0E-02	9.80E-01 6.4E-02
LZ09_F4	5.78E-01 6.0E-02	1.03E+00 4.8E-02	5.39E-01 9.1E-02	9.71E-01 1.7E-01	5.40E-01 4.9E-02	6.16E-01 7.6E-02	7.57E-01 1.2E-01
LZ09_F5	6.41E-01 6.2E-02	1.09E+00 7.7E-02	5.54E-01 1.0E-01	6.64E-01 8.7E-02	5.65E-01 4.8E-02	6.75E-01 8.2E-02	8.73E-01 7.5E-02
LZ09_F6	9.37E-01 8.7E-02	1.66E+00 4.1E-01	8.65E-01 8.4E-02	9.87E-01 2.8E-02	8.76E-01 8.4E-02	8.86E-01 8.3E-02	8.44E-01 1.0E-01
LZ09_F7	1.38E+00 1.2E-01	1.12E+00 1.5E-01	1.27E+00 2.0E-01	1.18E+00 1.8E-01	1.29E+00 2.5E01	1.31E+00 1.5E-01	9.36E-01 1.5E-01
LZ09_F8	1.26E+00 1.1E-01	1.18E+00 1.6E-01	1.31E+00 1.6E-01	1.27E+00 7.4E-02	1.26E+00 2.8E-01	1.19E+00 1.5E-01	7.99E-01 1.3E-01
LZ09_F9	1.59E+00 1.9E-01	1.67E+00 1.2E-01	1.58E+00 1.9E-01	9.77E-01 1.3E-01	1.31E+00 1.8E-01	1.74E+00 1.7E-01	1.56E+00 2.0E-01
Fonseca	4.06E-01 3.9E-02	4.06E-01 2.5E-02	7.72E-02 1.1E-02	1.46E-01 5.7E-04	2.87E-01 2.9E-02	4.03E-01 3.5E-02	1.10E-01 1.2E-02
Kursawe	5.66E-01 2.4E-02	8.66E-01 2.9E-02	4.17E-01 5.4E-03	7.31E-01 4.2E-03	5.05E-01 1.7E-02	5.68E-01 2.2E-02	4.20E-01 8.9E-03
Schaffer	6.23E-01 2.3E-01	6.53E-01 2.2E-01	5.99E-01 2.5E-01	1.15E+00 3.4E-02	6.39E-01 2.3E-01	7.34E-01 3.7E-02	1.08E-01 1.6E-02
Viennet2	8.41E-01 9.5E-02	9.80E-01 5.6E-02	8.48E-01 8.2E-02	1.00E+00 7.3E-04	8.54E-01 8.3E-02	8.15E-01 9.0E-02	7.93E-01 7.3E-02
Viennet3	7.26E-01 5.4E-02	8.09E-01 7.4E-02	6.78E-01 6.8E-02	1.00E+00 3.0E-05	7.00E-01 5.9E-02	7.49E-01 5.5E-02	6.76E-01 6.2E-02
Poloni	8.04E-01 1.8E-02	1.17E+00 4.3E-02	7.50E-01 1.3E-02	1.26E+00 4.7E-02	7.94E-01 2.1E-02	8.15E-01 2.3E-02	7.91E-01 3.4E-02
OKA1	1.12E+00 6.5E-02	1.63E+00 8.8E-02	1.04E+00 9.8E-02	1.17E+00 8.1E-02	1.34E+00 9.1E-02	1.04E+00 6.0E-02	6.29E-01 8.0E-02
OKA2	1.51E+00 9.0E-02	1.45E+00 3.3E-01	1.14E+00 1.8E-01	1.52E+00 3.1E-01	1.56E+00 1.4E-01	1.46E+00 7.5E-02	9.03E-01 1.1E-01

Algoritmaların Spread metriği ile elde edilen Çizelge 5.10'daki ortalama sonuçlara bakıldığında MOSG algoritmasının 17 fonksiyon üzerinde en iyi ve 11 fonksiyon üzerinde ikinci en iyi sonuçlar ürettiği görülmektedir. Geriye kalan algoritmalardan MOCell algoritması 13 fonksiyonda en iyi ve 12 fonksiyonda ikinci en iyi ortalama değerler ile MOSG algoritmasından sonra gelmektedir. Diğer algoritmalar

için bu sayılar NSGA-II için [0/0], IBEA için [1/2], MOEA/D için [3/0], MOAAA için [1/8] ve MOVS için [1/3] şeklindedir. Literatüre bakıldığında MOCell algoritmasının Spread metriğinde genel olarak başarılı sonuçlar ürettiği görülmektedir. MOSG algoritmasının bu metrik ile elde edilen değerlere göre MOCell algoritmasını başarı olarak geride bırakması uygulanan mempleks stratejisinin beklenen etkiyi gösterdiği söylenebilir.

Çizelge 5.11. Problem Seti-1: Epsilon metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II		**IBEA**		**MOCell**		**MOEA/D**		**MOAAA**		**MOVS**		**MOSG**	
	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.
ZDT1	1.32E-02	1.8E-03	8.99E-03	1.0E-03	6.57E-03	3.2E-04	2.41E-02	6.8E-03	1.44E-02	2.1E-03	1.20E-02	1.4E-03	5.38E-03	1.6E-04
ZDT2	1.38E-02	2.3E-03	1.64E-02	1.5E-03	5.92E-03	6.8E-04	4.23E-02	2.0E-02	1.40E-02	2.0E-03	1.54E-02	4.8E-03	5.32E-03	1.5E-04
ZDT3	8.67E-03	1.6E-03	7.79E-02	1.2E-01	7.56E-02	1.5E-01	1.36E-01	3.1E-02	9.05E-03	1.7E-03	8.20E-03	1.4E-03	5.15E-03	3.3E-04
ZDT4	1.60E-02	4.1E-03	7.51E-01	1.1E-01	1.08E-02	9.8E-03	4.52E-01	2.8E-01	8.02E-02	7.5E-02	1.07E+00	5.8E-01	5.47E-03	2.6E-04
ZDT6	1.51E-02	1.6E-03	1.22E-02	1.1E-03	8.03E-03	5.0E-04	5.18E-03	7.4E-04	1.31E-02	2.1E-03	1.26E-02	3.0E-03	4.71E-03	1.7E-04
WFG1	4.57E-01	2.6E-01	4.41E-01	2.3E-01	1.00E+00	2.1E-01	6.75E-01	1.5E-01	1.24E-01	4.4E-02	2.42E-01	9.0E-02	1.02E+00	3.0E-02
WFG2	3.83E-01	3.9E-01	6.60E-01	2.9E-01	6.39E-01	3.3E-01	3.09E-02	6.0E-03	1.43E-02	2.6E-03	1.45E-02	2.7E-03	3.50E-02	9.4E-03
WFG3	3.90E-02	7.0E-03	2.35E-02	1.9E-03	1.68E-02	1.2E-03	2.78E-02	1.4E-03	3.62E-02	6.5E-03	3.37E-02	5.8E-03	2.86E-02	3.1E-03
WFG4	3.53E-02	7.1E-03	4.34E-02	4.6E-03	1.54E-02	2.7E-03	6.59E-02	1.6E-02	3.38E-02	6.6E-03	3.04E-02	4.5E-03	3.09E-02	1.0E-02
WFG5	6.03E-02	2.1E-02	4.70E-02	1.8E-02	5.53E-02	2.8E-02	7.71E-02	3.2E-03	6.22E-02	1.8E-02	4.95E-02	2.1E-02	6.27E-02	2.2E-02
WFG6	4.61E-02	1.8E-02	6.26E-02	1.9E-02	6.09E-02	5.3E-02	2.41E-02	7.2E-04	3.21E-02	7.5E-03	3.33E-02	6.7E-03	2.31E-02	4.3E-03
WFG7	3.51E-02	6.5E-03	4.44E-02	5.7E-03	1.47E-02	6.8E-04	2.49E-02	5.7E-04	3.35E-02	5.7E-03	3.24E-02	5.3E-03	2.06E-02	1.7E-03
WFG9	3.48E-02	5.9E-03	3.37E-02	4.7E-03	1.84E-02	3.9E-03	3.13E-02	1.5E-03	3.41E-02	5.5E-03	3.52E-02	6.6E-03	2.17E-02	2.3E-03
DTLZ1	1.12E-01	9.6E-02	2.93E-01	2.0E-02	3.37E-01	2.8E-01	3.09E-01	2.3E-01	1.90E-01	1.7E-01	1.43E-01	1.3E-01	5.25E-01	4.8E-01
DTLZ2	1.28E-01	1.7E-02	8.95E-02	4.2E-03	1.28E-01	1.6E-02	5.77E-01	1.5E-05	1.28E-01	1.8E-02	1.28E-01	1.7E-02	1.22E-01	1.6E-02
DTLZ4	1.10E-01	1.8E-02	5.02E-01	3.0E-01	3.01E-01	3.6E-01	6.58E-01	9.7E-02	1.07E-01	1.3E-02	1.06E-01	1.4E-02	1.00E-01	1.5E-02
DTLZ5	1.09E-02	1.8E-03	3.20E-02	1.7E-03	4.97E-03	1.2E-03	5.77E-01	2.3E-06	1.02E-02	1.4E-03	1.05E-02	1.8E-03	5.60E-03	7.1E-04
DTLZ6	8.56E-01	7.8E-02	6.19E-02	2.4E-02	1.67E+00	1.4E-01	5.77E-01	5.4E-08	9.81E-03	1.6E-03	8.44E-03	1.3E-03	4.16E-03	2.1E-04
DTLZ7	1.46E-01	4.8E-02	1.20E+00	9.6E-01	9.06E-01	8.0E-01	2.02E+00	6.6E-01	1.38E-01	4.5E-02	1.65E-01	4.4E-02	2.00E-01	4.3E-01
LZ09_F1	1.87E-02	2.0E-03	3.85E-02	1.9E-02	4.43E-02	2.8E-02	8.81E-03	8.8E-04	1.82E-02	1.8E-03	2.33E-02	4.0E-03	4.65E-02	1.1E-02
LZ09_F2	2.17E-01	5.7E-02	2.50E-01	7.4E-02	3.30E-01	1.3E-01	1.99E-01	6.3E-02	1.45E-01	3.4E-02	1.90E-01	2.2E-02	1.97E-01	2.7E-02
LZ09_F3	1.39E-01	3.0E-02	2.02E-01	7.5E-02	2.50E-01	7.9E-02	2.14E-01	9.2E-02	1.16E-01	2.0E-02	1.65E-01	2.9E-02	1.58E-01	1.4E-02
LZ09_F4	1.61E-01	2.4E-02	2.03E-01	2.7E-02	1.94E-01	2.0E-02	1.65E-01	3.1E-02	1.33E-01	3.1E-02	1.84E-01	1.7E-02	2.04E-01	2.0E-02
LZ09_F5	1.24E-01	3.0E-02	1.55E-01	5.4E-02	1.85E-01	6.1E-02	1.74E-01	7.6E-02	9.07E-02	1.9E-02	1.36E-01	1.8E-02	1.46E-01	1.2E-02
LZ09_F6	3.34E-01	7.6E-02	4.51E-01	1.3E-01	4.43E-01	1.8E-01	7.06E-01	9.2E-02	2.86E-01	4.7E-02	2.89E-01	5.2E-02	4.18E-01	1.6E-01
LZ09_F7	3.55E-01	1.2E-01	4.66E-01	1.3E-01	5.24E-01	1.5E-01	3.73E-01	2.2E-01	4.24E-01	1.3E-01	4.06E-01	1.0E-01	5.99E-01	2.3E-01
LZ09_F8	3.57E-01	9.7E-02	4.42E-01	1.2E-01	5.51E-01	1.4E-01	3.81E-01	9.0E-02	4.35E-01	1.1E-01	3.92E-01	7.5E-02	5.47E-01	1.9E-01
LZ09_F9	2.54E-01	6.5E-02	2.79E-01	8.0E-02	3.74E-01	1.2E-01	2.31E-01	7.7E-02	1.63E-01	4.4E-02	2.04E-01	4.1E-02	2.30E-01	5.8E-02
Fonseca	1.32E-02	1.6E-03	8.38E-03	7.6E-04	6.77E-03	5.9E-04	6.34E-03	3.5E-05	1.19E-02	1.7E-03	1.31E-02	1.5E-03	1.02E-02	3.3E-03
Kursawe	7.76E-02	1.2E-02	2.72E-01	2.0E-02	4.43E-02	4.2E-03	7.76E-02	4.9E-03	7.15E-02	9.7E-03	7.83E-02	1.3E-02	6.79E-02	2.2E-02
Schaffer	1.61E+00	1.0E+00	1.67E+00	1.6E+00	1.72E+00	1.3E+00	1.75E+00	1.1E+00	1.57E+00	1.0E+00	5.34E-02	1.0E-02	2.00E-02	9.6E-04
Viennet2	3.26E-02	8.2E-03	3.02E-02	1.0E-02	2.98E-02	4.9E-03	1.26E-01	7.2E-05	3.37E-02	7.5E-03	3.29E-02	7.7E-03	2.66E-02	3.9E-03
Viennet3	4.90E-02	1.2E-02	6.68E-02	2.2E-02	5.26E-02	1.2E-02	4.78E-01	2.2E-04	4.91E-02	1.2E-02	5.22E-02	1.2E-02	4.52E-02	9.1E-03
Poloni	1.46E-01	3.6E-02	4.18E-01	3.3E-02	7.17E-02	6.4E-03	2.55E-01	1.7E-01	1.30E-01	3.5E-02	1.53E-01	3.1E-02	1.21E-01	2.7E-02
OKA1	3.39E-01	6.2E-02	5.02E-01	1.3E-01	5.16E-01	1.7E-01	3.21E-01	7.0E-02	4.23E-01	6.1E-02	3.36E-01	7.8E-02	4.10E-01	5.8E-02
OKA2	4.73E-01	8.5E-02	7.26E-01	2.0E-01	7.90E-01	2.3E-01	7.10E-01	1.8E-01	6.21E-01	6.1E-02	5.26E-01	9.4E-02	6.87E-01	4.3E-02

Algoritmaların Epsilon metriği ile elde edilen Çizelge 5.11'deki ortalama sonuçlara bakıldığında MOSG algoritmasının 11 fonksiyon üzerinde en iyi ve 6 fonksiyon üzerinde ikinci en iyi sonuçlar ürettiği görülmektedir. Diğer algoritmalar için bu sayılar NSGA-II için [4/5], IBEA için [2/1], MOCell için [7/5], MOEA/D için [3/4], MOAAA için [9/1] ve MOVS için [0/14] şeklindedir. Epsilon metriği için algoritmaların ortalama sonuçlarına bakıldığında MOSG algoritmasının diğer algoritmalara karşı sayısal bir üstünlük yakaladığı görülmektedir.

Çizelge 5.8 – 5.11'deki sonuçlara genel olarak bakıldığında MOSG algoritmasının en iyi ve ikinci en iyi olduğu fonksiyonların sayısı bakımından diğer algoritmalardan daha başarılı olduğu görülmektedir. Bu sonuçları sadece ortalama değer açısından ele almak yetersiz kalacağından farklı kıyaslama yöntemleri de kullanılmıştır. Friedman (Friedman, 1937) istatistiksel testi algoritmaların her bir çalıştırma sonucunu kullanarak ortalama bir başarı sıralaması yapmaktadır. Algoritmaların dört metrik için elde edilen sonuçlarına göre ortalama bir başarı sıralaması elde etmek adına Friedman testi uygulanmıştır. Bu doğrultuda algoritmalar için elde edilen ortalama Friedman başarı sıralaması Çizelge 5.12'de verilmiştir. Her bir metrik için en iyi sıralama değerine sahip algoritmanın değeri koyu gri ile işaretlenirken ikinci en iyi değere sahip algoritmanın sıralama değeri açık gri şeklinde işaretlenmiştir.

Çizelge 5.12. Problem Seti-1: Algoritmaların dört metrik üzerindeki Friedman test sonuçları

	HV	**IGD**	**Spread**	**Epsilon**
NSGA-II	4.1528	3.6389	4.4445	3.8889
IBEA	3.3333	5.7778	5.6667	5.0278
MOCell	4.0695	4.1111	2.4167	4.3056
MOEA/D	3.3889	4.5001	5.2501	4.8056
MOAAA	4.4722	2.9722	3.7501	3.3056
MOVS	4.3056	3.6111	4.2501	3.5001
MOSG	4.2778	3.3889	2.2222	3.1667

Çizelge 5.12'deki sonuçlara bakıldığında MOSG algoritmasının Spread ve Epsilon metriklerinde en iyi, IGD metriğinde ikinci en iyi ortalama Friedman başarı sıralamasına sahip olduğu görülmektedir. Öte yandan MOAAA algoritmasının HV ve IGD metriklerinde en iyi, Epsilon metriğinde ikinci en iyi Friedman başarı sıralaması yakaladığı görülmektedir. Geriye kalan algoritmalar içerisinde MOVS ve MOCell algoritmaları sırası ile HV ve Spread metriklerinde en iyi ikinci Friedman sıralama başarısı yakaladıkları gözlemlenmektedir. Friedman testi sonuçlarına genel olarak

bakıldığında MOSG ve MOAAA algoritmalarının diğer algoritmalara üstünlük sağladığı, kendi aralarında rekabetçi sonuçlar ürettiği görülmektedir.

MOSG algoritması ve kıyas yapılan algoritmaların dört metrik üzerinde elde ettikleri değerler önceki çizelgelerde sunulmuştur. Bu değerlerin istatistiksel olarak birbirinden farklı olup olmadığını kontrol etmek amacıyla Wilcoxon sıra toplamı (*Wilcoxon rank sum* (Epanechnikov, 1969; Biswas ve ark., 2014)) testi uygulanmıştır. Wilcoxon istatistiksel testinde güven aralığı %95 olarak alınmıştır. MOSG algoritması ve kıyas algoritmaları ile yapılan Wilcoxon testi sonuçları HV, IGD, Spread ve Epsilon metrikleri için sırasıyla Çizelge 5.13 – 5.16'da verilmiştir. Çizelgelerde verilen p-değerinin 0.05 değerinden küçük olması MOSG tarafından üretilen sonuçların kıyas yapılan algoritmanın ürettiği sonuçlardan farklı ve istatistiksel olarak anlamlı olduğunu göstermektedir. Farklı ve anlamlı olan sonuçlar "+" simgesi ile verilmiştir.

Çizelge 5.13. Problem Seti-1: HV metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG	**NSGA-II**		**IBEA**		**MOCell**		**MOEA/D**		**MOAAA**		**MOVS**	
vs.	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
ZDT1	4.24E-29	+	4.00E-10	+	1.26E-17	+	2.44E-12	+	1.24E-17	+	3.41E-23	+
ZDT2	2.40E-17	+	2.58E-16	+	2.56E-34	+	2.56E-34	+	9.29E-26	+	6.93E-24	+
ZDT3	4.08E-20	+	5.00E-14	+	3.46E-06	+	2.56E-34	+	9.99E-30	+	3.04E-01	-
ZDT4	2.25E-28	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	7.22E-25	+	9.83E-27	+
ZDT6	5.64E-39	+	5.64E-39	+	2.56E-34	+	1.85E-34	+	2.56E-34	+	2.56E-34	+
WFG1	1.04E-31	+	9.26E-32	+	2.84E-31	+	4.08E-33	+	1.36E-24	+	9.26E-32	+
WFG2	2.56E-34	+	4.90E-18	+	7.75E-02	-	2.56E-34	+	2.56E-34	+	2.56E-34	+
WFG3	3.16E-34	+	2.89E-34	+	2.56E-34	+	7.31E-34	+	7.35E-32	+	2.98E-34	+
WFG4	2.56E-34	+	9.30E-33	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
WFG5	1.78E-01	-	1.26E-15	+	2.30E-05	+	5.33E-03	+	5.11E-30	+	7.57E-04	+
WFG6	5.02E-33	+	1.09E-03	+	1.40E-18	+	1.17E-17	+	2.56E-34	+	3.30E-29	+
WFG7	2.56E-34	+	1.20E-27	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	3.07E-34	+
WFG9	9.03E-33	+	5.82E-15	+	6.93E-24	+	3.25E-20	+	2.80E-34	+	1.93E-32	+
DTLZ1	6.95E-17	+	1.88E-20	+	5.01E-25	+	2.56E-34	+	1.97E-01	-	1.79E-07	+
DTLZ2	2.46E-30	+	5.45E-16	+	3.82E-25	+	2.36E-20	+	4.73E-12	+	1.60E-21	+
DTLZ4	2.13E-33	+	9.77E-22	+	4.12E-31	+	7.71E-22	+	3.87E-04	+	1.56E-25	+
DTLZ5	8.32E-05	+	1.32E-10	+	5.69E-05	+	4.98E-01	-	1.61E-23	+	5.70E-08	+
DTLZ6	2.56E-34	+	4.06E-22	+	2.21E-20	+	2.56E-34	+	3.88E-08	+	2.56E-34	+
DTLZ7	4.27E-34	+	8.58E-01	-	9.57E-03	+	1.48E-05	+	3.48E-20	+	7.36E-33	+
LZ09_F1	1.89E-29	+	3.26E-34	+	2.56E-34	+	2.56E-34	+	3.33E-18	+	6.18E-32	+
LZ09_F2	2.56E-34	+	2.56E-34	+	2.55E-34	+	2.55E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F3	6.04E-30	+	4.99E-02	+	2.56E-34	+	2.56E-34	+	2.25E-24	+	1.02E-32	+
LZ09_F4	7.99E-34	+	1.96E-02	+	5.83E-32	+	2.56E-34	+	9.58E-33	+	3.16E-34	+
LZ09_F5	2.56E-34	+	5.81E-33	+	2.56E-34	+	1.40E-32	+	2.56E-34	+	2.56E-34	+
LZ09_F6	4.13E-04	+	8.21E-02	-	4.76E-32	+	4.27E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F7	1.03E-18	+	2.98E-34	+	2.64E-34	+	2.56E-34	+	1.62E-32	+	3.78E-34	+
LZ09_F8	1.06E-29	+	2.56E-34	+	1.26E-24	+	8.52E-33	+	5.12E-26	+	3.78E-34	+
LZ09_F9	2.56E-34	+	3.28E-23	+	3.32E-33	+	2.56E-34	+	8.02E-28	+	9.10E-18	+
Fonseca	1.86E-07	+	1.49E-08	+	1.95E-14	+	5.81E-33	+	8.41E-25	+	2.06E-02	+
Kursawe	3.35E-22	+	2.56E-34	+	1.44E-05	+	1.95E-26	+	4.49E-21	+	1.82E-25	+
Schaffer	9.30E-02	-	3.70E-05	+	1.58E-02	+	9.10E-22	+	3.49E-01	-	1.66E-05	+
Viennet2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Viennet3	2.56E-34	+	2.72E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Poloni	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA1	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.55E-34	+	2.56E-34	+	1.02E-34	+
OKA2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+

Çizelge 5.13'teki, HV metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 36 fonksiyondan; NSGA-II, IBEA ve MOAAA algoritmaları için 34, MOCell, MOEA/D ve MOVS algoritmaları için 35 tanesinden farklı ve anlamlı sonuçlar ürettiği görülmektedir.

Çizelge 5.14. Problem Seti-1: IGD metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG	**NSGA-II**		**IBEA**		**MOCell**		**MOEA/D**		**MOAAA**		**MOVS**	
vs.	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
ZDT1	3.58E-29	+	9.11E-10	+	5.60E-20	+	6.33E-07	+	3.50E-17	+	4.29E-23	+
ZDT2	1.18E-30	+	8.66E-24	+	2.56E-34	+	2.56E-34	+	2.05E-26	+	1.96E-28	+
ZDT3	1.81E-07	+	5.82E-10	+	8.91E-23	+	2.56E-34	+	1.82E-10	+	1.77E-04	+
ZDT4	1.96E-23	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	1.15E-26	+	2.13E-21	+
ZDT6	2.56E-34	+	2.56E-34	+	2.56E-34	+	1.36E-35	+	2.56E-34	+	2.56E-34	+
WFG1	9.99E-01	-	4.93E-16	+	9.26E-32	+	9.26E-32	+	8.27E-01	-	1.77E-01	-
WFG2	1.21E-33	+	2.44E-32	+	1.94E-16	+	2.56E-34	+	5.39E-26	+	1.89E-33	+
WFG3	2.56E-34	+	8.03E-33	+	2.56E-34	+	2.56E-34	+	9.56E-34	+	2.56E-34	+
WFG4	2.56E-34	+	4.37E-11	+	6.88E-14	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
WFG5	1.47E-01	-	2.89E-24	+	2.07E-22	+	1.68E-10	+	3.32E-33	+	5.09E-04	+
WFG6	2.07E-17	+	1.13E-13	+	1.06E-06	+	4.14E-01	-	4.96E-34	+	1.85E-05	+
WFG7	2.09E-24	+	1.93E-06	+	5.01E-24	+	3.65E-30	+	3.56E-32	+	7.40E-17	+
WFG9	1.36E-15	+	1.44E-01	-	2.57E-01	-	9.56E-03	+	1.89E-33	+	7.69E-20	+
DTLZ1	2.03E-02	+	5.45E-05	+	2.98E-34	+	2.56E-34	+	2.10E-08	+	2.70E-11	+
DTLZ2	5.23E-19	+	8.23E-01	-	4.58E-01	-	8.21E-21	+	3.79E-16	+	1.32E-10	+
DTLZ4	7.47E-24	+	3.55E-02	+	7.95E-01	-	8.66E-24	+	5.39E-10	+	1.20E-18	+
DTLZ5	5.66E-06	+	1.34E-28	+	1.82E-18	+	2.16E-08	+	2.61E-31	+	3.57E-03	+
DTLZ6	2.49E-24	+	6.76E-03	+	2.64E-03	+	7.16E-23	+	1.22E-01	-	7.36E-16	+
DTLZ7	4.32E-30	+	1.69E-01	-	1.51E-17	+	5.08E-02	-	1.65E-17	+	6.76E-30	+
LZ09_F1	2.07E-31	+	4.87E-33	+	2.56E-34	+	2.56E-34	+	3.55E-30	+	3.78E-32	+
LZ09_F2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F3	5.86E-08	+	6.18E-03	+	2.56E-34	+	2.56E-34	+	4.24E-04	+	1.32E-11	+
LZ09_F4	8.59E-03	+	4.67E-01	-	2.56E-34	+	2.56E-34	+	1.45E-02	+	2.59E-13	+
LZ09_F5	2.56E-34	+	7.36E-33	+	2.72E-34	+	4.27E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F6	4.64E-01	-	1.39E-12	+	2.56E-34	+	5.43E-13	+	2.56E-34	+	2.56E-34	+
LZ09_F7	2.72E-29	+	2.56E-34	+	1.22E-10	+	8.14E-01	-	6.33E-18	+	1.38E-14	+
LZ09_F8	6.91E-03	+	2.56E-34	+	2.56E-34	+	7.48E-21	+	1.79E-10	+	1.00E-10	+
LZ09_F9	3.11E-20	+	4.05E-04	+	2.56E-34	+	1.17E-17	+	4.29E-21	+	5.66E-21	+
Fonseca	1.72E-17	+	1.06E-03	+	2.56E-34	+	9.46E-01	-	5.04E-01	-	2.19E-03	+
Kursawe	3.81E-01	-	2.56E-34	+	2.56E-34	+	1.05E-07	+	3.69E-04	+	1.16E-03	+
Schaffer	4.96E-34	+	4.66E-11	+	2.56E-34	+	2.56E-34	+	7.31E-34	+	7.53E-34	+
Viennet2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Viennet3	2.56E-34	+	1.28E-01	-	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Poloni	2.56E-34	+	9.03E-33	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA1	2.56E-34	+	3.67E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+

Çizelge 5.14'teki, IGD metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 36 fonksiyondan; algoritmaların tablodaki dizilişlerine göre sırasıyla 32, 31, 33, 32, 33 ve 35 tanesinden farklı ve anlamlı sonuçlar ürettiği görülmektedir.

Çizelge 5.15. Problem Seti-1: Spread metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG	**NSGA-II**		**IBEA**		**MOCell**		**MOEA/D**		**MOAAA**		**MOVS**	
vs.	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
ZDT1	4.55E-28	+	2.62E-33	+	2.56E-34	+	1.29E-33	+	4.42E-09	+	1.46E-16	+
ZDT2	1.13E-18	+	1.23E-19	+	1.72E-11	+	2.56E-34	+	1.25E-07	+	1.18E-06	+
ZDT3	5.44E-07	+	1.31E-05	+	1.73E-03	+	2.56E-34	+	6.38E-02	-	6.84E-01	-
ZDT4	7.31E-34	+	9.86E-33	+	2.56E-34	+	2.56E-34	+	1.45E-33	+	2.56E-34	+
ZDT6	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.55E-35	+	2.56E-34	+	2.56E-34	+
WFG1	4.32E-24	+	1.88E-04	+	9.49E-18	+	1.08E-32	+	4.59E-29	+	9.03E-09	+
WFG2	2.56E-34	+	9.81E-32	+	2.56E-34	+	4.87E-33	+	2.56E-34	+	2.56E-34	+
WFG3	2.56E-34	+	5.96E-03	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
WFG4	6.53E-17	+	1.27E-11	+	1.71E-22	+	3.67E-34	+	6.94E-33	+	5.54E-24	+
WFG5	1.09E-03	+	2.67E-01	-	5.04E-07	+	5.58E-34	+	1.60E-31	+	5.06E-11	+
WFG6	4.08E-33	+	2.44E-32	+	3.59E-19	+	5.32E-33	+	2.56E-34	+	4.47E-29	+
WFG7	3.87E-28	+	1.75E-26	+	6.56E-29	+	2.15E-15	+	4.37E-32	+	2.97E-20	+
WFG9	5.02E-33	+	1.80E-31	+	2.33E-31	+	8.40E-29	+	2.56E-34	+	4.00E-29	+
DTLZ1	1.45E-10	+	6.21E-02	-	1.23E-19	+	3.53E-23	+	9.36E-03	+	9.02E-04	+
DTLZ2	8.77E-33	+	8.66E-24	+	9.28E-17	+	2.41E-18	+	1.03E-23	+	1.45E-28	+
DTLZ4	2.55E-33	+	1.41E-33	+	6.35E-33	+	3.42E-33	+	4.57E-30	+	7.57E-32	+
DTLZ5	4.19E-01	-	5.45E-01	-	1.97E-04	+	2.17E-32	+	4.02E-15	+	1.04E-11	+
DTLZ6	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
DTLZ7	2.89E-34	+	8.24E-24	+	2.01E-28	+	2.64E-29	+	3.07E-34	+	2.56E-34	+
LZ09_F1	3.39E-04	+	1.96E-30	+	2.56E-34	+	2.56E-34	+	2.20E-02	+	3.52E-07	+
LZ09_F2	2.89E-34	+	9.53E-32	+	5.02E-33	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F3	4.86E-05	+	4.23E-06	+	1.18E-32	+	2.56E-34	+	3.61E-07	+	4.68E-02	+
LZ09_F4	1.53E-08	+	9.11E-01	-	1.11E-23	+	2.56E-34	+	3.74E-03	+	2.69E-14	+
LZ09_F5	6.94E-29	+	2.00E-27	+	4.04E-01	-	1.34E-16	+	1.31E-22	+	5.34E-19	+
LZ09_F6	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F7	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F8	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	3.85E-33	+	4.67E-34	+
LZ09_F9	2.56E-34	+	7.46E-07	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Fonseca	2.56E-34	+	6.21E-29	+	2.56E-34	+	2.56E-34	+	1.21E-33	+	5.42E-34	+
Kursawe	2.64E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	1.29E-33	+	2.56E-34	+
Schaffer	2.56E-34	+	1.24E-31	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Viennet2	2.56E-34	+	3.05E-01	-	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Viennet3	2.56E-34	+	1.45E-07	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Poloni	2.56E-34	+	2.34E-02	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA1	2.56E-34	+	2.90E-20	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA2	2.56E-34	+	9.58E-33	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+

Çizelge 5.15'teki, Spread metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 36 fonksiyondan; algoritmaların tablodaki dizilişlerine göre sırasıyla 35, 31, 35, 36, 35 ve 35 tanesinden farklı ve anlamlı sonuçlar ürettiği görülmektedir.

Çizelge 5.16. Problem Seti-1: Epsilon metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG	NSGA-II		IBEA		MOCell		MOEA/D		MOAAA		MOVS	
vs	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
ZDT1	8.94E-30	+	2.15E-07	+	3.87E-15	+	8.91E-12	+	1.93E-19	+	1.77E-25	+
ZDT2	2.48E-02	+	1.44E-02	+	5.17E-33	+	2.56E-34	+	3.24E-02	+	3.01E-02	+
ZDT3	1.04E-04	+	4.20E-10	+	9.94E-08	+	2.56E-34	+	1.34E-03	+	2.84E-03	+
ZDT4	6.88E-34	+	6.66E-16	+	2.56E-34	+	2.56E-34	+	1.25E-33	+	7.76E-34	+
ZDT6	2.56E-34	+	2.56E-34	+	2.56E-34	+	1.33E-34	+	2.56E-34	+	2.56E-34	+
WFG1	9.46E-01	-	1.01E-15	+	1.61E-11	+	9.26E-32	+	2.38E-01	-	7.47E-06	+
WFG2	7.51E-16	+	1.61E-27	+	1.14E-05	+	5.02E-33	+	1.48E-10	+	1.36E-15	+
WFG3	7.81E-09	+	1.58E-28	+	2.56E-34	+	2.34E-10	+	1.02E-04	+	4.35E-09	+
WFG4	2.56E-34	+	1.54E-03	+	4.76E-06	+	2.56E-34	+	2.56E-34	+	4.14E-34	+
WFG5	4.29E-02	+	2.36E-19	+	4.55E-09	+	3.63E-01	-	5.93E-21	+	2.31E-01	-
WFG6	8.75E-14	+	1.12E-16	+	2.34E-02	+	1.02E-02	+	5.11E-30	+	5.21E-01	-
WFG7	1.06E-24	+	9.58E-05	+	9.97E-01	-	2.11E-18	+	1.31E-31	+	8.67E-11	+
WFG9	1.71E-14	+	8.84E-03	+	5.78E-03	+	1.93E-01	-	5.17E-33	+	1.81E-11	+
DTLZ1	7.98E-04	+	2.24E-01	-	8.57E-05	+	2.75E-21	+	3.87E-22	+	9.12E-23	+
DTLZ2	8.90E-18	+	2.04E-01	-	6.68E-04	+	8.28E-14	+	3.05E-09	+	3.41E-11	+
DTLZ4	3.67E-19	+	1.41E-01	-	1.50E-04	+	2.28E-15	+	1.46E-06	+	1.57E-15	+
DTLZ5	9.43E-03	+	3.36E-23	+	6.66E-07	+	7.59E-02	-	7.99E-25	+	1.93E-13	+
DTLZ6	2.11E-15	+	1.20E-06	+	2.59E-08	+	1.53E-20	+	3.84E-02	+	1.84E-15	+
DTLZ7	1.90E-31	+	3.94E-01	-	3.27E-01	-	3.84E-05	+	2.02E-22	+	3.58E-29	+
LZ09_F1	9.98E-10	+	4.81E-34	+	2.56E-34	+	5.02E-33	+	2.68E-02	+	3.00E-15	+
LZ09_F2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
LZ09_F3	1.06E-09	+	8.66E-06	+	9.50E-03	+	2.56E-34	+	6.88E-14	+	1.40E-09	+
LZ09_F4	4.42E-02	+	7.30E-06	+	1.21E-14	+	2.56E-34	+	3.20E-02	+	1.60E-05	+
LZ09_F5	4.70E-30	+	1.55E-01	-	7.14E-33	+	1.67E-32	+	2.56E-34	+	2.56E-34	+
LZ09_F6	4.75E-01	-	1.39E-12	+	1.18E-30	+	1.35E-04	+	2.56E-34	+	2.72E-34	+
LZ09_F7	8.02E-28	+	2.56E-34	+	1.50E-26	+	8.83E-02	-	9.54E-22	+	5.42E-12	+
LZ09_F8	7.73E-06	+	9.03E-33	+	1.30E-17	+	1.02E-30	+	2.63E-04	+	9.39E-02	-
LZ09_F9	1.11E-04	+	1.19E-03	+	9.46E-03	+	1.13E-33	+	4.43E-01	-	9.19E-05	+
Fonseca	4.12E-31	+	1.17E-04	+	3.26E-34	+	2.82E-07	+	6.12E-24	+	1.25E-26	+
Kursawe	2.56E-34	+	2.72E-34	+	2.56E-34	+	8.02E-28	+	3.36E-34	+	5.58E-34	+
Schaffer	4.46E-33	+	1.19E-15	+	6.11E-34	+	2.80E-34	+	2.40E-33	+	4.81E-34	+
Viennet2	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Viennet3	2.56E-34	+	2.57E-29	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
Poloni	2.56E-34	+	3.32E-33	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA1	2.56E-34	+	2.80E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+	2.56E-34	+
OKA2	2.56E-34	+	2.56E-34	+	2.56E-34	+	7.36E-26	+	2.56E-34	+	2.56E-34	+

Çizelge 5.16'daki, Epsilon metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 36 fonksiyondan; algoritmaların tablodaki dizilişlerine göre sırasıyla 34, 34, 35, 35, 34 ve 35 tanesinden farklı ve anlamlı sonuçlar ürettiği görülmektedir.

Wilcoxon sıra toplamı testi sonuçlarına genel olarak bakıldığında MOSG algoritmasının ürettiği sonuçların kıyaslama yapılan algoritmaların ürettiği sonuçlardan farklı ve anlamlı olduğu açık bir şekilde söylenebilir.

Bu aşamaya kadar yapılan performans karşılaştırmalarında MOSG algoritması ve diğer altı algoritmanın birlikte ele alınması şeklinde olmuştur. Bu karşılaştırmalara ek olarak önerilen algoritmanın diğer altı algoritma ile bire bir kıyaslanarak ortaya çıkan

sonuçlar değerlendirmeye alınmıştır. Problem setindeki 36 fonksiyon üzerinde yapılan 100 çalıştırmanın ortalama sonuçları (Çizelge 5.8 – 5.11) kullanılarak bu bire bir karşılaştırmalar dört metrik için de yapılmıştır. Bu karşılaştırma sonuçlarının olduğu çizelgelerde MOSG algoritması söz konusu fonksiyon için, kıyaslanan algoritmadan daha iyi sonuç üretti ise "+", eşit sonuç üretti ise "=" ve daha kötü sonuç üretti ise "–" simgesi kullanılmıştır. Çizelgelerin en alt satırında MOSG algoritmasının diğer algoritmalar ile kıyaslanması sonucunda galip (g), berabere (b) ve mağlup (m) olduğu fonksiyon sayısı verilmiştir.

MOSG algoritmasının HV metriği için diğer algoritmalar ile yapılan bire bir kıyaslama sonuçları Çizelge 5.17'de verilmiştir. MOSG algoritması MOAAA ve MOVS algoritması ile kıyaslandığında bu algoritmalara 17 fonksiyonda galip gelirken 2 fonksiyonda berabere kalıp 17 fonksiyonda ise mağlup olmuştur. Friedman testi sonuçlarına bakıldığında (Çizelge 5.12) bu iki algoritmanın da MOSG algoritmasından daha iyi ortalama başarıya sahip olduğu görülmektedir. Ancak bire bir karşılaştırmada bu metrik için MOSG algoritmasının bu iki algoritma ile eş değer sonuçlar ürettiği görülmektedir. Diğer algoritmalar ile yapılan bire bir karşılaştırma sonuçlarına bakıldığında MOSG algoritmasının bariz bir üstünlüğünün olduğu görülmektedir.

Çizelge 5.17. Problem Seti-1: HV metriği için MOSG algoritmasının diğer algoritmalarla bire bir karşılaştırma sonuçları

MOSG vs.	NSGA-II	IBEA	MOCell	MOEA/D	MOAAA	MOVS
ZDT1	+	=	+	+	+	+
ZDT2	+	+	+	+	+	+
ZDT3	+	+	+	+	+	+
ZDT4	+	+	+	+	+	+
ZDT6	+	+	+	=	+	+
WFG1	-	-	-	-	-	-
WFG2	-	+	+	-	-	-
WFG3	-	-	-	-	-	-
WFG4	-	-	-	+	-	-
WFG5	+	+	+	+	+	+
WFG6	+	+	+	-	-	=
WFG7	-	=	-	-	-	-
WFG9	=	=	=	+	=	+
DTLZ1	-	-	-	-	-	-
DTLZ2	+	-	+	+	+	+
DTLZ4	+	+	+	+	+	+
DTLZ5	+	+	-	+	+	+
DTLZ6	+	+	+	+	+	+
DTLZ7	+	+	+	+	+	+
LZ09_F1	-	-	-	-	-	-
LZ09_F2	+	+	+	-	-	-
LZ09_F3	-	-	-	-	-	-
LZ09_F4	-	-	-	-	-	-
LZ09_F5	-	-	-	-	-	-
LZ09_F6	+	+	+	+	+	-
LZ09_F7	-	-	-	-	-	-
LZ09_F8	-	-	-	-	-	-
LZ09_F9	+	+	+	+	-	-
Fonseca	+	=	-	-	+	+
Kursawe	+	+	=	+	+	+
Schaffer	+	+	+	+	+	+
Viennet2	+	+	+	+	+	+
Viennet3	+	+	=	+	+	+
Poloni	=	+	=	+	=	=
OKA1	-	-	-	-	-	-
OKA2	-	-	-	-	-	-
#(g/b/m)	**20/2/14**	**19/4/13**	**17/4/15**	**19/1/16**	**17/2/17**	**17/2/17**

MOSG algoritmasının IGD metriği için diğer algoritmalar ile yapılan bire bir kıyaslama sonuçları Çizelge 5.18'de verilmiştir. MOSG algoritmasın MOAAA ile bire bir karşılaştırılması ile elde edilen sonuçlara göre 17 fonksiyonda daha iyi ortalama sonuçlar üretirken 19 fonksiyonda daha kötü ortalama sonuçlar ürettiği görülmektedir. Bununla beraber MOVS algoritması ile yapılan bire bir kıyaslama 18 fonksiyonda daha iyi 18 fonksiyonda ise daha kötü ortalama sonuçlar ortaya koymuştur. Geriye kalan dört algoritma ile yapılan bire bir karşılaştırmalarda baskın olarak daha iyi ortalama sonuçlar ürettiği görülmektedir.

Çizelge 5.18. Problem Seti-1: IGD metriği için MOSG algoritmasının diğer algoritmalarla bire bir karşılaştırma sonuçları

MOSG vs.	NSGA-II	IBEA	MOCell	MOEA/D	MOAAA	MOVS
ZDT1	+	+	+	+	+	+
ZDT2	+	+	=	+	+	+
ZDT3	+	+	+	+	+	+
ZDT4	+	+	+	+	+	+
ZDT6	+	+	+	+	+	+
WFG1	-	-	-	-	-	-
WFG2	+	+	+	+	-	-
WFG3	+	-	-	=	+	+
WFG4	-	+	-	+	-	-
WFG5	+	+	-	+	+	+
WFG6	+	+	+	-	-	+
WFG7	-	+	-	-	-	-
WFG9	+	+	-	+	+	+
DTLZ1	-	-	-	-	-	-
DTLZ2	+	+	+	+	+	+
DTLZ4	+	+	+	+	+	+
DTLZ5	-	+	-	+	-	-
DTLZ6	+	+	+	+	+	+
DTLZ7	-	+	+	+	-	-
LZ09_F1	-	-	-	-	-	-
LZ09_F2	+	+	+	-	-	-
LZ09_F3	-	+	+	+	-	-
LZ09_F4	-	+	-	-	-	-
LZ09_F5	-	+	+	+	-	-
LZ09_F6	-	+	+	+	-	-
LZ09_F7	-	-	-	-	-	-
LZ09_F8	-	-	+	-	-	-
LZ09_F9	+	+	+	-	-	-
Fonseca	+	+	-	-	+	+
Kursawe	+	+	-	+	+	+
Schaffer	+	+	+	+	+	+
Viennet2	+	+	-	+	+	+
Viennet3	+	+	+	+	+	+
Poloni	+	+	-	+	+	+
OKA1	-	+	+	-	-	-
OKA2	-	+	+	+	-	-
#(g/b/m)	**21/0/15**	**30/0/6**	**20/1/15**	**23/1/12**	**17/0/19**	**18/0/18**

MOSG algoritmasının Spread metriği için diğer algoritmalar ile yapılan bire bir kıyaslama sonuçları Çizelge 5.19'da verilmiştir. MOSG algoritmasının bu metrik için diğer algoritmalar ile yapılan bire bir karşılaştırma sonuçlarına göre tüm algoritmalardan genel olarak daha başarılı ortalama sonuçlar ürettiği görülmektedir. Bununla beraber MOSG algoritmasının 19 fonksiyonda galip geldiği 17 fonksiyonda ise mağlup olduğu MOCell algoritması en yakın ortalama sonuçlara sahip olmuştur. MOCell algoritmasının Spread metriği ile yapılan performans kıyaslamalarında genel olarak başarılı olduğu düşünülünce MOSG algoritmasının başarısı ön plana çıkmaktadır.

Çizelge 5.19. Problem Seti-1: Spread metriği için MOSG algoritmasının diğer algoritmalarla bire bir karşılaştırma sonuçları

MOSG vs.	NSGA-II	IBEA	MOCell	MOEA/D	MOAAA	MOVS
ZDT1	+	+	+	+	+	+
ZDT2	+	+	+	+	+	+
ZDT3	+	+	+	+	+	+
ZDT4	+	+	+	+	+	+
ZDT6	+	+	+	+	+	+
WFG1	-	-	-	+	-	-
WFG2	-	+	-	+	-	-
WFG3	+	+	-	+	+	+
WFG4	+	+	-	+	+	+
WFG5	+	+	-	+	+	+
WFG6	+	+	-	+	+	+
WFG7	+	+	-	+	+	+
WFG9	+	+	-	+	+	+
DTLZ1	+	+	+	+	+	+
DTLZ2	+	-	+	+	+	+
DTLZ4	+	+	+	+	+	+
DTLZ5	+	+	-	+	+	+
DTLZ6	+	+	+	+	+	+
DTLZ7	+	+	+	+	+	+
LZ09_F1	-	+	-	-	-	-
LZ09_F2	+	+	-	-	-	+
LZ09_F3	-	+	-	-	-	-
LZ09_F4	-	+	-	+	-	-
LZ09_F5	-	+	-	-	-	-
LZ09_F6	+	+	+	+	+	+
LZ09_F7	+	+	+	+	+	+
LZ09_F8	+	+	+	+	+	+
LZ09_F9	+	+	+	-	-	+
Fonseca	+	+	-	+	+	+
Kursawe	+	+	-	+	+	+
Schaffer	+	+	+	+	+	+
Viennet2	+	+	+	+	+	+
Viennet3	+	+	+	+	+	+
Poloni	+	+	-	+	+	+
OKA1	+	+	+	+	+	+
OKA2	+	+	+	+	+	+
#(g/b/m)	**30/0/6**	**34/0/2**	**19/0/17**	**31/0/5**	**28/0/8**	**30/0/6**

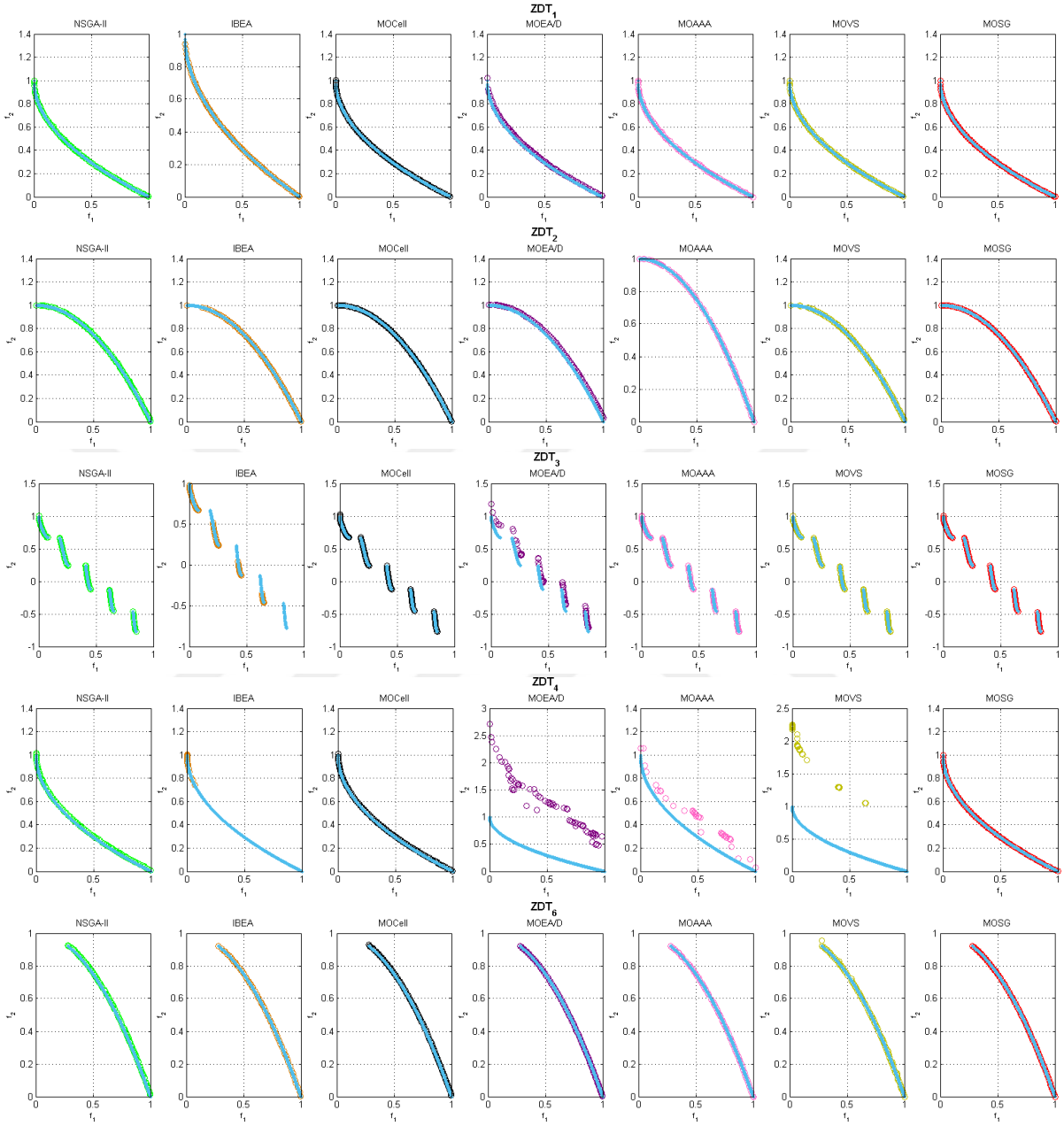
MOSG algoritmasının Epsilon metriği için diğer algoritmalar ile yapılan bire bir kıyaslama sonuçları Çizelge 5.20'de verilmiştir. Çizelgedeki sonuçlara bakıldığında MOSG algoritmasının bu metrik için yapılan bire bir kıyaslamada diğer algoritmalara açık bir şekilde üstünde sağladığı görülmektedir.

Çizelge 5.20. Problem Seti-1: Epsilon metriği için MOSG algoritmasının diğer algoritmalarla bire bir karşılaştırma sonuçları

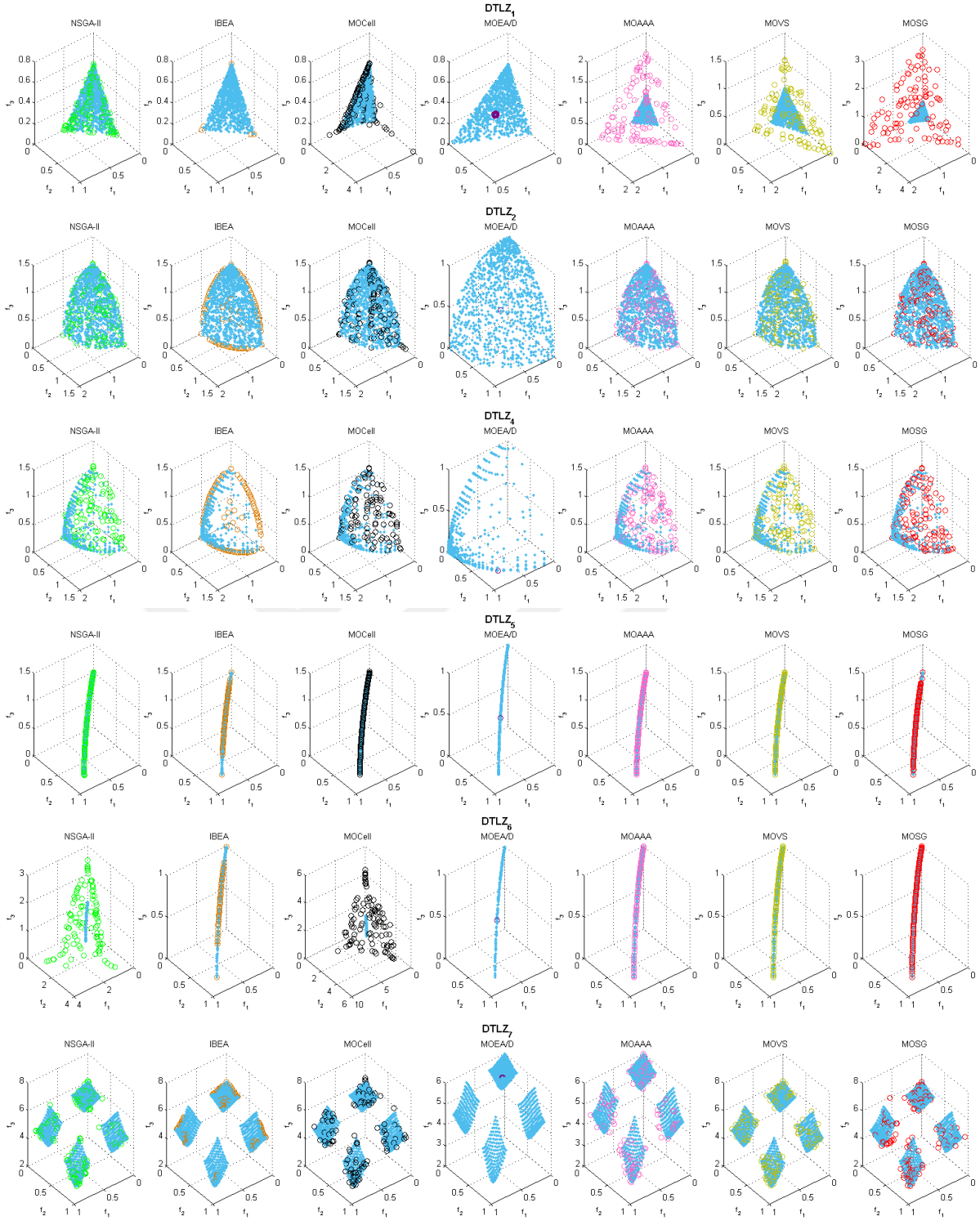
MOSG vs.	NSGA-II	IBEA	MOCell	MOEA/D	MOAAA	MOVS
ZDT1	+	+	+	+	+	+
ZDT2	+	+	+	+	+	+
ZDT3	+	+	+	+	+	+
ZDT4	+	+	+	+	+	+
ZDT6	+	+	+	+	+	+
WFG1	-	-	-	-	-	-
WFG2	+	+	+	-	-	-
WFG3	+	-	-	-	+	+
WFG4	+	+	-	+	+	-
WFG5	-	-	-	+	-	-
WFG6	+	+	+	+	+	+
WFG7	+	+	-	+	+	+
WFG9	+	+	-	+	+	+
DTLZ1	-	-	-	-	-	-
DTLZ2	+	-	+	+	+	+
DTLZ4	+	+	+	+	+	+
DTLZ5	+	+	-	+	+	+
DTLZ6	+	+	+	+	+	+
DTLZ7	-	+	+	+	-	-
LZ09_F1	-	-	-	-	-	-
LZ09_F2	+	+	+	+	-	-
LZ09_F3	-	+	+	+	-	+
LZ09_F4	-	-	-	-	-	-
LZ09_F5	-	+	+	+	-	-
LZ09_F6	-	+	+	+	-	-
LZ09_F7	-	-	-	-	-	-
LZ09_F8	-	-	+	-	-	-
LZ09_F9	+	+	+	+	-	-
Fonseca	+	-	-	-	+	+
Kursawe	+	+	-	+	+	+
Schaffer	+	+	+	+	+	+
Viennet2	+	+	+	+	+	+
Viennet3	+	+	+	+	+	+
Poloni	+	+	-	+	+	+
OKA1	-	+	+	-	+	-
OKA2	-	+	+	+	-	-
#(g/b/m)	**23/0/13**	**26/0/10**	**22/0/14**	**27/0/9**	**21/0/15**	**20/0/16**

Çok amaçlı optimizasyon problemlerinde, algoritmaların temel amacının PF_t çözümlerine en yakın çözümler kümesini elde etmek olduğu bilinmektedir. Gerek çeşitlilik gerekse yakınsama bakımından PF_t'ye yakın çözümler elde eden algoritmaların daha başarılı olduğu söylenebilir. Algoritmaların elde ettiği *PF* çözümlerinin PF_t çözümleri ile görsel olarak karşılaştırılmasını sağlamak amacıyla her algoritma için grafik oluşturulmuştur. ZDT ve DTLZ problem ailesi için algoritmaların ürettiği *PF* çözümleri ve PF_t çözümleri sırasıyla Şekil 5.1 ve 5.2'de verilmiştir. *PF* ve PF_t'lerin birlikte verildiği şekillerde; mavi renk PF_t çözümlerini, diğer renk ise algoritma tarafından üretilen *PF* çözümlerini temsil etmektedir. MOSG algoritması ile

beraber diğer algoritmaların 36 fonksiyondan geriye kalanlar için ürettiği *PF* çözümleri ve PF_t çözümlerinin görselleri EK-1'de verilmiştir.



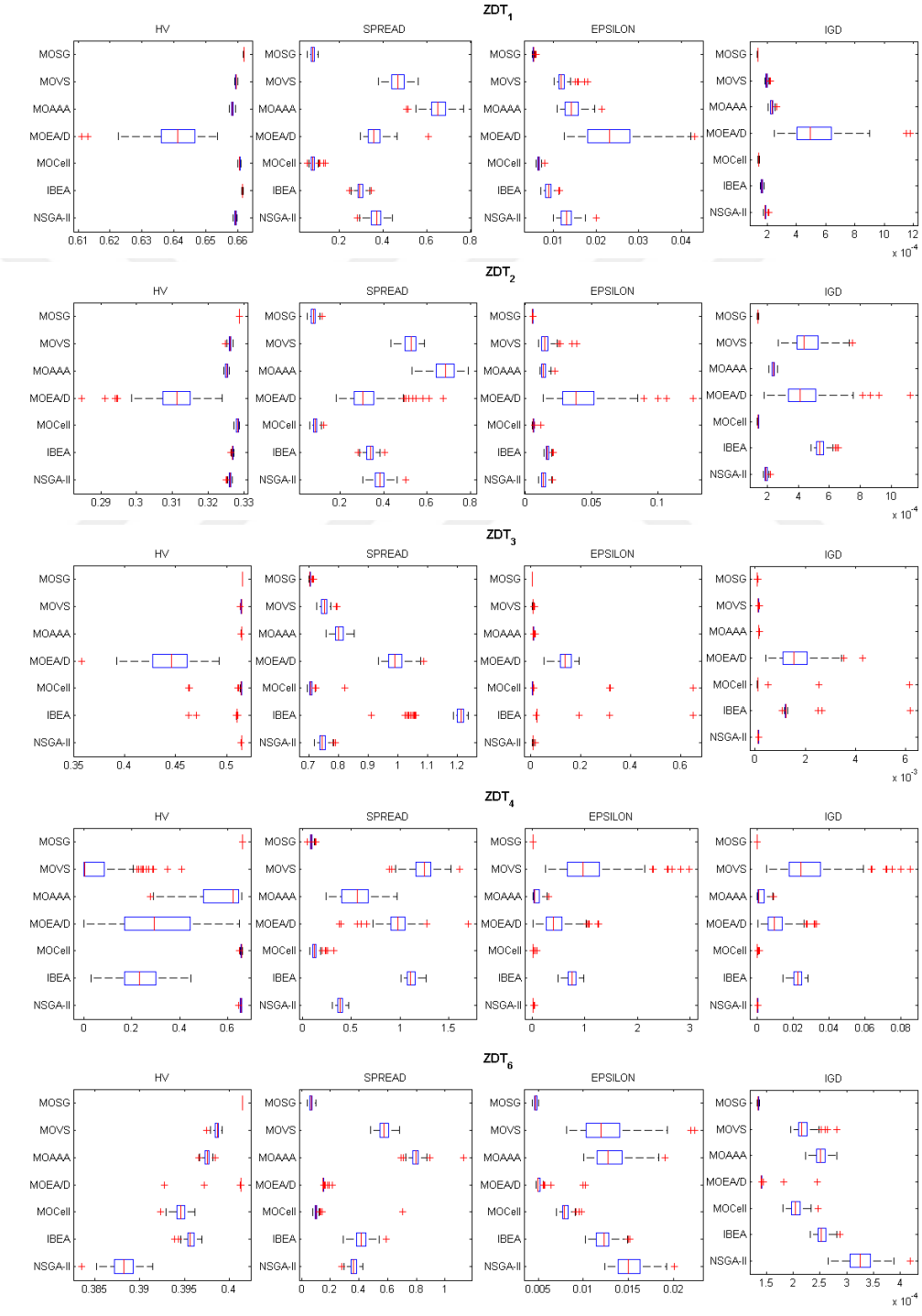
Şekil 5.1. ZDT problem ailesi için algoritmaların ürettiği *PF*'lerin PF_t'ler ile karşılaştırılması



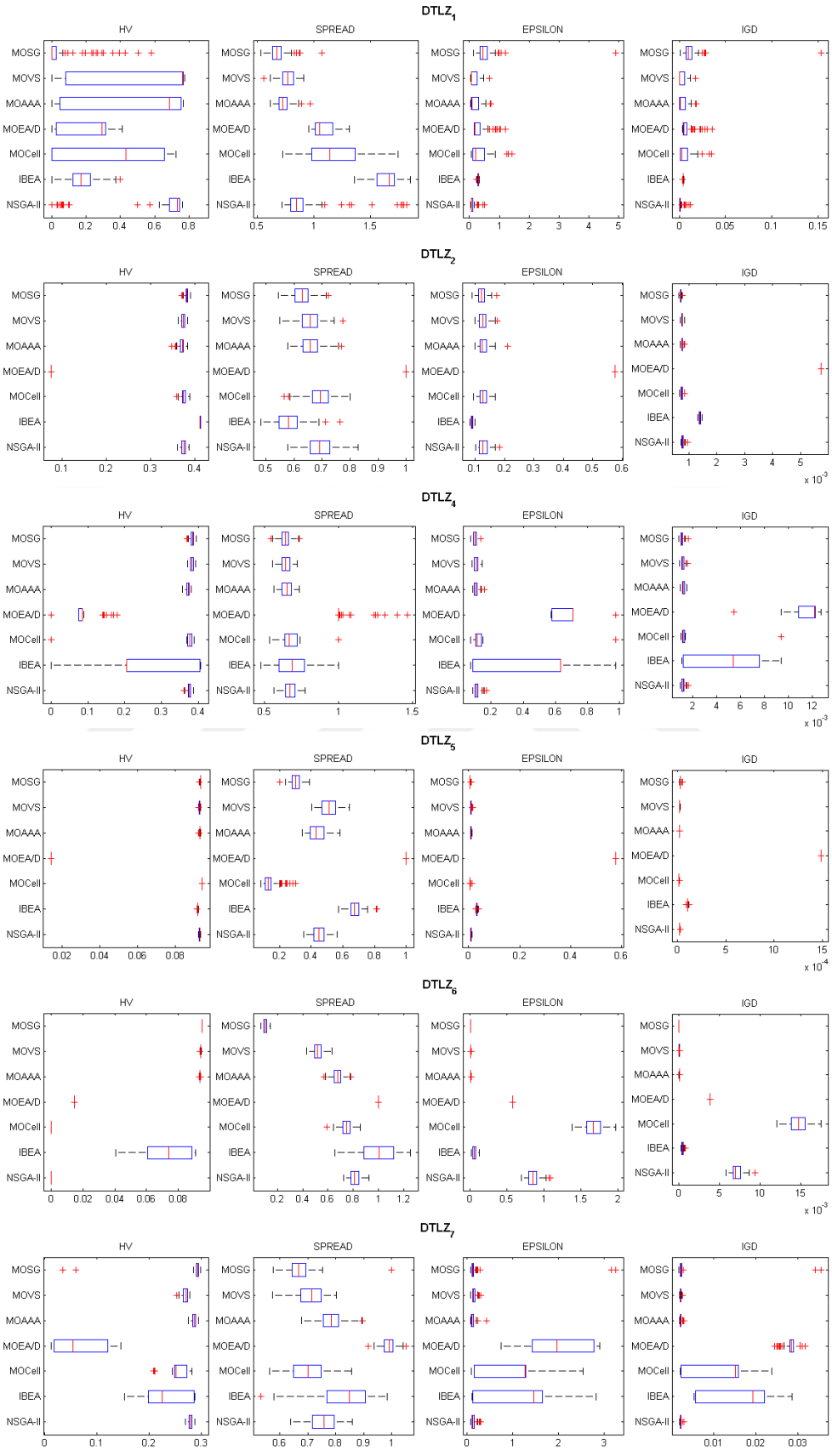
Şekil 5.2. DTLZ problem ailesi için algoritmaların ürettiği *PF*'lerin PF_t'ler ile karşılaştırılması

Öte yandan bir veri seti hakkında bilgi sahibi olabilmek için kullanılan görsel yöntemlerden bir tanesi de kutu grafiği (*box plot*) yaklaşımıdır. MOSG ve diğer algoritmalar bu problem setindeki 36 fonksiyon için 100 tekrar ile çalıştırılmıştır. Bu 100 çalıştırmanın hakkında genel bilgi edinmek adına algoritmaların ürettiği sonuçların kutu grafikleri çizdirilmiştir. Şekil 5.3 ve 5.4, algoritmaların sırasıyla ZDT ve DTLZ

problem ailesi üzerinde elde ettikleri 100 çalıştırma sonuçlarının kutu grafikleri çizimini göstermektedir. Bu çizimlerde algoritmaların elde ettikleri sonuçların kullanılan dört metrik için karşılıkları ele alınmıştır. Algoritmaların elde ettiği sonuçlara göre çizilen kutu grafiklerine bakıldığında MOSG algoritmasının genel olarak kararlı bir duruş sergilediği görülmektedir. MOSG algoritması ve diğer algoritmaların 36 fonksiyondan geriye kalanlar için ürettiği sonuçların kutu grafikleri EK-1'de verilmiştir.



Şekil 5.3. ZDT problem ailesi için algoritmaların ürettiği sonuçların kutu grafikleri



Şekil 5.4. DTLZ problem ailesi için algoritmaların ürettiği sonuçların kutu grafikleri

5.2.2. Problem seti -2 deney sonuçları

Mühendislik tasarımı ve kısıtlı fonksiyonlardan oluşan bu problem setinde MOSG algoritmasının performansı dört farklı çok amaçlı algoritma (NSGA-II, IBEA, MOCell, PAES) ile karşılaştırılmıştır. Bu problem seti üzerinde yapılan deneylerde MAXFes değeri 5000 ile sınırlı tutulmuştur. Bunun nedeni kullanılan problemlerin karar değişken sayılarının az olmasıdır. Algoritmalar her problem için 100 tekrar ile çalıştırılmıştır. Performans karşılaştırması için bir önceki problem setinde kullanılan dört metrik (HV, IGD, Spread, Epsilon) kullanılmıştır. Algoritmaların bu problem seti üzerinde 100 tekrarlı çalıştırma sonucu ürettikleri ortalama ve standart sapma değerleri Çizelge 5.21-5.24'te verilmiştir. Kullanılan problem setindeki her bir problem için en iyi ortalama değeri bulan algoritmanın bulduğu değer koyu gri ile işaretlenirken ikinci en iyi ortalama değer ise açık gri ile işaretlenmiştir.

Çizelge 5.21'deki sonuçlar algoritmaların problem seti-2 üzerindeki çalışmalarının HV metriğindeki karşılıklarını göstermektedir. HV metriği üzerinde elde edilen sonuçlara bakıldığında MOSG algoritmasının 10 problemden 9 tanesi için en iyi ortalama değerleri üretirken 1 tane problem için de en iyi ikinci ortalama değeri ürettiği görülmektedir. Diğer algoritmaların bu problemler üzerinde en iyi ve ikinci en iyi ortalama değerleri bulma sayıları tablodaki sıralamalarına göre sırasıyla [1/5], [1/1], [1/5] ve [0/0] şeklindedir. HV metriği üzerinde elde edilen ortalama sonuçlara bakıldığında MOSG algoritmasının diğer algoritmalara bariz bir üstünlük sağladığı görülmektedir.

Çizelge 5.21. Problem Seti-2: HV metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II		**IBEA**		**MOCell**		**PAES**		**MOSG**	
	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.
Binh2	8.10E-01	3.2E-04	8.08E-01	1.8E-03	8.11E-01	7.1E-05	8.07E-01	3.1E-03	8.12E-01	4.3E-05
ConstrEx	7.73E-01	4.5E-04	7.59E-01	5.1E-03	7.71E-01	3.0E-03	7.36E-01	3.9E-02	7.73E-01	2.2E-03
Srinivas	5.32E-01	3.2E-04	5.34E-01	1.6E-04	5.34E-01	1.4E-04	5.29E-01	8.7E-04	5.35E-01	6.9E-05
KITA	6.42E-01	1.4E-03	6.36E-01	2.0E-03	6.41E-01	4.0E-03	6.22E-01	1.7E-02	6.45E-01	1.8E-03
Water	3.90E-01	9.7E-03	2.60E-01	5.3E-02	4.03E-01	8.5E-03	2.91E-01	8.3E-02	4.32E-01	1.7E-03
FourBarTruss	1.82E-01	9.4E-04	1.86E-01	3.0E-04	1.85E-01	5.8E-04	1.57E-01	4.6E-02	1.85E-01	2.8E-04
DiskBrake	8.73E-01	1.4E-03	8.69E-01	3.7E-03	8.71E-01	4.1E-03	8.70E-01	5.0E-03	8.77E-01	3.8E-04
CentileverBeam	8.71E-01	2.9E-04	7.91E-01	3.0E-02	8.72E-01	3.4E-05	8.70E-01	3.7E-04	8.72E-01	2.5E-05
Spring	7.46E-01	6.5E-03	6.70E-01	3.8E-02	7.29E-01	1.6E-02	6.99E-01	3.2E-02	7.63E-01	9.0E-04
GearTrain	9.02E-01	1.3E-02	9.00E-01	6.5E-03	8.93E-01	1.9E-02	7.85E-01	1.3E-01	9.05E-01	1.1E-06

Çizelge 5.22'deki sonuçlar algoritmaların problem seti-2 üzerindeki çalışmalarının IGD metriğindeki karşılıklarını göstermektedir. Çizelge 5.22'deki

sonuçlara bakıldığında MOSG algoritmasının 10 problemden 9 tanesi için en iyi ortalama değerleri üretirken 1 tane problem için de en iyi ikinci ortalama değeri ürettiği görülmektedir. Diğer algoritmaların bu problemler üzerinde ortalama en iyi ve ikinci en iyi değerleri bulma sayıları tablodaki sıralamalarına göre sırasıyla [1/5], [1/0], [0/5] ve [0/0] şeklindedir. IGD metriği üzerinde elde edilen ortalama sonuçlara bakıldığında MOSG algoritmasının diğer algoritmalardan daha iyi ortalama sonuçlar ürettiği açıkça görülmektedir.

Çizelge 5.22. Problem Seti-2: IGD metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II		**IBEA**		**MOCell**		**PAES**		**MOSG**	
	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.
Binh2	1.50E-04	8.9E-06	1.42E-03	4.2E-04	1.11E-04	3.2E-06	6.85E-04	5.9E-04	1.07E-04	2.0E-06
ConstrEx	2.01E-04	3.5E-05	4.40E-03	1.4E-03	3.21E-04	2.3E-04	2.40E-03	2.2E-03	2.47E-04	1.1E-04
Srinivas	1.44E-04	7.7E-06	1.25E-04	8.9E-06	1.04E-04	4.9E-06	2.12E-04	3.5E-05	9.85E-05	1.9E-06
KITA	5.44E-04	6.2E-05	9.65E-04	1.9E-04	6.43E-04	2.6E-04	1.63E-03	1.1E-03	5.19E-04	1.4E-04
Water	2.51E-03	1.4E-04	1.52E-02	1.9E-03	2.35E-03	1.9E-04	9.62E-03	3.0E-03	1.99E-03	1.1E-04
FourBarTruss	2.02E-02	2.1E-05	2.01E-02	1.3E-05	2.02E-02	3.4E-05	2.22E-02	4.0E-03	2.01E-02	5.8E-06
DiskBrake	8.60E-04	5.8E-04	4.92E-03	3.9E-04	2.08E-03	1.1E-03	2.60E-03	1.1E-03	2.89E-04	3.0E-05
CentileverBeam	3.09E-04	2.3E-05	1.14E-02	1.5E-03	2.24E-04	4.2E-06	4.15E-04	5.4E-05	2.23E-04	4.8E-06
Spring	2.85E-03	1.1E-03	1.40E-02	4.2E-03	5.83E-03	3.1E-03	9.36E-03	4.9E-03	1.43E-03	8.6E-04
GearTrain	1.31E-02	7.0E-03	3.28E-02	6.2E-03	2.10E-02	9.1E-03	3.71E-02	2.0E-02	6.82E-03	3.6E-03

Çizelge 5.23'teki sonuçlar MOSG ve diğer algoritmaların problem seti-2 üzerindeki çalışmalarının Spread metriğindeki karşılıklarını göstermektedir. Spread metriği üzerinde elde edilen sonuçlara bakıldığında MOSG algoritmasının 10 problemden 8 tanesi için en iyi ortalama değerleri üretirken 1 tane problem için de en iyi ikinci ortalama değeri ürettiği görülmektedir. Diğer algoritmaların bu problemler üzerinde en iyi ve ikinci en iyi ortalama değerleri bulma sayıları tablodaki sıralamalarına göre sırasıyla [0/1], [0/0], [2/8] ve [0/0] şeklindedir. Spread metriği üzerinde elde edilen ortalama sonuçlara bakıldığında MOSG algoritmasının diğer algoritmalara bariz bir üstünlük sağladığı görülmektedir. Bununla beraber Spread metriği üzerinde genellikle iyi sonuçlar elde eden MOCell algoritması, MOSG algoritmasından sonra diğer üç algoritmaya üstünlük sağlamıştır.

Çizelge 5.23. Problem Seti-2: Spread metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II		IBEA		MOCell		PAES		MOSG	
	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.
Binh2	3.98E-01	3.5E-02	5.40E-01	1.4E-02	1.59E-01	1.7E-02	6.79E-01	4.7E-02	1.29E-01	1.6E-02
ConstrEx	4.53E-01	3.2E-02	1.11E+00	4.1E-02	3.80E-01	7.9E-02	8.84E-01	8.3E-02	2.91E-01	5.2E-02
Srinivas	4.00E-01	3.4E-02	3.61E-01	2.9E-02	1.17E-01	1.5E-02	6.28E-01	5.0E-02	7.64E-02	1.1E-02
KITA	6.74E-01	1.3E-01	1.19E+00	4.2E-02	6.09E-01	1.6E-01	1.17E+00	9.4E-02	5.81E-01	1.6E-01
Water	5.60E-01	5.2E-02	7.51E-01	1.3E-01	5.57E-01	4.2E-02	6.60E-01	6.3E-02	5.79E-01	3.7E-02
FourBarTruss	6.57E-01	1.9E-02	6.58E-01	1.6E-02	5.47E-01	1.8E-02	8.18E-01	4.5E-02	5.20E-01	8.7E-03
DiskBrake	5.35E-01	5.4E-02	7.54E-01	4.2E-02	5.10E-01	4.9E-02	7.84E-01	5.9E-02	3.23E-01	3.8E-02
CentileverBeam	4.03E-01	3.6E-02	6.71E-01	4.1E-02	1.28E-01	1.6E-02	6.40E-14	4.4E-02	1.24E-01	1.7E-02
Spring	1.26E+00	6.3E-02	1.17E+00	1.1E-01	6.42E-01	8.3E-02	1.45E+00	9.3E-02	3.29E-01	3.5E-02
GearTrain	1.44E+00	6.4E-02	1.43E+00	8.3E-02	7.72E-01	3.8E-02	1.31E+00	1.2E-01	7.76E-01	3.0E-02

Çizelge 5.24'teki sonuçlar algoritmaların problem seti-2 üzerindeki çalışmalarının Epsilon metriğindeki karşılıklarını göstermektedir. Çizelge 5.22'deki sonuçlara bakıldığında MOSG algoritmasının 10 problemden 7 tanesi için en iyi ortalama değerleri üretirken 3 tane problem için de en iyi ikinci ortalama değerleri ürettiği görülmektedir. Diğer algoritmaların bu problemler üzerinde ortalama en iyi ve ikinci en iyi değerleri bulma sayıları tablodaki sıralamalarına göre sırasıyla [2/3], [1/1], [0/3] ve [0/0] şeklindedir. Epsilon metriği üzerinde elde edilen ortalama sonuçlara bakıldığında MOSG algoritmasının diğer algoritmalardan daha iyi ortalama sonuçlar ürettiği görülmektedir.

Çizelge 5.24. Problem Seti-2: Epsilon metriği için algoritmaların ortalama ve standart sapma değerleri

	NSGA-II		IBEA		MOCell		PAES		MOSG	
	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.	Ortalama	Std. Sap.
Binh2	1.01E+00	2.5E-01	1.08E+02	2.8E-01	4.69E-01	5.5E-02	1.60E+00	1.4E+00	4.06E-01	1.9E-02
ConstrEx	1.65E-02	2.5E-03	3.95E-01	1.2E-01	2.25E-02	8.2E-03	1.08E-01	1.6E-01	1.84E-02	7.6E-03
Srinivas	3.21E+00	5.7E-01	2.21E+00	4.4E-01	1.79E+00	3.0E-01	5.49E+00	1.7E+00	1.42E+00	1.6E-01
KITA	4.72E-02	8.2E-03	7.38E-02	1.1E-02	5.32E-02	1.6E-02	1.50E-01	8.5E-02	4.79E-02	1.3E-02
Water	8.23E+04	1.8E+04	1.09E+06	3.6E+05	5.63E+04	1.5E+04	8.52E+05	6.9E+05	2.39E+04	2.5E+03
FourBarTruss	6.90E+01	5.7E+00	7.03E+01	1.3E+01	9.14E+01	2.5E+01	2.06E+02	1.4E+02	6.36E+01	0.0E+00
DiskBrake	7.34E-02	1.9E-02	2.81E-01	8.4E-02	1.40E-01	7.8E-02	1.21E-01	1.3E-01	3.12E-02	4.6E-03
CentileverBeam	7.00E-04	6.8E-04	2.94E-04	8.0E-05	5.59E-04	5.7E-04	4.52E-03	3.5E-03	3.09E-04	2.9E-04
Spring	2.58E+03	3.0E+03	2.71E+04	1.0E+04	8.59E+03	7.9E+03	1.61E+04	1.2E+04	2.41E+02	2.1E+02
GearTrain	7.22E-01	1.0E+00	6.65E-01	8.4E-01	1.56E+00	1.5E+00	6.29E+00	5.6E+00	4.30E-06	2.9E-05

Çizelge 5.21-5.24'teki sonuçlar MOSG ve diğer algoritmaların, dört metrik üzerinde, ortalama değerlerini göstermektedir. Algoritmaların her problem üzerindeki 100 bağımsız çalıştırılması değerlendirilerek elde edilen Friedman testi sonuçları ise Çizelge 5.25'te verilmiştir. Her bir metrik için en iyi Friedman testi sonucuna sahip algoritmanın değeri koyu gri, ikinci en iyi değere sahip algoritmanın değeri ise açık gri

ile işaretlenmiştir. Çizelge 5.25'teki sonuçlara bakıldığında MOSG algoritmasının dört metrik için de en iyi Friedman sıralamasına sahip olduğu görülmektedir. Öte yandan NSGA-II ve MOCell algoritmaları HV ve IGD metrikleri için ikinci en iyi dereceyi paylaşırken Spread metriğinde MOCell, Epsilon metriğinde NSGA-II ikinci en iyi Friedman skoruna sahiptir.

Çizelge 5.25. Problem Seti-2: Algoritmaların dört metrik üzerindeki Friedman test sonuçları

	HV	**IGD**	**Spread**	**Epsilon**
NSGA-II	3.30	2.60	3.30	2.50
IBEA	2.10	4.20	4.20	3.70
MOCell	3.30	2.60	1.80	3.00
PAES	1.40	4.40	4.40	4.50
MOSG	4.90	1.20	1.30	1.30

MOSG algoritması ve diğer algoritmaların dört metrik için elde ettikleri ortalama değerler ve bu değerlere bağlı Friedman sonuçları önceki çizelgelerde gösterilmiştir. Elde edilen bu ortalama sonuçların birbirinden farklı ve anlamlı sonuçlar olup olmadığını kontrol etmek amacıyla Wilcoxon sıra toplamı testi uygulanmıştır. %95 güven aralığı ile yapılan Wilcoxon testi sonuçları HV, IGD, Spread ve Epsilon metrikleri için sırasıyla Çizelge 5.26-5.29'de verilmiştir. MOSG algoritması ve kıyas yapılan algoritma için anlamlı bir sonuç elde edilmesi durumunda "+" simgesi ile işaretlenmiştir.

Çizelge 5.26'daki sonuçlar algoritmaların HV metriği üzerinde elde ettikleri değerlere Wilcoxon testi uygulanması sonucu elde edilen bulgulardır. Çizelgedeki sonuçlara bakıldığında MOSG algoritmasının problem setindeki 10 problem için elde ettiği değerlerin tamamının diğer algoritmaların ürettiği değerlerden farklı ve anlamlı olduğu görülmektedir.

Çizelge 5.26. Problem Seti-2: HV metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG vs.	NSGA-II		IBEA		MOCell		PAES	
	p-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
Binh2	7.07E-18	+	1.01E-17	+	7.07E-18	+	7.07E-18	+
ConstrEx	7.07E-18	+	7.80E-11	+	7.07E-18	+	7.07E-18	+
Srinivas	9.26E-03	+	7.15E-09	+	2.07E-17	+	2.20E-17	+
KITA	7.07E-18	+	7.97E-18	+	7.07E-18	+	7.07E-18	+
Water	7.07E-18	+	2.80E-03	+	3.13E-17	+	5.64E-17	+
FourBarTruss	1.49E-16	+	1.86E-14	+	6.12E-18	+	6.14E-18	+
DiskBrake	4.20E-13	+	5.40E-11	+	7.97E-18	+	7.07E-18	+
CentileverBeam	7.07E-18	+	7.07E-18	+	7.07E-18	+	7.07E-18	+
Spring	7.07E-18	+	7.07E-18	+	7.07E-18	+	7.07E-18	+
GearTrain	7.07E-18	+	7.07E-18	+	7.07E-18	+	7.07E-18	+

Çizelge 5.27'deki, IGD metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 10 problemden; NSGA-II algoritması için 9, IBEA algoritması için 8, MOCell ve PAES algoritmaları için 10 tanesinden farklı ve anlamlı sonuçlar ürettiği görülmektedir.

Çizelge 5.27. Problem Seti-2: IGD metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG vs.	NSGA-II		IBEA		MOCell		PAES	
	p-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
Binh2	7.07E-18	+	4.12E-10	+	7.07E-18	+	7.07E-18	+
ConstrEx	7.07E-18	+	4.06E-01	-	7.07E-18	+	7.07E-18	+
Srinivas	1.12E-01	-	7.82E-02	-	7.07E-18	+	1.55E-14	+
KITA	7.07E-18	+	7.07E-18	+	7.07E-18	+	7.07E-18	+
Water	7.07E-18	+	1.01E-17	+	1.31E-15	+	7.97E-18	+
FourBarTruss	3.24E-08	+	4.55E-15	+	6.82E-18	+	4.07E-17	+
DiskBrake	8.71E-03	+	2.35E-04	+	2.54E-16	+	4.46E-17	+
CentileverBeam	1.15E-11	+	1.13E-16	+	8.46E-18	+	2.47E-17	+
Spring	7.07E-18	+	6.31E-13	+	7.07E-18	+	7.07E-18	+
GearTrain	1.29E-17	+	3.02E-15	+	7.07E-18	+	7.07E-18	+

Çizelge 5.28'deki, Spread metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 10 problemden; IBEA algoritması için 7, NSGA-II, MOCell ve PAES algoritmaları için tamamından farklı ve anlamlı sonuçlar ürettiği görülmektedir.

Çizelge 5.28. Problem Seti-2: Spread metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

MOSG vs.	NSGA-II		IBEA		MOCell		PAES	
	p-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
Binh2	7.07E-18	+	8.59E-12	+	7.07E-18	+	7.07E-18	+
ConstrEx	7.07E-18	+	1.99E-01	-	7.07E-18	+	7.07E-18	+
Srinivas	1.07E-16	+	1.12E-09	+	7.07E-18	+	7.07E-18	+
KITA	7.07E-18	+	8.99E-18	+	7.07E-18	+	7.07E-18	+
Water	7.07E-18	+	3.10E-12	+	7.07E-18	+	7.07E-18	+
FourBarTruss	6.85E-18	+	2.68E-01	-	6.85E-18	+	6.85E-18	+
DiskBrake	7.39E-04	+	4.14E-01	-	7.07E-18	+	7.97E-18	+
CentileverBeam	7.07E-18	+	7.07E-18	+	7.07E-18	+	7.07E-18	+
Spring	7.07E-18	+	1.54E-17	+	7.07E-18	+	7.07E-18	+
GearTrain	5.89E-03	+	5.29E-03	+	1.76E-13	+	4.28E-11	+

Çizelge 5.29'daki, Epsilon metriği için yapılan Wilcoxon testi sonuçlarına bakıldığında MOSG algoritmasının problem setindeki 10 problemden; algoritmaların tablodaki dizilişlerine göre sırasıyla 8, 10, 9 ve 10 tanesinden farklı ve anlamlı sonuçlar ürettiği görülmektedir.

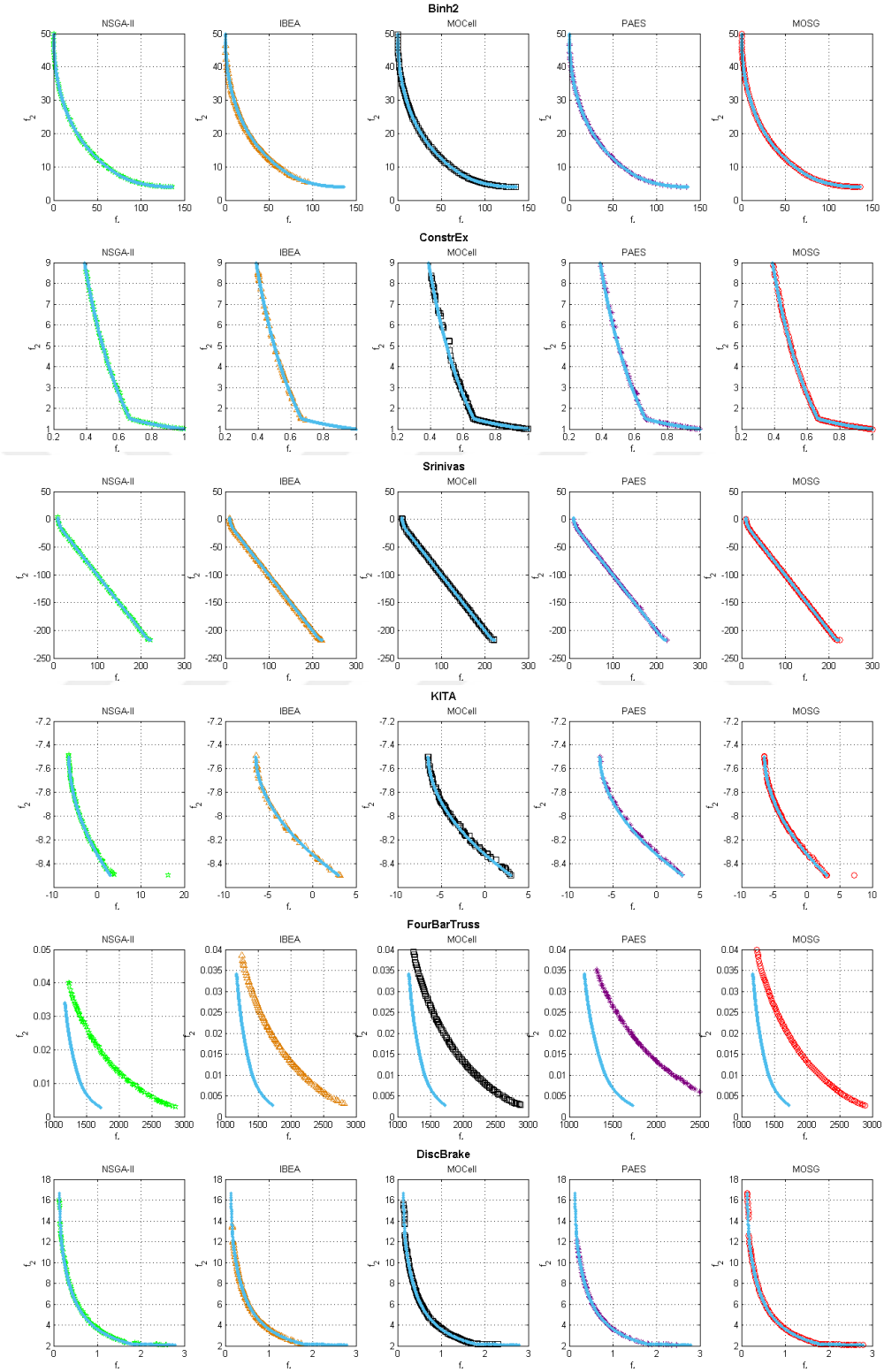
Çizelge 5.29. Problem Seti-2: Epsilon metriği için MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

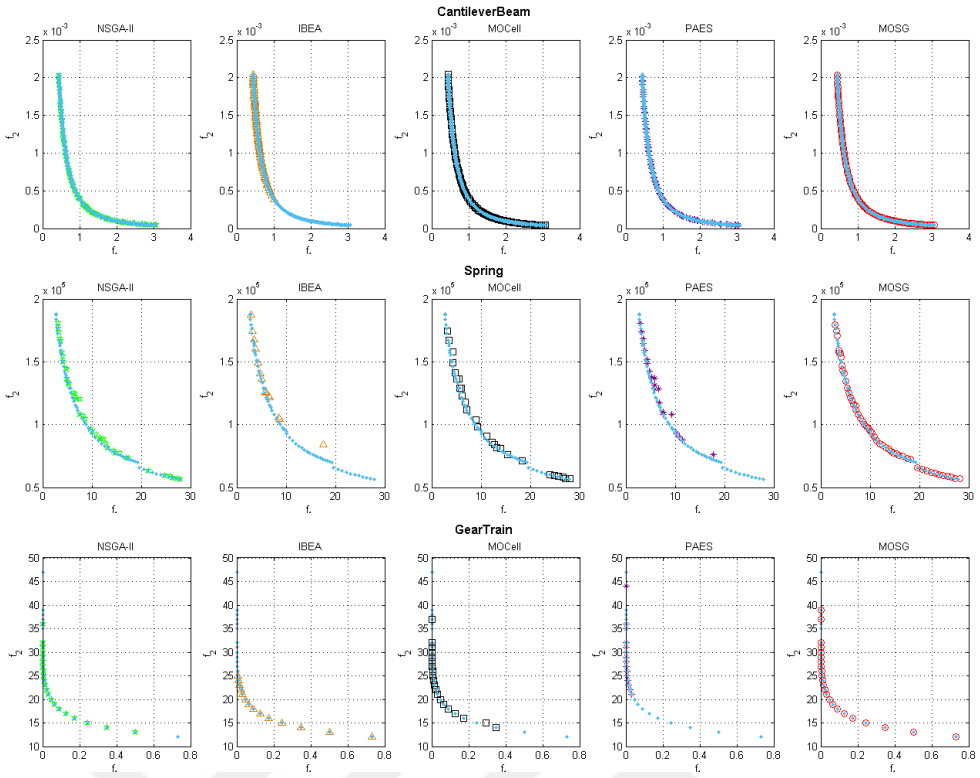
MOSG vs.	NSGA-II		IBEA		MOCell		PAES	
	p-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
Binh2	7.07E-18	+	6.53E-14	+	7.07E-18	+	7.07E-18	+
ConstrEx	6.20E-04	+	1.06E-02	+	3.40E-01	-	8.38E-16	+
Srinivas	4.80E-01	-	2.87E-03	+	7.07E-18	+	3.46E-14	+
KITA	7.07E-18	+	2.33E-17	+	7.07E-18	+	7.07E-18	+
Water	3.31E-20	+	3.31E-20	+	3.31E-20	+	3.31E-20	+
FourBarTruss	4.24E-16	+	4.16E-16	+	5.14E-18	+	5.87E-18	+
DiskBrake	5.79E-01	-	3.58E-02	+	3.99E-13	+	4.46E-17	+
CentileverBeam	8.56E-15	+	1.10E-13	+	7.07E-18	+	1.29E-17	+
Spring	7.07E-18	+	2.22E-11	+	2.07E-17	+	7.07E-18	+
GearTrain	7.07E-18	+	7.97E-18	+	7.07E-18	+	7.07E-18	+

Problem seti-2 için yapılan Wilcoxon sıra toplamı testi sonuçlarına genel olarak bakıldığında MOSG algoritmasının ürettiği sonuçların kıyaslama yapılan algoritmaların ürettiği sonuçlardan farklı ve anlamlı olduğu açık bir şekilde söylenebilir.

MOSG ve diğer algoritmaların elde ettikleri *PF* çözümlerinin problemlerin PF_t çözümleri ile ne kadar benzeştiğini görmek adına elde edilen *PF* ve PF_t çözümleri Şekil 5.5'te grafiksel olarak sunulmuştur. Bu çizimlerde PF_t çözümleri mavi renk ile temsil

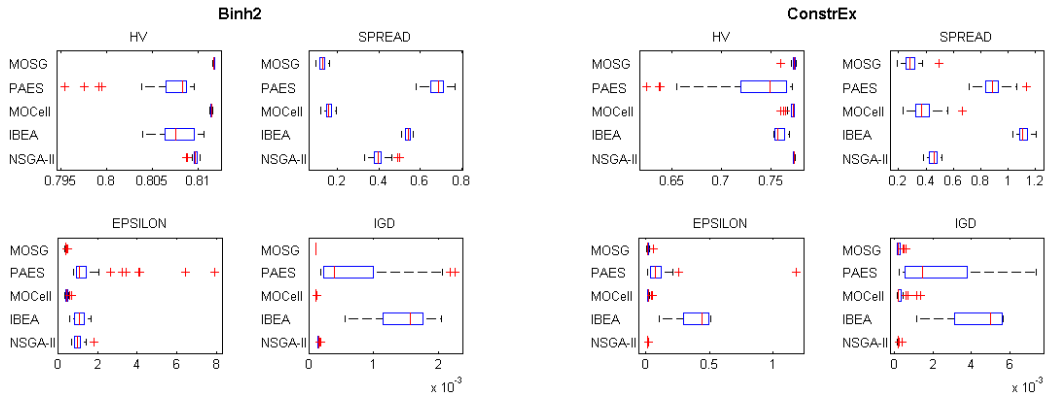
edilirken diğer renkteki çözümler ilgili algoritmanın elde ettiği *PF* çözümlerini temsil etmektedir.

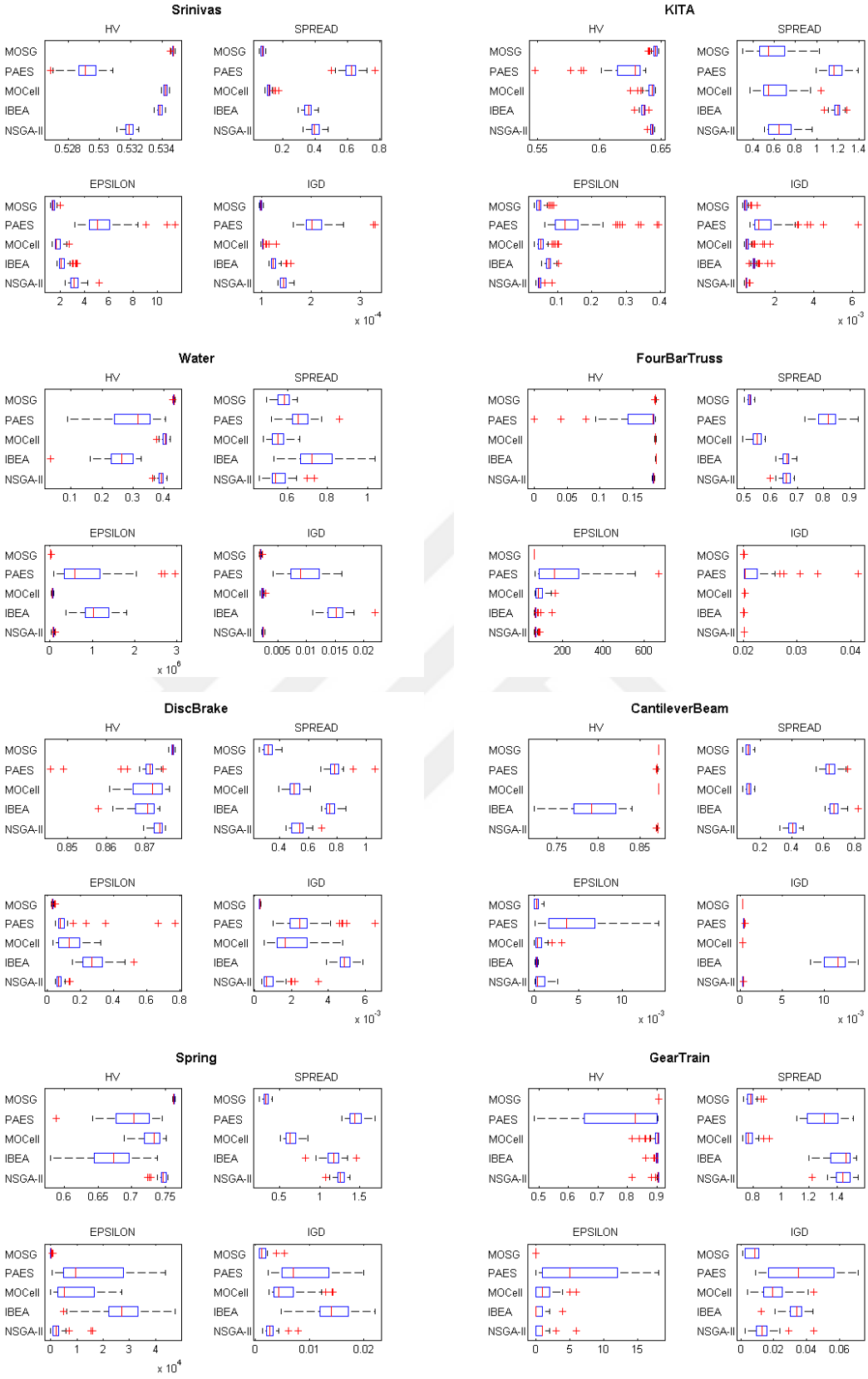




Şekil 5.5. Problem seti-2 için algoritmaların ürettiği *PF*'lerin PF_t'ler ile karşılaştırılması

Son olarak MOSG ve diğer algoritmaların problem seti-2 üzerinde yaptıkları 100 bağımsız çalıştırmanın hakkında genel bir bilgi edinmek amacıyla bu çalıştırmalar sonucunda elde edilen değerlerin kutu grafikleri Şekil 5.6 ile verilmiştir. Problem seti-1'de olduğu gibi bu problem setinde de MOSG algoritmasının elde ettiği sonuçlar itibari ile kararlı bir yapıda çalıştığı görülmektedir.





Şekil 5.6. Problem seti-2 için algoritmaların ürettiği sonuçların kutu grafikleri

5.2.3. Problem seti -3 deney sonuçları

Bu bölümde MOSG algoritması modifiye edilerek kesikli problemlere uygulanacak hale getirilmiştir. D-MOSG olarak isimlendirilen modifiye algoritma eşikleme yaklaşımı ile görüntü segmentasyonu problemine uygulanmıştır. Şekil 3.17'de verilen görüntüler üzerinde yapılan çalışmada farklı eşik sayıları (2, 3, 4, 5, 6, 8, 10, 12, 15) kullanılarak algoritmaların farklı seviyelerdeki başarıları incelenmiştir. D-MOSG algoritması bileşenleri olan SFLA ve GWO algoritmaları ile performans olarak karşılaştırılmıştır. Eşikleme orijinalinde tek amaçlı bir problemdir ve algoritmalar belirlenen uygunluk fonksiyonunu optimize ederek en iyi eşik değerlerini bulmaya çalışmaktadır. Uygunluk fonksiyonu olarak Otsu metodu ve Kapur entropisi seçilmiştir. SFLA ve GWO algoritmaları bu uygunluk fonksiyonları ile ayrı ayrı çalıştırılırken, D-MOSG algoritması bu iki uygunluk birlikte optimize etmeye çalışarak problemi çok amaçlı olarak ele almıştır. Algoritmaların çalışması sonucu elde edilen segmente edilmiş görüntülerin kalitesini ölçmek amacıyla PSNR metriği kullanılmıştır. D-MOSG ve diğer algoritmalar problem seti üzerinde her eşik sayısı için 50 bağımsız tekrar ile çalıştırılmıştır. Algoritmaların 50 bağımsız çalıştırması sonucu elde ettikleri ortalama PSNR değerleri Çizelge 5.30'da verilmiştir. Referans görüntü ile elde edilmiş görüntü arasında üretilen PSNR değerinin yüksek olması hedeflenmektedir ve büyük değer iyi sonuç anlamına gelmektedir. Çizelgede algoritmaların elde ettiği değerler içerisinde en iyi değere sahip olan kalın olarak işaretlenmiştir. Çizelge 5.30'daki tek amaçlı algoritmaların PSNR sonuçlarına bakıldığında aynı algoritmanın farklı uygunluk fonksiyonu kullanarak aynı görüntü üzerinde farklı sonuçlar elde ettiği görülmektedir. Bununla beraber uygunluk fonksiyonlarının, görüntünün karakteristiğine göre, farklı eşik sayısı seviyelerinde birbirine üstünlük sağladığı durumlar mevcuttur. Sonuçlara genel olarak bakıldığında ise iki uygunluk fonksiyonunu birlikte optimize ederek problemi çok amaçlı olarak ele alan D-MOSG algoritmasının diğer algoritmalardan daha iyi sonuçlar ürettiği görülmektedir.

Çizelge 5.30. Algoritmaların problem seti-3'teki görüntüler üzerinde elde ettiği ortalama PSNR değerleri

I	TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG	I	TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
	2	19.9388	21.0552	19.9388	21.0552	**21.1289**		2	20.2346	19.9711	20.2354	19.9691	**20.3705**
	3	22.9262	22.5255	22.9262	22.5255	**23.5901**		3	22.8397	22.7196	22.8397	22.7272	**23.1431**
	4	24.7015	24.9453	24.7015	24.9366	**25.4721**		4	25.0962	25.1119	25.0962	25.1172	**25.2803**
	5	26.7864	26.8521	26.7863	27.0148	**27.3006**		5	26.7272	26.0372	26.7394	26.1852	**26.8788**
I_1	6	28.1133	27.1736	27.9735	27.0333	**28.7148**	I_6	6	28.1940	27.5461	28.1487	27.6103	**28.2860**
	8	**31.0111**	29.7403	30.7341	29.6227	30.9751		8	30.3593	29.6090	30.3397	29.6511	**30.3852**
	10	**32.7216**	31.6456	32.5671	31.5100	32.5581		10	**32.0551**	31.3343	31.8904	31.3689	31.9954
	12	**34.0275**	32.9156	33.8991	32.8010	33.8606		12	**33.6427**	32.8790	33.3176	32.9145	33.4428
	15	**35.8050**	34.6693	35.2106	34.4094	35.4979		15	**35.4091**	34.6342	35.0555	34.6462	34.9207
	2	18.8060	19.2953	18.8060	19.2953	**19.7974**		2	18.8672	17.9846	18.8672	17.9846	**19.5929**
	3	20.8962	22.6416	20.8962	22.6416	**23.1122**		3	19.9103	22.0029	19.9103	22.0029	**23.5419**
	4	22.4564	22.7426	22.4564	22.7542	**25.6822**		4	23.0992	26.3049	23.1001	26.3094	**26.5102**
	5	23.7937	25.4118	23.7937	25.4114	**27.3853**		5	24.4155	27.1249	24.4159	27.1219	**28.2103**
I_2	6	27.9040	27.4093	25.4210	27.3491	**28.8563**	I_7	6	29.7953	27.6848	29.7830	27.6418	**29.9148**
	8	30.5971	29.2520	30.5284	29.3887	**31.0511**		8	31.9635	29.8830	31.9618	29.7504	**32.2279**
	10	32.4730	31.1340	32.3258	31.5516	**32.9939**		10	33.6640	32.1511	33.5046	32.4912	**34.0240**
	12	34.2351	33.0570	33.5398	33.1439	**34.4734**		12	**35.6689**	34.0331	34.4325	33.8037	34.9593
	15	**36.2094**	34.8479	35.2242	35.0573	36.1920		15	**36.7666**	35.2065	36.2657	35.1609	36.0841
	2	20.3280	**21.2370**	20.3280	**21.2370**	**21.2370**		2	20.1237	18.3457	20.1237	18.3457	**20.2523**
	3	23.7448	21.9154	23.7448	21.8030	**23.9251**		3	21.6721	22.2025	21.6721	22.2025	**22.3544**
	4	25.4259	23.0145	25.4259	22.6177	**25.4465**		4	23.0425	24.1778	23.0425	24.1778	**24.4531**
	5	26.3044	25.7693	26.3114	25.7708	**26.5102**		5	24.1153	25.1851	24.0976	25.1851	**26.3132**
I_3	6	26.8621	27.4144	26.8621	27.3728	**27.8666**	I_8	6	25.2895	27.2851	25.2901	27.2067	**27.8419**
	8	27.9004	30.3664	27.8422	30.3552	**30.7771**		8	27.5244	30.1327	27.5142	29.9662	**30.9463**
	10	29.2602	31.8212	29.4430	31.8932	**32.4443**		10	29.5771	31.7023	29.3392	31.8203	**32.8155**
	12	30.1344	33.3518	30.5745	33.1710	**34.0644**		12	31.1493	33.0785	30.6879	32.9673	**34.2338**
	15	32.4144	35.4510	32.3386	34.8857	**36.0202**		15	33.4495	35.3046	31.9986	34.6698	**35.6968**
	2	22.2688	22.8015	22.2688	22.8015	**23.5554**		2	19.1358	19.8723	19.1358	19.8723	**19.9156**
	3	22.8957	23.7759	22.8957	23.7759	**23.9879**		3	20.9404	22.5262	20.9404	22.5262	**22.6168**
	4	24.1971	26.3856	24.1971	26.3856	**26.4416**		4	22.6356	25.6773	22.6356	25.6794	**25.8892**
	5	25.0832	27.7747	25.1107	27.6186	**27.8067**		5	23.8099	27.7912	23.8110	27.8117	**28.0996**
I_4	6	26.2674	28.5740	26.2981	27.9086	**29.0930**	I_9	6	24.8026	28.8001	24.9026	29.1825	**29.4414**
	8	28.3589	29.6657	28.5099	29.1444	**30.7445**		8	25.9344	30.8532	26.0148	31.2626	**31.6379**
	10	29.9504	31.4023	29.9088	30.6606	**32.4442**		10	27.1540	32.5583	27.0789	32.5278	**33.1969**
	12	30.3115	32.7512	30.3106	32.0624	**34.0298**		12	28.5144	34.0142	28.4826	33.8326	**34.3917**
	15	31.2540	35.1550	31.1927	33.7868	**36.0750**		15	29.4219	**36.0218**	29.4187	35.3477	36.0131
	2	20.3309	20.3541	20.3309	20.3541	**20.4293**		2	20.2145	18.4092	20.2145	18.4092	**20.4431**
	3	**22.8005**	22.1027	**22.8005**	22.1004	22.7760		3	22.3805	21.5412	22.3805	21.5412	**22.8076**
	4	23.8704	24.3194	23.8704	24.3194	**25.0432**		4	23.4600	23.6983	23.4600	23.0847	**24.5351**
	5	25.3302	26.2421	25.3302	26.2465	**27.0231**		5	25.0132	25.9201	25.0132	25.9496	**26.2383**
I_5	6	26.1378	27.4958	26.1233	27.4912	**28.4231**	I_{10}	6	25.5514	27.3315	25.5489	27.2375	**28.0031**
	8	29.6496	30.2123	29.5045	30.1888	**30.3722**		8	27.1260	29.9865	27.1482	30.1619	**30.8162**
	10	31.4608	32.0339	31.2143	32.0405	**32.2946**		10	30.1947	32.3668	29.5666	31.9388	**32.9299**
	12	32.4298	33.1267	32.2359	33.4302	**33.7322**		12	31.4537	33.8344	31.3706	33.7857	**34.4385**
	15	34.8544	34.7631	33.3522	34.8632	**35.5122**		15	33.3987	35.6720	32.7056	35.6925	**36.3925**

$SFLA_O$ = SFLA ile Otsu, $SFLA_K$ = SFLA ile Kapur, GWO_O = GWO ile Otsu, GWO_K = GWO ile Kapur çiftlerinin uygulanması olmak üzere.

Algoritmaların 50 bağımsız tekrar ile çalışması sonucu elde edilen sonuçların Friedman testine tabi tutulması amaçlanmıştır. Friedman testi iki farklı açıdan uygulanmıştır. İlk önce algoritmaların her görüntü üzerindeki her bir eşik sayısı seviyesi için elde ettikleri tüm sonuçlar ile Friedman testi sıralamaları elde edilmiş ve Çizelge

5.31'de gösterilmiştir. İkinci olarak her bir eşik sayısı seviyesi için algoritmaların başarısını ölçmek amacıyla Friedman testi uygulanmış ve elde edilen sıralama sonuçları Çizelge 5.32'de verilmiştir.

Çizelge 5.31. Algoritmaların genel Friedman sıralaması

Algoritma	**Sıralama**
D-MOSG	**4.80**
$SFLA_{Otsu}$	2.51
$SFLA_{Kapur}$	2.86
GWO_{Otsu}	2.11
GWO_{Kapur}	2.73

Çizelge 5.31'de verilen genel Friedman testi sonuçlarına göre D-MOSG algoritmasının en iyi sıralama derecesi ile diğer algoritmalardan daha başarılı sonuçlar ürettiği görülmektedir. Diğer algoritmaların başarı sıralamasına bakıldığında ise en iyiden kötüye doğru $SFLA_{Kapur}$, GWO_{Kapur}, $SFLA_{Otsu}$ ve GWO_{Otsu} şeklinde olduğu görülmektedir.

Çizelge 5.32. Algoritmaların genel eşik sayısı tabanlı Friedman sıralaması

TH	D-MOSG	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K
2	**4.90**	2.25	2.80	2.35	2.70
3	**4.80**	2.60	2.55	2.60	2.45
4	**5.00**	1.75	3.25	1.85	3.15
5	**5.00**	1.70	2.95	2.10	3.25
6	**5.00**	2.45	3.10	1.95	2.50
8	**4.90**	2.60	2.80	2.10	2.60
10	**4.70**	2.90	2.60	2.00	2.80
12	**4.60**	3.00	2.90	2.00	2.50
15	**4.30**	3.30	2.80	2.00	2.60

Çizelge 5.32'deki eşik sayısı tabanlı Friedman testi sonuçlarına bakıldığında D-MOSG algoritmasının her eşik seviyesi için diğer algoritmalardan daha başarılı sıralama değerine sahip olduğu görülmektedir. Bununla beraber eşik sayısı arttığında D-MOSG algoritmasının başarı derecesinin kısmi olarak azaldığı görülebilmektedir. Öte yandan diğer algoritmaların elde ettikleri sıralamalara bakıldığında farklı eşik seviyelerinde birbirlerine üstünlük sağlayabildikleri görülmektedir.

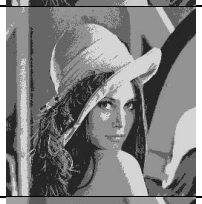
D-MOSG ve diğer algoritmalar tarafından elde edilen sonuçların birbiri ile karşılaştırıldığında aralarında anlamlı bir fark olup olmadığını öğrenmek amacıyla Wilcoxon sıra toplamı testi uygulanmıştır. D-MOSG algoritmasının elde ettiği

sonuçların diğer algoritmalar ile kıyaslanması sonucunda elde edilen Wilcoxon testi sonuçları Çizelge 5.33'te verilmiştir. Wilcoxon testi sonuçlarına bakıldığında D-MOSG algoritmasının elde ettiği sonuçlar ile diğer algoritmaların sonuçları arasında anlamlı bir fark olduğu görülmektedir.

Çizelge 5.33. D-MOSG algoritmasının diğer algoritmalarla yapılan Wilcoxon testi sonuçları

D-MOSG vs.	**SFLA$_{Otsu}$**		**SFLA$_{Kapur}$**		**GWO$_{Otsu}$**		**GWO$_{Kapur}$**	
Görüntü	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S	*p*-değeri	S
I_1	4.65E-01	-	4.02E-04	+	7.13E-02	-	2.43E-04	+
I_2	1.32E-04	+	6.95E-07	+	7.02E-08	+	2.23E-06	+
I_3	4.65E-09	+	2.31E-03	+	7.74E-09	+	1.54E-03	+
I_4	1.15E-20	+	3.13E-03	+	8.26E-21	+	4.31E-05	+
I_5	1.30E-03	+	4.92E-03	+	2.70E-05	+	6.76E-03	+
I_6	3.45E-01	-	6.59E-03	+	9.16E-02	-	6.98E-03	+
I_7	4.67E-03	+	2.07E-06	+	1.88E-04	+	1.14E-06	+
I_8	1.32E-11	+	3.99E-03	+	1.46E-14	+	1.68E-03	+
I_9	9.40E-37	+	1.60E-02	+	3.40E-37	+	6.37E-03	+
I_{10}	2.26E-10	+	9.48E-03	+	1.95E-11	+	6.38E-03	+

D-MOSG ve diğer algoritmaların örnek bir görüntü (I_3) üzerinde elde ettiği eşik değerleri kullanılarak yapılan segmentasyon sonucu oluşan görüntüler Şekil 5.7'de verilmiştir. Algoritmaların her eşik sayısı için elde ettiği sonuçların görüntüleri verilmiştir. Diğer görüntülerin segmentasyon sonuçları EK-2'de verilmiştir.

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

Şekil 5.7. I_3 görüntüsü için elde edilen segmentasyon sonuçları

6. SONUÇ

Tez çalışması kapsamında, tek amaçlı problemlerin çözümü için geliştirilmiş olan kurbağa sıçrama (SFLA) ve gri kurt optimizasyonu (GWO) algoritmaları hibrit bir şekilde kullanılarak çok amaçlı optimizasyon problemlerine uygulanmıştır. Bu hibrit yapının yanı sıra önerilen algoritmanın performansını arttırmak amacıyla bazı modifikasyonlar yapılmıştır. Önerilen algoritmanın (MOSG) değerlendirilmesi üç aşamada farklı problem setleri üzerinde yapılmıştır.

İlk aşamada özellikleri birbirinden farklı 36 kısıtsız çok amaçlı optimizasyon problemi kullanılmıştır. Bu aşamada algoritmanın performansı çok amaçlı optimizasyon algoritmalarından NSGA-II, IBEA, MOCell, MOEA/D, MOAAA ve MOVS algoritmalarının performansı ile kıyaslanmıştır. Performans metriği olarak farklı açılardan ölçüm yapan HV, IGD, Spread ve Epsilon metrikleri kullanılmıştır. Algoritmaların bu problem seti üzerinde, dört metrik için, elde ettikleri ortalama değerler sunulmuştur. Bununla beraber her metrik için genel bir başarı sıralaması elde etmek adına Friedman istatistiksel testi uygulanmıştır. Elde edilen Friedman testi sonuçlarına göre MOSG ve MOAAA algoritmalarının birbirine yakın başarı elde ettiği görülmüştür. Bu nedenle hem bu iki algoritma arasında hem de MOSG algoritmasının diğer algoritmalar ile ikili bir karşılaştırılması yapılmıştır. Dört metrik için yapılan bu ikili karşılaştırma sonuçlarına göre MOSG algoritmasının daha başarılı performans gösterdiği görülmüştür. Ayrıca MOSG algoritmasının ürettiği sonuçların diğer algoritmaların sonuçları ile karşılaştırıldığında anlamlı bir fark olup olmadığını test etmek için Wilcoxon sıra toplamı testi kullanılmıştır. Bu testin sonucunda MOSG algoritmasının sonuçlarının diğer algoritmaların sonuçlarından anlamlı bir şekilde farklı olduğu görülmüştür. Algoritmalar tarafından elde edilen PF çözümlerinin PF_t çözümlerine ne kadar benzediğini göstermek amacıyla bu çözümler grafiksel olarak sunulmuştur. Son olarak elde edilen sonuçların kutu grafikleri çizdirilmiştir.

İkinci aşamada mühendislik tasarım problemleri ve kısıtlı problemlerden oluşan farklı özellikteki 10 adet çok amaçlı optimizasyon problemi kullanılmıştır. MOSG algoritmasının performansı bu aşamada NSGA-II, IBEA, MOCell ve PAES algoritmaları ile kıyaslanmıştır. Sonuçların karşılaştırılması birinci aşamada kullanılan metrik ve testler ile yapılmıştır. Bu aşamada MOSG algoritması ortalama değerler ve Friedman başarı sıralaması bakımından diğer algoritmalara açık bir üstünlük sağladığından ikili bir karşılaştırma yapma gereği duyulmamıştır.

Üçüncü ve son aşamada ise önerilen algoritma, görüntü işleme çalışmalarında sıklıkla kullanılan 10 adet gri seviye görüntünün segmentasyonunda test edilmiştir. Bu aşamada MOSG algoritmasının performansı kendi bileşenleri olan SFLA ve GWO algoritmaları ile karşılaştırılmıştır. Bu bölümde tek amaçlı bir problem olan eşikleme, önerilen algoritma kullanılarak çok amaçlı bir problem olarak ele alınmıştır. Görüntü segmentasyonu yapılan bu aşamada performans metriği olarak tepe sinyali gürültü oranı (PSNR) kullanılmıştır. Elde edilen deneysel sonuçlar Friedman ve Wilcoxon istatistik testleri ile analiz edilmiştir. Ayrıca elde edilen sonuç görüntüleri görsel olarak sunulmuştur. Elde edilen deneysel sonuçlara bakıldığında MOSG algoritmasının performansı açık bir şekilde öne çıkmıştır.

Üç aşamada elde edilen deneysel sonuçlara bakıldığında önerilen algoritmanın genel olarak karşılaştırma yapılan algoritmalardan daha başarılı olduğu görülmüştür.

Bundan sonraki çalışmalarda, mempleks yapısından dolayı paralel olarak çalıştırılmaya uygun olan MOSG algoritması, çalışma süresi uzun olan problemleri daha hızlı çözmek adına paralel kullanılarak bu problemlere uygulanabilir. Bununla beraber ikili (binary), hibrit veya dinamik çok amaçlı problemler üzerinde algoritmanın performansı test edilebilir. Tez kapsamında kullanılan Pareto yaklaşımından farklı olarak ayrıştırma veya indikatör tabanlı yaklaşımlar kullanılabilir.

KAYNAKLAR

Akbari, R., Hedayatzadeh, R., Ziarati, K. ve Hassanizadeh, B., 2012, A multi-objective artificial bee colony algorithm, *Swarm and Evolutionary Computation*, 2, 39-52.

Al-Dujaili, A. ve Suresh, S., 2016, Bmobench: Black-box multi-objective optimization benchmarking platform, *arXiv preprint arXiv:1605.07009*.

Al-Tashi, Q., Abdulkadir, S. J., Rais, H. M., Mirjalili, S., Alhussian, H., Ragab, M. G. ve Alqushaibi, A., 2020, Binary multi-objective grey wolf optimizer for feature selection in classification, *IEEE Access*, 8, 106247-106263.

Ali, S. Q. ve Hasanien, H. M., 2016, Shuffled frog leaping algorithm for multi-objective design optimization of transverse flux linear motor, *Electric Power Components and Systems*, 44 (11), 1307-1315.

Altınöz, Ö. T., 2015, Çok Amaçlı Optimizasyon Problemlerinde Pareto Cephesinin Maliyet Etkin Yöntemlerle Temsil Edilmesi, Doktora Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü.

Arslan, S. ve Ozturk, C., 2019, Multi hive artificial bee colony programming for high dimensional symbolic regression with feature selection, *Applied Soft Computing*, 78, 515-527.

Asadzadeh, M., 2016, Desired Precision in Multi-Objective Optimization: Epsilon Archiving or Rounding Objectives?

Babalik, A., Ozkis, A., Uymaz, S. A. ve Kiran, M. S., 2018, A multi-objective artificial algae algorithm, *Applied Soft Computing*, 68, 377-395.

Bhandari, A. K., Kumar, A. ve Singh, G. K., 2015, Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur's, Otsu and Tsallis functions, *Expert Systems with Applications*, 42 (3), 1573-1601.

Bhandari, A. K., Singh, N. ve Shubham, S., 2019, An efficient optimal multilevel image thresholding with electromagnetism-like mechanism, *Multimedia Tools and Applications*, 78 (24), 35733-35788.

Biswas, S., Das, S., Suganthan, P. N. ve Coello, C. A. C., 2014, Evolutionary multiobjective optimization in dynamic environments: A set of novel benchmark functions, *2014 IEEE Congress on Evolutionary Computation (CEC)*, 3192-3199.

Chaharsooghi, S. K. ve Kermani, A. H. M., 2008, An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP), *Applied mathematics and computation*, 200 (1), 167-177.

Chan, F. T., Jha, A. ve Tiwari, M. K., 2016, Bi-objective optimization of three echelon supply chain involving truck selection and loading using NSGA-II with heuristics algorithm, *Applied Soft Computing*, 38, 978-987.

Chen, X., Du, W. ve Qian, F., 2014, Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization, *Chemometrics and Intelligent Laboratory Systems*, 136, 85-96.

Coello, C. A. C., Pulido, G. T. ve Lechuga, M. S., 2004, Handling multiple objectives with particle swarm optimization, *IEEE transactions on Evolutionary Computation*, 8 (3), 256-279.

Coello, C. A. C. ve Cortés, N. C., 2005, Solving multiobjective optimization problems using an artificial immune system, *Genetic programming and evolvable machines*, 6 (2), 163-190.

Coello, C. A. C., Lamont, G. B. ve Van Veldhuizen, D. A., 2007, Evolutionary algorithms for solving multi-objective problems, Springer, p.

Coello, C. C. ve Lechuga, M. S., 2002, MOPSO: A proposal for multiple objective particle swarm optimization, *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, 1051-1056.

Coello, C. C., 2006, Evolutionary multi-objective optimization: a historical view of the field, *IEEE computational intelligence magazine*, 1 (1), 28-36.

Coello, C. C., Dhaenens, C. ve Jourdan, L., 2009, Advances in multi-objective nature inspired computing, Springer, p.

Dai, C., Wang, Y. ve Ye, M., 2015, A new multi-objective particle swarm optimization algorithm based on decomposition, *Information Sciences*, 325, 541-557.

Deb, K. ve Agrawal, R. B., 1995, Simulated binary crossover for continuous search space, *Complex systems*, 9 (2), 115-148.

Deb, K., Agrawal, S., Pratap, A. ve Meyarivan, T., 2000, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, *International conference on parallel problem solving from nature*, 849-858.

Deb, K. ve Jain, S., 2002, Running performance metrics for evolutionary multi-objective optimization.

Deb, K., Pratap, A., Agarwal, S. ve Meyarivan, T., 2002, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE transactions on Evolutionary Computation*, 6 (2), 182-197.

Deb, K., Thiele, L., Laumanns, M. ve Zitzler, E., 2005, Scalable test problems for evolutionary multiobjective optimization, In: Evolutionary multiobjective optimization, Eds: Springer, p. 105-145.

Deb, K., 2011, Multi-objective optimisation using evolutionary algorithms: an introduction, In: Multi-objective evolutionary optimisation for product design and manufacturing, Eds: Springer, p. 3-34.

Dikmen, H., Dikmen, H., Elbir, A., Eksi, Z. ve Çelik, F., 2014, Gezgin satıcı probleminin karınca kolonisi ve genetik algoritmalarla eniyilemesi ve karşılaştırılması, *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 18 (1), 8-13.

Dilip, L., Bhesdadiya, R., Trivedi, I. ve Jangir, P., 2018, Optimal power flow problem solution using multi-objective grey wolf optimizer algorithm, In: Intelligent Communication and Computational Technologies, Eds: Springer, p. 191-201.

Ding, M., Chen, H., Lin, N., Jing, S., Liu, F., Liang, X. ve Liu, W., 2017, Dynamic population artificial bee colony algorithm for multi-objective optimal power flow, *Saudi journal of biological sciences*, 24 (3), 703-710.

Doğan, B. ve Ölmez, T., 2015a, A new metaheuristic for numerical function optimization: Vortex Search algorithm, *Information Sciences*, 293, 125-145.

Doğan, B. ve Ölmez, T., 2015b, Vortex search algorithm for the analog active filter component selection problem, *AEU-International Journal of Electronics and Communications*, 69 (9), 1243-1253.

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

Du, P., Wang, J., Hao, Y., Niu, T. ve Yang, W., 2020, A novel hybrid model based on multi-objective Harris hawks optimization algorithm for daily PM2. 5 and PM10 forecasting, *Applied Soft Computing*, 96, 106620.

Durillo, J. J. ve Nebro, A. J., 2011, jMetal: A Java framework for multi-objective optimization, *Advances in engineering software*, 42 (10), 760-771.

Edgeworth, F. Y., 1881, Mathematical psychics: An essay on the application of mathematics to the moral sciences, 10, CK Paul, p.

Elbeltagi, E., Hegazy, T. ve Grierson, D., 2005, Comparison among five evolutionary-based optimization algorithms, *Advanced engineering informatics*, 19 (1), 43-53.

Epanechnikov, V. A., 1969, Non-parametric estimation of a multivariate probability density, *Theory of Probability & Its Applications*, 14 (1), 153-158.

Erdoğmuş, P., 2016, Doğadan esinlenen optimizasyon algoritmaları ve optimizasyon algoritmalarının optimizasyonu, *Düzce üniversitesi bilim ve teknoloji dergisi*, 4 (1).

Ergul, E. U. ve Eminoglu, I., 2014, DOPGA: A new fitness assignment scheme for multi-objective evolutionary algorithms, *International Journal of Systems Science*, 45 (3), 407-426.

Ergül, E., 2015, ÇOK AMAÇLI GENETİK ALGORİTMA YÖNTEMLERİNİN BAŞARIMININ BELİRLENMESİ İÇİN İKİ YENİ ÖLÇÜT ÖNERİSİ, *İleri Teknoloji Bilimleri Dergisi*, 4 (1), 1-13.

Eusuff, M., Lansey, K. ve Pasha, F., 2006, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, *Engineering optimization*, 38 (2), 129-154.

Eusuff, M. M. ve Lansey, K. E., 2003, Optimization of water distribution network design using the shuffled frog leaping algorithm, *Journal of Water Resources planning and management*, 129 (3), 210-225.

Fang, G., Guo, Y., Wen, X., Fu, X., Lei, X., Tian, Y. ve Wang, T., 2018, Multi-objective differential evolution-chaos shuffled frog leaping algorithm for water resources system optimization, *Water Resources Management*, 32 (12), 3835-3852.

Fonseca, C. M. ve Fleming, P. J., 1993, Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization, *Icga*, 416-423.

Fonseca, C. M. ve Fleming, P. J., 1995, Multiobjective optimization and multiple constraint handling with evolutionary algorithms 1: A unified formulation.

Friedman, M., 1937, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *Journal of the american statistical association*, 32 (200), 675-701.

Golberg, D. E., 1989, Genetic algorithms in search, optimization, and machine learning, *Addion wesley*, 1989 (102), 36.

Gong, M., Liu, F., Zhang, W., Jiao, L. ve Zhang, Q., 2011, Interactive MOEA/D for multi-objective decision making, *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 721-728.

Guzmán, M. A., Delgado, A. ve De Carvalho, J., 2010, A novel multiobjective optimization algorithm based on bacterial chemotaxis, *Engineering Applications of Artificial Intelligence*, 23 (3), 292-301.

Haklı, H. ve Uğuz, H., 2014, A novel particle swarm optimization algorithm with Levy flight, *Applied Soft Computing*, 23, 333-345.

Hancer, E., Xue, B., Zhang, M., Karaboga, D. ve Akay, B., 2015, A multi-objective artificial bee colony approach to feature selection using fuzzy mutual information, *2015 IEEE congress on evolutionary computation (CEC)*, 2420-2427.

Heidari, A. A. ve Pahlavani, P., 2017, An efficient modified grey wolf optimizer with Lévy flight for optimization tasks, *Applied Soft Computing*, 60, 115-134.

Hidalgo-Paniagua, A., Vega-Rodríguez, M. A., Ferruz, J. ve Pavón, N., 2015, MOSFLA-MRPP: multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning, *Engineering Applications of Artificial Intelligence*, 44, 123-136.

Holland, J. H., 1992, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, p.

Horn, J., Nafpliotis, N. ve Goldberg, D. E., 1994, A niched Pareto genetic algorithm for multiobjective optimization, *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, 82-87.

Huband, S., Hingston, P., Barone, L. ve While, L., 2006, A review of multiobjective test problems and a scalable test problem toolkit, *IEEE transactions on Evolutionary Computation*, 10 (5), 477-506.

Huo, Y., Zhuang, Y., Gu, J. ve Ni, S., 2015, Elite-guided multi-objective artificial bee colony algorithm, *Applied Soft Computing*, 32, 199-210.

Images, 2021, DATASET OF STANDARD 512X512 GRAYSCALE TEST IMAGES, https://ccia.ugr.es/cvg/CG/base.htm: [Feb 2021].

Ishibuchi, H., Murata, T. ve Türkşen, I., 1997, Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems, *Fuzzy sets and systems*, 89 (2), 135-150.

Janson, S., Merkle, D. ve Middendorf, M., 2008, Molecular docking with multi-objective particle swarm optimization, *Applied Soft Computing*, 8 (1), 666-675.

Jensi, R. ve Jiji, G. W., 2016, An enhanced particle swarm optimization with levy flight for global optimization, *Applied Soft Computing*, 43, 248-261.

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization, *Citeseer*.

Karakoyun, M., 2015, Kurbağa sıçrama algoritmasının kümeleme problemlerine uygulanması, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Karakoyun, M., Baykan, N. A. ve Hacibeyoglu, M., 2017, Multilevel thresholding for image segmentation with swarm optimization algorithms, *International Research Journal of Electronics & Computer Engineering*, 3 (3), 1.

Karakoyun, M., 2019, A New Approach Based on K-means Clustering and Shuffled Frog Leaping Algorithm to Solve Travelling Salesman Problem, *Academic Perspective Procedia*, 2 (3), 446-453.

Karakoyun, M., Onur, I. ve İhtisam, A., 2019, Grey Wolf Optimizer (GWO) Algorithm to Solve the Partitional Clustering Problem, *International Journal of Intelligent Systems and Applications in Engineering*, 7 (4), 201-206.

Kaya, M. ve Güngör, S., 2007, Çok-Amaçlı Genetik Algoritma Kullanan Bulanık Sınıflandırıcı Etmenlerle Hastalık Teşhisi, *ELEKTRİK-ELEKTRONİKBİLGİSAYAR MÜHENDİSLİĞİ 12. ULUSAL KONGRESİ VE FUARI*.

Kennedy, J. ve Eberhart, R., 1995, Particle swarm optimization, *Proceedings of ICNN'95-international conference on neural networks*, 1942-1948.

Khorsandi, A., Hosseinian, S. ve Ghazanfari, A., 2013, Modified artificial bee colony algorithm based on fuzzy multi-objective technique for optimal power flow problem, *Electric Power Systems Research*, 95, 206-213.

Kishor, A., Singh, P. K. ve Prakash, J., 2016, NSABC: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering, *Neurocomputing*, 216, 514-533.

Knowles, J. ve Corne, D., 1999, The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation, *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 98-105.

Komaki, G. ve Kayvanfar, V., 2015, Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, *Journal of Computational Science*, 8, 109-120.

Kumawat, I. R., Nanda, S. J. ve Maddila, R. K., 2017, Multi-objective whale optimization, *Tencon 2017-2017 ieee region 10 conference*, 2747-2752.

Kursawe, F., 1990, A variant of evolution strategies for vector optimization, *International Conference on Parallel Problem Solving from Nature*, 193-197.

Łapa, K., 2019, Meta-optimization of multi-objective population-based algorithms using multi-objective performance metrics, *Information Sciences*, 489, 193-204.

Li, H. ve Zhang, Q., 2008, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, *IEEE transactions on Evolutionary Computation*, 13 (2), 284-302.

Li, J.-Q., Pan, Q.-K. ve Tasgetiren, M. F., 2014, A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities, *Applied Mathematical Modelling*, 38 (3), 1111-1132.

Li, J., Pan, Q. ve Xie, S., 2012, An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems, *Applied mathematics and computation*, 218 (18), 9353-9371.

Li, W., Özcan, E., John, R., Drake, J. H., Neumann, A. ve Wagner, M., 2017, A modified indicator-based evolutionary algorithm (mIBEA), *2017 IEEE Congress on Evolutionary Computation (CEC)*, 1047-1054.

Li, X., 2003, A non-dominated sorting particle swarm optimizer for multiobjective optimization, *Genetic and evolutionary computation conference*, 37-48.

Liang, J., Guo, Q., Yue, C., Qu, B. ve Yu, K., 2018, A self-organizing multi-objective particle swarm optimization algorithm for multimodal multi-objective problems, *International Conference on Swarm Intelligence*, 550-560.

Liang, J. J., Qin, A. K., Suganthan, P. N. ve Baskar, S., 2006, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE transactions on Evolutionary Computation*, 10 (3), 281-295.

Liu, J. ve Liu, J., 2019, Applying multi-objective ant colony optimization algorithm for solving the unequal area facility layout problems, *Applied Soft Computing*, 74, 167-189.

Liu, Y., 2008, A fast and elitist multi-objective particle swarm algorithm: NSPSO, *2008 IEEE International Conference on Granular Computing*, 470-475.

Lu, C., Xiao, S., Li, X. ve Gao, L., 2016, An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production, *Advances in engineering software*, 99, 161-176.

Lu, C., Gao, L., Li, X. ve Xiao, S., 2017, A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry, *Engineering Applications of Artificial Intelligence*, 57, 61-79.

Lu, C., Gao, L., Pan, Q., Li, X. ve Zheng, J., 2019, A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution, *Applied Soft Computing*, 75, 728-749.

Luo, J., Li, X., Chen, M.-R. ve Liu, H., 2015, A novel hybrid shuffled frog leaping algorithm for vehicle routing problem with time windows, *Information Sciences*, 316, 266-292.

Meza, J., Espitia, H., Montenegro, C., Giménez, E. ve González-Crespo, R., 2017, MOVPSO: vortex multi-objective particle swarm optimization, *Applied Soft Computing*, 52, 1042-1057.

Mirjalili, S., Mirjalili, S. M. ve Lewis, A., 2014, Grey wolf optimizer, *Advances in engineering software*, 69, 46-61.

Mirjalili, S., Saremi, S., Mirjalili, S. M. ve Coelho, L. d. S., 2016, Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization, *Expert Systems with Applications*, 47, 106-119.

Mirjalili, S., Jangir, P. ve Saremi, S., 2017, Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems, *Applied intelligence*, 46 (1), 79-95.

Na, W., Yuchao, S., Xiaohong, C., Xia, L. ve Dui, L., 2020, A∈-indicator-based shuffled frog leaping algorithm for many-objective optimization problems, *Journal of Systems Engineering and Electronics*, 31 (1), 142-155.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B. ve Alba, E., 2007, Design issues in a multiobjective cellular genetic algorithm, *International Conference on Evolutionary Multi-Criterion Optimization*, 126-140.

Nebro, A. J., Durillo, J. J., Garcia-Nieto, J., Coello, C. C., Luna, F. ve Alba, E., 2009a, SMPSO: A new PSO-based metaheuristic for multi-objective optimization, *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, 66-73.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B. ve Alba, E., 2009b, Mocell: A cellular genetic algorithm for multiobjective optimization, *International Journal of Intelligent Systems*, 24 (7), 726-746.

Niknam, T., Farsani, E. A. ve Nayeripour, M., 2011, An efficient multi-objective modified shuffled frog leaping algorithm for distribution feeder reconfiguration problem, *European Transactions on Electrical Power*, 21 (1), 721-739.

Ning, J., Zhang, B., Liu, T. ve Zhang, C., 2018, An archive-based artificial bee colony optimization algorithm for multi-objective continuous optimization problem, *Neural Computing and Applications*, 30 (9), 2661-2671.

Nuaekaew, K., Artrit, P., Pholdee, N. ve Bureerat, S., 2017, Optimal reactive power dispatch problem using a two-archive multi-objective grey wolf optimizer, *Expert Systems with Applications*, 87, 79-89.

Okabe, T., Jin, Y., Olhofer, M. ve Sendhoff, B., 2004, On test functions for evolutionary multi-objective optimization, *International Conference on Parallel Problem Solving from Nature*, 792-802.

Oltean, M., Grosan, C., Abraham, A. ve Koppen, M., 2005, Multiobjective optimization using adaptive pareto archived evolution strategy, *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, 558-563.

Osyczka, A., 1985, Multicriteria optimization for engineering design, In: Design optimization, Eds: Elsevier, p. 193-227.

Otsu, N., 1979, A threshold selection method from gray-level histograms, *IEEE transactions on systems, man, and cybernetics*, 9 (1), 62-66.

Özkış, A., 2017, Girdap arama ve yapay alg algoritmalarının çok amaçlı optimizasyon problemlerine uyarlanması, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Özkış, A. ve Babalık, A., 2017, A novel metaheuristic for multi-objective optimization problems: The multi-objective vortex search algorithm, *Information Sciences*, 402, 124-148.

Özsağlam, M. Y. ve Çunkaş, M., 2008, Optimizasyon problemlerinin çözümü için parçaçık sürü optimizasyonu algoritması, *Politeknik Dergisi*, 11 (4), 299-305.

Pareto, V., 1896, Cours d'Economie Politique, *Lausanne, Switzerland*.

Peng, W. ve Zhang, Q., 2008, A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems, *2008 IEEE international conference on granular computing*, 534-537.

Riquelme, N., Von Lücken, C. ve Baran, B., 2015, Performance metrics in multi-objective optimization, *2015 Latin American Computing Conference (CLEI)*, 1-11.

Rodríguez, L., Castillo, O., Soria, J., Melin, P., Valdez, F., Gonzalez, C. I., Martinez, G. E. ve Soto, J., 2017, A fuzzy hierarchical operator in the grey wolf optimizer algorithm, *Applied Soft Computing*, 57, 315-328.

Rosenberg, R. S., 1967, Simulation of genetic populations with biochemical properties, PhD, University of Michigan, Ann Harbor Michigan.

Roshanian, J., Bataleblu, A. A., Farghadani, M. H. ve Ebrahimi, B., 2017, Multiobjective multidisciplinary design optimization of a general aviation aircraft, *Modares Mechanical Engineering*, 17 (2), 199-210.

Rudolph, G. ve Agapie, A., 2000, Convergence properties of some multi-objective evolutionary algorithms, *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, 1010-1016.

Sağ, T., 2008, Çok kriterli optimizasyon için genetik algoritma yaklaşımları, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Sağ, T., 2015, Sürü zekâsı kullanarak renkli görüntü segmentasyon tekniklerinin geliştirilmesi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*.

Saremi, S., Mirjalili, S. Z. ve Mirjalili, S. M., 2015, Evolutionary population dynamics and grey wolf optimizer, *Neural Computing and Applications*, 26 (5), 1257-1263.

Satapathy, S. C., Raja, N. S. M., Rajinikanth, V., Ashour, A. S. ve Dey, N., 2018, Multi-level image thresholding using Otsu and chaotic bat algorithm, *Neural Computing and Applications*, 29 (12), 1285-1307.

Schaffer, J. D., 1984, Multiple objective optimization with vector evaluated genetic algorithm, PhD, Vanderbilt University.

Schaffer, J. D., 1985, Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proceedings of the 1st International Conference on Genetic Algorithms*, 93-100.

Sierra, M. R. ve Coello, C. A. C., 2005, Improving PSO-based multi-objective optimization using crowding, mutation and∈-dominance, *International conference on evolutionary multi-criterion optimization*, 505-519.

Singh, N. ve Singh, S., 2017, A novel hybrid GWO-SCA approach for optimization problems, *Engineering Science and Technology, an International Journal*, 20 (6), 1586-1601.

Srinivas, N. ve Deb, K., 1994, Muiltiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary computation*, 2 (3), 221-248.

Tan, K. C., Lee, T. H. ve Khor, E. F., 2002, Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons, *Artificial intelligence review*, 17 (4), 251-290.

Tang, D., Yang, J., Dong, S. ve Liu, Z., 2016, A lévy flight-based shuffled frog-leaping algorithm and its applications for continuous optimization problems, *Applied Soft Computing*, 49, 641-662.

Tawhid, M. A. ve Savsani, V., 2018, A novel multi-objective optimization algorithm based on artificial algae for multi-objective engineering design problems, *Applied intelligence*, 48 (10), 3762-3781.

Tripathi, A. K., Sharma, K. ve Bala, M., 2018, A novel clustering method using enhanced grey wolf optimizer and mapreduce, *Big data research*, 14, 93-100.

Tripathi, P. K., Bandyopadhyay, S. ve Pal, S. K., 2007, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, *Information Sciences*, 177 (22), 5033-5049.

Tuba, E., Alihodzic, A. ve Tuba, M., 2017, Multilevel image thresholding using elephant herding optimization algorithm, *2017 14th international conference on engineering of modern electric systems (EMES)*, 240-243.

Uymaz, S. A., Tezel, G. ve Yel, E., 2015a, Artificial algae algorithm (AAA) for nonlinear global optimization, *Applied Soft Computing*, 31, 153-171.

Uymaz, S. A., Tezel, G. ve Yel, E., 2015b, Artificial algae algorithm with multi-light source for numerical optimization and applications, *Biosystems*, 138, 25-38.

Van Veldhuizen, D. A. ve Lamont, G. B., 1998, Multiobjective evolutionary algorithm research: A history and analysis, *Citeseer*.

Van Veldhuizen, D. A., 1999, Multiobjective evolutionary algorithms: classifications, analyses, and new innovations, *AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING*.

Vlennet, R., Fonteix, C. ve Marc, I., 1996, Multicriteria optimization using a genetic algorithm for determining a Pareto set, *International Journal of Systems Science*, 27 (2), 255-260.

Wang, L., Zhou, G., Xu, Y. ve Liu, M., 2012, An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling, *The International Journal of Advanced Manufacturing Technology*, 60 (9), 1111-1123.

Xiang, Y. ve Zhou, Y., 2015, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, *Applied Soft Computing*, 35, 766-785.

Xiang, Y., Zhou, Y. ve Liu, H., 2015, An elitism based multi-objective artificial bee colony algorithm, *European Journal of Operational Research*, 245 (1), 168-193.

Xu, Z. ve Xia, M., 2011, Distance and similarity measures for hesitant fuzzy sets, *Information Sciences*, 181 (11), 2128-2138.

Yagmahan, B. ve Yenisey, M. M., 2010, A multi-objective ant colony system algorithm for flow shop scheduling problem, *Expert Systems with Applications*, 37 (2), 1361-1368.

Yang, X.-S. ve Deb, S., 2009, Cuckoo search via Lévy flights, *2009 World congress on nature & biologically inspired computing (NaBIC)*, 210-214.

Yang, X.-S. ve Deb, S., 2013, Multiobjective cuckoo search for design optimization, *Computers & Operations Research*, 40 (6), 1616-1624.

Yu, C.-L., Lu, Y. ve Chu, J., 2012, Multi-objective optimization with combination of particle swarm and extremal optimization for constrained engineering design, *WSEAS Trans Syst Control*, 4 (7), 129-138.

Zapotecas-Martinez, S., Garcia-Najera, A. ve Lopez-Jaimes, A., 2019, Multi-objective grey wolf optimizer based on decomposition, *Expert Systems with Applications*, 120, 357-371.

Zapotecas-Martínez, S., López-Jaimes, A. ve García-Nájera, A., 2019, Libea: A lebesgue indicator-based evolutionary algorithm for multi-objective optimization, *Swarm and Evolutionary Computation*, 44, 404-419.

Zhang, Q. ve Li, H., 2007, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE transactions on Evolutionary Computation*, 11 (6), 712-731.

Zhen, Z., Wang, D. ve Liu, Y., 2009, Improved shuffled frog leaping algorithm for continuous optimization problem, *2009 IEEE Congress on Evolutionary Computation*, 2992-2995.

Zhong, Y.-B., Xiang, Y. ve Liu, H.-L., 2014, A multi-objective artificial bee colony algorithm based on division of the searching space, *Applied intelligence*, 41 (4), 987-1011.

Zhou, J. ve Yao, X., 2017, A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition, *International Journal of Production Research*, 55 (16), 4765-4784.

Zhu, Z. ve Zhou, X., 2020, An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints, *Computers & Industrial Engineering*, 140, 106280.

Zitzler, E. ve Thiele, L., 1998, Multiobjective optimization using evolutionary algorithms—a comparative case study, *International conference on parallel problem solving from nature*, 292-301.

Zitzler, E., 1999, Evolutionary algorithms for multiobjective optimization: Methods and applications, Citeseer, p.

Zitzler, E. ve Thiele, L., 1999, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE transactions on Evolutionary Computation*, 3 (4), 257-271.

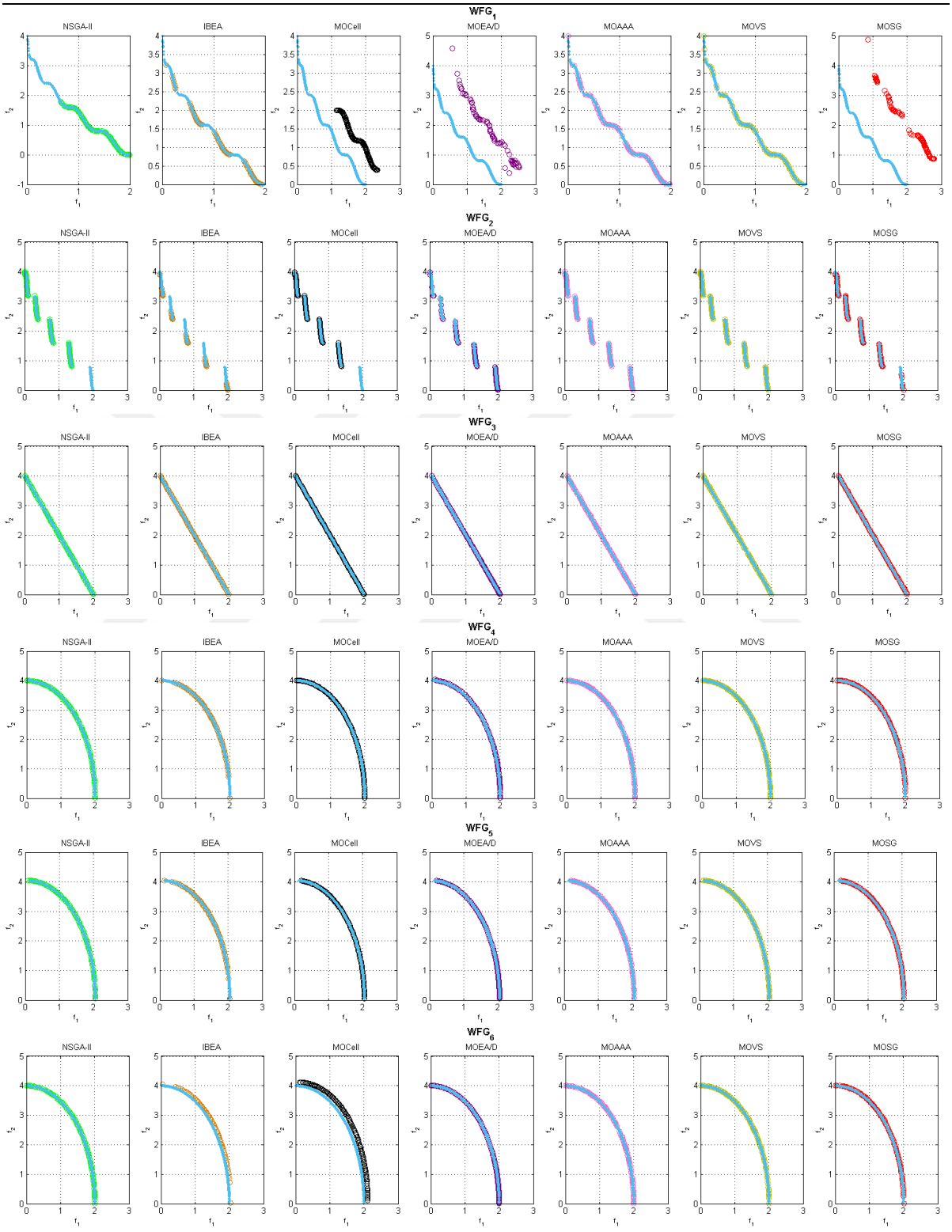
Zitzler, E., Deb, K. ve Thiele, L., 2000, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary computation*, 8 (2), 173-195.

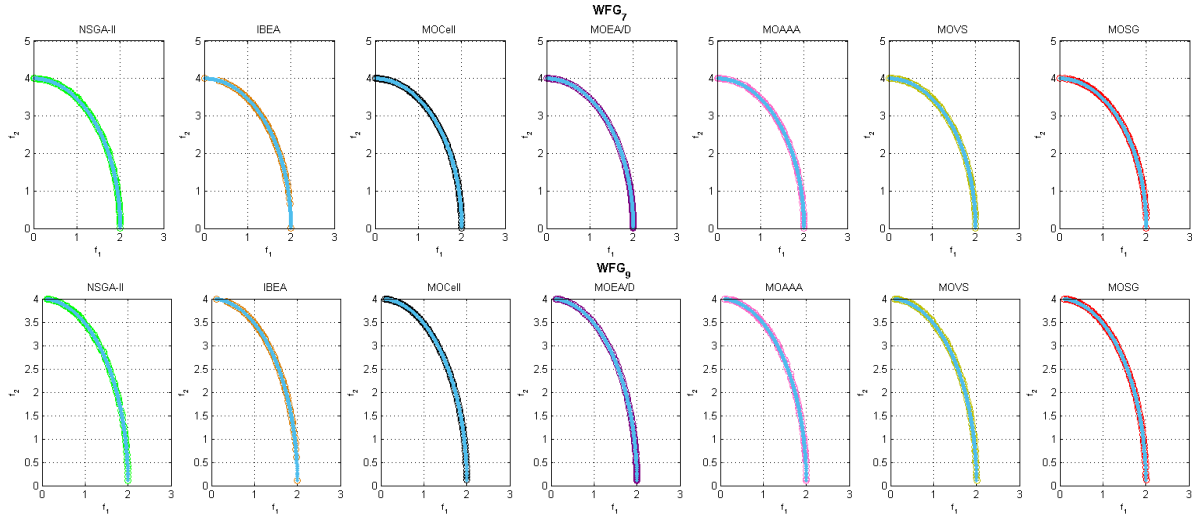
Zitzler, E., Laumanns, M. ve Thiele, L., 2001, SPEA2: Improving the strength Pareto evolutionary algorithm, *TIK-report*, 103.

Zitzler, E. ve Künzli, S., 2004, Indicator-based selection in multiobjective search, *International conference on parallel problem solving from nature*, 832-842.

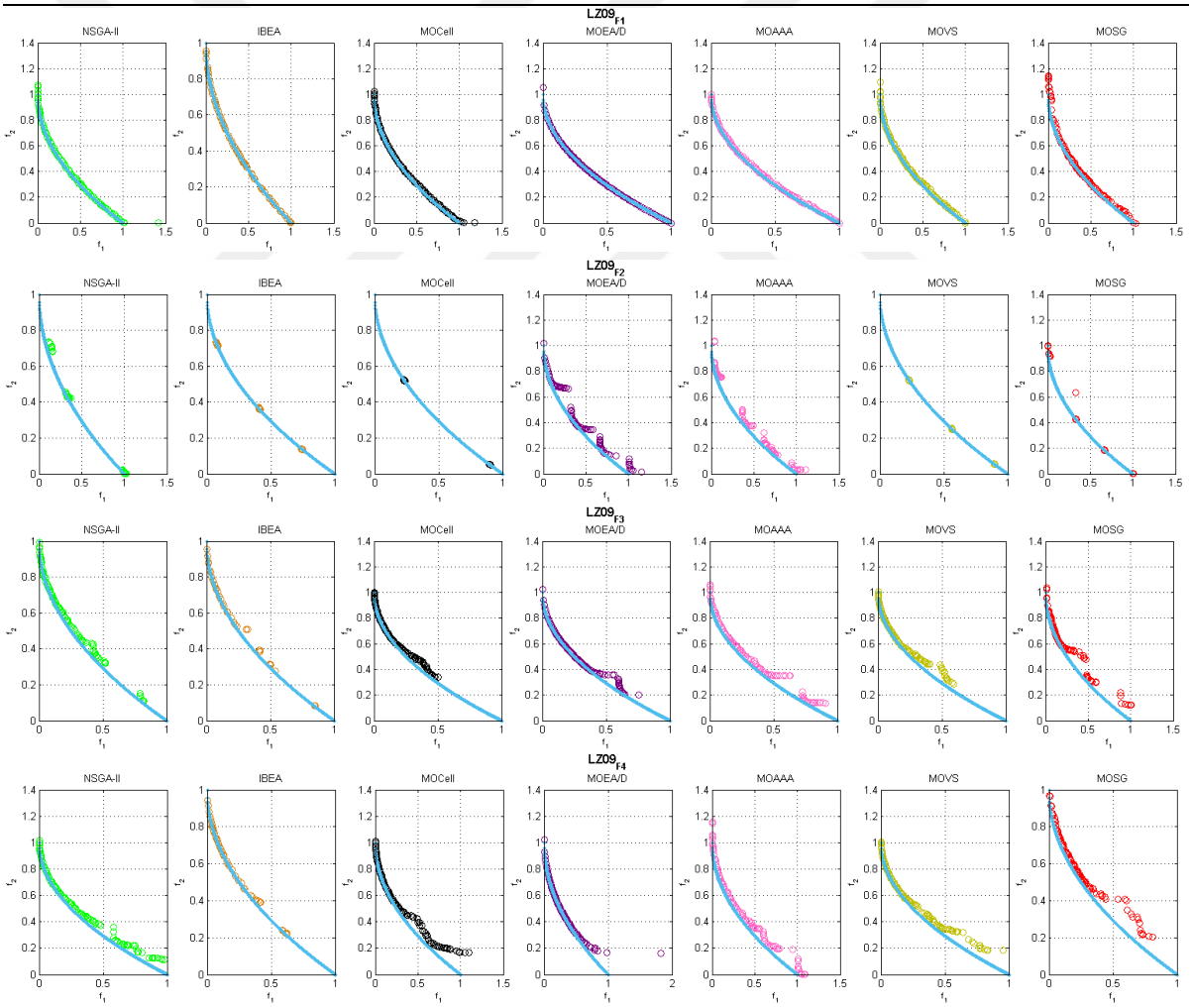
EKLER

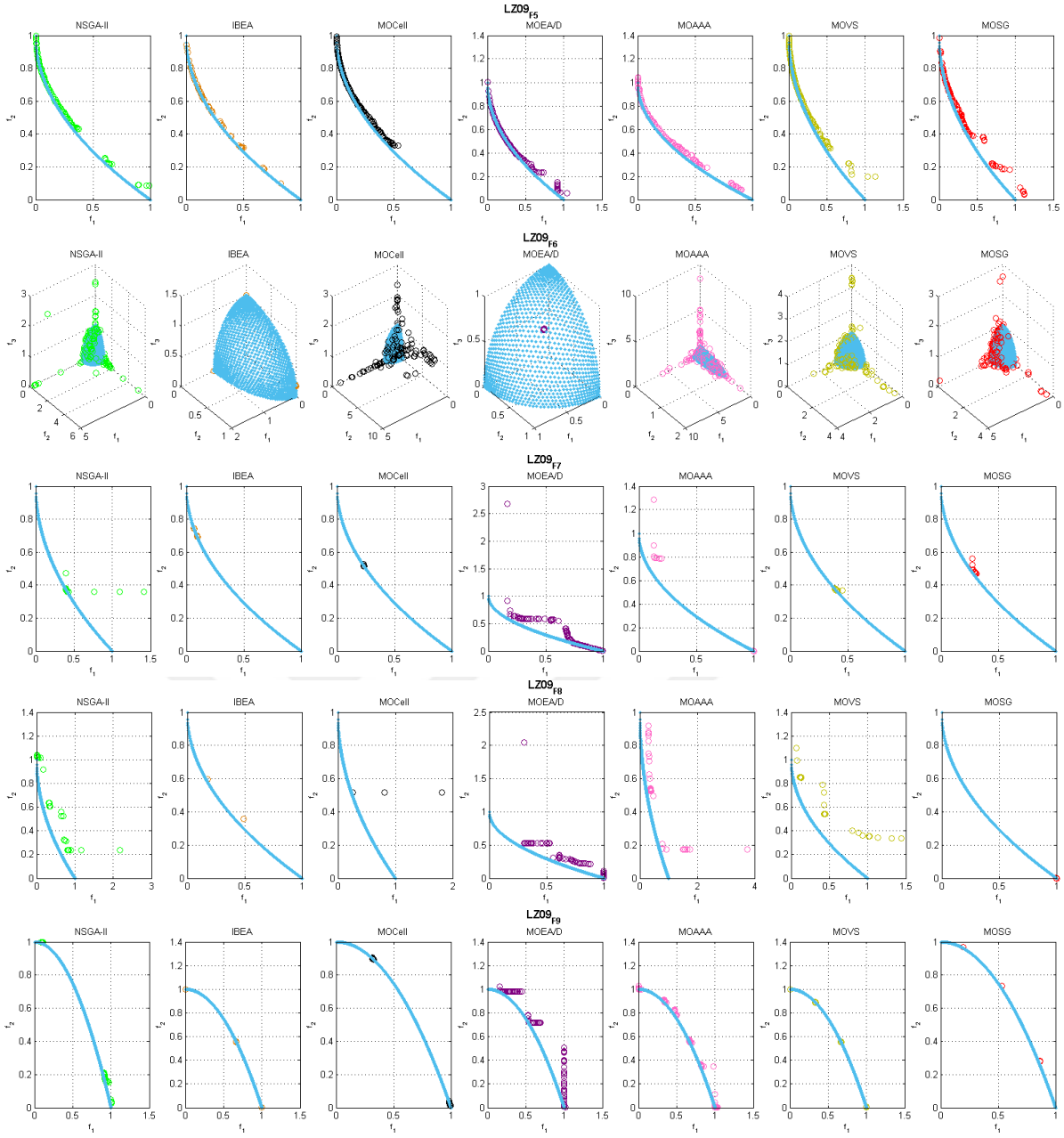
EK-1: Problem Seti-1 için Algoritmaların Ürettiği *PF* Çözümleri ve Sonuçların Kutu Grafikleri



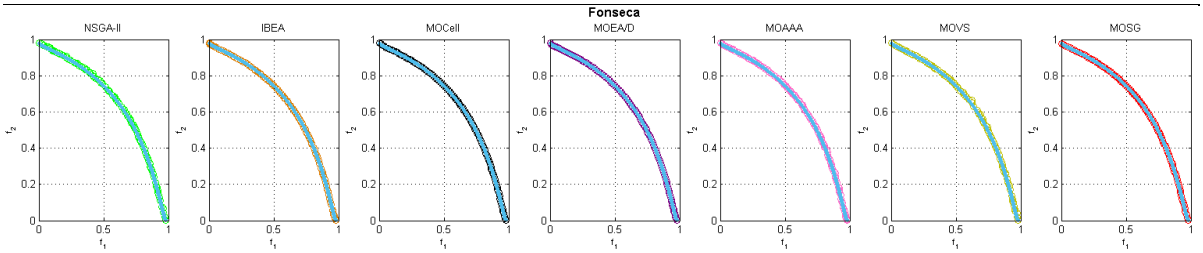


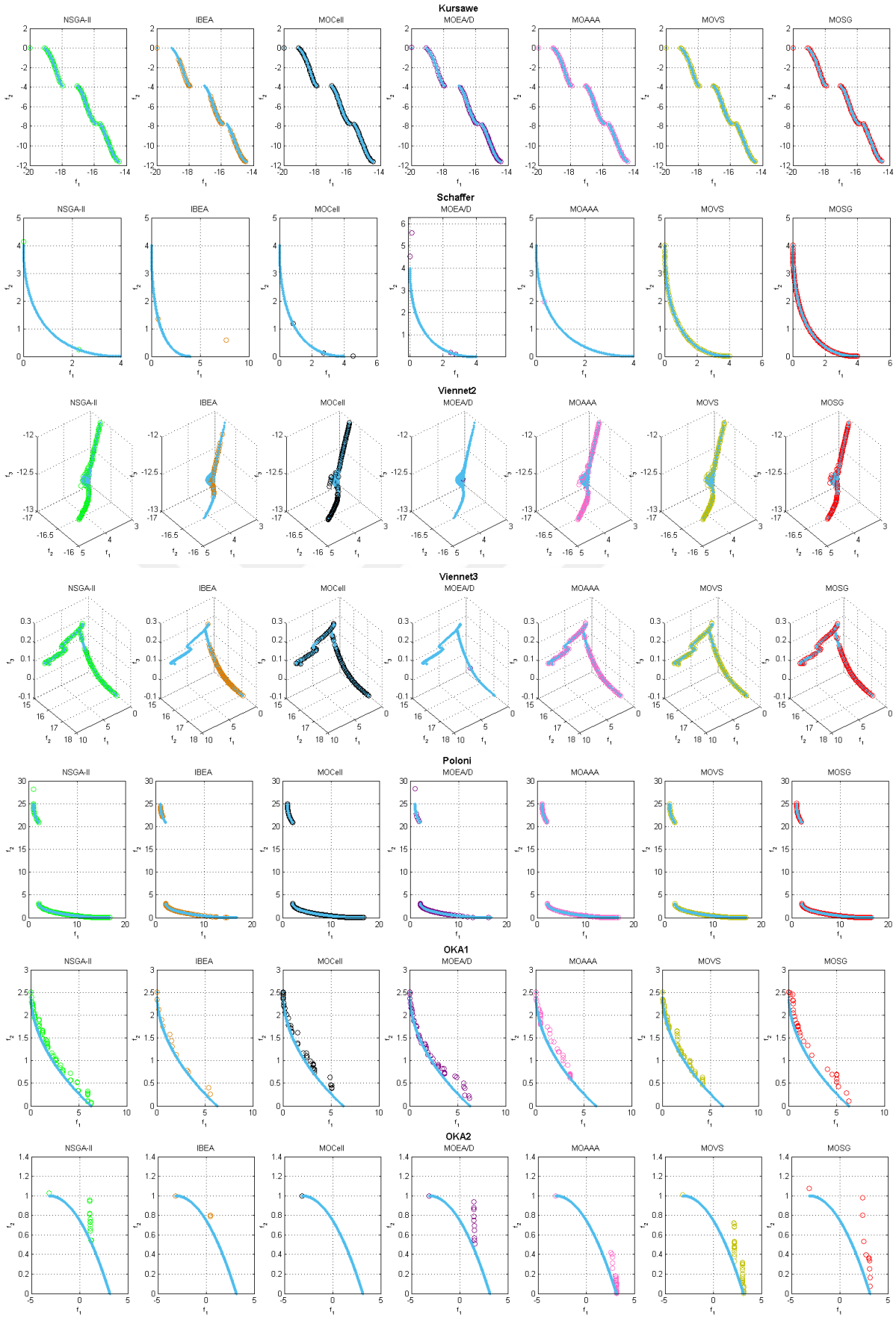
Şekil Ek-1.1. WFG problem ailesi için algoritmaların ürettiği *PF*'lerin PF_t'ler ile karşılaştırılması



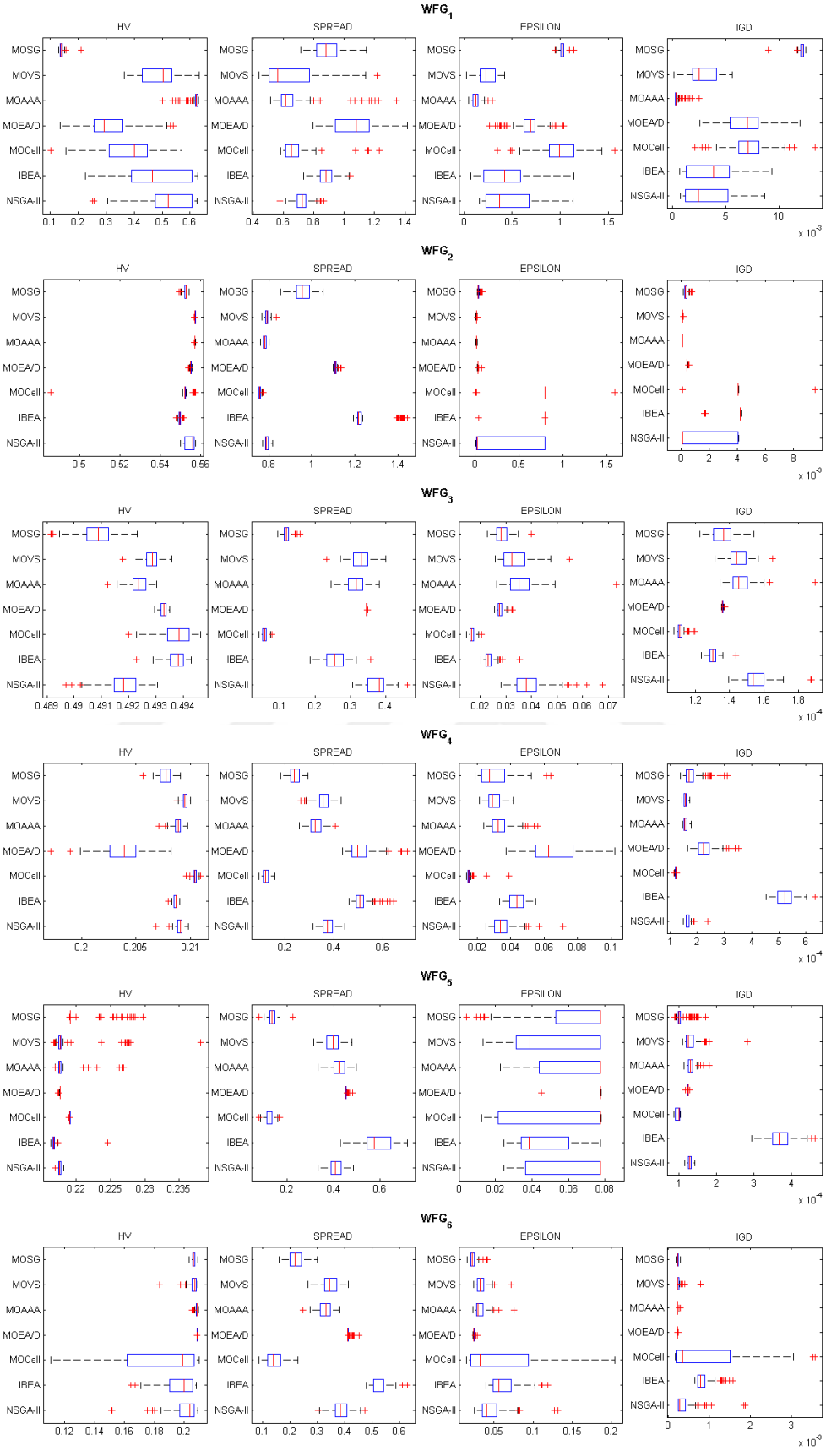


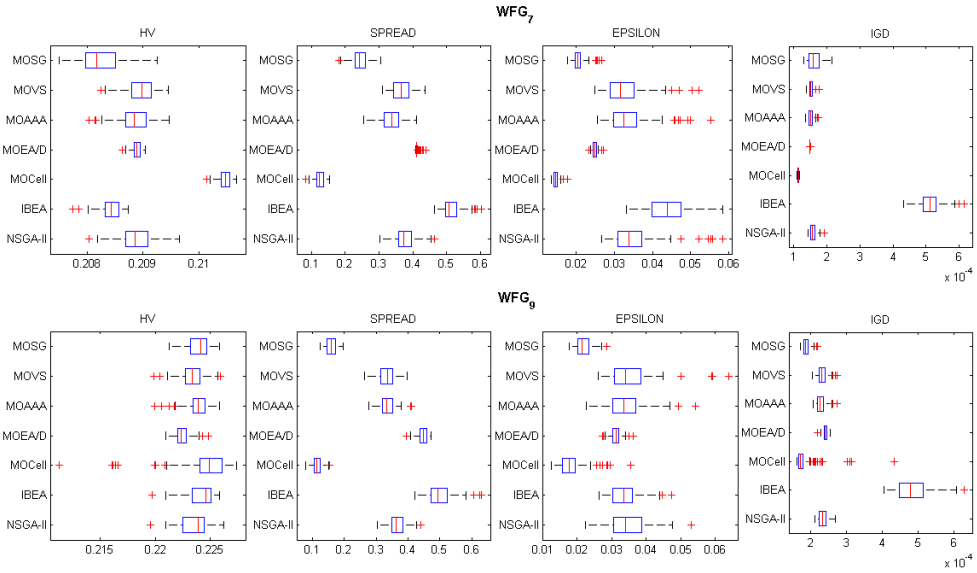
Şekil Ek-1.2. LZ09F problem ailesi için algoritmaların ürettiği *PF*'lerin PF_t'ler ile karşılaştırılması



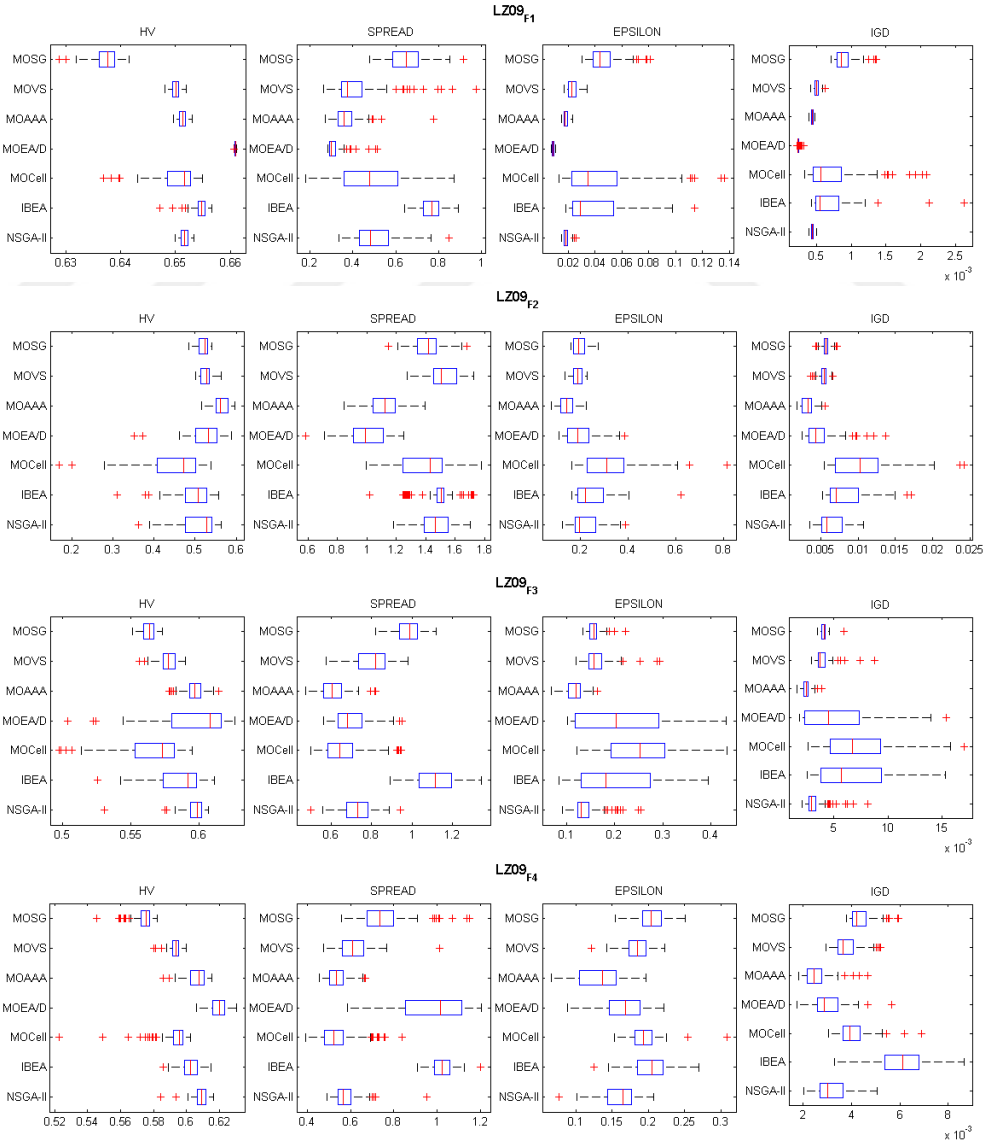


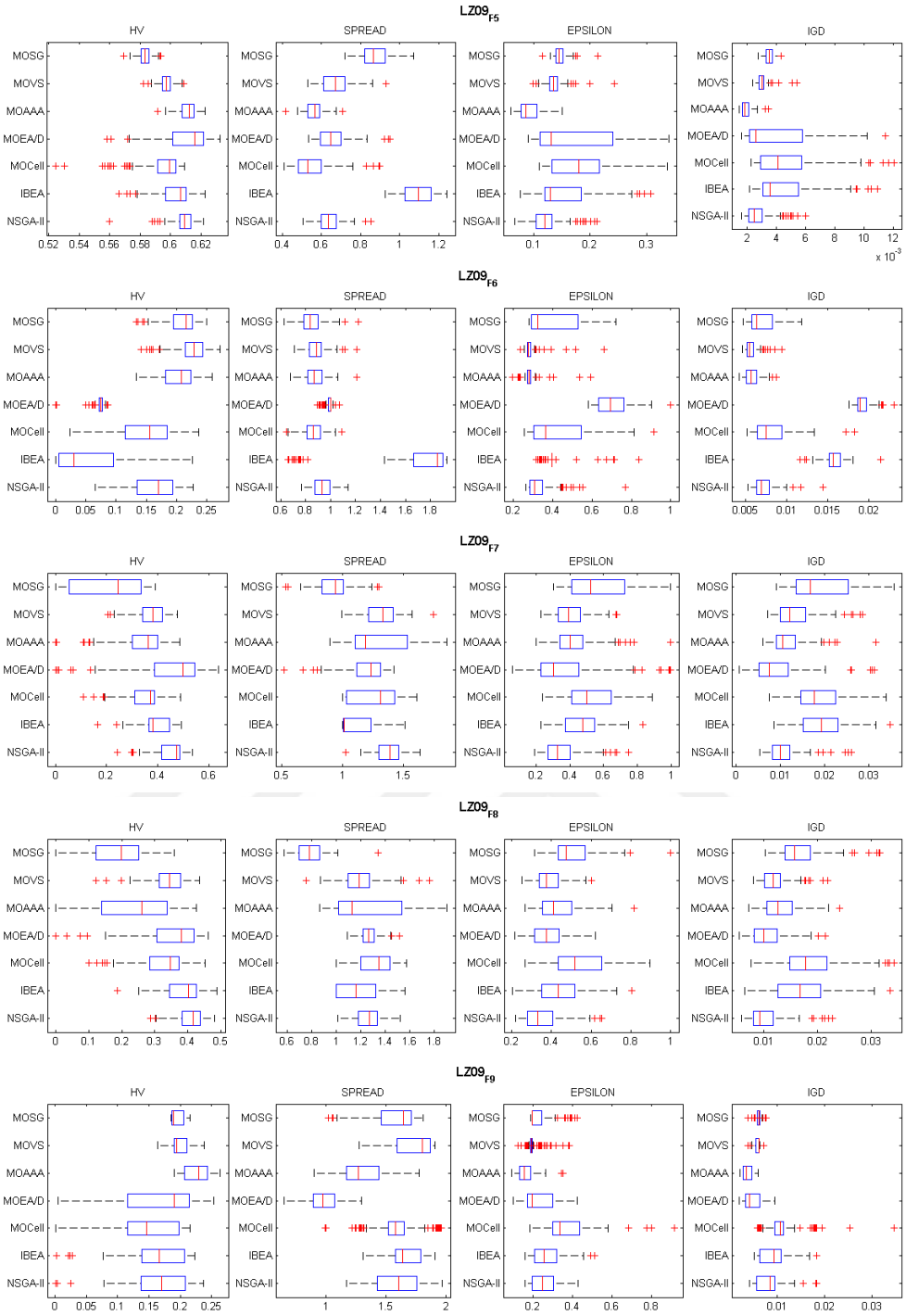
Şekil Ek-1.3. Klasik problemler için algoritmaların ürettiği *PF*'lerin PF_t'ler ile karşılaştırılması


WFG1
HV
SPREAD
EPSILON
IGD
WFG2
WFG3
WFG4
WFG5
WFG6
MOSG
MOVS
MOAAA
MOEA/D
MOCell
IBEA
NSGA-II

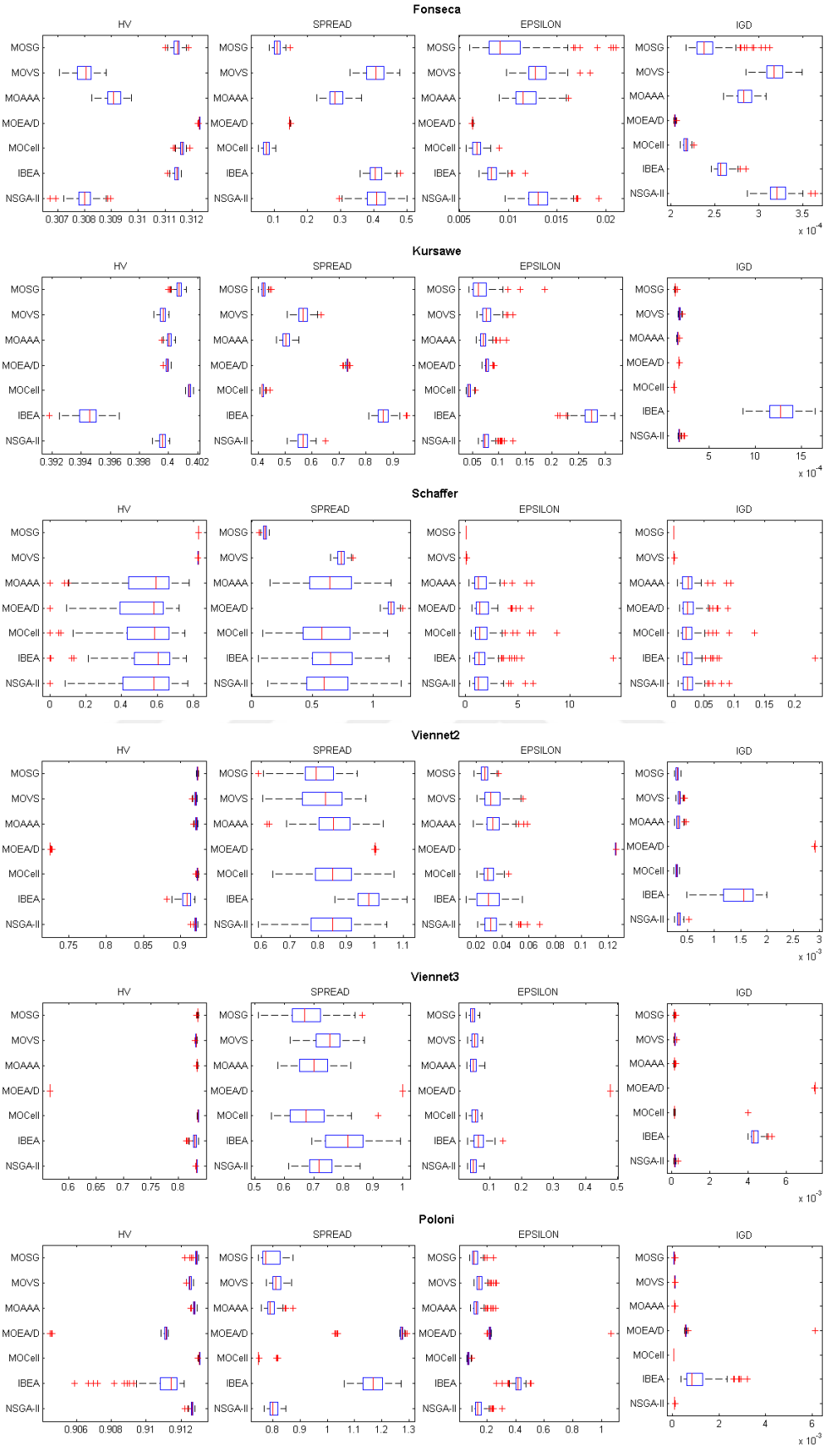


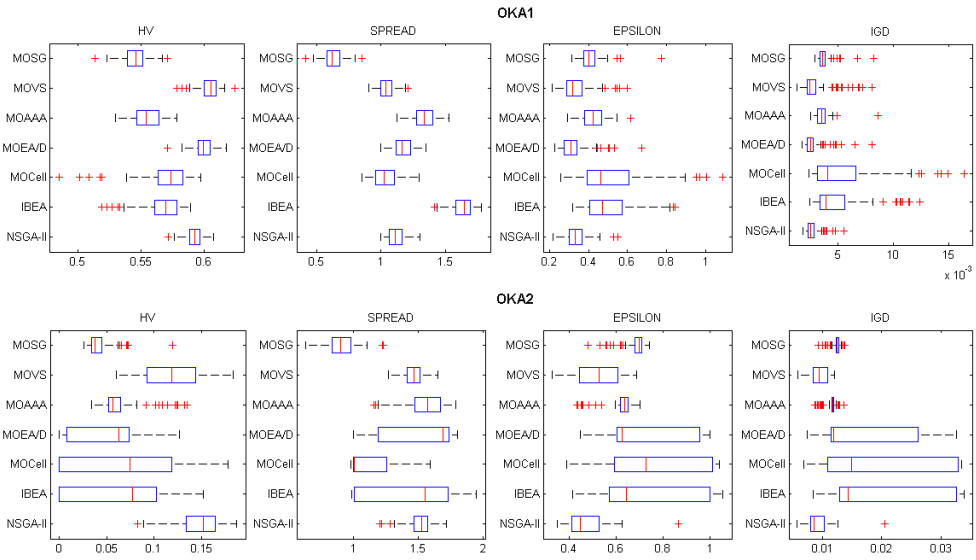
Şekil Ek-1.4. WFG problem ailesi için algoritmaların ürettiği sonuçların kutu grafiği





Şekil Ek-1.5. LZ09F problem ailesi için algoritmaların ürettiği sonuçların kutu grafiği


Fonseca
HV
SPREAD
EPSILON
IGD
Kursawe
HV
SPREAD
EPSILON
IGD
Schaffer
HV
SPREAD
EPSILON
IGD
Viennet2
HV
SPREAD
EPSILON
IGD
Viennet3
HV
SPREAD
EPSILON
IGD
Poloni
HV
SPREAD
EPSILON
IGD
MOSG
MOVS
MOAAA
MOEA/D
MOCell
IBEA
NSGA-II



Şekil Ek-1.6. Klasik problemler için algoritmaların ürettiği sonuçların kutu grafiği

EK-2: Problem Seti-3 için Algoritmaların Elde Ettiği Segmentasyon Görüntüleri

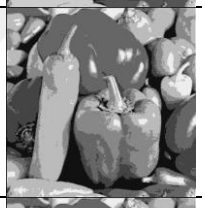
TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

Şekil Ek-2.1. I_1 görüntüsü için elde edilen segmentasyon sonuçları

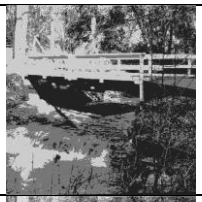
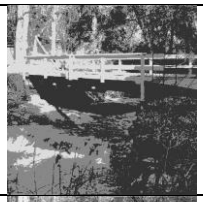
Şekil Ek-2.2. I_2 görüntüsü için elde edilen segmentasyon sonuçları

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

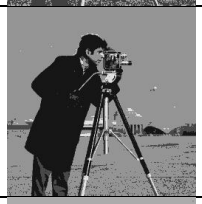
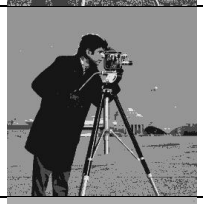
Şekil Ek-2.3. I_4 görüntüsü için elde edilen segmentasyon sonuçları

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

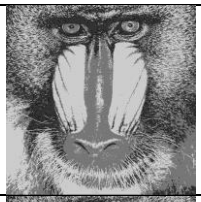
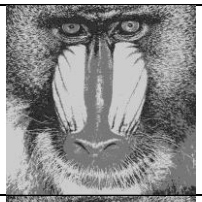
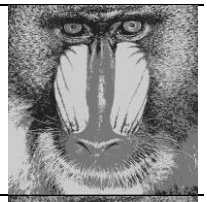
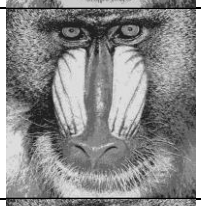
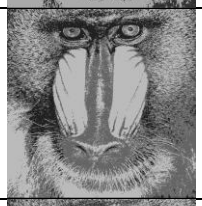
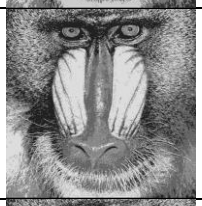
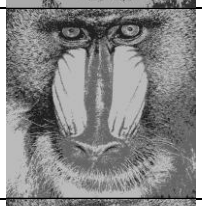
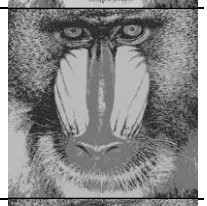
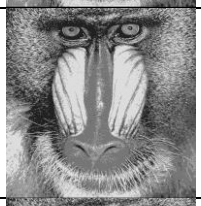
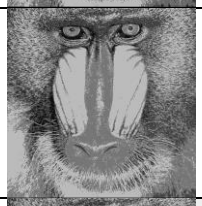
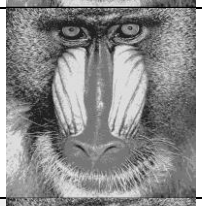
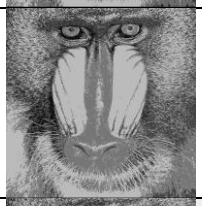
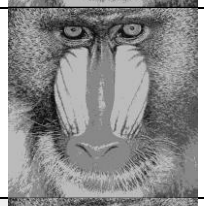
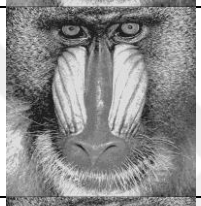
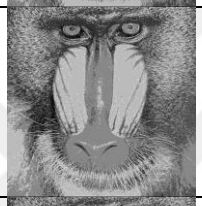
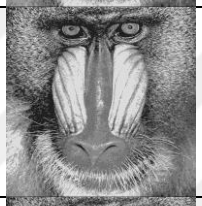
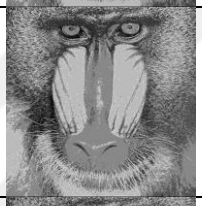
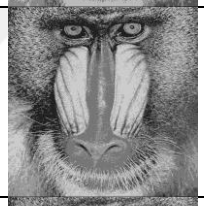
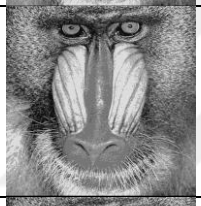
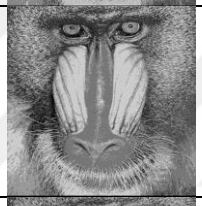
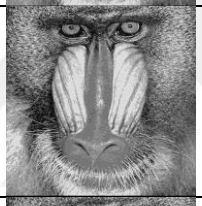
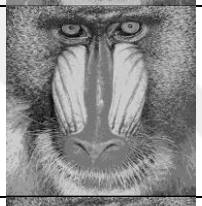
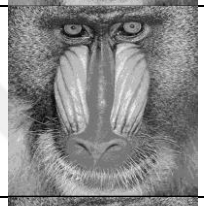
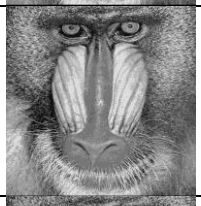
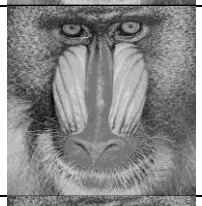
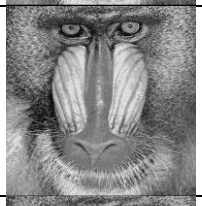
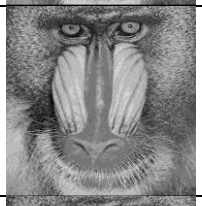
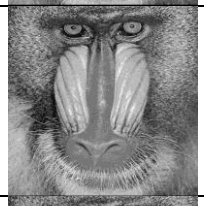
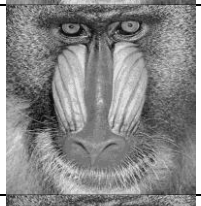
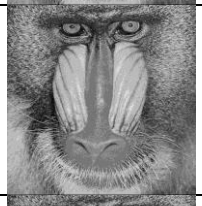
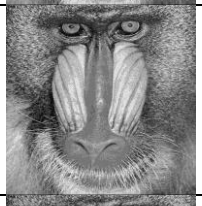
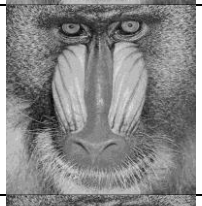
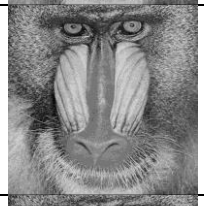
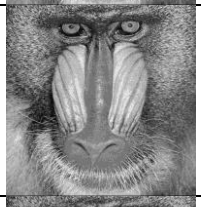
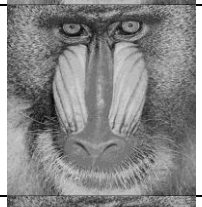
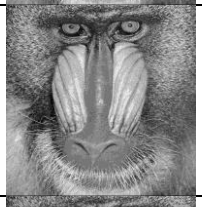
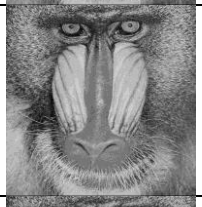
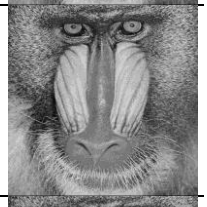
Şekil Ek-2.4. I_5 görüntüsü için elde edilen segmentasyon sonuçları

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

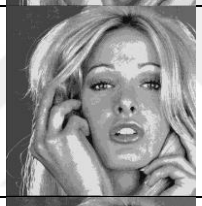
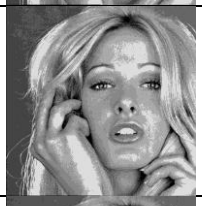
Şekil Ek-2.5. I_6 görüntüsü için elde edilen segmentasyon sonuçları

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

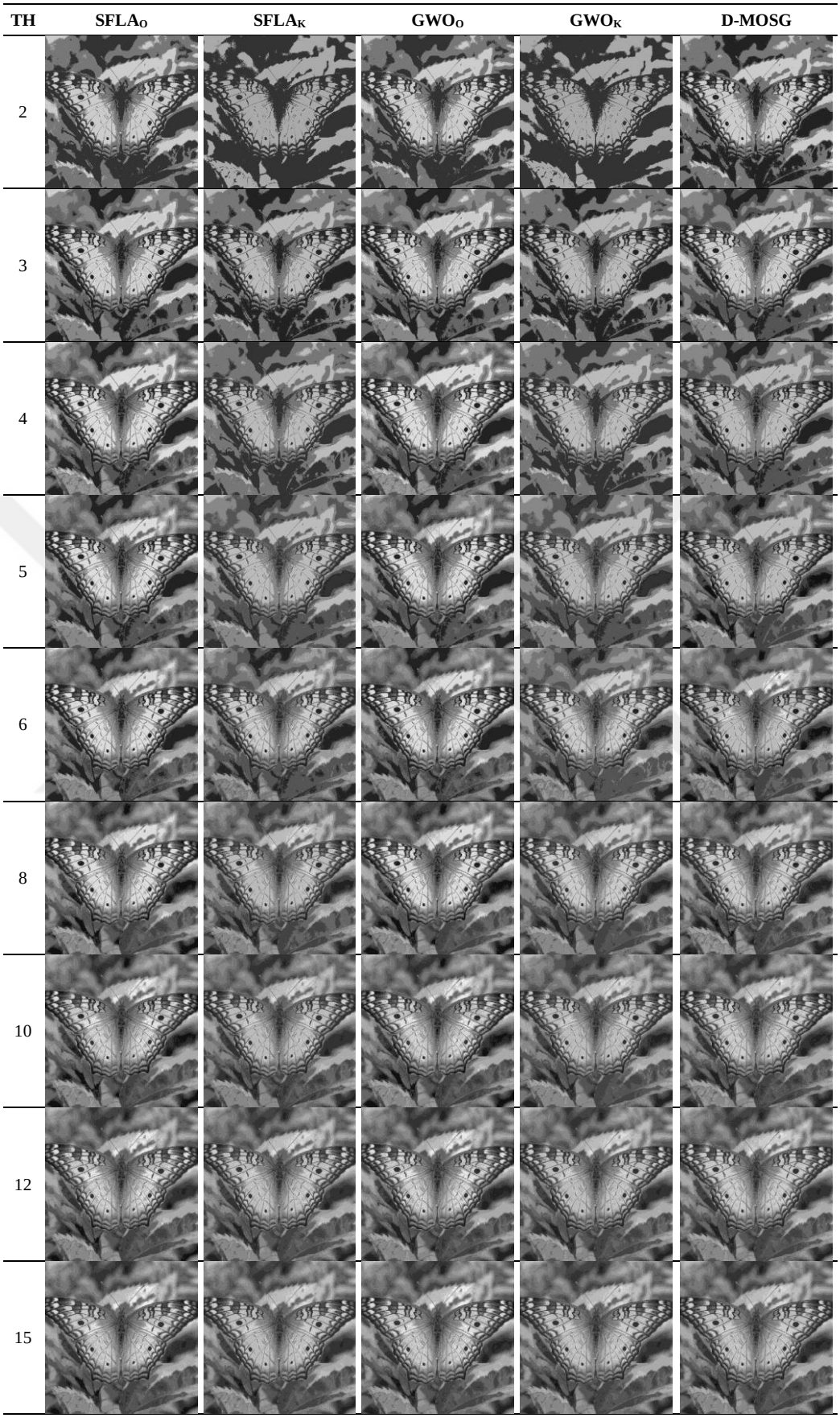
Şekil Ek-2.6. I_7 görüntüsü için elde edilen segmentasyon sonuçları

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

Şekil Ek-2.7. I_8 görüntüsü için elde edilen segmentasyon sonuçları

TH	$SFLA_O$	$SFLA_K$	GWO_O	GWO_K	D-MOSG
2					
3					
4					
5					
6					
8					
10					
12					
15					

Şekil Ek-2.8. I_9 görüntüsü için elde edilen segmentasyon sonuçları



Şekil Ek-2.9. I_{10} görüntüsü için elde edilen segmentasyon sonuçları