POPÜLASYON TEMELLİ METASEZGİSELLER İÇİN GENELLEŞTİRİLMİŞ ÇATILARIN TASARIMI VE GERÇEKLEŞTİRİMİ

Doktora Tezi

Gürcan YAVUZ

Eskişehir 2019

POPÜLASYON TEMELLİ METASEZGİSELLER İÇİN GENELLEŞTİRİLMİŞ ÇATILARIN TASARIMI VE GERÇEKLEŞTİRİMİ

Gürcan YAVUZ

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Doğan AYDIN

Eskişehir
Eskişehir Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Haziran 2019

JÜRİ VE ENSTİTÜ ONAYI

Gürcan YAVUZ'un "Popülasyon Temelli Metasezgiseller İçin Genelleştirilmiş Çatıların Tasarımı ve Gerçekleştirimi" başlıklı tezi 20/06/2019 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca, Bilgisayar Mühendisliği Anabilim dalında Doktora tezi olarak kabul edilmiştir.

Jüri Üyeleri	**Unvanı Adı Soyadı**	**İmza**
Üye (Tez Danışmanı)	: Doç. Dr. Doğan AYDIN	……
Üye	: Doç. Dr. Serkan GÜNAL	……
Üye	: Dr. Öğr. Üyesi Burhanettin DURMUŞ	……
Üye	: Dr. Öğr. Üyesi Alper BİLGE	……
Üye	: Dr. Öğr. Üyesi Sevcan YILMAZ GÜNDÜZ	……

Prof. Dr. Murat TANIŞLI
Lisansüstü Eğitim Enstitüsü Müdürü

ÖZET

POPÜLASYON TEMELLİ METASEZGİSELLER İÇİN GENELLEŞTİRİLMİŞ ÇATILARIN TASARIMI VE GERÇEKLEŞTİRİMİ

Gürcan YAVUZ

Bilgisayar Mühendisliği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Haziran 2019

Danışman: Doç. Dr. Doğan AYDIN

Popülasyon tabanlı metasezgisel algoritmaların problem çözmede gösterdiği başarı bu algoritmalara olan ilginin artmasına sebep olmuştur. Yapay Arı Kolonisi ve Parçacık Sürü Optimizasyon algoritmaları, öne çıkan popülasyon tabanlı metasezgisel algoritmalardan iki tanesidir. Araştırmacılar, bu algoritmaların yapısına dahil ettikleri ufak değişiklikler ile gelişmiş yeni varyantlar ortaya koymuşlardır. Bu yeni önerilmiş varyantlar genellikle araştırmacıların kullandıkları problem setine ve araştırmacıların tecrübelerine bağlı olmaktadır. Bu da algoritmaların karşılaşmadıkları problem türlerinde yeterli başarıyı gösterememesine yol açmaktadır. Ortaya çıkan bu problemi aşmak için bu tez çalışmasında ABC ve PSO algoritmaları için genelleştirilmiş algoritma çatıları önerilmiştir. Çeşitli ABC varyantlarının özelliklerini içinde barındıran ve bir otomatik yapılandırma aracı ile probleme özgü ABC algoritması üretmesini sağlayan ABC-X isminde bir genelleştirilmiş algoritma çatısı önerilmiştir. Bu algoritmanın başarısı elli adet problem içeren ölçüt seti ile test edilmiştir. Ayrıca algoritma bir gerçek dünya problemi olan valf etkili güç dağıtım probleminin çözümünde kullanılmıştır. Bir diğer yapılan çalışma PSO algoritması için Şablon PSO isminde bir algoritma önerilmesidir. Algoritmanın performansı CEC'17 ölçüt seti ve SOCO yüksek boyut ölçüt seti kullanılarak analiz edilmiştir. Son olarak, algoritmaların performansındaki en büyük pay sahibi olan bileşenin algoritmalarda yer alan arama denklemi olduğu düşüncesiyle kendinden uyarlanabilir arama denklemi tabanlı ABC (SSEABC) algoritması önerilmiştir. SSEABC'nin performansı, CEC'17 ve SOCO ölçüt setleri ile test edilmiştir. Ayrıca SSEABC, bir mühendislik problemi olan filtre tanıma probleminin çözümünde de kullanılmıştır. Önerilmiş olan üç algoritma da karşılaştırıldıkları algoritmalardan daha iyi ve rekabetçi sonuçlar elde etmiştir.

Anahtar kelimeler: Yapay arı kolonisi, Parçacık sürü optimizasyonu, Algoritma çatısı, Sürekli optimizasyon.

ABSTRACT

DESIGN AND IMPLEMENTATION OF GENERALIZED FRAMEWORKS FOR POPULATION-BASED METAHEURISTICS

Gürcan YAVUZ

Department of Computer Engineering

Eskişehir Technical University, Institute of Graduate Programs, June 2019

Supervisor: Assoc. Prof. Dr. Doğan AYDIN

The success of population-based meta-heuristic algorithms in problem solving has increased the interest in these algorithms. Artificial Bee Colony and Particle Swarm Optimization algorithms are two of population-based meta-heuristic algorithms. Researchers have introduced new variants that have been developed with minor changes in the structure of these algorithms. These newly proposed variants generally depend on the set of problems used by the researchers and the experience of the researchers. This leads to insufficient algorithms for the types of problems they do not encounter. In order to overcome this problem, generalized algorithm frameworks for ABC and PSO algorithms are proposed in this thesis. A generalized algorithm framework called ABC-X, which incorporates the properties of various ABC variants and enables it to generate a problem-specific ABC algorithm with an automatic configuration tool, has been proposed. The success of this algorithm was tested with a set of criteria including fifty problems. It has also been used to solve the problem of power dispatch problem with a real-world problem. Another study is to propose an algorithm called Template PSO for the PSO algorithm. The performance of the algorithm was analyzed using the CEC'17 criterion set and SOCO high dimension criterion set. Finally, a self-adaptive search equation-based ABC (SSEABC) algorithm has been proposed, considering that the component that has the greatest effect on the performance of the algorithms is the search equation. The performance of SSEABC has been tested with CEC'17 and SOCO benchmark sets. In addition, SSEABC has been used to solve an engineering problem, the filter identification problem. All three proposed algorithms have achieved better and more competitive results than the algorithms they compare.

Keywords: Artificial bee colony, Particle swarm optimization, Algorithm framework, Continuous optimization.

TEŞEKKÜR

İlk olarak, tez çalışması boyunca deneyimlerini, bilgilerini ve desteğini esirgemeyen ve en değerli zamanlarını bana ayıran danışman Hocam Sayın Doç. Dr. Doğan Aydın'a en derin teşekkürlerimi sunarım.

Bu çalışmada tez izleme jürisinde yer alan ve verdikleri fikirler, öneriler ile bana yeni bakış açıları kazandıran, Sayın Doç. Dr. Serkan GÜNAL ve Dr. Öğr. Üyesi Burhanettin DURMUŞ'a teşekkür ederim.

Eğitim hayatım boyunca bütün olanakları bana sunan, bugünlere gelmemde en büyük paya sahip olan babam Mehmet Ali YAVUZ'a ve annem Zeynep YAVUZ'a saygı ve sevgilerimi sunarım.

Tez yazım süresince bana gösterdiği sabır ve anlayışı için eşim Hatice YAVUZ'a teşekkürü bir borç bilirim.

Gürcan YAVUZ

Haziran, 2019

20/06/2019

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

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

Gürcan YAVUZ

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

TABLOLAR DİZİNİ

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

α : Feromonun etkisini kontrol eden parametre

β : Sezgisel etkiyi kontrol eden parametre

$\Delta\tau_{qj}^{k}$: k karıncasının, q ile j'nin arasına bıraktığı feromon miktarı

η_{qj} : q ve j arasındaki sezgisel bilgi

γ_{qj}^{k} : k karıncasının q'den j'ye geçme olasılığı

$\hat{x}_i$: Yeni aday çözüm

$\hat{y}(n)$: n örnek girişi için bilinmeyen sistemin çıkışı

ω : Atalet ağırlığı

ω_{min} : Atalet ağırlığının minimum değeri

ω_{max} : Atalet ağırlığının maksimum değeri

ρ : Buharlaşma oranı

τ_{qj} : q ve j arasındaki feromon miktarı

ε : Güç farkı hassasiyeti

$\phi 1$: Düzgün dağılım kullanılarak üretilen rastgele bir sayı

$\phi 2$: Düzgün dağılım kullanılarak üretilen rastgele bir sayı

$\phi 3$: Düzgün dağılım kullanılarak üretilen rastgele bir sayı

$\phi 1_{i,j}$: Düzgün dağılım kullanılarak üretilen rastgele bir sayı

$\phi 2_{i,j}$: Düzgün dağılım kullanılarak üretilen rastgele bir sayı

$\phi 3_{i,j}$: Düzgün dağılım kullanılarak üretilen rastgele bir sayı

e_{CLPSO} : CLPSO örnek üretme yöntemi ile üretilmiş örnek

e_{GLPSO} : GLPSO örnek üretme yöntemi ile üretilmiş örnek

$e_{GGLPSOD}$: GGLPSOD örnek üretme yöntemi ile üretilmiş örnek

e_{PSOTD} : PSOTD örnek üretme yöntemi ile üretilmiş örnek

a_i : i derecesindeki filtre çıkış katsayısı

b_i : i derecesindeki filtre giriş katsayısı

$c_{1,min}$: c1 öğrenme faktörünün minimum değeri

$c_{1,max}$: c1 öğrenme faktörünün maksimum değeri

$c_{2,min}$: c2 öğrenme faktörünün minimum değeri

$c_{2,max}$: c2 öğrenme faktörünün maksimum değeri

$c1, c2$: Öğrenme faktörleri

D : Problem boyutu

D_{jer} : Toplam jeneratör sayısı

d_n, e_n : Valf noktası etkileri

$dist(x,y)$: x ve y arasındaki mesafe

$f(x)$: Aday çözüm

$f(x^*)$: Optimum çözüm

$F_n(Pow_{G,n})$: n'inci üretim biriminin yakıt maliyeti fonksiyonu

g : Büyüme periyodu

$gbest$: Popülasyondaki en iyi parçacık

$H_{filter}(z)$: Filtrenin transfer fonksiyonu

$H_{system}(z)$: Bilinmeyen sistemin transfer fonksiyonu

i : i. yiyecek kaynağının [1, SN] aralığındaki indeks değeri

$itr_{current}$: Mevcut iterasyon sayısı

itr_{max} : Maksimum iterasyon sayısı

j : j. problem değişkeninin [1, D] aralığındaki indeks numarası

L_i : Parçacığın pozisyonu

M , M_e : Arama denkleminde değiştirilecek boyut sayısı

M_o , m : Arama denkleminde değiştirilecek boyut sayısı

MOD : Modülasyon işlemi

N : Popülasyondaki parçacık sayısı

O, T : IIR filtre denklemindeki pay ve paydanın derecesi

p_{GD} : Komşu x_{GD}'nin seçilme olasılığı

P_{terim1_i} : Genelleştirilmiş pozisyon denkleminin muhtemel bileşeni

P_{terim2_i} : Genelleştirilmiş pozisyon denkleminin muhtemel bileşeni

P_{terim3_i} : Genelleştirilmiş pozisyon denkleminin muhtemel bileşeni

p_i : Bir aday çözümün seçim olasılık değeri

$pbest$: Parçacığın kişisel en iyisi

$pbest_r$: Popülasyondan rastgele seçilmiş parçacığın kişisel en iyisi

$pbest_{r1}$: Popülasyondan rastgele seçilmiş parçacığın kişisel en iyisi

$pbest_{r2}$: Popülasyondan rastgele seçilmiş parçacığın kişisel en iyisi

$Pow_{G,n}$: n. üretim biriminin MW cinsinden çıkış gücü

$Pow_{G,n}^{min}$: Minimum güç sınırları

$Pow_{G,n}^{max}$: Maksimum güç sınırları

Pow_{load} : Güç yükü

Pow_{loss} : Güç kayıpları

r_i : x_i'nin uygunluk durumuna göre popülasyondaki sıralaması

Rf : Yer değiştirme faktörü

S : Giriş örneklerinin toplam sayısı

SF : Rastgele seçilen pozitif reel sayı

SN : Popülasyondaki yiyecek kaynaklarının sayısı

SN_{Max} : Maksimum popülasyon boyutu

SP : $SP \in [1.0, 2.0]$, seçim baskısı

$trail_i$: Yiyecek kaynağı deneme sayısı

V : Parçacığın hızı

$V_{i,min}$: i. parçacığın minimum hız değeri

$V_{i,max}$: i. parçacığın maksimum hız değeri

V_i : i. parçacığın hız değeri

V_{terim1_i} : Genelleştirilmiş hız denkleminin muhtemel bileşenleri

V_{terim2_i} : Genelleştirilmiş hız denkleminin muhtemel bileşenleri

V_{terim3_i} : Genelleştirilmiş hız denkleminin muhtemel bileşenleri

w : Çözümün ağırlığı

w_{min} : Kaşif arının pozisyon ayarlamasının minimum yüzdesi

w_{max} : Kaşif arının pozisyon ayarlamasının maksimum yüzdesi

X : Bulunan en iyi aday çözüm

$x(n)$: Filtrenin giriş değeri

x^* : Optimum çözüm

$x_{AVE,j}$: Popülasyondaki çözümlerin j boyundaki ortalaması

$x_{G,j}$: Global-en iyi aday çözüm

x_{gbest} : Global-en iyi aday çözüm

$x_{GD,j}$: En yakın komşu çözüm

x_{GD} : En yakın komşu çözüm

$x_{i,j}$: i. çözümün j. boyutu

$x_{MD,j}$: Ortanca çözümün j boyutu

x_{min} : Problemin minimum değeri

x_{max} : Problemin maksimum değeri

x_{new} : Üretilen yeni çözüm

$x_{r1,j}$, x_{r2} : Popülasyondan rastgele seçilmiş çözüm

$x_{SC,j}$: Popülasyonun en iyi ikinci çözümün j. boyutu

$x_{WO,j}$: Popülasyonun en kötü çözümün j. boyutu

x_i : Popülasyondaki i. çözüm

$y(n)$: n örnek girişi için IIR filtrenin çıkışı

ABC : Artificial Bee Colony (Yapay Arı Kolonisi)

ABC-X : Genelleştirilmiş Yapay Arı Kolonisi

AC : Alternative Current (Alternatif Akım)

ACO : Ant Colony Optimization
(Karınca Kolonisi Optimizasyonu)

ACOR : Sürekli Optimizasyon için Karınca Koloni Optimizasyonu

AYP : Algoritma Yapılandırma Problemi

BABC : Best-so-far Selection Artificial Bee Colony
(Şimdiye kadarki en iyi seçim Yapay Arı Kolonisi)

BFO : Bacteria Foraging Optimization
(Bakteri Yiyecek Arama Optimizasyonu)

BHO : Black Hole Optimization (Kara Delik Optimizasyonu)

CABC : Chaotic Artificial Bee Colony (Kaotik Yapay Arı Kolonisi)

CEC'05 : 2005 Congress on Evolutionary Computation
(2005 IEEE Evrimsel Hesaplama Kongresi)

CEC'14 : 2014 Congress on Evolutionary Computation
(2014 IEEE Evrimsel Hesaplama Kongresi)

CEC'17 : 2017 Congress on Evolutionary Computation
(2017 IEEE Evrimsel Hesaplama Kongresi)

CLPSO : Comprensive Learning Particle Swarm Optimization
(Kapsamlı Öğrenme Parçacık Sürü Optimizasyonu)

CMAES : Covariance Matrix Adaptation Evolution Strategy
(Kovaryans Matrisi Adaptasyon Gelişim Stratejisi)

CompABC : Composite Artificial Bee Colony
(Kompozit Yapay Arı Kolonisi)

CompBudget : Computational Budget (Sabit fonksiyon çağırım sayısı)

CSO : Cat Swarm Optimization
(Kedi Sürüsü Optimizasyonu)

dB : Decibel (Desibel)

DE : Differential Evolution (Diferansiyel Gelişim)

EABC : Enhanced Artificial Bee Colony
(Geliştirilmiş Yapay Arı Kolonisi)

ELPSO : Example-Based Particle Swarm Optimization
(Örnek Tabanlı Parçacık Sürü Optimizasyonu)

EPDP : Ekonomik Güç Dağıtım Problemi

ERABC : Efficient and Robust Artificial Bee Colony
(Verimli ve Sağlam Yapay Arı Kolonisi)

ES : Evolution Strategy (Evrimsel Strateji)

FA : Firefly Algorithm (Ateş Böceği Algoritması)

FEs : Function Evaluations (Fonksiyon Çağrım Sayısı)

FFO : Fruit Fly Optimization
(Meyve Sineği Optimizasyonu)

FrankPSO : Frankenstein Parçacık Sürü Optimizasyonu

GA : Genetic Algorithm (Genetik algoritma)

GABC : Global-best Artificial Bee Colony
(Global-en-iyi Yapay Arı Kolonisi)

GB : Gigabyte

GDABC : Global-best Distance guided Artificial Bee Colony
(Global-en iyi mesafe rehberli Yapay Arı Kolonisi)

GGA : Gender based Genetic Algorithm
(Cinsiyete dayalı Genetik Algoritma)

GGLPSOD : Çeşitlilik ile Global Genetik Parçacık Sürü Optimizasyonu

GHz : Gigahertz

GLPSO : Genetic Learning Particle Swarm Optimization
(Genetik Öğrenme Parçacık Sürü Optimizasyonu)

HS : Harmony Search (Harmoni Arama)

IABC : Incremental Artificial Bee Colony (Artan Yapay Arı Kolonisi)

IIR : Infinitive Impulse Response (Sonsuz Dürtü Yanıtı)

ILS : Iterated Local Search (Yinelemeli Yerel Arama)

IMABC : Improved Artificial Bee Colony
(Geliştirilmiş Yapay Arı Kolonisi)

IMPABC : Improved Artificial Bee Colony
(Geliştirilmiş Yapay Arı Kolonisi)

ISL : Incremental Social Learning (Artan Sosyal Öğrenme)

MABC : Modified Artificial Bee Colony
(Değiştirilmiş Yapay Arı Kolonisi)

MAXFEs : Maximum Function Evaluations
(Maksimum çağırım sayısı)

MB : Megabyte

MEABC : Multi-strategy Ensemble Artificial Bee Colony
(Çok-strateji Topluluğu Yapay Arı Kolonisi)

MMAS : Max-Min Ant System (Max-Min Karınca Sistemi)

MS : Metaheuristics (Metasezgisel)

MSE : Mean Square Error
(Ortalama Kare Hatası)

MTSLS1 : Multiple Trajectory Search
(Çoklu Yörünge Araması)

MW : MegaWatt

NFL : No Free Lunch (Bedavaya yemek yok)

OABC : Opposition-based Artificial Bee Colony
(Karşıtlık-tabanlı Yapay Arı Kolonisi)

OPIABC : One-Position Inheritance Artificial Bee Colony
(Tek-Nokta Kalıtım Yapay Arı Kolonisi)

PAP : Population-Based Algorithm Portfolios
(Popülasyon-tabanlı Algoritma Portföy)

ParamILS : Parameter Iterated Local Search
(Parametre Yinelemeli Yerel Arama)

PSO : Particle Swarm Optimization
(Parçacık Sürü Optimizasyonu)

RAM : Random Access Memory (Rasgele Erişimli Bellek)

Revac : Relevance Estimation and Value Calibration of Parameters
(Alaka Tahmini ve Parametrelerin Değer Kalibrasyonu)

SMAC : Sequential Model-based Algorithm Configuration
(Sıralı Model-tabanlı Algoritma Yapılandırması)

SOCO : Soft Computing

SPO : Sequential Parameter Optimization
(Sıralı Parametre Optimizasyonu)

SSEABC : Self-adaptive Search Equation-Based Artificial Bee Colony
(Kendinden-uyarlamalı Arama Denklemi-Tabanlı Yapay Arı Kolonisi)

TempPSO : Template Particle Swarm Optimization
(Şablon Parçacık Sürü Optimizasyonu)

UACOR : Unified Ant Colony Optimization
(Tekilleştirilmiş Karınca Kolonisi Optimizasyonu)

1. GİRİŞ

Optimizasyon, belirli şartlar ve kısıtlamalar altındaki kaynakların en etkin şekilde tahsisinin yönetilmesidir. Gerçek hayatta yer alan endüstri, mühendislik ve ekonomi alanlarında karşılaşılan zaman, maliyet, risk gibi unsurların en uygun hale getirilmeye çalışılması bir optimizasyon problemi olarak görülebilir (Talbi, 2009).

Optimizasyon problemlerinin çözümü için önerilen yöntemler klasik ve sezgisel yöntemler olmak üzere ikiye ayrılmaktadır. Klasik yöntemler probleme özgüdür ve kesin bir sonucu garanti ederler. Bu algoritmalar bütün çözüm uzayını tarayarak çözüm üretmektedirler. Bundan dolayı problem büyüdükçe hesaplama maliyeti artmaktadır. Yapısal ve analitik metotlar olmak üzere iki farklı türü vardır. Optimizasyon problemlerinin çözümü için önerilen bir diğer yöntem ise Sezgisel yöntemleridir. Bunlar kesin bir sonucu garanti etmezler. Yapılarında rassallık barındırmakta ve yeterli bir süre içerisinde yaklaşık sonucu üretmektedirler. Sezgisel yöntemler de kendi içerisinde sezgisel ve metasezgisel olarak ikiye ayrılmaktadır. Bu tezin konusu da olan metasezgisel (MS) metotlar, çeşitli sınıflandırma yöntemlere göre gruplandırılmaktadır. Sınıflandırılmalardan bir tanesi çözüm sayısına göre sınıflandırmadır. Buna göre MS'ler tek çözüm ve popülasyon tabanlı algoritmalar olarak ikiye ayrılmaktadır. Bu tez kapsamında popülasyon tabanlı metasezgiseller üzerine çalışmalar yapılmıştır.

Optimizasyon problemlerinin çözümü için geliştirilen çok sayıda metasezgisel algoritmalar bulunmaktadır. Bu algoritmalar; doğadan, insan yapımı işlerden, kimya ve fizik gibi çok çeşitli kaynaklardan esinlenmektedir (Fister vd., 2013). Bir diğer ifade ile bu algoritmalar bir çıkış kaynağı olarak bir metaforu kullanmaktadır ve dolayısıyla bunlara metafor tabanlı metasezgiseller de denilmektedir (Sörensen, 2015). Bu tarz metafor algoritmaların sayısı gün geçtikçe artmaktadır. Fister Jr vd.'nin 2016 tarihli çalışmalarına göre çalışmanın yayınlama tarihinde 200'den fazla doğadan esinlenen algoritmanın var olduğunu ve her ay bunlara bir kaç tane daha yeni doğadan esinlenen algoritmanın katıldığını iddia etmektedirler (Fister Jr vd., 2016). Bu sayı o kadar dikkat çekecek seviyelere gelmiştir ki Sörensen bunu "optimizasyon alanı, doğadaki metaforu kullanan yeni (MS) tsunamisine maruz kalmaktadır (Sörensen, 2015)" şeklinde tanımlamıştır.

Sörensen, 2000'li yılların ortalarından sonra arı, termit ve ateş böceği gibi sosyal böcekleri içeren veya başka bir doğa olayından esinlenen çok sayıda algoritmanın araştırmacılar tarafından ortaya konulduğunu ama bunların sayılarının çok fazla olmasına

rağmen literatürde bıraktığı etkilerin çok az olduğunu savunmuştur. Ayrıca bu metafor algoritmaların sunulduğu makaleler incelendiğinde öne çıkan unsurun önerdikleri yeniliklerden ziyade esinlendikleri metafor olduğunu iddia etmiştir. Sörensen, bunun nedenini araştırmacıların optimizasyon terminolojisi yerine temel aldıkları metaforun terminolojisini kullanmalarına bağlamaktadır (Sörensen, 2015). Bununla birlikte, yeni metafor algoritma olduğu iddiası ile ortaya çıkan algoritmalarda var olan bir başka durum ise bazı araştırmacıların bunların aslında "yeni" olmadığını savunmasıdır. Piotrowski, Napiorkowski ve Rowinski'nin yayınladıkları çalışma; yeni olduğunu iddasıyla ortaya konulan Kara Delik Optimizasyon (Black Hole Optimization, BHO) metafor algoritması ile ilgilidir. BHO'un, yeni bir algoritma olmadığını ve aslında eskiden beri var olan Parçacık Sürü Optimizasyonu algoritmasının daha basitleştirilmiş ve atalet ağırlığı içeren bir varyantı olduğunu savunmaktadır (Piotrowski vd., 2014). Bir başka durum ise, BHO algoritmasının yayınlanmasından sonra başka araştırmacıların yeni olduğunu düşündükleri BHO algoritmasını problemlerinin çözümlerinde kullanmalarıdır. BHO algoritması dışında bir başka örnek vermek gerekirse, Harmoni Arama (Harmony Search, HS) 2001 yılında Geem vd. tarafından önerilmiş, yeni olduğu iddası ile ortaya çıkarılmış bir metasezgisel algoritmadır (Geem vd., 2001). Çıkışından bu yana HS, çok sayıda çalışmada kullanılmıştır. Weyland'ın, 2015 yılında yapmış olduğu çalışması, HS algoritmasının aslında bir başka metasezgisel algoritma olan Evrimsel Strateji (Evolution Strategy, ES) algoritmasının bir başka varyantı olduğunu göstermiştir (Weyland, 2015). Yine Weyland ilgili çalışmasında, yeni olarak çıkarılan algoritmaların arasındaki farklılıkların yok denecek kadar az olduğunu vurgulamıştır. Buna örnek olarak, Ateş Böceği Algoritması (Firefly Algorithm, FA) (X. Yang, 2010), Meyve Sineği Optimizasyonu (Fruit Fly Optimization, FFO) (Pan, 2012) veya Kedi Sürüsü Optimizasyonu (Cat Swarm Optimization, CSO) (Chu vd., 2006) gibi algoritmalarının aslında Parçacık Sürü Optimizasyonu algoritmasına çok benzediğini iddia etmektedir. Bu örnekleri çoğaltmak mümkündür.

Algoritmaların kullandıkları metafor sayesinde yeni bir algoritma gibi davranmaları, literatürde birçok niteliksiz yayının ortaya çıkmasına sebep olmaktadır. Sonuç olarak, var olan yüzlerce metasezgisel algoritma içerisine yeni bir metasezgiselin eklenmesi literatürde bir özgünlük değeri oluşturmamaktadır. Literatürde yüzlerce MS'nin olması, bir başka sorunu daha doğurmaktadır. Araştırmacılar, bir problemi çözmek için MS kullanmak istediklerinde karşılarına taranması gereken yüzlerce MS

algoritması çıkmaktır. Bu algoritmaların büyük çoğunluğu, Sörensen'in de ifade ettiği gibi optimizasyon terminolojisini kullanmayan ve metafor terminolojisini kullanan algoritmalar olmaktadır. Dolayısıyla bu algoritmaların anlaşılması da zor olmaktadır (Sörensen, 2015). Nihayetinde, araştırmacıların yüzlerce seçenek arasında problemlerini çözmek için bir algoritmayı seçmeleri çok kolay olmamaktadır.

Literatürde önerilen her yeni metasezgisel, önerildiği problemler için iyi olduğu iddiası ile ortaya çıkmaktadır. Öte yandan Wolpert ve Macready (Wolpert ve Macready, 1997) tarafından ortaya atılan "Bedavaya Yemek Yok (No Free Lunch, NFL)" teoremi ise; hiçbir algoritmanın bütün problemler için diğer yaklaşımlardan daha iyi olmadığını ileri sürmektedir. Örnek vermek gerekirse, bir A ve B algoritması bulunsun ve verilen bir problem için A da B'den daha iyi olsun. NFL, her zaman B algoritmasının A algoritmasından daha iyi olduğu bir problemin var olduğunu savunmaktadır (Talbi, 2009). NFL teoremi göstermektedir ki, farklı problem türleri için tek bir algoritma bütün problemlerin çözümü için yetersiz kalmaktadır. NFL teoremine ek olarak; Ho ve Pepyne'nde benzer bir görüş savunarak "Evrensel bir optimizasyon algoritması tasarımının mümkün olmadığını" belirtmişlerdir (Ho ve Pepyne, 2002).

Sörensen, yeni bir metafora dayalı MS'ye gerek olmadığını, araştırmacıların daha farklı teknikler kullanmasının gerekliliğini savunmuştur (Sörensen, 2015). Bu sayede daha nitelikli çalışmaların olabileceğini düşünmüştür. NFL teorimi de göz önüne alındığında tek bir algoritmanın bütün problem türlerinin çözümleri için yeterli olmadığı görülmektedir. Öte yandan, literatüre yeni bir metafor algoritması eklemenin dışında optimizasyon alanında başka yaklaşımlar da mevcuttur. Bunlar, literatürde temel olarak kabul edilen bazı metasezgisellerin kullanılması ile oluşturulan algoritma portföy yaklaşımları, toplu algoritma çatıları, birleşik algoritmalar ve kendinden uyarlanabilen (self-adaptive) algoritma yapılarıdır. Bu tip yaklaşımlar, yapısal olarak birçok algoritma çeşidini barındırdıklarından yeni metasezgisel algoritma önerilerinin önlenmesinde önemli rol oynamaktadır. Bununla birlikte, optimizasyon problemleri için birden fazla algoritmanın bir araya getirilmesi sayesinde algoritmaların farklı problem türlerindeki yetersizliklerinin giderilmesi sağlanmış olur.

1.1. Tez Çalışmasının Amacı

Bu tez kapsamında yukarıda bahsedilen problemlerin üstesinden gelinmesine yardımcı olacak, tek amaçlı sayısal optimizasyon problemlerinin çözümünde

kullanılabilecek, etkin ve esnek toplu algoritma çatılarının geliştirilmesi amaçlanmıştır. Bunun için popülasyon tabanlı algoritmalar olan Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) ve Yapay Arı Kolonisi (Artificial Bee Colony, ABC) algoritmaların bileşenlerinin sınıflandırılması ve bunların bir araya getirilmesiyle iki toplu (ensemble) algoritma çatısı geliştirilmiştir. Böylece geliştirilen çatılar çok sayıda algoritma çeşidinin farklı kombinasyonunu içerisinde barındırmaktadır. Kullanıcılara da istedikleri algoritmayı manuel veya otomatik parametre yapılandırma araçları vasıtasıyla oluşturmalarına imkân sağlanmıştır. Manuel ayarlama ile var olan bir algoritma varyantı oluşturulabilmektedir. Otomatik parametre ayarlama yöntemleri ile de verilen bir problem veya problem kümesi için iyi çalışan bir algoritma varyantı çatılardan türetilebilmektedir. Oluşturulan iki algoritma çatıları esnek ve genişletilebilir olarak tasarlanmıştır. Bunlara ek olarak, çatı tasarımları sırasında görülmüştür ki algoritmaların gösterdiği etkinliklerin büyük bir kısmı algoritmalarda yer alan arama denklemlerine bağlıdır. Buradan yola çıkarak kendinden uyarlamalı arama denklemi mekanizması içeren bir ABC varyantı da geliştirilmiştir.

1.2. Tez Çalışmasının Organizasyonu

Tezin ilerleyen kısımları ise şu şekilde organize edilmiştir:

İkinci bölümde, algoritma üretme alanında önerilmiş ve bu tez kapsamındaki yaklaşımlara benzer olan çalışmalara kısaca değinilmiştir.

Üçüncü bölümde, optimizasyon metotları ve bunların sınıflandırılmaları anlatılmıştır. Daha sonra popülasyon tabanlı algoritmalar hakkında bilgiler verilmiştir. Ardından ise bu tezde yer alan algoritmaların da dahil olduğu kategori olan sürü zekasının tanımına ve biyolojik alt yapısına değinilerek bu alanda yer alan birkaç algoritmalardan bahsedilmiştir.

Dördüncü bölüm, algoritmaların parametrelerinin önemini ve bunlara ilk değerlerinin verilme şekillerini kapsamaktadır. Parametrelerin ilk değerlerinin belirlenmesi için literatürde önerilmiş olan çeşitli yaklaşımlara yer verilmiştir. Bununla birlikte, tezde kullanılmış olan Yinelemeli F-Race yönteminden de kısaca bahsedilmiştir.

Beşinci bölümde, tekilleştirilmiş bir Parçacık Sürü Optimizasyon algoritması olan Şablon PSO algoritmasının yapısına yer verilmiştir. Bu algoritmanın, CEC'17 ölçüt setine uygulanışına ve elde edilen sonuçlarına yer verilmiştir. Daha sonra algoritmanın

ölçeklenebilirlik yeteneğini sınamak için Soft Computing (SOCO) dergisinin özel sayınına ait olan ölçüt seti kullanılmış ve bunların deney sonuçları paylaşılmıştır.

Altıncı bölümde, tekilleştirilmiş Yapay Arı Algoritmasının (Artificial Bee Colony, ABC) detayları anlatılmıştır. Daha sonra, çeşitli ölçüt setlerinden alınan fonksiyonlar ile oluşturulan karma ölçüt seti kullanılarak deneyler yapılmış ve sonuçlarına yer verilmiştir. Algoritmanın performansı, literatürde yer alana ABC varyantları ile karşılaştırılmıştır. Bununla birlikte, bir gerçek dünya problemi olan valf nokta etkili konveks olmayan ekonomik güç dağıtım probleminin ABC-X algoritması kullanılarak çözülmesi sağlanmış ve literatürdeki farklı algoritmaların sonuçları ile karşılaştırılması yapılmıştır.

Yedinci bölümde, metasezgisel algoritmalarda yer alan arama denklemlerinin algoritmaların performansına katkısının çok fazla olduğu düşüncesi ile kendinden uyarlanabilir arama denklem yapısına sahip SSEABC algoritması önerilmiştir. Bu bölümde SSEABC algoritmasının detaylarına yer verilmiştir. Algoritmanın performansı CEC'17 ölçüt seti ile test edilmiştir. Daha sonra algoritmanın ölçeklenebilirlik performansı Soft Computing özel sayısına ait olan ölçüt seti kullanılarak araştırılmıştır. Ardından SSEABC, bir gerçek dünya problemi olan IIR filtre tanımlama problemine uygulanmıştır.

Son bölümde ise tezin genel bir değerlendirmesi ve elde edilen sonuçların tartışılmasına yer verilmiştir.

2. YAPILAN ÇALIŞMALAR

Popülasyon-tabanlı metasezgisel algoritmaların problem çözmede gösterdikleri başarılar araştırmacıların ilgi odağı haline gelmelerini sağlamıştır. Bunlar, birçok endüstri ve mühendislik problemine uygulanmıştır. Uygulanmış olduğu bazı alanlar; araç rotalama problemi, görüntü işleme, gezgin satıcı problemi, network topoloji tasarımı, filtre tasarımı, protein yapısı optimizasyonu, makine öğrenimi gibidir (Gotmare vd., 2017; Karaboga vd., 2014; Saad vd., 2018). Bunlara ek olarak, problem çözümünde göstermiş oldukları başarıları artırmak için metasezgisel algoritmaları üretme alanında da çeşitli çalışmalar yapılmaktadır. Bu alandaki özgün ve değerli çalışmalar, temel olarak iki grup altında toplanmaktadır.

Bunlardan birincisi, literatürde uzun süredir yer alan ve bu alanda temel olarak kabul edilen metasezgisel algoritmaların (DE, PSO, ACO gibi) geliştirilerek iyileştirilmesidir. Örnek olarak, Orijinal PSO algoritması 1995 yılında Kennedy ve Eberhart tarafından geliştirilmiştir. O günden bu yana PSO algoritmasının çok fazla sayıda geliştirilmiş varyantı önerilmiştir. Algoritmaya yeni eklentiler, yeni adımlar veya mevcut yapıların değiştirilmesi gibi iyileştirmeler ile PSO'nun performansının arttırılması amaçlanmıştır. Aynı zamanda, farklı bir metasezgiselde yer alan ve başarılı olduğu düşünülen tekniklerin PSO'ya adapte edilmesi ile de algoritmanın iyileştirilmesine çalışılır.

Algoritma üretme alanındaki ikinci önemli çalışmalar ise farklı metasezgisel algoritmaları veya metasezgisel algoritmaların varyantlarını bir araya getirerek oluşturulan yapılardır. Bunlar; algoritma portföy yapıları, birleşik (composite) ve toplu (ensemble) algoritmalardır. Bu tür yaklaşımlar farklı algoritmaların güçlerinden faydalanmayı amaçlamaktadır. Algoritmaların bir araya getirilmesi ile problemlerin çözülmesi sağlanır. Tez kapsamında, ikinci kategoriye giren algoritmalar önerildiği için bu alandaki çalışmalara kısaca değinilecektir.

2.1. Algoritma Portföy Yaklaşımı

Mühendislik ve gerçek dünya problemlerinin çözümleri için birçok popülasyon tabanlı algoritma önerilmiştir. Bir algoritma, problem türlerinin tamamında iyi olamamaktadır. Bir problemde iyi olabilirken diğerinde ise kötü olabilmektedir. Dolayısıyla performansları değişkenlik göstermektedir. Bununla birlikte, farklı algoritmalar da farklı problemlerde iyi çözümler üretebilmektedir. Hangi algoritmanın o

problem için daha iyi olacağı bilinemez. Bundan dolayı bir problemin çözümü için tek bir algoritmanın seçilmesi riskli olmaktadır. Ortaya çıkan bu sorunu gidermek için araştırmacılar tarafından algoritma portföy yaklaşımı önerilmiştir.

Algoritma portföy, farklı problem türlerinde iyi olan farklı algoritmaların bir araya toplanmasıyla ortaya çıkarılan bir çatı yaklaşımıdır. Toplanan algoritmalar genellikle zeki veya istatistiksel yöntemler kullanılarak seçilirler. Böylece bir probleme özgü belirlenen iyi birkaç algoritma, problem için etkin sonuçlar elde etmede kullanılır.

Algoritma portföy yaklaşımı ile ilgili ilk çalışmalar, kombinatoryel problemlerin çözümünde olmuştur (Gomes ve Selman, 1997; Huberman vd., 1997). Huberman vd. kombinatoryel problemlerin çözümünde birden fazla algoritmayı bir araya getirmek için ekonomik bir yaklaşım olan portföylerden yararlanmışlardır (Huberman vd., 1997). Gomes ve Selman; Las Vegas algoritma portföyünü kullanılarak bir kombinatoryel problemin çözümünde önemli performans iyileşmesi elde etmişlerdir (Gomes ve Selman, 1997). Peng vd., portföy algoritma yaklaşımını sayısal optimizasyon alanında uygulamışlardır (Peng vd., 2010). Dört popülasyon tabanlı algoritmayı bir araya getiren bir portföy çatı önermişler ve buna Popülasyon-Tabanlı Algoritma Portföy (Population-Based Algorithm Portfolios, PAP) ismini vermişlerdir. PAP yaklaşımı; bir problemin çözümünde, çözüm için ayrılan zamanın kısıtlı olduğunu ve tek bir algoritmaya bu zaman tahsisinin riskli bir durum olduğunu kabul eder. Bundan dolayı zamanın birden fazla algoritmaya paylaştırılarak riskin dağıtılmasının daha doğru olduğunu iddia etmektedir (Peng vd., 2010). Ayrıca Akay vd.; PSO, DE ve ABC algoritmalarını bir araya toplayarak portföy yaklaşımı oluşturmuşlardır. Mesaj Gönderim Arayüzünü, kullanarak da ilk paralelleştirilmiş portföy yapısını meydana getirmişlerdir. Böylelikle, daha hızlı çalışan bir çatı elde edilmiştir (Akay vd., 2017).

2.2. Birleşik Algoritmalar

Birleşik algoritmalar (composite), bir metasezgisel algoritmanın çeşitli varyantlarında yer alan bileşenlerine dayanmaktadır. Birleşik algoritmalar yaklaşımında ilk olarak, birden çok algoritmalar bir havuzda toplanır. Daha sonra bu algoritmaların belirli bir problem kümesi üzerindeki en iyi bileşenleri sistematik olarak yapılan analizler vasıtasıyla tespit edilmeye çalışılır. Örnek olarak, havuzda yer alan bütün algoritmaların ilklendirme adımındaki yöntemleri karşılaştırılır ve en iyisi belirlenir. Bu işlem algoritmanın diğer adımları için de gerçekleştirilir. Bu en iyi bileşenlerin bir araya

getirmesi sonucunda bir birleşik algoritma oluşturulur. Birleştirilmiş algoritmalara verilebilecek en önemli örnekler Frankenstein'ın Parçacık Sürü Optimizasyonu (Frankenstein Particle Swarm Optimization, FrankPSO) (Montes de Oca vd., 2009), ParadisEO-MOEO çatısı (Cahon, Melab ve Talbi 2004) ve Kompozit Yapay Arı Kolonisi (Composite Artificial Bee Colony, CompABC) (Aydın, 2015) algoritmalarıdır.

Bu tür yaklaşımların dezavantajı, doğru bileşen seçimlerinin karar verilmesinde her problem örneği için bileşen analizlerinin tekrar yapılması gerekliliğidir. Dolayısıyla en iyi bileşenlerinin tespit edilmesi için uzun zaman gerektirmektedir. Diğer bir dezavantaj ise, bileşik algoritmaların tasarımında bileşenlerin birbirleri ile ilgili etkileşimin hesaba katılmamasıdır. Çünkü bileşenlerin seçim aşaması birbirlerinden bağımsız olarak en iyi bileşen seçilerek yapılır. Fakat diğer aşamalarda seçilen yöntemle uyumlu çalışıp çalışmadığı dikkate alınmaz.

FrankPSO geliştirilirken, çeşitli parçacık sürü optimizasyonu varyantlarında yer alan bileşenler, bir problem seti ile çalıştırılarak analiz edilmiştir. Yapılan analizler yardımıyla en iyi bileşenler tespit edilmiştir. Daha sonra belirlenen bu bileşenler tek bir algoritma altında bir araya getirilerek birleştirilmiş bir algoritma yapısı oluşturulmuştur. Analizlerin sonucunda ortaya çıkan FrankPSO algoritması, bileşenleri analiz edilen ve FrankPSO'ya temel oluşturan PSO varyantlarından daha iyi sonuç verdiği gözlemlenmiştir. Bir başka bileşik algoritmaya örnek ParadisEO-MOEO'dur. Bu da çok amaçlı problemlerin çözümünde kullanılabilecek bir algoritma çatısıdır. Bu çatıda önemli olan nokta çatı içerisinde birçok algoritma bileşenini barındırması ve kullanıcıların bu algoritma bileşenlerini bir araya getirerek yeni algoritmalar üretilmesine olanak tanımasıdır. Böylelikle, problem üzerinde bileşen analizi yapılarak birleştirilmiş algoritmaların üretilmesi sağlanmış olur. Diğer bir bileşik algoritması da CompABC algoritmasıdır. CompABC oluşturulmasında iki farklı problem seti kullanılmıştır. İlk olarak, ABC varyantları bu problem setleri ile çalıştırılıp bileşenleri analiz edilmiştir. Bunun sonucunda en iyi bileşenler belirlenerek iki farklı ABC algoritması varyantı elde edilmiştir. Bu CompABC algoritmaları, on farklı ABC varyantı ile karşılaştırılmış ve ABC varyantlarına karşı üstün performans gösterdiği görülmüştür.

2.3. Toplu veya Tekilleştirilmiş Algoritmalar

Algoritma üretme alanında yer alan diğer yaklaşım ise bir metasezgisel için her bir algoritma adımında önerilmiş olan bileşenleri bir araya toplayarak bir algoritma çatısı

tasarlamaktır. Bu tip algoritmalara tekilleştirilmiş (unified) veya toplu (ensemble) algoritmalar denilmektedir.

Tekilleştirilmiş algoritmalar, tek bir algoritma olmakla beraber çok sayıda algoritmayı içerisinde barındırabildiğinden birleşik algoritmalara oranla daha esnektir. Diğer bir deyişle, tekilleştirilmiş algoritmalar her bir problem karşısında uygun birleşik algoritmayı oluşturabilen bir yapıdadır. Dolayısıyla bu yaklaşım diğerine oranla daha popülerdir.

Vrugt, Robinson, ve Hyman; CMAES, PSO ve genetik algoritmayı bir araya toplayarak uyarlanabilir metot aracılığı ile kullanan bir yaklaşım önermişlerdir (Vrugt vd., 2009). Dubois-Lacoste, Lopez-Ibanez, ve Stützle; iki arama metodunu bir araya getirerek iki amaçlı akış tipi çizelgeleme problemini çözmüşlerdir (Dubois-Lacoste vd., 2011). Bunun yanında Lopez-Ibanez ve Stützle; çok amaçlı kombinatoryel optimizasyon problemleri için tekilleştirilmiş karınca kolonisi optimizasyon çatısı geliştirmişlerdir (Lopez-Ibanez ve Stützle, 2012). Liao vd.; sürekli optimizasyon için ACOR, DACOR ve IACOR-LS algoritmalarının bileşenlerini içeren tekilleştirilmiş bir karınca kolonisi optimizasyon algoritması (unified ant colony optimization, UACOR) önermişlerdir (Liao vd., 2014). Önerilen algoritmanın performansı iki farklı ölçüt seti ile test edilmiştir. Parametre ayarlama araçları kullanan çalışmada, tekilleştirilmiş algoritma her bir ölçüt seti için farklı algoritmalar gibi davranmış ve çok iyi sonuçlar elde etmiştir. Lynn ve Suganthan, farklı problem çözme davranışlarını gösteren ve birbirlerinin eksiklerini tamamlayacak şekilde seçilen PSO varyantlarını bir havuzda toplamışlardır. Toplanan PSO varyantlarını sabit iterasyon sayısı kadar çalıştırıp, performanslarının ölçülmektedirler. Daha sonra belirli bir mantığa göre havuzdan seçilen algoritmalar problem çözme amaçlı kullanılmaktadır (Lynn ve Suganthan, 2017). Wu vd. (Wu vd., 2018); başarıları kanıtlanmış, güçlü DE varyantları olan JADE, EPSDE ve CoDE'yi bir araya getirerek EDEV isminde çoklu-popülasyonlu bir çatı geliştirmişlerdir. Ayrıca CEC'05 ve CEC'14 ölçüt setler ile önerdikleri algoritmanın performansını test etmişlerdir. EDEV, birçok DE varyantına karşı üstün performans göstermiştir. Bunlara ek olarak Franzin ve Stützle, en eski metasezgisellerden biri olan Benzetimli Tavlama (Simulated Annealing, SA) algoritması için ve algoritma tasarımcılarının gereksiz detaylar ile uğraşmasını önleyerek uygulama geliştirmesini sağlayan otomatik yapılandırılabilir çatı geliştirmişlerdir (Franzin ve Stützle, 2019). Geliştirilen çatının performansı üç tane kombinatoryel optimizasyon problemiyle test edilmiştir. Bir diğer

tekilleştirilmiş algoritmaya örnek olarak, ABC varyantlarını temel alan ve geliştirilmesine yazarında dahil olduğu ABC-X algoritması da verilebilir. ABC-X algoritmasına ait detaylar Bölüm 6'da yer verilecektir (Aydın, Yavuz, ve Stützle, 2017).

3. OPTİMİZASYON METOTLARI

Optimizasyon, belirli şartlar ve kısıtlamalar altında en iyi sonucu alma sürecidir (Boussaïd vd., 2013; Talbi, 2009; X.-S. Yang, 2010). Harcanan çaba ve istenen kazanç bazı karar değişkenlerinin bir fonksiyonu olarak ifade edilebileceği için, optimizasyon, bir fonksiyonun belirli şartlar altında en az ya da en çok olduğu durumu bulmak şeklinde tanımlanabilir.

Bir optimizasyon problemi belirli bir çözüm uzayında, belirli bir amaç fonksiyonu değerinin en küçük (veya en büyük) yapan elemanını bulma işlemidir. Gerçek dünyada farklı alanlarda birçok problem türü mevcuttur. Bunların çoğunluğunun çözümü karmaşık ve zordur. Bu problemleri çözmek için literatürde farklı optimizasyon metotları önerilmiştir. Bunlar, kesin sonuç veren Klasik metotlar (deterministik) ve yaklaşık sonuç veren Sezgisel metotlar (stokastik) olmak üzere ikiye ayrılmaktadır.

Klasik metotlar, probleme özel metotlardır. Bu yüzden problemde meydana gelebilecek değişimlere duyarlıdırlar. Klasik metotlar sonlu bir sürede kesin sonuç vermeyi garanti ederler. Bunu da çözüm uzayındaki yer alan bütün muhtemel sonuçları tarayarak gerçekleştirirler (Sörensen, 2015). Bu yüzden problem türü ve boyutuna göre hesaplama maliyeti yüksek olabilmektedir. Ayrıca, birden fazla en iyiye sahip bir problemde yerel en iyiye takılabilmektedirler.

Optimizasyon metotlarının bir diğer kategorisi olan sezgisel metotlar da kesin sonuçtan ziyade doğru sonuca en yakın bir sonuç sağlamaktadır. Böylelikle yüksek hesaplama maliyeti olabilecek durumlarda daha düşük maliyet ile kabul edilebilir sonuçlar üretebilirler. Bunlar yapıları gereği rassallık içermektedirler. Problemin türü ve boyutundan fazlaca etkilenmeden uygun sürelerde problemleri çözebilmektedirler.

Sezgisel algoritmalar, Metasezgisel ve Probleme-özel sezgisel olmak üzere iki sınıfa ayrılır (Talbi, 2009). Problem-özel sezgisel algoritmalar, belirli bir problem için geliştirilen probleme özel yöntemlerdir. Bu algoritmalar problemden elde ettiği bilgilerden faydalanırlar. Bu yüzden bu algoritmalardan sadece bu problem için iyi sonuçlar vermesi beklenmektedir. Buna karşın Metasezgiseller ise; çeşitli optimizasyon problemlerine uygulanabilen, problemden bağımsız daha genel yöntemlerdir ve birçok optimizasyon problemi üzerinde başarı ile uygulanmıştır. Bir sonraki alt bölümde metasezgiseller irdelenecektir.

3.1. Metasezgiseller

Metasezgisel (MS) kavramı ilk olarak 1986 yılında araştırmacı Fred Glover tarafından kullanılmıştır. Yunanca "üst seviye metot" manasına gelen "meta" ile "problem çözmek için yeni yöntemler bulma sanatı" anlamına gelen Yunanca "heuriskein" kelimesinden türeyen "heuristics" kelimelerinin birleşimden meydana gelmektedir (Talbi, 2009). Metasezgiseller, mühendislikte yer alan çözülmesi zor ve karmaşık problemlere yaklaşık çözümler üretmektedirler. Yapılarının basit ve öğrenilmesinin kolay olması araştırmacıların ilgi odağı haline gelmesini sağlamıştır. Literatürde yer alan MS teknikler, bazı ortak özelliklere sahiptir (Boussaïd vd., 2013; Parejo vd., 2012; Talbi, 2009):

- Sezgisellere yol göstermektedirler.
- Rassal(stochastics) bileşenlerden faydalanırlar.
- Probleme özgü değildirler.
- Yerel iyilerden (Local optima) kurtulma yeteneklerine sahiptirler.
- Belirli zamanda optimal çözümü bulmayı garanti etmezler.

MS'ler çeşitli sınıflandırmalara göre kategorilendirilmişlerdir. Bu sınıflandırmalardan bir tanesi aramada kullanılan çözüm sayısına göre yapılan sınıflandırmadır. Buna göre metasezgiseller, tek çözüm tabanlı ve popülasyon tabanlı olmak üzere iki gruba ayrılmaktadırlar.

3.1.1. Tek çözüm tabanlı metasezgiseller

Tek çözüm tabanlı metasezgiseller, kendilerine verilen tek bir çözümü iyileştirerek çalışırlar. Bu iyileştirme komşuluklar üzerinde yürümek ya da çözüm uzayında yer alan arama yollarında ilerlemek olarak tanımlanabilir. Yürümeler yinelemeli bir prosedürle çözüm uzayında o anki çözümden yeni bir çözüme geçiş şeklinde gerçekleştirilir. Tek çözüm tabanlı metasezgisellere örnek olarak Tabu Arama (Tabu Search) (Glover ve Laguna, 1998), Yinelemeli Yerel Arama (Iterated Local Search) (Lourenço vd., 2003) ve Benzetimli Tavlama (Simulated Annealing) (Van Laarhoven ve Aarts, 1987) verilebilir.

3.1.2. Popülasyon tabanlı metasezgiseller

Popülasyon tabanlı metasezgiseller bir grup çözümü yinelenebilir şekilde daha iyi hale getiren yöntemlerdir. Sürü Zekâsı ve Evrimsel Algoritmalar olmak üzere ikiye ayrılmaktadır. Bu tezde, Sürü Zekâsı algoritmaları üzerine çalışmalar yapıldığından dolayı Sürü Zekâsının ve algoritmalarının detaylarına değinilecektir.

3.2. Sürü zekâsı

3.2.1. Sürü zekasının biyolojik altyapısı

Görme yeteneği bulunmayan termit kolonilerinin birlikte hareket ederek inşa ettikleri yuvalar, kuş sürülerinin "V" şeklinde uçarak avcılardan ve havanın sürtünmesinden korunmaları veya karıncaların bir yem kaynağı bulduktan sonra kolonide yer alan diğer karıncalara bilgilerini aktararak en kısa yoldan yiyecekten faydalanması gibi doğadaki canlıların zeki davranışlarının tamamı araştırmacıların ilgi odağı olmuştur. Bu örnekleri çoğaltmak mümkündür. Karınca, balık, kuş, arı ve termit vb. canlıların gösterdikleri kolektif ve merkezi olmayan davranışlar incelendiğinde, bazı ortak özelliklerin olduğu görülmektedir. Bu özellikleri gösteren canlılar sürü veya koloni adı verilen gruplar halinde hareket etmektedirler. Bu canlıların öne çıkan ortak özellikleri şunlardır (Yu ve Gen, 2010) (Câmara, 2015):

- Bir grubun iş birliğinden ortaya çıkan bir davranış sunmak
- Büyük ölçüde homojen bireysel ajanlar,
- Her birinin basit işler gerçekleştirmesi
- Eşzamanlı olarak hareket edilmesi
- Hiç veya çok az merkezi koordinasyon
- Stigmergy

Kendi başına karmaşık davranışlar göstermeyen ancak birlikte yardımlaşarak karşılarına çıkan probleme ortak çözüm bulan canlılara “sosyal canlılar” veya “sosyal böcekler” (social insect) denilmektedir (X.-S. Yang vd., 2017). Sosyal böceklere örnek olarak karınca, termit ve arı gibi canlılar verilebilir. Bu böceklerin oluşturdukları gruplara sürü, koloni gibi farklı isimler verilmektedir.

Sürülerin veya kolonilerin merkezi kontrol sistemleri yoktur yani bir yönetici tarafından yönetilmezler. Bununla birlikte her bir eleman basit kurallar dâhilinde hareket

etmektedir. Kolonide yer alan bireyler belirli sayıda kuralları sürekli olarak takip etmektedirler. Tek başlarına karmaşık olarak bir eylem ortaya koyamazlar. Ancak benzer bireyler ile iş birliği yaparak karmaşık davranışlar gösterebilirler. Bunu gerçekleştirmek için sürünün her bir bireyi çevreleriyle ve birbirleriyle etkileşime geçerler. Bu canlılara, etkileşime geçtikleri çevreler ya da inşa etmeye çalıştıkları yapılar yol göstermektedir. Yani bu canlılar global bir bilgiye sahip olmadan yaptıkları işlerin kendilerine rehberlik etmesi sonucunda bir yapı ortaya koyabilirler.

Sosyal canlıların gösterdikleri zekâ iki kavrama dayanmaktadır. Bunlardan ilki bir iletişim türü olan stigmery ve diğeri ise öz-örgütlenmedir.

3.2.1.1. *Öz örgütlenme*

Sosyal canlıların gösterdiği kolektif davranış kaynaklarından en önemli unsuru öz örgütlenmedir. Öz örgütlenme (self-organization), bir sistemin alt seviyede yer alan bileşenlerinin, birbirleriyle etkileşimi sonucunda ortaya çıkan yapının dinamik mekanizmalar kümesini ifade eder (Garnier vd., 2007). Bu alt seviyedeki bileşenler, sistemin global düzeyindeki bilgilerden mahrumdurlar. Sadece yerel olarak elde ettikleri bilgiler onları yönlendirir. Yerel olarak elde edilen bilgiler ile birlikte basit davranışlar vasıtasıyla kolektif bir hareket üretebilirler. Öz örgütlenmeye doğadaki karıncaların yiyecek bulma ile arıların kovanlarını inşa ederken ki davranışları örnek olarak verilebilir. Bununla birlikte, öz örgütlenmenin dört temel unsuru vardır. Bunlar:

Pozitif geri besleme (Positive feedback): Karıncaların yiyecek aramada gösterdikleri davranışlar pozitif geri beslemeye örnektir. Yiyecek arayan bir karınca geçtiği yollara kimyasal bir madde olan feromonu (pheromone) salgılar. Bu yiyecek arayan karınca bir yiyecek kaynağı bulduğu an hemen yuvasına geri döner. Dönerken de feromon izlerini bırakmayı da sürdürür. Yuvaya geldiğinde yuvada bekleyen diğer karıncalar, feromon izlerini algılayıp bunları takip ederek yiyecek kaynağına ulaşırlar. Bu karıncalar da yiyecek kaynağına giderlerken geçtikleri yollara feromon bırakırlar ve böylelikle diğer karıncanın bırakmış olduğu izleri güçlendirirler. Bu izleri takip etme ve güçlendirme eylemi pozitif geri besleme olarak tanımlanabilir. Ayrıca karıncaların yerel seviyede bıraktıkları izler global düzeyde feromon ağının oluşmasına sebep olur.

Negatif geri besleme (Negative feedback): Negatif geri besleme sayesinde pozitif geri besleme dengelenmektedir. Yine karıncaların yiyecek arama davranışlarından örnek verilirse, yiyecek kaynağının tükenmesi sonucunda kaynağa giden karınca sayısı azalır.

Böylelikle yola bırakılan feromon izlerinin güçlendirilmesi de azalır. Sonucunda belirli bir süre güçlendirilmeyen feromon izleri buharlaşarak kaybolur. Bu buharlaşma süreci negatif geri beslemeye örnek verilebilir.

Dalgalanmanın yükseltilmesi (Amplification of fluctuations): Sosyal böceklerin davranışları rastgelelik barındırır. Bu da yeni, daha önce keşfedilmemiş alanların ve yiyecek kaynaklarının bulunmasını sağlar. Mesela, yiyecek ararken yolunu kaybeden bir karıncanın şans eseri bir yiyecek kaynağı bulması ve yuvada bekleyen diğer karıncalara haber vermesi buna güzel bir örnektir.

Çok etkileşimler (Multiple interactions): Sosyal böcekler çevreleriyle ve sürüde yer alan bireyler arasında etkileşim halindedirler. Bu etkileşim, sürü içerisinde bilginin dağılmasına da imkân sağlar. Sonucunda global düzeyde bir eşgüdümlü hareket ortaya çıkmaktadır. Öz örgütlenmede iki farklı etkileşim yöntemi vardır. Bunlardan ilki doğrudan etkileşim bir diğeri ise stigmergy yolu ile etkileşimdir. Doğrudan etkileşimde sosyal böcekler herhangi bir aracı olmadan birbirleri ile etkileşime geçerler. Buna örnek olarak, bir yiyecek kaynağı bulan arının kovana gelip sallanma dansı (waggle dansı) yaparak diğer arılara bilgisini aktarması verilebilir. Diğer etkileşim yöntemi stigmergy kavramı bir sonraki alt bölümde anlatılacaktır.

3.2.1.2. *Stigmergy*

Fransız araştırmacı Pierre-Paul Grassé, termit kolonilerinin yuvalarını inşa ederken göstermiş oldukları davranışları açıklamak için "stigmergy" kavramını ileriye sürmüştür. Yunanca "stigma" yani "işaret" ve "ergon" yani "iş/hareket" kelimelerinin birleşiminden meydana gelmektedir (Câmara, 2015). Anlamı, ajanlar arasında kendiliğinden olan ve dolaylı koordinasyon olarak tanımlanır. Grassé göre kolooninin kolektif olarak gösterdiği zeki davranışların kaynağının sebebi, kolooninin içinde yer alan bireylerin kendisinden ziyade inşa edilen yapıdan kaynaklandığıdır. Yani termitlere rehberlik eden inşa edilen yuvanın kendisidir.

Stimergy terimini Grasse, çalışan termit işçilerinin göstermiş olduğu performans ile orantılı olarak uyarıldığı bir iletişim şekli olarak tanımlamaktadır. Stimergy kavramını diğer iletişim yollarından ayıran iki öne çıkan özelliği vardır. Bunlar;

- Dolaylı bir iletişim şeklidir. Sosyal canlılar çevrelerinde değişimler yaparak iletişim kurmaktadırlar.

- Bilgi yereldir. Sadece erişebilen canlılar tarafından ziyaret edilir.

Termitlerin davranışlarını açıklamak için kullanılan stigmergy kavramı daha sonra başka canlılar için de kullanılmıştır. Örnek olarak, karıncalar yiyecek kaynağı ile yuvaları arasında gidip gelirken yola bıraktıkları feromon ile haberleşmeleri verilebilir. Bu maddenin fazlaca olduğu yolları algılayıp takip eden diğer karıncalar yiyeceklere ulaşabilmektedirler. Bu sayede de karıncalar yiyecek kaynağına giden en kısa yolu bulabilmektedirler.

3.2.2. Sürü zekâsı algoritmaları

Sürü zekâsı, doğada yer alan canlıların koloni olarak hareketleri sonucunda ortaya koydukları karmaşık davranışların modellenmesiyle ortaya çıkan tekniklerdir. Canlıların gösterdiği merkezi olmayan, öz-örgütlü ve kolektif yapı modellenerek bilim adamları tarafından problemlerin çözümlerinde kullanılmaya çalışılmıştır. Böylelikle, Sürü Zekâsı (SZ) doğmuştur. "Sürü Zekâsı" kavramı ilk olarak 1989 yılında Gerardo Beni ve Jing Wang tarafından hücresel robotik sistemler ile ilgili bir makalesinde kullanılmıştır (Blum ve Li, 2008).

SZ, bir algoritma değildir ancak doğadan esinlenen algoritmaların ait oldukları bir kategoridir. Bu kategori içerisinde yer alan algoritmalar doğadaki kuş, balık, karınca gibi kolonileri taklit etmektedirler. Bu algoritmalar birbirleri ile etkileşim halinde olan basit bireylerin oluşturdukları popülasyonları problem çözmede kullanırlar. Popülasyonda yer alan her bir birey problemlerin çözümlerini temsil eder. Sürü zekâsı denilince iki algoritma öne çıkmaktadır, bunlar Karınca Kolonisi Optimizasyon Algoritması (Ant Colony Optimization, ACO) (Dorigo ve Birattari, 2010) ve Parçacık Sürü Optimizasyonudur (Particle Swarm Optimizastion, PSO) {Formatting Citation}. Bu iki algoritmaya ek olarak, Yapay Arı Koloni Algoritması (Artificial Bee Colony, ABC) (Karaboga ve Basturk, 2007), Bakteri Yiyecek Arama Algoritması (Bacterial Foraging Optimization, BFO) (Boussaïd vd., 2013), Ateş Böceği Algoritması (Firefly Algorithm, FA) (X. S. Yang, 2010) da örnek olarak verilebilir. Bundan sonraki alt bölümlerde ACO, PSO ve ABC algoritmalarının detaylarına yer verilecektir.

3.2.2.1. *Karınca kolonisi optimizasyonu algoritması*

Doğada yer alan karıncaların, yiyecek aramada birbirleri ile iletişimleri ve yiyecek bulmadaki davranışları dikkat çekicidir. Karıncalar yiyecek aramaya başladıklarında ilk

olarak çevrelerini rastgele araştırırlar. Geçtikleri yolları karıncaların bedenlerinden üretilen feromon ismindeki bir salgıyla işaretlerler. Araştırma sırasında bir yiyecek kaynağına rastlanırsa, keşfeden karınca bulduğu kaynağın kalitesini belirler ve yuvaya geri döner. Dönerken de bulmuş olduğu yiyecek kaynağı ile orantılı olarak geçtiği yerlere yine feromon bırakır. Yuvadaki diğer karıncalarda feromonları algılar ve yiyecek kaynağından haberdar olurlar. Yuvada bulunan diğer karıncalar da, feromonları takip ederek yiyecek kaynağına ulaşırlar. Yiyecek kaynağına giderlerken bu karıncalar da geçtikleri yollara feromon bırakırlar. Bu şekilde karıncalar feromon vasıtasıyla birbirleri ile haberleşmiş ve yiyecek kaynağına en kısa yoldan ulaşmış olurlar.

Karınca Kolonisi Optimizasyonu, Marco Dorigo tarafından 1992 yılında karıncaların yukarıda bahsedilen yiyecek aramada gösterdikleri zeki davranışlardan esinlenerek geliştirilmiş bir Sürü Zekâsı algoritmasıdır. ACO, ilk olarak gezgin satıcı problemine uygulanmış ve olumlu sonuçlar elde edilmiştir (Dorigo vd., 1991). Daha sonra sürekli optimizasyon probleminin çözümünde de kullanılmıştır. Günümüzde ACO, birçok farklı alanlardaki çalışmalarda kullanılmıştır (Ertenlice ve Kalayci, 2018; Kashef ve Nezamabadi-pour, 2015). Ayrıca dünyanın farklı yerlerindeki çok sayıdaki araştırmacı tarafından birçok yeni geliştirilmiş sürümü ortaya çıkmıştır.

ACO için literatür özeti

ACO'nun ilk çıkışından bu yana çeşitli ACO varyantları da ortaya atılmıştır. İlk ACO algoritması olan ve en popüler Karınca Sistemi (Ant System, AS) olarak bilenen yaklaşımdır (Dorigo vd., 1991). Ayrıca, Gambardella ve Dorigo tarafından Karınca Kolonisi Sistemi (Ant Colony System) algoritması önerilmiştir (Dorigo ve Gambardella, 1997). Bullnheimer vd. Sıra-Tabanlı Karınca Sistemini (Rank-Based Ant System, RBAS) geliştirmişlerdir (Bullnheimer vd., 1997). Diğer bir ACO yaklaşımı ise Stützle ve Hoos tarafından önerilmiş olan ve feromon miktarının sınırlarını belirleyecek yaklaşımlar tanımlamış Max-Min Karınca Sistemi (Max-Min Ant System, MMAS)'dir (Stützle ve Hoos, 2000). Hiroyasu vd. tarafından sürekli optimizasyon için gezici ACO (Touring Ant Colony System, TACO) önerilmiştir (Hiroyasu vd., 2000). Öte yandan Socha ve Dorigo da karınca kolonisi algoritmasını sürekli optimizasyon problemlerine uygulamışlardır (Socha ve Dorigo, 2008).

Algoritma tanımı

Karınca koloni optimizasyonu, doğada yer alan karıncalardan esinlenilmiş bir algoritmadır. Marco Dorigo tarafından 1992 yılında ortaya atılmıştır (Dorigo vd., 1991).

Algoritma 3.1 *Karınca koloni optimizasyonu algoritmasının sözde kodu*

```
1: Parametreleri ayarla, feromon izlerini ilklendir.
2: while Sonlandırma kriteri sağlanıncaya kadar do
3:     Karınca çözümlerini oluştur.
4:     Yerel aramayı uygula (Opsiyonel)
5:     Feromon değerlerini güncelle.
```

ACO algoritması yapay karıncalardan oluşan yinelemeli bir algoritmadır ve sözde kodu Algoritma 3.1'de yer almaktadır. İlklendirme adımından sonra Karınca Çözümleri Üretimi ve Feromon Güncelleme ismindeki birbirini takip eden adımlardan meydana gelmektedir. Birde bu adımlara ek olarak tercihe bağlı olan Yerel Arama adımı yer almaktadır. Bu işlemler yinelemeli olarak bitme kriteri sağlanıncaya kadar devam eder.

ACO, kombinatoryel optimizasyon problemi göz önüne alınarak incelenirse algoritma çalışması şu şekilde gerçekleşmektedir:

İlklendirme adımında, algoritma parametreleri ve feromon değerleri ilkendirilir.

Bir diğer adım olan karınca çözümlerini oluşturma adımında, algoritmadaki yapay karıncalar bütün düğümlerden sadece bir kez dolaşacak şekilde çözüm inşa ederler. Dolaşma işlemini yapan k yapay karıncası bir q. düğümden j. düğümüne geçerken izlediği yolu aşağıdaki denklem yolu ile gerçekleştirmektedirler.

$$\gamma_{qj}^{k} = \begin{cases} \dfrac{\tau_{qj}^{\alpha}\eta_{qj}^{\beta}}{\sum_{c_{ql}\in N(s^p)} \tau_{ql}^{\alpha}\eta_{ql}^{\beta}} & c_{qj} \in N(s^p) \\ 0 & diğer, \end{cases} \tag{3.1}$$

γ_{qj}^{k}, k karıncasının q'den j'ye geçme olasılığıdır. τ_{qj} ve η_{qj}, q ve j arasındaki sırasıyla feromonun miktarını ve sezgisel bilgiyi temsil etmektedir. N_{s^p} de, çözüm kümesini göstermektedir. α ve β, feromon miktarının ve sezgisel değerlerinin katkılarını kontrol eden parametre değerleridir.

Yerel arama uygulama adımı tercihli bir adımdır. Algoritmanın sonuçlarını iyileştirmek için dâhil edilmiş bir adımdır. Burada araştırmacılar çözmeye çalıştıkları probleme göre çeşitli yerel aramalar kullanmışlardır. Bir önceki adımda elde edilen en iyi

çözüm bu adımda iyileştirilmeye çalışılır. Algoritmanın performansına önemli katkısı vardır.

Algoritmanın son adımı olan feromonları güncelleme adımında, çözümlerin kalitelerine göre feromon değerleri güncellenir. Güncelleme, aşağıda yer alan Denklem (3.2) ile gerçekleştirilir.

$$\tau_{qj} = (1-\rho) \times \tau_{qj}(t) + \sum_{k=1}^{m} \Delta\tau_{qj}^{k} \tag{3.2}$$

Burarada ρ buharlaşma oranını, m ise karınca sayısını göstermektedir. $\Delta\tau_{qj}^{k}$ k karıncasının, i ile q'nun arasına bıraktığı feromon miktarını göstermektedir.

Güncelleme işlemi çözümlerin kalitesine göre iki şekilde yapılır. Birincisi depolama adı verilen işlemdir. Buna göre, iyi çözümlere ait feromon değerleri arttırılır. Böylelikle bunların seçilme şansları arttırılır. Bu, algoritmanın pozitif geri beslemesine örnek olarak verilebilir. Diğeri ise, buharlaştırmadır. Bunda ise, kötü çözümlerin feromon değerleri, feromon buharlaştırılması ile azaltılır. Bu sayede seçilme şanslarının azalması sağlanır. Bu da negatif geri beslemesine örnek olarak verilebilir.

Bu adımlar bir döngü halinde algoritmanın tamamlanma kriteri sağlanıncaya kadar devam etmektedir.

3.2.2.2. *Parçacık sürü optimizasyon algoritması*

Parçacık Sürü Optimizasyon (Particle Swarm Optimization, PSO), 1995 yılında Eberhart ve Kennedy tarafından kuşların göç hareketlerinden (bird flocking) yola çıkılarak ortaya koyulmuş bir sürü zekâsı algoritmasıdır (Kennedy ve Eberhart, 1995). Sürü zekâsı algoritmalarının ilk örneklerinden biri sayılabilir. Basit ve kolay kodlanabilir oluşu araştırmacıların dikkatini çekmiştir. Böylece farklı alanlardaki çeşitli zor problemlerin çözümü için kullanılmıştır. Çok sayıdaki uygulama alanlarından bazıları; yapay sinir ağı (Leung vd., 2012), ders planlaması (Hossain vd., 2019) ve verilerin kümelenmesi (Alswaitti vd., 2018) .

PSO için literatür özeti

Farklı alanlardaki zor problemlerin PSO algoritması ile çözülmesinin yanında araştırmacılar, PSO'nun yeteneklerini geliştirmeye çalışmışlardır. Bununla alakalı olarak yapılan ilk çalışmalar algoritmanın parametre değerlerinin geliştirilmesi yönünde

olmuştur (Agarwalla ve Mukhopadhyay, 2017). Shi ve Eberhart, hız güncelleme denklemine atalet ağırlığını (ω) ekleyerek parçacığın önceki hız değerinin etkisini ayarlayarak algoritmanın arama davranışını kontrol etmeye çalışmışlardır (Shi ve Eberhart, 1998). Bununla birlikte, (Shi ve Eberhart, 1999) atalet ağırlığı iterasyonlar ilerledikçe doğrusal olarak azalan bir yapıya kavuşmuş ve olumlu sonuçlar almışlardır. Buna ek olarak, atalet ağırlığı rastgele olarak belirlenen, logaritmik fonksiyonlar kullanılarak güncellenen, doğal üstel fonksiyon ile güncellenen gibi farklı atalet ağırlığı güncelleme yöntemleri önerilmiştir (Harrison vd., 2016). Ayrıca atalet ağırlığı dışındaki parametreler için de farklı yaklaşımlar önerilmiştir. Bunlara örnek olarak, algoritmanın ilk çıkışında hız denkleminde yer alan hızlanma ya da öğrenme katsayıları adı verilen *c1* ve c2 çalışma süresince sabit değer yöntemi kullanılarak değerleri belirleniyordu. Buna karşın Ratnaweera, Halgamuge, ve Watson doğrusal zamanla değişen hızlanma katsayılarını kullanmışlardır (Ratnaweera vd., 2004). K. Chen vd., PSO'nun erken yakınsama eksiğini gidermek için sinüs ve kosinüs fonksiyonlarından oluşan hızlanma katsayılarından faydalanmışlardır (K. Chen vd., 2017). Bunlara ek olarak Clerc ve Kennedy, algoritmanın yakınsama hızını artırmak ve parçacıkların arama uzayını terk etmemesini sağlamak için daralma katsayısı (constriction coefficient) adı verilen yeni bir değişkeni hız arama denklemine eklemişlerdir (Clerc ve Kennedy, 2002),.

Parametrelerde yapılan geliştirmelerin yanında PSO popülasyondaki parçacıkların topoloji olarak tanımlanan komşuluk yapıları için de farklı alternatifler önererek algoritmanın arama yeteneğinin güçlendirilmesi amaçlanmıştır. Global en iyi (Eberhart ve Kennedy, 1995) ve yerel (local) en iyi öne çıkan iki topoloji yapısıdır. Bunlara ek olarak (Kennedy, 1999); yıldız, rastgele, tekerlek, piramit, Von Neumann, 4-Küme gibi komşuluk yapılarının performansını araştırmıştır. (Shi vd., 2011) hücresel otomat komşuluk yapısı olarak kullanarak yerel optimuma takılmayı önlemeye çalışmışlardır. (García-Nieto ve Alba, 2014) ise rastgele topoloji kullanarak iyi sonuçlar elde etmişlerdir.

Orijinal PSO'da parçacıklar, kendi kişisel en iyisi ile global en iyi parçacıktan öğrenmektedir. Bu yaklaşımdan farklı olarak yeni öğrenme yaklaşımları da önerilmiştir. (Liang vd., 2006) bütün parçacıkların geçmiş en iyilerinden faydalanan sonucunda popülasyondaki çeşitliliği sağlayan yeni bir öğrenme yöntemi önermişlerdir. (Zhan vd., 2011) ortogonal deneysel tasarımı kullanan bir öğrenme yöntemi ortaya koymuşlardır. (Ren vd., 2014) arama uzayına dağıtılmış yüksek kalitedeki çözümlerden oluşan havuzu kullanan bir öğrenme yöntemini tercih etmişlerdir. (Gong vd., 2016) öğrenme için

kullanılan örneği üretmek için genetik algoritmadan esinlenen bir strateji tercih etmişlerdir.

PSO algoritmasının global arama becerisinin yanında yerel arama yetenekleri karmaşık problemlerde yetersiz kalmaktadır. Bu nedenden dolayı araştırmacıların bazıları yerel arama teknikleri ile melezleştirerek melez PSO türevleri ortaya koymuşlardır. (Montes de Oca vd., 2011) Powell'ın yön belirleme yöntemini tercih etmişlerdir. (Ren vd., 2014) PSO varyantı algoritmalarını Solis ve Wets yerel arama algoritması ile melezleştirmişlerdir. (García-Nieto ve Alba, 2014) kendi PSO algoritmalarını çoklu yörünge araması (Multiple Trajectory Search, MTSLS1) ile birleştirmişlerdir. (Chen vd., 2018) yerel arama yöntemi olarak Quasi-Newton metodunu kullanmışlardır.

Algoritma tanımı

Parçacık sürü optimizasyonu algoritması, çözümleri arama uzayında yer alan parçacıklar (particle) olarak temsil etmektedir. Her bir çözüm, üç adet D boyutlu vektöre sahiptir. Bunlar; parçacığın hızını temsil eden $V_i = [V_{i1}, V_{i2}, \cdots, V_{iD}]$ vektörü, parçacığın pozisyonunu yani arama uzayındaki yerini gösteren $L_i = [L_{i1}, L_{i2}, \cdots, L_{iD}]$ vektörü ile parçacığın o ana kadarki kişisel en iyi pozisyonun saklandığı $pbest_i^t = [P_{i1}, P_{i2}, \cdots, P_{iD}]$ vektörüdür (Bonyadi ve Michalewicz, 2016).

Algoritma 3.2 *Orijinal parçacık sürü optimizasyonu algoritmasının sözde kodu*

```
1: Hız ve pozisyon değerlerini rastgele üret.
2: while Sonlandırma kriteri sağlanıncaya kadar do
3:     Kişisel en iyiyi güncelle
4:     Global en iyiyi güncelle
5:     for Her bir parçacık i do
6:         Hız vektörünü Denklem (3.3) ile güncelle
7:         Pozisyon vektörünü Denklem (3.4) ile güncelle
```

PSO, popülasyon (topluluk) tabanlı bir algoritmadır ve popülasyonda N adet parçacık yer almaktadır. Algoritma ilk başladığında; parçacıkların pozisyon ve hızları düzgün (uniform) dağılım kullanılarak rastgele üretilmektedir. İterasyonlar ilerledikçe tüm parçacıkların hızları ve pozisyonları güncellenmektedir. Hız vektörleri, Denklem (3.3)'de görüldüğü gibi hız değeri güncellenen referans parçacığının kişisel en iyisi *pbest* ile popülasyondaki en iyi parçacığın pozisyonu *gbest* kullanılarak belirlenir. Bununla birlikte; parçacıkların pozisyonları da güncellenen parçacığın mevcut pozisyon bilgisi ile

yeni elde edilmiş hız değeri kullanılarak güncellenmektedir ve Denklem (3.4)'de yer verilmiştir. Bu hız ve pozisyon güncelleme işlemi yinelemeli olarak algoritmanın bitme kriterine kadar devam eder. PSO'nun sözde kodu Algoritma 3.2'de verilmiştir.

$$V_i^{t+1} = V_i^t + c_1 rand(0,1)(pbest_i^t - L_i^t) + c_2 rand(0,1)(gbest^t - L_i^t) \quad (3.3)$$

$$L_i^{t+1} = L_i^t + V_i^{t+1} \quad (3.4)$$

Denklemde yer alan, V_i^t parçacığın mevcut hız değerini, L_i^t ise parçacığın mevcut pozisyonunu göstermektedir. c_1 ile c_2 sırasıyla bilişsel (cognitive) ve sosyal (social) öğrenme faktörleridir. $pbest_i^t$, *i* parçacığının *t* anındaki en iyi amaç (objective) değerine sahip yerini temsil etmektedir. $gbest^t$, *t* anındaki popülasyondaki tüm parçacıkların arasında en iyi amaç fonksiyon değerine sahip parçacığı temsil etmektedir.

3.2.2.3. Yapay arı kolonisi algoritması

Yapay arı kolonisi (Aritificial Bee Colony, ABC), 2005 yılında Derviş Karaboğa tarafından geliştirilen doğadan ilham alan bir algoritmadır. Bal arılarının zeki ve organize davranışlarına dayanır. Algoritmanın ortaya çıkmasından bu yana, çeşitli alanlardaki birçok problemlerin çözümünde birçok araştırmacı tarafından başarıyla kullanılmıştır.

ABC için literatür özeti

ABC algoritmasının geliştirilmesi için çok sayıda çalışma yapılmıştır. İlk yapılan çalışmalar işçi arı ve gözcü arı adımları için tek bir etkili arama denklemleminin kullanılmasına dayanmaktadır. Diwold vd. (Diwold vd., 2011)'de, arama denklemi rastgele seçilen çözümler yerine popülasyonun o ana kadar ki en iyi çözümü ile yönlendirmiştir. Gao ve Liu (Gao ve Liu, 2011), diferansiyel gelişim (Differential Evoluation, DE) algoritmasından ilham alan bir arama denklemi kullanmışlardır. Alataş (Alatas, 2010), kaotik bir harita kullanarak ABC'nin yerel en iyiye takılmamasını sağlamaya çalışmıştır. Bununla birlikte, Karaboğa ve Akay (Akay ve Karaboga, 2012), orijinal ABC'deki aday çözüm üretiminde kullanılan tek bir parametre değiştirme yaklaşımının yetersiz olduğunu savunmuşlardır. Böylece, algoritmanın arama kapasitesini, (MR) adlı parametre değeriyle değiştirerek arama yeteneğini geliştirmeyi amaçlamışlardır. Banharnsakun, Achalakul, ve Sirinaovakul (Banharnsakun vd., 2011), orijinal ABC'nin komşu çözüm temelli yaklaşımı yerine o ana kadarki en iyi çözüm

yaklaşımını önermişlerdir. Ayrıca; Gao, Liu, ve Huang (Gao vd., 2014), aday çözüm üretmek için işçi ve gözcü adımları için ayrı arama denklemleri tanımlamışlardır. Aydin vd. (Aydın vd., 2011), ABC algoritmasını artan sosyal öğrenme çatısı (Incremental Social Learning Framework, ISL) ile birlikte yerel arama yöntemiyle birleştirmişlerdir. Böylece algoritmanın arama ve faydalanma yeteneğini geliştirmeyi amaçlamışlardır.

Algoritma tanımı

Yapay arı kolonisi algoritmasının sözde kodu Algoritma 3.3'de verilmiştir. Sözde koddan anlaşılacağı üzere ABC dört bileşenden oluşmaktadır. Bunlar ilklendirme, işçi arı, gözcü arı ve kâşif arı adımlarıdır.

Algoritma 3.3. *Orijinal yapay arı kolonisi algoritması sözde kodu*

```
1: İlklendirme
2:
3: while bitirilme kriteri sağlanıncaya kadar do
4:        İşçi arı adımı
5:        Gözcü arı adımı
6:        Kâşif arı adımı
```

İlklendirme adımı: Bu adımda, algoritmanın kontrol parametreleri ayarlanır. Bu parametreler; çözümleri temsil eden yiyecek kaynaklarının sayısı (*SN*), yiyecek kaynaklarının ömrünü temsil eden *limit* değeri ve algoritmanın bitme kriteridir.

Popülasyonda yer alan her birey, yiyecek kaynağı olarak isimlendirilmektedir. Bu yiyecek kaynakları, muhtemel çözümleri temsil eden D-boyutlu vektörlerdir. Algoritmanın bu adımında popülasyonda yer alan her bir x_i çözümü aşağıdaki denklem ile arama uzayında düzgün (uniform) dağılım kullanılarak üretilmektedir:

$$x_{i,j} = x_{i,j} + rand(0,1)(x_j^{max} - x_j^{min}) \tag{3.5}$$

Burada, i yiyecek kaynağının [1, SN] aralığındaki indeks değerini ve j problem değişkeninin [1, D] aralığındaki indeks numarasını göstermektedir. x_j^{max} ve x_j^{min}, x_i sayısının j yönündeki sırasıyla maksimum ve minimum değerleridir. rand(0,1), belirtilen aralıkta düzgün dağılım kullanılarak rastgele bir sayı üretir. Burada j $(1 \geq j \geq D)$ problem değişkeninin indeks numarasıdır. Daha sonra, her bir çözüm x_i'nin amaç fonksiyonunun değerleri (f_i) hesaplanır. Algoritmanın ilerleyen adımlarında, her

bir çözüm en aza indirilmeye çalışılmaktadır. Her x_i çözümünün deneme sayısı *trial*$_i$'dir ve sıfır olarak ilklendirilir.

İşçi arı adımı: İşçi arı adımında, referans aday çözümlerinin her biri değiştirilir ve referansların her birinin etrafında bir aday çözüm üretilir.

İşçi arı ile aday çözümler arasında birebir bir eşleme mevcuttur. Bu yüzden işçi arı sayısı kadar çözüm vardır. Referans çözüm x_i'nin bir boyutu olan j değiştirilerek aşağıda yer alan denklem ile $\hat{x}_i$ aday çözümü üretilir.

$$\hat{x}_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{r,j}) \tag{3.6}$$

Burada j, $j \in \{1,2,\ldots,D\}$ olmak üzere rastgele bir sayıdır. $\phi_{i,j}$, [-1,1] aralığında rastgele oluşturulmuş bir sayıdır. $x_{r,j}$ ($r \neq i$ ve $r \in \{1,2,\ldots,SN\}$) rastgele seçilen başka bir çözümdür. Ardından yeni aday çözüm $\hat{x}_i$'nin amaç fonksiyonu (f_i) hesaplanır ve x_i ile $\hat{x}_i$ arasında açgözlü (greedy) seçim uygulanarak daha iyi olan seçilir. Eğer $\hat{x}_i$, x_i'den daha iyi değil ise *trial*$_i$ değeri bir artırılır.

Gözcü arı adımı: İşçi arı adımındaki tüm çözümlerin kalitesi değerlendirildikten sonra algoritma, gözcü adımında da *SN* kadar yeni aday çözüm üretir. Gözcü arı adımını işçi arı adımından ayıran en belirgin fark, referans aday çözümünün seçimindeki farklılıktır.

İşçi arı adımının aksine her gözcü arı, referans kaynağı seçerken kaynağın kalitesi ile alakalı bir olasılık değeri kullanmaktadır. Bir aday çözümün seçim olasılık değeri (p_i) aşağıdaki gibi hesaplanır:

$$p_i = \frac{\dfrac{1}{1+f(x_i)}}{\sum_{i}^{N} \dfrac{1}{1+f(x_n)}}. \tag{3.7}$$

Denklem (3.7)'de f_i, x_i'nin amaç fonksiyon değeri olmak üzere aday çözümün uygunluk fonksiyon değeri aşağıdaki eşitlikle hesaplanmaktadır:

$$fitness_i = \begin{cases} \dfrac{1}{1+f(x_i)}, & f(x_i) \geqslant 0, \\ 1+abs(f(x_i)), & f(x_i) < 0. \end{cases} \tag{3.8}$$

Yüksek kaliteli çözümler daha yüksek olasılıkla seçildiğinden, gözcü arılar daha kaliteli çözümleri olan komşuları araştırmayı tercih ederler. Bir gözcü arı bir çözüme ulaştıktan sonra, işçi arı adımında kullanılan Denklem (3.6) ile yeni çözümler arar.

Kâşif arı adımı: Aday çözüm x_i'ler, birçok defa ziyaret edilmesine rağmen, aday çözümde bir *limit* değeri kadar iyileşme meydana gelmeyebilir. Bu durumda aday çözüm terk edilir. Yeni bir aday çözüm Denklem (3.5) kullanılarak rastgele üretilir ve popülasyona dahil edilir. Bu strateji sayesinde arama uzayı araştırılmakta ve bu şekilde algoritmanın arama yeteneğinin artırılması sağlanmaktadır.

4. PARAMETRE BELİRLEME YAKLAŞIMLARI

Algoritma tasarımcıları, metasezgisel algoritmalarını geliştirirken algoritmalarının mümkün olduğunca esnek, problemden bağımsız çalışan, yüksek performanslı olmasını isterler (Sörensen, 2015; Sorensen vd., 2017; Talbi, 2009). Yüksek performans kriterini sağlamak için, tasarımcılar algoritma bileşenlerini titizlikle seçerler. Algoritmanın gücünü arttıracağına inandıkları bileşenleri bir araya getirerek bu bileşenler arasında bir uyum yakalamaya çalışırlar. Algoritmalarının problemden bağımsız ve esnek çalışması taleplerini karşılamak için ise değerlerini dışarıdan belirledikleri bileşenleri kullanırlar.

Yukarıda bahsi geçen bileşenlere atanacak olan değerler, algoritmaya gönderilen parametreler aracılığıyla sağlanır. Örnek vermek gerekirse; ABC'de yer alan yiyecek kaynaklarının sayısı veya PSO'da yer alan parçacıkların komşuluk yapısı, algoritmalara dışarıdan parametreler yardımıyla gönderilerek algoritmaların çalışmaları şekillendirilebilir. Parametreler için belirlenen değerler ise, çözülmesi istenen problemin yapısına ve zorluk derecesine göre seçilmektedir. Bu seçilen değerlerin, algoritmaların performansı üzerinde büyük etkileri vardır (Aydın, Yavuz, ve Stützle, 2017) ve bu tür parametrelere kontrol veya strateji parametreleri denilmektedir (Eiben vd., 1999).

Algoritmaların kontrol parametreleri türlerine göre ikiye ayrılmaktadır. Bunlar, sayısal ve kategorik parametrelerdir. Sayısal parametreler de kendi içerisinde tam sayılı ve reel sayılı olmak üzere iki gruba ayrılmaktadır. Tam sayılı parametreye PSO'daki popülasyonu oluşturan parçacıkların sayısı ve reel sayılı parametreye ise yine PSO'da bulunan öğrenme katsayıları (*c1* ve *c2*) örnek verilebilir. Sayısal parametrelerin aksine kategorik parametreler, algoritmanın içerisindeki fonksiyonların yöntemlerini belirleyen ve kesikli değerler alan parametrelerdir. Örnek olarak, PSO'da her bir parçacığın komşuları vardır ve topoloji adı verilen farklı şekillerde organize olmaktadırlar. Bu topolojilerin, halka (Ring), yıldız (Star) gibi farklı alternatifleri bir seçim kümesini oluşturmaktadır ve bunun algoritma üzerinde önemli etkisi vardır. Bu alternatifleri barındıran seçim kümesi, kategorik parametreye örnek olarak verilebilir.

MS'ler, doğaları gereği rassal (stochastic) yapıya sahiptirler. Dolayısıyla algoritmaların çalışırken ortaya koyacağı davranışlar çalışma öncesinde bilinememektedir. Öte yandan MS'lerin sahip olduğu kontrol parametrelerinin ilk değerleri, algoritmanın çalışmasına başlamadan önce verilmektedir. Araştırmacılar ilk değerlerin verilme işlemini, titizlikle yaparak ilgili problem için algoritmanın performansını artırmaya çalışırlar. Ancak bu parametrelere verilen değerlerin,

performansa olumlu veya olumsuz katkıları da algoritma çalıştırılmadan bilinememektedir. Bu parametrelerin ilk değer belirleme işlemini kolaylaştırmak ve algoritmaların performansını artırmak için literatürde çeşitli yöntemler önerilmiştir. Bu yöntemler, uygulanma şekillerine göre iki sınıfa ayrılmaktadır. İlk sınıf parametre kontrol yöntemleri (parameter control) diğeri ise parametre ayarlama yöntemleridir (parameter tuning) (Karafotias vd., 2013; Talbi, 2009) .

4.1. Parametre Kontrol Yöntemleri

Parametre kontrol yöntemine çevrim-içi (online) yöntem de denilmektedir. Bu yöntemde, algoritma çalışmaya başlamadan önce parametrelere başlangıç değerleri verilir ve çalışma süresince bu parametrelerin değerleri belirlenen bir yönteme göre değiştirilmektedir. Parametre kontrol de kendi içerisinde üçe ayrılmaktadır (Karafotias vd., 2013; Talbi, 2009):

Deterministik parametre kontrolü (Deterministic parameter control): Bu yöntemde kullanıcılar, parametreleri güncelleyecek yöntemi ve bu yöntemin çalışma süresini belirlerler. Ayrıca parametreler, belirlenen sürede herhangi bir geri besleme olmadan kontrol mekanizmasına göre güncellenir.

Uyarlanabilir parametre kontrolü (Adaptive parameter control): Parametre güncelleme işlemi, algoritmanın arama sürecinden elde edilen geri beslemesine göre parametre değerleri belirlenir.

Kendinden uyarlamalı parametre kontrolü (Self-adaptive parameter control): Bu yöntemde, parametre belirleme kontrol yöntemi algoritmanın içerisine dahil edilmiştir. Örneğin, parametreler yapay arı kolonisi algoritmasına birer yiyecek kaynağı olarak dahil edilir. Böylelikle, problem çözülürken parametreler de duruma göre güncellenir.

4.2. Parametre Ayarlama Yöntemleri

Parametre ayarlama (parameter tuning), aynı zamanda çevrim-dışı (offline) yöntem olarak da bilinmektedir. Bu yöntemde algoritmanın parametre değerleri, algoritma çalışması öncesinde belirlenir ve algoritmanın çalışması boyunca sabit kalmaktadır (Nannen ve Eiben, 2007). Bu parametre ayarlama yöntemine, literatürde "algoritma yapılandırma problemi" (Algorithm Configuration Problem, AYP) ismi de verilmektedir (Hutter vd., 2009) .

Parametre ayarlama yöntemleri, yapılış şekillerine göre sınıflandırılmaktadır. Sınıflandırma yaklaşımları şunlardır (Montero vd., 2014) :

El yordamı ile ayarlama (Hand-made tuning): Bu yöntem, algoritma tasarımcısı tarafından deneme-yanılma yolu ile yapılmaktadır. Bu, algoritma tasarımcısının içgüdüsüne ve tecrübelerine dayanmaktadır. İnsan emeği kullanılarak yapıldığı için, denenebilecek parametre uzayı sınırlıdır ve aynı zamanda zaman alıcı bir iştir. Tasarımcılar, parametreleri belirlerken tek tek denerler. Diğer bir ifade ile bir parametrenin değerini değiştirirken diğer parametrelerin değerlerini sabit tutarlar. Bundan dolayı parametreler arasındaki ilişki değerlendirilemediğinden ve kontrol parametre sayısı arttıkça yeterli miktarda parametre değer örneği denenemediğinden bu yöntem ile belirlenen parametre değerlerinin algoritmaya katkısı sınırlıdır.

Kıyas yolu ile ayarlama (Tuning by analogy): Bu yaklaşım, başka araştırmacıların benzer problemlerde uyguladıkları ve başarısı kanıtlanmış olan yöntemlerin, parametre ayarlama yöntemi olarak kullanılmasıdır (Eiben vd., 1999).

Deneysel tasarım tabanlı ayarlama (Experimental desing based tuning): Bu kategoriye dahil olan yöntemler, parametre ayarlamayı deneylere bağlı olarak gerçekleştirirler. Yarışma metotlarından olan yinelemeli F-race bu sınıflandırma içerisinde yer almaktadır (Birattari vd., 2010). Buna ek olarak, Sıralı Parametre Optimizasyonu (Sequential Parameter Optimization, SPO) yöntemini de örnek olarak verilebilir (Bartz-Beielstein vd., 2005).

Arama tabanlı ayarlama (Search based tuning): Arama tabanlı yöntemler, parametrelerin yüzey analizini yaparak en uygun parametreleri belirlemeye çalışırlar (Wang vd., 2017). Bunlara örnek olarak, yinelemeli yerel arama stratejisini kullanan ParamILS (Hutter vd., 2009) ve parametre dağılımlarını tahmin eden Revac algoritması (Nannen ve Eiben, 2007) verilebilir.

Melez ayarlama (Hybrid tuning): Son iki kategoride yer alan farklı yöntemleri birlikte kullanan ayarlama yöntemleridir. Buna örnek olarak ise, iyi parametreleri elde etmek için, deneysel tasarım ile yerel arama yöntemini bir arada kullanan Calibra (Adenso-Díaz ve Laguna, 2006) verilebilir. Ne yazık ki, Calibra, sürekli ve kesikli beş parametreye kadar çalışabilmektedir.

4.2.1. Otomatik parametre ayarlama (yapılandırma) yaklaşımları

Parametre ayarlama işlemini otomatikleştirmek için araştırmacılar tarafından çeşitli araçlar geliştirilmiştir. Bu kısımda otomatik parametre ayarlama araçlarından bazılarına kısaca değinilecektir.

4.2.1.1. *Parametre yinelemeli yerel arama*

Parametre Yinelemeli Yerel Arama (Parameter Iterated Local Search, ParamILS) (Hutter vd., 2009), parametre yapılandırma uzayını yinelemeli yerel arama tekniği (Iterated Local Search, ILS) ile tarayan bir yöntemdir. İlklendirme için varsayılan ve rastgele ayarlar kombinasyonunu kullanır. Bunlara ek olarak, tek komşu değişim tabanlıdır yani bir anda sadece tek bir kontrol parametresinin değeri değiştirilir (Hutter vd., 2009). Ayrıca algoritmaların parametreleri kategorik olarak tanımlanması gerekmektedir. Sayısal ve sürekli parametrelere de kategorik parametre olarak davranılmaktadır (Pérez Cáceres ve Stützle, 2017). BasicILS ve FocusILS adında iki varyantı vardır.

4.2.1.2. *Cinsiyete dayalı genetik algoritma*

Cinsiyete Dayalı Genetik Algoritma (Gender based Genetic Algorithm, GGA) (Ansótegui vd., 2009), özelleştirilmiş bir genetik algoritma kullanan bir yapılandırıcıdır. ParamILS gibi küçük parametre uzayı ile sınırlı değildir ve reel sayılar gibi sürekli değerli parametreleri de doğrudan kullanabilir (Hoos H. H., 2011).

4.2.1.3. *Sıralı model tabanlı algoritma yapılandırması*

Sıralı model tabanlı algoritma yapılandırması (Sequential Model-based Algorithm Configuration, SMAC), parametre yapılandırmalarının performanslarını tahmin etmek için bir vekil model (surrogate-model) kullanan yapılandırıcıdır (Hutter vd., 2011). SMAC'ın modelleri, regresyon ve sınıflandırma için standart bir makine öğrenme aracı olan rassal ormana (random forest) dayanmaktadır. ParamILS'nin aksine reel değer ve kategorik parametreleri desteklemektedir. Parametreleri belirlenmeye çalışılan algoritmanın, her bir çalıştırmasına ait maksimum zamanı dinamik olarak kontrol eder. Ayrıca, kötü yapılandırmaları daha öncesinden tahmin eder. Böylelikle irace aracının aksine hesaplama zamanından tasarruf sağlayabilmektedir (López-Ibáñez vd., 2016).

4.2.1.4. *Alaka tahmini ve parametrelerin değer kalibrasyonu*

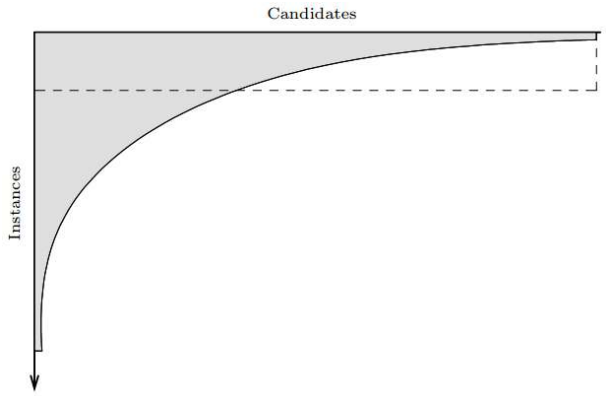
Alaka tahmini ve parametrelerin değer kalibrasyonu (Relevance Estimation and Value Calibration of Parameters, REVAC) Mannen ve Eiben tarafından 2006 yılında önerilmiştir (Nannen ve Eiben, 2007). Aslında REVAC, bir evrimsel yöntemdir (Smit ve Eiben, 2010). Parametrenin dağılımlarını tahmin etmek için operatörlerden faydalanır. Parametre yapılandırılmalarından oluşan bir popülasyon kullanır. Mutasyon ve çaprazlama yöntemlerini iterasyonlar boyunca kullanarak parametre yapılandırmalarını iyileştirmeye çalışır. Bu yöntem, sadece sürekli parametreler ile çalışmaktadır.

4.2.1.5. *Yinelemeli f-race*

Birattari, bir parametre ayarlama yaklaşımı önermiştir. F-race adındaki bu yaklaşım, yarışma ve Friedman'ın parametrik olmayan iki yönlü varyans analiz testini temel alan bir yöntemdir (López-Ibáñez vd., 2016). Bu yöntem, bir aday yapılandırma kümesi içerisinden en iyi olan yapılandırmayı yarışma ile belirlemeye çalışır.

F-race, ilk olarak rastgele yapılandırmalar üretir ve bunları bir problem seti üzerinde dener. Daha sonra en kötü olan yapılandırmaların belirlenmesi ve bunların elenmesi için Friedman'ın parametrik olmayan iki yönlü varyans analiz testi kullanılır. Bu test sayesinde, yapılandırmalar arasında istatistiksel olarak bir farklılık var mı diye kontrol edilir. Eğer bir farklılık varsa, kötü olan yapılandırma adayları elenir. F-race, yarışmada tek bir yapılandırma kalana kadar çalışmasını sürdürür. Şekil 4.1'de de bu elenme işleminin sonucundaki yapılandırma sayılarının değişimi görülmektedir. Son olarak, en iyi olan yapılandırma belirlenir.

Yinelemeli F-race (Iterated F-race) ise, F-race yöntemini tekrarlı olarak uygulamaktadır (Birattari vd., 2010). Her iterasyonda yinelemeli F-race, son kalmış olan en iyi yapılandırmanın etrafında yeni bir yapılandırma kümesi örnekler. Böylelikle iyi olan yapılandırılmalara odaklanılması sağlanır. Daha sonra yine F-race yöntemi uygulanır. Bu işlem; yinelemeli F-race'nin bitme kriteri olan maksimum algoritma çalıştırma sayısına kadar devam eder. Geriye kalan son yapılandırma, en iyi yapılandırma olarak sunulur.



Şekil 4.1. *Yarışma yöntemindeki yapılandırmaların, iterasyonlar ilerledikçe sayılarının değişimine ait grafiksel gösterim. İlk önce, aday yapılandırmalar hakkında bilgiler toplanmaktadır. Daha sonra yetersiz olduğu düşünülen adaylar elenir. Yarışma yöntemi, böylelikle daha ümit vadeden adaylara odaklanır. Grafikte aday çözümlerin miktarının değişimi görülmektedir* (Birattari vd., 2010).

irace, yinelemeli F-race yöntemini gerçekleyen bir yazılım paketidir (Birattari vd., 2010). "irace" aracı son yıllarda en çok kullanılan üç parametre yapılandırma aracıdan (irace, paramils, smac) birisi olarak öne çıkmaktadır. Bu tez de parametre değerleri belirlenirken irace paketinden yararlanılmıştır. Tezin ilerleyen bölümlerinde anlatılacak olan önerilen algoritmalar ile birlikte deneylerde karşılaştırma amaçlı kullanılan algoritmaların kontrol parametreleri irace yazılım paketi ile belirlenmiştir. Böylelikle deneylerdeki tüm karşılaştırmaların daha adil bir şekilde yapılması sağlanmıştır.

5. ŞABLON PARÇACIK SÜRÜ OPTİMİZASYONU

5.1. Algoritma Tanımı

Bu tezde; Orijinal PSO algoritmasını temel alan, esnek bir şablon algoritma geliştirilmesi amaçlanmıştır. Bunun için ilk olarak, literatürde yer alan çeşitli performanslı PSO varyantları belirlenmiştir. Daha sonra, bu algoritmaların ortak özellikleri belirlenerek algoritmaların bileşen grupları oluşturulmuştur. İlerleyen bölümlerde algoritma bileşen gruplarına değinilecektir.

Literatürde ortaya çıkan yeni PSO varyantı algoritmalar Orijinal PSO algoritmasına çeşitli yenilikler getirmektedir. Böylelikle PSO'nun performansını geliştirmeyi amaçlamışlardır. Bununla birlikte, bu yenilikler PSO'ya, yeni hız denklemi veya yeni komşuluk topolojisi gibi ufak değişiklikler getirmiştir. Ayrıca her bir yeni değişiklik üstesinden gelmeye çalıştıkları problem türlerine göre tasarlanmıştır. Bu tez kapsamında geliştirilen TempPSO için, bu izlenen yolun dışında farklı bir yol izlenmesi amaçlanmıştır. Buna göre, literatürde ortaya konmuş PSO varyantlarında yer alan iyileştirmelerin bir araya getirilerek algoritma şablonu altında toplanması hedeflenmiştir. Böylelikle varyantların bilgi birikimlerinden faydalanılması amaçlanmıştır. Bunun için, dokuz adet gelişmiş PSO varyantı seçilmiştir. Bu PSO varyantları : Kapsamlı Öğrenme Parçacık Sürü Optimizasyonu (Comprensive Learning Particle Swarm Optimization, CLPSO) (Liang vd., 2006), Örnek Tabanlı Parçacık Sürü Optimizasyonu (Example-Based Learning Particle Swarm Optimization, ELPSO) (Huang vd., 2012), Frankenstein'ın Parçacık Sürü Optimizasyonu (Frankenstein's Particle Swarm Optimization, FrankPSO) (Montes de Oca vd., 2009), Genetik Öğrenme Parçacık Sürü Optimizasyonu (Genetic Learning Particle Swarm Optimization, GLPSO) (Gong vd., 2016), Çeşitlilik ile Global Genetik Parçacık Sürü Optimizasyonu (Global Genetic Learning Particle Swarm Optimization with Diversity, GGLPSOD) (Lin vd., 2018), Parçacık Sürü Optimizasyonu (Orijinal Particle Swarm Optimization, PSO) Tamamen Bilgilendirilmiş Parçacık Sürüsü (Fully informed particle swarm, FIPS) (Mendes vd., 2004), İki Diferansiyel Mutasyonlu Parçacık Sürü Optimizasyon (Particle swarm optimizer with two differential mutations, PSOTD) (Y. Chen vd., 2017) ve PSOk (García-Nieto ve Alba, 2014)'dır. Bu algoritmaların özellikleri gruplandırılarak mevcut ve yeni algoritmaları üretilebilecek şablon algoritması geliştirilmiştir. Bu yeni algoritmaya "Şablon Parçacık Sürü Optimizasyonu" (Template Particle Swarm Optimization, TempPSO) ismi verilmiştir.

Oluşturduğumuz “TempPSO” algoritması, dışarıdan yani komut satırından parametre alabilecek yapıda tasarlanmıştır. Her bir parametre, algoritmanın farklı bir özelliğini temsil etmektedir. Bu parametreler ile algoritmanın herhangi bir özelliği seçilebilmektedir. Bu sayede, TempPSO bir otomatik algoritma yapılandırma aracıyla şekillendirilebilir hale getirilmiştir. Bu tezde detayları Bölüm 4.2.1’de verilen otomatik algoritma yapılandırma aracı olan irace (López-Ibánez vd., 2011) kullanılmıştır.

Algoritmanın her bir bileşenin seçimi için, algoritma ve bir örnek problem seti, parametre ayarlama aracına gönderilmektedir. Bu araç (irace), algoritmaya farklı parametreler göndererek en iyi parametre yapılandırmasını belirlemeye çalışır. Parametre yapılandırması tamamladığında, problem setine en uygun bir parametre yapılandırması ortaya çıkmaktadır. Bu parametre yapılandırması artık yeni bir PSO varyantına karşılık gelecektir.

TempPSO algoritmasının iskelet yapısı Algoritma 5.1’de verilmiştir. Algoritma; ilk olarak, algoritmanın başlangıç popülasyonunu çeşitlendirmesini sağlayan ilklendirme aşaması ile çalışmaya başlar. Bu aşamada parçacıkların; hız, pozisyon alanları ve atalet ağırlığının ilk değer atamaları için ayrı stratejiler uygulanmaktadır. Bununla birlikte, algoritmada eğer kullanılacaksa parçacıkların öğrenmesinde etkili olan topoloji adındaki komşuluk yapıları belirlenir. Bu işlemler tamamlandıktan sonra, probleme özgü belirlenmiş olan hız, pozisyon ve atalet ağırlığı güncelleme denklemleri ile ilklendirme aşamasında belirlenmiş alanlar güncellenir. Daha sonra algoritma popülasyonun en iyi parçacığını günceller. Bu işlem tamamlanma kriteri sağlanıncaya kadar devam eder. Algoritma bileşenlerinin ayrıntılı açıklamaları ilerleyen kısımlarda yer alacaktır.

Algoritma 5.1. *Şablon Parçacık Sürü Optimizasyon Algoritmasının iskelet yapısı*

```
1: procedure TemplatePSO
2:     initialization(), iteration = 0
3:     while Termination criteria is not met do
4:         for each particle i do
5:             applyRefreshingGapStrategy(iteration)
6:             updateVelocity(i)
7:             updatePosition(i)
8:             updateInertiaWeight()
9:             updateLearningFactor(LfStr1; LfStr2)
10:    applyLocalSearch()                          # Optional
11:    iteration = iteration + 1
```

Tablo 5.1. *TempPSO algoritmasının parametreleri. i: tam sayı, r: reel sayı, c: kategorik değerleri göstermektedir.*

İsim	Tür	Değer	Açıklama
nparticles	i	(3, 100)	Popülasyon boyutu
iStr	c	uniform, kaotik, karşıtlık	Pozisyon ilklendirme yöntemleri
chaoticmap	c	(1, 2, 3, 6,7,8)	Kaotik harita çeşitleri
velinitstr	c	sıfır, uniform, küçük rastgele	Hız ilklendirme yöntemleri
iwinitstr	c	sabit, uniform	Atalet ağırlığı ilklendirme yöntemleri
min_omega	r	(0, 0.5)	Atalet ağırlığının minimum değeri
iw	r	(0, 1)	Atalet ağırlığı değeri
max_omega	r	(0.5, 0.9)	Atalet ağırlığının maksimum değeri
lf1	r	(0, 4)	Birinci öğrenme faktörü
lf2	r	(0, 4)	İkinci öğrenme faktörü
lf1upstr	c	kullanma, sabit, doğrusal	Birinci öğrenme faktörü güncelleme stratejileri
lf2upstr	c	kullanma, sabit, doğrusal	İkinci öğrenme faktörü güncelleme stratejileri
min_lf1	r	(0.1, 1)	Birinci öğrenme faktörünün minimum değeri
max_lf1	r	(1, 2.5)	Birinci öğrenme faktörünün maksimum değeri
min_lf2	r	(1, 2.5)	İkinci öğrenme faktörünün minimum değeri
max_lf2	r	(0.1, 1)	İkinci öğrenme faktörünün maksimum değeri
velclampstr	c	yutan, sınır, rastgele	Hız değeri sınırlandırma yöntemleri
posboundstr	c	nearest, reflect, random, pbest, periodic, mirror, no action	Pozisyon sınırlandırma yöntemleri
iwupdstr	c	sabit, doğrusal, frankenstein	Atalet ağırlığı güncelleme stratejileri
refgapstr	c	kullanma, CLPSO, GLPSO	Yenilenme boşluğu stratejileri
refgap	i	(1, 100)	Yenilenme boşluğu iterasyon sayısı
topostr	c	kullanma, fully connected, ring, random	Topoloji stratejileri
rns	c	(10, 20, 30, 40, 50, 60, 70, 80, 90)	Rastgele komşuluk sayısı
pm	c	(0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2)	Mutasyon olasılığı

Tablo 5.1. (Devam) *TempPSO algoritmasının parametreleri i: tam sayı, r : reel sayı, c : kategorik değerleri göstermektedir.*

İsim	Tür	Değer	Açıklama
CR1	c	(0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)	Birinci çaprazlama olasılığı
CR2	c	(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)	İkinci çaprazlama olasılığı
iwschedule	c	(1, 2, 3, 4, 5)	Atalet ağırlığı için planlama (FrankPSO)
tschedule	c	(1, 2, 3, 4, 5)	Topoloji için planlama (FrankPSO)
vu1term	c	3 seçenek (Tablo 5.2)	Hız güncelleme denkleminin birinci terimi
vu2term	c	12 seçenek (Tablo 5.2)	Hız güncelleme denkleminin ikinci terimi
vu3term	c	12 seçenek (Tablo 5.2)	Hız güncelleme denkleminin üçüncü terimi
pu1term	c	5 seçenek (Tablo 5.4)	Pozisyon güncelleme denkleminin birinci terimi
pu2term	c	7 seçenek (Tablo 5.4)	Pozisyon güncelleme denkleminin ikinci terimi
pu3term	c	7 seçenek (Tablo 5.4)	Pozisyon güncelleme denkleminin üçüncü terimi
lsID	c	Kullanma, MTSLS1, CMAES, Rekabetçi yerel arama	Yerel arama stratejileri
ttunea	r	(1, 10)	CMAES'ye ait bir kontrol parametresi
ttuneb	r	(1, 5)	CMAES'ye ait bir kontrol parametresi
ttunec	r	(0.1, 1)	CMAES'ye ait bir kontrol parametresi
ttuned	r	(1, 4)	CMAES'ye ait bir kontrol parametresi
ttunee	r	(-20, -6)	CMAES'ye ait bir kontrol parametresi
ttunef	r	(-20, -6)	CMAES'ye ait bir kontrol parametresi
ttuneg	r	(-20, -6)	CMAES'ye ait bir kontrol parametresi
MTSLS_ITR	i	(3, 100)	MTSLS1'in iterasyon sayısı
CompBudget	c	(0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3) * MAXFES	Rekabet aşaması için fonksiyon çağrım sayısı
CMAESFES	c	(0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5) *MAXFES	CMAES için maksimum fonksiyon çağırım sayısı

5.1.1. TempPSO algoritma bileşenleri

TempPSO'nun geliştirilmesinin ilk adımı olarak, Orijinal PSO algoritmasının yapısında yer alan temel adımları içeren, gerçeklenmesi kolay ve saygın dergilerde yayınlanmış dokuz PSO varyantı algoritma literatürden seçilmiştir. Seçilen algoritmalar ayrı ayrı kodlanarak performansları incelenmiştir. Daha sonra bu algoritmaların yapıları incelenerek ortak özellikleri çıkarılmıştır. Ardından da bu ortak özellikleri içeren gruplar oluşturulmuştur. Oluşturulan gruplar, TempPSO'nun temelini oluşturan algoritmaların bileşenlerini barındırmaktadır. Sonraki alt başlıklarda algoritma grupları ve TempPSO'nun bileşenleri detaylandırılacaktır.

5.1.1.1. *İlklendirme adımı*

PSO algoritmalarında çözümler, parçacık ismindeki bireyler ile temsil edilmektedir. Bu parçacıkların; uzaydaki durumunu belirleyen pozisyon (L) ve hareketini temsil eden hız (V) vektörleri vardır. Literatürde hız ve pozisyon vektörlerinin başlangıç değerlerini belirlemek için çeşitli yaklaşımların olduğu görülmüştür (Cazzaniga vd., 2015; Engelbrecht, 2012; W. feng Gao vd., 2012; García-Nieto ve Alba, 2014). Bu çalışmalardan elde edilen bilgilerin ışığında önerilen algoritmaya çeşitli alternatifler dâhil edilmiştir.

Popülasyondaki her bir parçacığın pozisyonlarının ilk değer ataması için Tablo 5.1'de görüldüğü gibi üç bileşen seçenek olarak sunulmuştur. Bunlardan ilki, Orijinal PSO algoritması ile gelen "düzgün dağılım (uniform)" ilklendirme yöntemidir. Bu stratejiye göre, aday çözümler arama uzayında "düzgün dağılım" yöntemi kullanılarak rastgele üretilmektedir. İkinci bileşen "kaotik (chaotic)", rastgele sayı üreteci olarak kaotik haritaları kullanmaktadır. Son ilklendirme bileşeni "karşıtlık (opposition)"'dır. Bu yönteme göre rastgele üretilen bir aday ile birlikte, çözüm uzayının merkezine göre yansıması da üretilir. N, popülasyon boyutu olmakta üzere elde edilen $2 \times N$ çözüm içinden kötü olan N tanesi elenir ve geriye iyi olanlar kalır.

Parçacıkların bir diğer bileşeni olan hız vektörü için ise farklı ilklendirme mekanizmaları mevcuttur. TempPSO geliştirilmesine temel olan algoritmalardan üç alternatif belirlenmiş ve Tablo 5.1'de listelenmiştir. Belirlenen bileşenlerden ilki olan "sıfır (zero)" yönteminde, parçacığın başlangıçta herhangi bir hareketinin olmadığını kabul edilir. Bundan dolayı tüm parçacıkların hız değeri "0" olarak atanmaktadır. İkinci bileşen olan "düzgün dağılım (uniform)", Orijinal PSO da yer alan standart yöntemdir ve

düzgün dağılım kullanılarak ilk hız değerleri belirlenmektedir. Son bileşen ise "küçük rastgele (small random)"'dir. Bu yöntemde hız değerleri [0, 0.1] aralığından rastgele değerler ile ilklenmektedir.

Bu kısımda yer alan bir diğer ilklendirme işlemi ise, hız güncelleme denklemlerinde yer alan atalet ağırlığına ilk değerlerin verilmesidir. Atalet ağırlığı algoritmada eğer yer alıyorsa, ilk değer ataması için iki yaklaşım mevcuttur. Bunlardan ilki, "sabit" yani dışarıdan bir değerin verilmesidir ve bu değer algoritma çalışması boyunca sabit kalmaktadır. Diğer yaklaşım ise "düzgün dağılım (uniform)", yani atalet ağırlığının düzgün dağılım kullanılarak rastgele belirlenmesidir.

5.1.1.2. *Atalet ağırlığı güncellemesi*

TempPSO'nun temelini oluşturan dokuz algoritmada yer alan atalet ağırlığı güncellemeleri, üç alternatifli bir grup oluşturulmuştur. Tablo 5.1 ile listelenen bileşenlerden ilki "sabit (fix)"'dir. Bu ilk bileşene göre atalet ağırlığı, herhangi bir güncelleme yapılmadan algoritma çalışması boyunca sabit kalmaktadır. İkinci bileşen ise GGLPSOD'de yer alan "doğrusal (linear)" yaklaşımdır. Denklem (5.1)'deki bu yaklaşımda atalet ağırlığı, optimizasyon işlemi boyunca doğrusal olarak ayarlanmaktadır. Son olan bileşen ise "Frankenstein", Denklem (5.2)'de yer almaktadır. Bu yöntem ise, Frankenstein'ın PSO'unda yer alan atalet ağırlığı güncelleme mekanizmasıdır.

$$\omega = \omega_i - \frac{itr_{current}}{itr_{max}}(\omega_{max} - \omega_{min}) \tag{5.1}$$

$$\omega = \frac{\omega itr_{max} - itr_{current}}{\omega itr_{max}}(\omega_{max} - \omega_{min}) + \omega_{min} \tag{5.2}$$

ω_{max} ve ω_{min} atalet ağırlığının alabileceği sırasıyla en büyük ve en küçük değerleri temsil etmektedir. $itr_{current}$, mevcut iterasyonu, itr_{max} maksimum iterasyon sayısını göstermektedir.

5.1.1.3. *Hız güncelleme stratejisi*

Çözümleri temsil eden parçacıkların hızlarını güncellemek için çeşitli alternatifler, TempPSO'nun temelini oluşturan algoritmalarda sunulmuştur. Bu yaklaşımları, bu çalışma da gerçekleştirmek için genelleştirilmiş hız güncelleme denklemi önerilmiş ve bu

Algoritma 5.2'de gösterilmiştir. Bu denklemde yer alan V_{terim1_i}, V_{terim2_i} ve V_{terim3_i} bileşenleri için çeşitli değerler önerilmiştir. Bunlar, Tablo 5.2'de yer almaktadır. Bu bileşenlerin değerleri otomatik algoritma yapılandırma aracı (bu tezde irace kullanılmıştır.) tarafından algoritma yapılandırma aşamasında seçilmektedir.

Algoritma 5.2'de yer alana genelleştirilmiş denklem yaklaşımı TempPSO'ya esneklik ve çeşitlilik kazandırmaktadır. TempPSO, bu yaklaşım sayesinde algoritmanın temelini meydana getiren PSO varyantlarının hız güncelleme denklemlerini de örnekleyebilmektedir. Dahası, literatürde yer alan üstün performanslı farklı algoritmalardaki hız güncelleme denklemlerinde bulunan bileşenlerin bir arada kullanılabilmesine imkân sağlanmaktadır. Ayrıca, daha önce önerilmemiş yeni hız güncelleme denklemlerinin üretilmesini sağlaması gibi faydaları da vardır.

Algoritma 5.2. *TempPSO'nun hız güncelleme adımı*

```
1: procedure UPDATEVELOCITY(i)
2:     V_i = V_terim1_i + V_terim2_i + V_terim3_i
3:     APPLYVELOCITYLIMITSTRATEGY(ITERATION)
```

5.1.1.4. *Hız sınırlandırma stratejileri*

Parçacıklar hız güncelleme denklemleri ile güncellendikten sonra elde edilen yeni değerler sonucunda parçacıklar arama uzayını terk edebilmektedirler. Yani yeni hız değerleri problemin sınır değerlerini aşabilmektedir. Böylelikle parçacıklar istenmeyen yerlere doğru gidebilmekte ve algoritmanın performansını kötüleştirmektedir. Bundan dolayı literatürde hız sınırlama (velocity clamping) stratejileri önerilmiştir.

TempPSO'da hız sınırlandırma yöntemi olarak üç farklı seçenek algoritmaya dâhil edilmiştir. Bunlar, "yutan (absorb)", "sınır (boundry)" ve "rastgele (random)" bileşenleridir (Helwig vd., 2013). Bileşenlere ait denklemler Tablo 5.3'de listelenmiştir.

$V_{i,max}$ değeri problemin maksimum değeri ve $V_{i,min}$ değeri ise problemin minimum değeri olarak alınmıştır

Tablo 5.2. *Hız güncelleme denklem bileşenleri. c1 ve c2 birinci ve ikinci öğrenme faktörüdür. rand(0,1), 0 ile 1 aralığında üretilen rastgele sayıyı göstermektedir. $pbest_i$ mevcut parçacıktır. $pbest_r$, $pbest_{r1}$ ve $pbest_{r2}$ popülasyondan rastgele seçilmiş olan parçacıkların kişisel en iyilerini göstermektedir. $pbest_{neigh}$ komşuluk topolojisine göre en iyi parçacığın kişisel en iyisidir. e_{CLPSO}, e_{GLPSO}, $e_{GGLPSOD}$ ve e_{PSOTD} CLPSO, GLPSO, GGLPSOD ve PSOTD algoritmalarındaki örnek üretme yöntemleri ile üretilmiş örnek parçacıkları göstermektedir. L_i mevcut parçacığın mevcut pozisyonunu, L_r popülasyondan rastgele seçilen parçacığın mevcut pozisyonun temsil etmektedir. gbest, popülasyondaki en iyi parçacıktır. V_i, mevcut parçacığın hızıdır.*

id	V_{terim1}	V_{terim2}	V_{terim3}
0	0	0	0
1	V_i	$c1rand(0,1)*(pbest_i - L_i)$	$c2rand(0,1)*(pbest_i - L_i)$
2	$\omega * V_i$	$\sum_{j \in N_i} rand^t[0,\phi_j]*(pbest_j - L_j)$	$\sum_{j \in N_i} rand^t[0,\phi_j]*(pbest_j - L_j)$
3		$c1rand(0,1)*(pbest_r - L_i)$	$c2rand(0,1)*(pbest_r - L_i)$
4		$c1rand(0,1)*(gbest - L_i)$	$c2rand(0,1)*(gbest - L_i)$
5		$c1rand(0,1)*(gbest - L_r)$	$c2rand(0,1)*(gbest - L_r)$
6		$c1rand(0,1)*(pbest_{neigh} - pbest_r)$	$c2rand(0,1)*(pbest_{neigh} - pbest_r)$
7		$c1rand(0,1)*(e_{CLPSO} - L_i)$	$c2rand(0,1)*(e_{CLPSO} - L_i)$
8		$c1rand(0,1)*(e_{GLPSO} - L_i)$	$c2rand(0,1)*(e_{GLPSO} - L_i)$
9		$c1rand(0,1)*(e_{GGLPSOD} - L_i)$	$c2rand(0,1)*(e_{GGLPSOD} - L_i)$
10		$c1rand(0,1)*(e_{PSOTD} - L_i)$	$c2rand(0,1)*(e_{PSOTD} - L_i)$
11		$c1rand(0,1)*(pbest_{r1} - pbest_{r2})$	$c2rand(0,1)*(pbest_{r1} - pbest_{r2})$

Tablo 5.3. *Hız sınırlandırma seçenekleri. $V_{i,max}$ ile $V_{i,min}$ hız değerinin en büyük en küçük değeridir. L_{max} ile L_{min} probleminin en büyük ve en küçük değeridir. rand(0,1), 0 ile 1 arasından rastgele seçilen bir sayıdır.*

id	Metotlar	Denklemler
1	absorb	$\dot{V}i,t+1 = \begin{cases} 0: & V_{i,t+1} > V_{i,max} \\ 0: & V_{i,t+1} < V_{i,min} \end{cases}$
2	random	$\dot{V}i,t+1 = \begin{cases} -rand(0,1)\ V_{i,t+1} : V_{i,t+1} > V_{i,max} \\ -rand(0,1)\ V_{i,t+1} : V_{i,t+1} < V_{i,min} \end{cases}$
3	boundary	$\dot{V}i,t+1 = \begin{cases} X_{max} : V_{i,t+1} > V_{i,max} \\ X_{min} : V_{i,t+1} < V_{i,min} \end{cases}$

5.1.1.5. *Pozisyon güncelleme stratejisi*

Bu tezde, mevcut algoritmaların pozisyon güncelleme yöntemlerinden elde edilen tecrübe ile genelleştirilmiş pozisyon güncelleme denklemi oluşturulmuştur. Genel yapısı Algoritma 5.3'de verilmiştir. Denklemde yer alan her bir terim için farklı alternatifler önerilmiş ve bunlar Tablo 5.4'de gösterilmiştir. Bu genelleştirilmiş pozisyon denkleminin her bir terimi, probleme özgü olarak otomatik algoritma yapılandırma aracı "irace" yardımıyla belirlenmektedir.

Algoritma 5.3. *TempPSO pozisyon güncelleme adımı. Algoritmanın başında her bir terim için Tablo 5.4'den rastgele bir bileşen seçilir. Seçilen terimler algoritmanın hız güncelleme denklemini oluşturur.*

1: **procedure** UPDATEPOSITION(i)
2: $P_i = P_{terim1_i} + P_{terim2_i} + P_{terim3_i}$
3: APPLYPOSITIONLIMITSTRATEGY(ITERATION)

Tablo 5.4. *Pozisyon güncelleme denklemi bileşenleri. L_{max} ile L_{min} probleminin en büyük ve en küçük değeridir. V_i, L_i mevcut parçacığın hız ve pozisyon değerleridir. $pbest_i$, mevcut parçacığın kişisel en iyisidir.*

id	P_{terim1}	P_{terim2}	P_{terim3}
0	0	0	0
1	$rand(L_{min},L_{max})$	$rand(L_{min},L_{max})$	$rand(L_{min},L_{max})$
2	V_i	V_i	V_i
3	$pbest_i$	$pbest_i$	$pbest_i$
4	L_i	L_i	L_i
5		$rand(0,1)*(L_{max} - L_{min})$	$rand(0,1)*(L_{max} - L_{min})$
6		$rand(0,1)*(pbest_i – L_i)$	$rand(0,1)*(pbest_i – L_i)$

5.1.1.6. *Pozisyon sınırlandırma stratejileri*

Parçacıkların pozisyon güncelleme denklemleri ile güncellendikten sonra elde edilen yeni aday pozisyon değerlerini, yani problemin sınırlarını aşabilmektedir. Böylelikle parçacık istenmeyen yerlere doğru gidebilmekte ve algoritmanın performansı düşebilmektedir. Bundan dolayı literatürde çeşitli pozisyon sınırlama (position bounding) stratejileri önerilmiştir.

TempPSO'ya hız sınırlandırma stratejisi olarak yedi farklı bileşen dâhil edilmiştir. Temel alınan PSO varyantlarından ve literatürden elde edilen hız sınırlandırma bileşen alternatifleri Tablo 5.5'de listelenmiştir (Helwig vd., 2013) .

Tablo 5.5. *Pozisyon sınırlandırma seçenekleri. MOD, mod işlemini temsil etmektedir.* $\dot{L}$ *, sınırlandırılmış olan yeni pozisyon değerini göstermektedir.*

id	Metotlar	Denklemler
0	nearest	$\dot{L}_{i,t+1} = \begin{cases} L_{max} : L_{i,t+1} > L_{max} \\ L_{min} : L_{i,t+1} < L_{min} \end{cases}$
1	pbest	$\dot{L}_{i,t+1} = \begin{cases} pbest_i : L_{i,t+1} > L_{max} \\ pbest_i : L_{i,t+1} < L_{min} \end{cases}$
2	reflect	$\dot{L}_{i,t+1} = \begin{cases} L_{max} - (L_{i,t+1} - L_{max}) : L_{i,t+1} > L_{max} \\ L_{min} + (L_{min} - L_{i,t+1}) : L_{i,t+1} < L_{min} \end{cases}$
3	random	$\dot{L}_{i,t+1} = \begin{cases} rand(L_{min}, L_{max}) : L_{i,t+1} > L_{max} \\ rand(L_{min}, L_{max}) : L_{i,t+1} < L_{min} \end{cases}$
4	no action	$\dot{L}_{i,t+1} = \begin{cases} L_{i,t} : L_{i,t+1} > L_{max} \\ L_{i,t} : L_{i,t+1} < L_{min} \end{cases}$
5	periodic	$\dot{L}_{i,t+1} = \begin{cases} L_{min} + (x_{i,t} \ MOD \ (L_{max} - L_{min})) : L_{i,t+1} > L_{max} \\ L_{min} + (L_{i,t} \ MOD \ (L_{max} - L_{min})) : L_{i,t+1} < L_{min} \end{cases}$
6	mirror	$\dot{L}_{i,t+1} = \begin{cases} L_{i,t+1} : L_{min} \le L_{i,t+1} \le L_{max} \\ 2L_{max} - L_{min} : L_{max} \le L_{i,t+1} \le 2L_{max} \end{cases}$

5.1.1.7. *Öğrenme faktörü güncelleme stratejisi*

Orijinal PSO algoritmasının hız güncelleme denkleminde iki adet öğrenme faktörü (learning factor) yer almaktadır. Bunlar *c1* ve *c2* olarak isimlendirilmiş ve algoritmanın başlangıcında tanımlanmıştır ve algoritma çalışması boyunca sakladıkları değerler sabit kalmaktadır. Bununla birlikte, araştırmacılar öğrenme faktörü değeri için çeşitli yöntemler kullanarak olumlu sonuçlar da elde etmişlerdir. TempPSO'nun temel algoritmalarından biri olan GGLPSOD algoritmasında, öğrenme faktörleri için "doğrusal (linear)" yaklaşımı kullanılmıştır. Bu yöntemde, öğrenme faktörlerinin değerleri zamanla doğrusal değişmektedir. Bu yöntem TempPSO'ya dâhil edilmiş ve Denklem (5.3) ile (5.4) 'de listelenmiştir.

$$c_1 = c_{1,i} - \frac{itr_{current}}{itr_{max}}(c_{1,max} - c_{1,min}) \tag{5.3}$$

$$c_2 = c_{2,i} - \frac{itr_{current}}{itr_{max}}(c_{2,max} - c_{2,min}) \tag{5.4}$$

$c_{1,min}$ ve $c_{1,max}$, c1 öğrenme faktörünün en küçük ve en büyük değerlerini göstermektedir. $c_{2,min}$ ve $c_{2,max}$, c2 öğrenme faktörünün en küçük ve en büyük değerlerini temsil etmektedir. $itr_{current}$, mevcut iterasyonu, itr_{max} maksimum iterasyon sayısını göstermektedir.

5.1.1.8. *Yenilenme aralığı stratejileri*

Bazı PSO varyantları, algoritmanın yerel optimuma takılmasını önlemek için yenilenme boşluğu (refreshing gap) isminde bir strateji uygulamaktadırlar. Algoritmalar parçacıkların güncellenmesinde örnek parçacık (exemplar) ismini verdikleri rehberleri kullanmaktadırlar. Bu örnek vasıtasıyla, parçacıkların amaç fonksiyonlarının iyileşmesi amaçlanmaktadır. Ancak örnek parçacıklar, rehberlik ettiği parçacıkları her zaman iyileştiremezler. Algoritma, bu örnek parçacığın faydasız olduğunu düşünür. Eğer iyileştirme, önceden tanımlanan yenilenme boşluğu miktarı (genellikle yedi iterasyon sayısı olarak belirlenir) kadar gerçekleşmez ise algoritma parçacıklar için yeni örnek parçacıklar oluşturur. Bunun için CLPSO ve GLPSO ile GGLPSO algoritmaları farklı iki yaklaşım önermişlerdir. CLPSO algoritması, popülasyondan iki rastgele parçacık seçer. Bu iki rastgele parçacık ile mevcut parçacığın kişisel en iyilerinin değişkenlerinden bir olasılık değerine göre seçim yapılır. Bu seçim ile yeni bir örnek oluşturulur. Bu yeni örnek, üç parçacığın bilgilerini taşıyan farklı bir parçacık gibi düşünülebilir. GLPSO yaklaşımında ise, genetik algoritmada bulunan çaprazlama, mutasyon ve seçme operatörlerini kullanarak parçacıklar için örnekler oluşturulur. Sonuç olarak, bu iki yaklaşım TempPSO çatısına eklenmiştir.

5.1.1.9. *Yerel arama stratejisi*

Yukarıda anlatılan algoritma bileşenlerine ek olarak TempPSO'ya isteğe bağlı bir mekanizma eklenmiştir. Bu mekanizma, çeşitli PSO varyantlarında ve farklı meta-sezgisellerde bulunan yerel arama ile melezleştirme yöntemidir. Bu yöntem, yerel arama tekniklerinin gücünü kullanarak PSO algoritması arama yeteneğindeki eksikliği gidermek için TempPSO'ya eklenmiştir. İsteğe bağlı olan bu yaklaşım için alternatifler Tablo 5.1'de görüldüğü gibi dört adettir. Bunlardan ilki, yerel arama kullanılmadan algoritmanın tek başına çalışmasıdır. İkincisi, TempPSO'nun Lin-Yu Tseng'ye ait MTSLS1 (Tseng ve Chen, 2008) yerel arama tekniği ile melezleştirilmesidir. Üçüncü olarak güçlü bir yerel arama tekniği olan CMAES (Liao vd., 2013) yöntemi melezleştirme

seçeneği olarak sunulmuştur. Son olarak ise, (Yavuz vd., 2016) bir ABC varyantı ve Bölüm 7'de açıklanmış olan SSEABC algoritmasında bulunan "rekabetçi (competition) yerel arama stratejisi" dâhil edilmiştir. Bu yöntemin detayları Bölüm 7.1.2.' de belirtilmiştir.

5.2. Deney Sonuçları

5.2.1. CEC'17 deney sonuçları ve analizleri

Önerilen algoritmanın performansını değerlendirmek için CEC'17 ölçüt seti (Awad vd., 2016) tercih edilmiştir. CEC'17, evrimsel algoritmaların sürekli optimizasyon alanındaki performansın ölçülmesinde ve karşılaştırılmasında kullanılmaktadır. Ayrıca, ölçüt seti dört ayrı kategori olmak üzere 30 adet fonksiyon içermektedir. Bunlar, üç adet unimodal (f1-f3), yedi adet multimodal (f4-f10), onar adet hibrit (f11-f20) ve kompozit (f21-f30) fonksiyonlarıdır. Ayrıca CEC'17 ölçüt setinin özeti, Tablo 5.6'da listelenmiştir. Bununla birlikte, tüm fonksiyonların arama uzayı $[-100,100]^D$ (D problem boyutu) olarak belirlenmiştir.

Önerilen TempPSO algoritmasında "rekabetçi yerel arama stratejisi" yer almaktadır. Bu strateji sayesinde, algoritmanın zayıf olan yerel arama yönünün güçlendirilmesi sağlanmıştır. Bu bileşenin algoritmaya katkısını görmek için, TempPSO'nun yerel arama içermeyen haliyle birlikte rekabetçi yerel arama stratejisini içeren hali deneylere dâhil edilmiştir ve sırasıyla TempPSO ve TempPSO-ls adları verilmiştir.

TempPSO ve TempPSO-ls'nin değerlendirilmesi için iki ayrı deney yapılmıştır. İlk deneydeki karşılaştırmada, literatürde yer alan gelişmiş PSO varyantları kullanılmıştır. Bu algoritmalar, kendi makaleleri incelenerek kodlanmıştır. Daha sonraki deney ise, TempPSO ile CEC'17 yarışmasına katılan yarışmacıların sonuçları karşılaştırılarak yapılmıştır. Burada yer alan algoritmaların sonuçları kendi makalelerinden elde edilmiştir. Deney sonuçları bir sonraki bölümde verilecektir.

Testler, (Awad vd., 2016)'da yer alan şartlara bağlı kalınarak gerçekleştirilmiştir. Buna göre, her algoritma her bir fonksiyon için 51 defa bağımsız olarak çalıştırılmıştır. Bununla birlikte, deneyler *D* problem boyutunu göstermek üzere $D\times 10000$ fonksiyon çağrımında bitirilmiştir. Her çalıştırmada elde edilen ortalama hata değeri (average error) hesaplanmıştır. Burada hata değeri, aday çözüm ile optimal çözümün farkı $f(x)-f(x^*)$ olarak tanımlanır. Ayrıca, hata değeri 10^{-8}'den küçük değerleri 10^{-8} olarak kabul

edilmiştir. Deneyler tamamlandıktan sonra algoritmaların elde ettikleri sonuçların birbirinden farklılıklarını göstermek için wilcoxon testi uygulanmıştır. Tüm deneyler 8 GB RAM ve Intel Xeon E5-2620 2 GHz işlemciye sahip bir bilgisayarda yapılmıştır.

Tablo 5.6. *CEC'17 ölçüt setinin fonksiyon tanımları*

Türler	Id	Fonksiyonlar*
Unimodal Fonksiyonlar	f1	Shifted and Rotated Bent Cigar Function
	f2	Shifted and Rotated Sum of Different Power Function
	f3	Shifted and Rotated Zakharov Function
Multimodal Fonksiyonlar	f4	Shifted and Rotated Rosenbrock's Function
	f5	Shifted and Rotated Rastrigin's Function
	f6	Shifted and Rotated Expanded Scaffer's F6 Function
	f7	Shifted and Rotated Lunacek Bi Rastrigin Function
	f8	Shifted and Rotated Non-Continuous Rastrigin's Function
	f9	Shifted and Rotated Levy Function
	f10	Shifted and Rotated Schwefel's Function
Hibrit Fonksiyonlar	f11	Hybrid Function 1 (N=3)
	f12	Hybrid Function 2 (N=3)
	f13	Hybrid Function 3 (N=3)
	f14	Hybrid Function 4 (N=4)
	f15	Hybrid Function 5 (N=4)
	f16	Hybrid Function 6 (N=4)
	f17	Hybrid Function 6 (N=5)
	f18	Hybrid Function 6 (N=5)
	f19	Hybrid Function 6 (N=5)
	f20	Hybrid Function 6 (N=6)
Kompozit Fonksiyonlar	f21	Composition Function 1 (N=3)
	f22	Composition Function 2 (N=3)
	f23	Composition Function 3 (N=4)
	f24	Composition Function 4 (N=4)
	f25	Composition Function 5 (N=5)
	f26	Composition Function 6 (N=5)
	f27	Composition Function 7 (N=6)
	f28	Composition Function 8 (N=6)
	f29	Composition Function 9 (N=3)
	f30	Composition Function 10 (N=3)
		*Arama aralığı: [-100, 100]

5.2.1.1. *Parametre ayarları*

CEC'17 ile yapılan deneylerden önce TempPSO ile TempPSO-ls ve diğer PSO varyantlarının parametreleri Bölüm 4.2.1'de detayları anlatılmış olan irace aracı ile belirlenmiştir. Bu sayede adil bir karşılaştırma yapılması sağlanmıştır.

irace ile elde edilen parametreler, algoritmaların CEC'17 ölçüt seti için en iyi performans gösteren değerleridir ve Tablo 5.7'de gösterilmiştir. TempPSO algoritma çatısı ile CEC'17 ölçüt seti kullanılarak iki algoritma üretilmiştir. Bunlardan ilki yerel arama bulunmayan TempPSO'dur. Diğeri ise rekabetçi yerel arama strateji dâhil edilerek yapılan ve parametre yapılandırma işlemi sonucunda üretilen TempPSO-ls'dir. CEC'17 ölçüt setine özgü PSO varyantı yeni algoritmaların sözde kodları Algoritma 5.4 ve Algoritma 5.5'de yer almaktadır. Böylelikle, TempPSO'nun farklı algoritmalar üretebilme yeteneği incelenecektir.

Tablo 5.7. *Deneye katılan algoritmaların ayarlanmış parametre değerleri*

Algoritmalar	Parametre değerleri
TempPSO-ls	lsID = 4 nparticles = 61 min_omega = 0.0421 iw = 0.3812 max_omega = 0.6763 iStr = 2 chaoticmap = 7 iwinitstr = 1 velinitstr = 1 velmaxstr = 1 velclampstr = 3 vu1term = 2 vu2term = 11 vu3term = 13 pm = 0.001 lf1 = 2.1594 lf2 = 2.8668 isc = 1 1 isc2 = 1 lf1upstr = 2 lf2upstr = 1 min_lf2 = 1.2235 max_lf2 = 0.6971 isconst = 0 iwupdstr = 1 posboundstr = 4 pu1term = 2 pu2term = 4 pu3term = 3 isref = 1 refgap = 8 refgapstr = 1 ttunea = 8.5927 ttuneb = 1.8721 ttunec = 0.3867 ttuned = 3.9024 ttunee = -10.5492 ttunef = -12.714 ttuneg = -11.8541 lsItr = 83 tr = 0.3 cmar = 0.15
TempPSO-nols	lsID = 0 nparticles = 36 iStr = 1 velinitstr = 1 velmaxstr = 1 velclampstr = 1 vu1term = 1 vu2term = 6 vu3term = 5 lf2 = 1.485 isc1 = 0 isc2 = 1 lf2upstr = 1 min_lf2 = 1.8081 max_lf2 = 0.4088 isconst = 0 posboundstr = 3 pu1term = 3 pu2term = 0 pu3term = 2
GLPSO	nparticles = 66 iw = 0.423 lf1 = 2.0155 refgap = 9 pm = 0.01
GGLPSOD	nparticles = 63 iw = 0.7314 min_omega = 0.3158 max_omega = 0.5673 refgap = 6 pm = 0.05 min_lf1 = 0.8849 max_lf1 = 2.4108 min_lf2 = 1.6384 max_lf2 = 0.5718
ELPSO	nparticles = 51 iw = 0.0127 refgap = 6 lf1 = 1.1573 lf2 = 1.4434
FIPS	nparticles = 72 lf1 = 2.3283 lf2 = 1.7284 topostr = 2
CLPSO	nparticles = 20 iw = 2.8914 min_omega = 0.284 max_omega = 0.6924 lf1 = 1.6211 refgap = 1
PSO	nparticles = 24 min_omega = 0.0809 max_omega = 0.7725 lf1 = 1.035 lf2 = 3.5816
PSOTD	nparticles = 16 iw = 0.268 lf1 = 2.8286 CR1 = 0.05 CR2 = 0.6 F = 0.1
FrankPSO	nparticles = 22 iw = 1.8546 min_omega = 0.0016 max_omega = 0.5804 lf1 = 0.1195 lf2 = 2.8565 iwschedule = 4 tschedule = 1 topostr = 1
PSOk	nparticles = 68 lf1 = 1.2471 lf2 = 2.9214 topostr = 3 rns = 10

Algoritma 5.4 detaylı olarak incelenirse; irace aracı, yeni önerilen algoritma parçacık pozisyonlarının ilk değer ataması için Orijinal PSO'da bulunan "uniform" yöntemini seçmiştir. Ayrıca parçacıkların hız vektörü için ilk değer belirleme yöntemi

olarak ise, “zero” yöntemi tercih edilmiş yani parçacıkların ilk başta hareket etmediği kabul edilmiştir. Bununla birlikte, hız vektörü güncelleme denklemi için ise *gbest* ve $pbest_r$ değişkenleri tercih edilmiştir. Böylece algoritmanın global en iyi parçacık yardımıyla algoritmanın arama yönü desteklenmiştir. Ayrıca popülasyondan rastgele seçilen bir parçacığın kişisel en iyisi kullanılarak çözümün çeşitlendirilmesi sağlanmıştır. Daha sonra, yeni aday hız değerleri problem sınırlarını aştığında makul değerlere çekmek için kullanılacak sınırlama yöntemi olarak ise, “absorb” stratejisi seçilmiştir. Bununla birlikte, yeni bir parçacık pozisyon güncelleme denklemi önerilmiştir. Karşılaştırmada yer alan PSO varyantlarının denklemleri incelendiğinde bu denklemin farklı olduğu görülmüştür. Ayrıca, Orijinal PSO’nun hız güncelleme denkleminde yer alan mevcut parçacığın pozisyon değerinin, o anki pozisyona eklenmesi yönteminin aksine parçacığın kişisel en iyi değerinin eklendiği görülmüştür. Böylelikle, algoritma mevcut parçacığın kişisel en iyisini kullanarak daha iyi bir çözüm elde etmeyi amaçladığı görülmektedir. Daha sonra, pozisyon sınırlandırma yöntemi olarak ise “random” seçilerek algoritmanın arama uzayını çeşitlendirilmesi sağlanmıştır. Son olarak, hız güncelleme denkleminde kullanılan öğrenme faktörünün (*c*) güncellenmesi için “linear” yöntemi seçilmiştir. Böylelikle, iterasyonlar ilerledikçe rassallık ağırlığı azaltılarak global en iyinin etkisinin arttırılması sağlanmıştır.

Rekabetçi yerel arama stratejisini içeren TempPSO algoritmasının sözde kodu olan Algoritma 5.4 incelendiğinde, Algoritma 5.5’den farklı olduğu görülmektedir. Parçacıkların pozisyonlarına ilk değer ataması için “chaotic” yöntemi seçilmiştir. Hız bileşenlerinin ilk değer ataması için ise “zero” yöntemini tercih edilmiştir. Ayrıca hız güncelleme denkleminde, GGLPSOD algoritmasının bileşeninden ve algoritmanın arama çeşitliliğini sağlayan iki rastgele parçacığın kişisel en iyisinden faydalanılmıştır. Bununla birlikte, atalet ağırlığı kullanılarak mevcut hız değerinin ağırlığının azaltılması sağlanmıştır. Ayrıca “*c1*” ve “*c2*” öğrenme faktörleri kullanmıştır. Algoritma 5.4’deki pozisyon güncelleme denkleminden farklı olarak buradaki pozisyon güncelleme denklemine mevcut pozisyon değeri de eklemiştir. Hız sınırlandırma yöntemi “boundary”, pozisyon sınırlandırma yöntemi olarak da “pbest” seçilmiştir. Atalet ağırlığı ve öğrenme faktörü (*c2*) güncellemesi içinse “linear” yöntemi tercih edilmiştir. Yenilenme boşluğu stratejisi olarak da GLPSO yöntemi seçilmiştir. Son olarak, yerel arama stratejisi için irace tarafından rekabetçi yerel arama stratejisi seçilmiştir.

Algoritma 5.4. *irace aracı ile CEC'17 için üretilen yerel arama içermeyen TempPSO varyantının sözde kodu*

1: Pozisyon değerlerini uniform dağılım kullanarak ata
2: Hız değerlerini "sıfır" olarak ata
3: c2 = 1.485
4: **while** sonlandırma kriteri sağlanıncaya kadar **do**
5: **for** her bir parçacık i **do**
6: $V_{i+1} = V_i + rand(0,1)(gbest - L_i) + c2 * rand(0,1)(pbest_R - L_i)$
7: Hız değerlerini "absorb" stratejisi ile sınırlandır
8: $P_{i+1} = V_i + pbest_i$
9: Pozisyon değerlerini "random" stratejisi ile sınırlandır
10: Öğrenme faktörü 2'yi "doğrusal" strateji ile güncelle
11: Kişisel en iyileri güncelle
12: Global en iyiyi güncelle
13: iteration = iteration + 1

Algoritma 5.5. *irace aracı ile CEC'17 için üretilen yerel arama stratejisi içeren TempPSO varyantının sözde kodu*

1: Pozisyon değerlerini henon kaotik harita ile ata
2: Hız değerlerini "sıfır" olarak ata
3: c2 = 2.8668 c1 = 2.1594
4: $lf2_{min} = 1.2235$ $lf2_{max} = 0.6971$
5: **while** sonlandırma kriteri sağlanıncaya kadar **do**
6: **for** her bir parçacık i **do**
7: $V_{i+1} = \omega V_i + c1 * rand(0,1)(e_{GGLPSOD} - L_i) + c2 * rand(0,1)(pbest_{R1} - pbest_{R2})$
8: Hız değerlerini "boundary" stratejisi ile sınırlandır
9: $P_{i+1} = V_i + L_i + pbest_i$
10: Pozisyon değerlerini "pbest" stratejisi ile sınırlandır
11: Öğrenme faktörü 2'yi "doğrusal" strateji ile güncelle
12: Atalet ağırlığını "doğrusal" strateji ile güncelle
13: GLPSO yenilenme stratejisini uygula
14: Global ve Kişisel en iyileri güncelle
15: Rekabetçi yerel arama stratejisi uygula
16: iteration = iteration + 1

5.2.1.2. *Algoritmanın zaman karmaşıklığı*

CEC 2017 ölçüt seti kullanılarak TempPSO algoritma karmaşıklığı Awad vd.'nin belirttiği gibi hesaplanmıştır (Awad vd., 2016) ve Tablo 5.8'de verilmiştir. Çalışma zamanı CEC 2017 ölçüt setinde yer alan f18 fonksiyonu çalıştırılarak hesaplanmıştır. T0 zamanı aşağıdaki program çalıştırılarak hesaplamıştır:

```
x= 0.55;
for i=1:1000000
x=x + x; x=x/2; x=x*x;
x=sqrt(x); x=log(x); x=exp(x); x=x/(x+2);
end
```

T1 zamanı, D boyut için f18 fonksiyonunun 200000 defa çalıştırılması ile hesaplanmıştır. T2' TempPSO ile f18'in çözülmesine harcanan ortalama zamanı göstermektedir.

Tablo 5.8. *TempPSO CEC'17 için algoritma karmaşıklığı sonuçları. Sonuçları saniye olarak verilmiştir.*

Boyut	T0 (s)	T1 (s)	T2' (s)	(T2' – T1)/T0
D=10	0.016	0.109	0.509	25.000
D=30	0.016	0.344	1.378	64.612
D=50	0.016	0.797	2.643	115.400
D=100	0.016	2.844	7.962	319.887

5.2.1.3. *PSO varyantları ile karşılaştırma*

Önerilen algoritmanın performans karşılaştırma sonuçları bu bölümde yer almaktadır. Deneylerde, PSO algoritmasının temel yapısına benzer özellik gösteren CLPSO (Liang vd., 2006), FrankPSO (Montes de Oca vd., 2009), GGLPSOD (Lin vd., 2018), GLPSO (Gong vd., 2016), ELPSO (Huang vd., 2012), FIPS (Mendes vd., 2004), PSOk (García-Nieto ve Alba, 2014) ve PSOTD (Y. Chen vd., 2017) algoritmaları yer almaktadır.

TempPSO, TempPSO-ls ve PSO varyantları CEC'17 ölçüt setinin her bir fonksiyonu için 51 defa bağımsız olarak çalıştırılmışlardır. Deneyler 30 ve 50 boyut için gerçekleştirilmiştir. Her çalıştırmanın sonucunda elde edilen ortalama hata değerleri 30 boyut için Tablo 5.9'de verilmiştir. 50 boyut ortalama hata değerleri de Tablo 5.10'de listelenmiştir. Tablolar incelendiğinde, tabloların dört kısımdan oluştuğu görülmektedir. İlk kısımda, algoritmaların 30 fonksiyonun her biri için elde ettiği ortalama hata değerleri yer almaktadır. İkinci kısımda, TempPSO algoritmasının diğer PSO varyantları ile yapılan karşılaştırılma bilgileri bulunmaktadır. Bu bilgilerden ilki, algoritmaların bütün fonksiyonlarda elde ettiği sıralamanın ortalama değerini gösteren "sıralama (rank)" değeridir. Bir diğeri ise, TempPSO ile PSO varyantlarının ikili karşılaştırılması sonucu elde edilen "kazanma (win)", "kaybetme (lost)" ve "berabere (draw)" sayılarıdır. Bu kısımda yer alan son bilgi ise, TempPSO ve PSO varyantlarının ikili yapılan wilcoxon testi sonucunda elde edilen "*p-value*" değeridir. Üçüncü kısımda, TempPSO-ls ve PSO varyantları ile yapılan karşılaştırmanın bilgileri yer almaktadır. Yine burada da "sıralama (rank)", "kazanma (win)", "kaybetme (lost)", "berabere (draw)" ve "*p*-value" değerleri vardır. Son kısımda ise, TempPSO, TempPSO-ls ve PSO varyantlarının tamamının

katılımıyla bütün fonksiyonlarda elde edilen sıralamaların ortalama sıralama bilgisi yer almaktadır.

Tablo 5.9'de, deneylerde yer alan algoritmaların 30 boyutta elde ettikleri ortalama hata değerleri listelenmektedir. f4, f11, f22, f25, f27 ve f30 fonksiyonları hariç kalan problemlerin tamamında TempPSO-ls diğer algoritmalara karşı üstün gelmiştir. f6 fonksiyonunda, TempPSO-ls ile FIPS algoritmaları optimum değeri bulmuşlardır. Ayrıca TempPSO, multimodal fonksiyon f4 ve hibrit fonksiyon f11'de en iyi algoritma olmuştur. Bununla birlikte, FrankPSO algoritması kompozit fonksiyonlarda iyi performans göstermiştir. FrankPSO; f22, f25, f27 ve f30 fonksiyonlarda ilk sırayı almıştır. Ayrıca; TempPSO-ls karşılaştırmaya dahil edilmeden sadece TempPSO ile diğer PSO varyantları karşılaştırılmıştır. Buna göre; TempPSO'nun, 2.07 ortalama sıralama değeri ile ilk sırada yer aldığı ve istatistiksel testlere göre de CLPSO ve FrankPSO hariç diğer algoritmalardan daha iyi olduğu görülmüştür. CLPSO ve FrankPSO algoritmalarına göre ise sonuçlarında bir farklılık olmadığı söylenebilir. Bu testlere ek olarak; TempPSO-ls, TempPSO olmadan sadece diğer PSO varyantları karşılaştırılmış ve sonuç olarak 1.23 ortalama sıralama değerini elde ettiği görülmüştür. Wilcoxon testine göre de bütün algoritmalara karşı üstün geldiği söylenebilir. Son olarak, TempPSO, TempPSO-ls ile diğer PSO varyantlarının sonuçlarının ortalama sıralaması yapıldığında ilk sırayı TempPSO-ls'nin ve ikinci sırayı da TempPSO'nun aldığı görülmüştür.

Tablo 5.9. *CEC'17 30 boyut PSO varyantları ile karşılaştırma sonuçları*

Fonk.	TempPSO-ls	TempPSO	GLPSO	GGLPSOD	ELPSO	FIPS	CLPSO	PSO	PSOTD	FrankPSO	PSOk
f1	**1.00E-08**	1.36E+03	6.62E+03	4.50E+03	3.86E+03	1.35E+03	7.75E+02	1.24E+10	3.90E+02	3.81E+03	4.18E+03
f2	**1.00E-08**	1.42E-04	2.22E+00	8.77E+01	1.50E+09	1.27E-03	3.12E+05	1.41E+43	2.37E+12	8.91E+06	2.31E+14
f3	**1.00E-08**	3.34E+00	1.12E+04	9.64E+03	7.80E+02	1.19E+04	9.76E+02	7.07E+04	1.28E+05	2.47E+01	1.64E+04
f4	5.86E+01	**2.82E+01**	6.53E+01	7.10E+01	6.30E+01	7.38E+01	6.47E+01	2.24E+03	8.06E+01	3.05E+01	9.27E+01
f5	**3.54E+00**	2.66E+01	5.06E+01	6.25E+01	1.71E+02	1.15E+02	6.32E+01	2.64E+02	9.13E+01	3.20E+01	1.70E+01
f6	**1.00E-08**	2.09E-05	3.33E-03	4.43E-02	1.26E-02	**1.00E-08**	3.39E-05	6.00E+01	4.78E-06	4.07E-04	3.37E-02
f7	**3.57E+01**	5.97E+01	9.37E+01	9.29E+01	2.06E+02	1.78E+02	9.38E+01	4.80E+02	1.33E+02	9.32E+01	5.41E+01
f8	**3.90E+00**	2.63E+01	5.53E+01	6.12E+01	1.74E+02	1.09E+02	6.31E+01	2.26E+02	9.51E+01	2.62E+01	1.47E+01
f9	**1.00E-08**	4.18E-01	2.20E+01	1.41E+02	3.07E+00	2.10E+02	1.94E+02	6.54E+03	2.14E+03	5.14E-01	6.41E-02
f10	**2.06E+02**	2.67E+03	2.92E+03	2.70E+03	6.64E+03	6.23E+03	2.79E+03	5.27E+03	3.31E+03	4.42E+03	1.30E+03
f11	4.16E+01	**1.77E+01**	6.02E+01	5.30E+01	8.98E+01	8.22E+01	5.01E+01	1.34E+03	7.63E+02	3.16E+01	9.53E+01
f12	**1.09E+03**	1.19E+04	6.92E+04	1.40E+05	1.43E+05	4.36E+05	3.60E+05	9.92E+08	1.62E+06	6.39E+04	1.10E+05
f13	**2.15E+01**	8.79E+03	1.87E+04	2.66E+04	1.31E+04	2.27E+04	2.47E+03	2.99E+08	2.01E+04	9.99E+03	1.84E+04
f14	**1.13E+02**	1.16E+03	1.61E+04	2.26E+04	3.50E+04	1.06E+04	2.19E+04	1.07E+05	2.13E+05	7.53E+03	8.03E+03
f15	**2.61E+01**	4.00E+03	1.05E+04	4.86E+03	4.64E+03	9.34E+03	2.42E+02	1.48E+05	3.93E+03	1.67E+03	2.78E+03
f16	**1.75E+01**	4.01E+02	8.33E+02	9.78E+02	1.01E+03	8.92E+02	6.35E+02	1.78E+03	8.75E+02	5.17E+02	3.93E+02
f17	**5.01E+01**	7.90E+01	2.65E+02	3.12E+02	1.87E+02	2.27E+02	1.52E+02	7.13E+02	3.63E+02	1.59E+02	1.42E+02
f18	**8.94E+01**	5.09E+04	1.66E+05	1.84E+05	7.00E+05	1.82E+05	1.62E+05	1.13E+06	5.78E+05	1.87E+05	1.18E+05
f19	**8.98E+01**	4.59E+03	7.92E+03	1.17E+04	5.36E+03	4.71E+03	4.58E+02	2.70E+06	2.96E+03	4.58E+03	5.22E+03
f20	**3.64E+01**	1.36E+02	3.06E+02	3.33E+02	1.77E+02	3.14E+02	2.19E+02	5.29E+02	3.94E+02	2.04E+02	1.72E+02
f21	**2.05E+02**	2.28E+02	2.55E+02	2.62E+02	3.65E+02	3.19E+02	2.61E+02	4.60E+02	2.96E+02	2.25E+02	2.23E+02
f22	1.05E+02	1.42E+02	7.56E+02	1.70E+03	1.10E+03	3.33E+02	5.51E+02	4.07E+03	1.11E+03	1.00E+02	**1.00E+02**
f23	**3.49E+02**	3.73E+02	4.14E+02	4.28E+02	5.00E+02	4.70E+02	4.17E+02	9.45E+02	4.47E+02	3.78E+02	4.33E+02
f24	**4.22E+02**	4.46E+02	5.00E+02	5.18E+02	5.88E+02	5.63E+02	5.16E+02	1.05E+03	5.63E+02	4.77E+02	5.17E+02
f25	3.87E+02	3.87E+02	3.91E+02	3.91E+02	3.93E+02	3.87E+02	3.87E+02	8.80E+02	3.87E+02	**3.79E+02**	3.88E+02
f26	**7.83E+02**	1.15E+03	1.68E+03	1.63E+03	1.80E+03	1.80E+03	1.36E+03	4.34E+03	1.87E+03	8.57E+02	1.60E+03
f27	5.13E+02	5.14E+02	5.23E+02	5.20E+02	5.25E+02	5.14E+02	5.22E+02	7.37E+02	5.21E+02	**4.56E+02**	5.36E+02
f28	**3.04E+02**	3.52E+02	3.60E+02	3.88E+02	3.41E+02	3.33E+02	4.11E+02	1.45E+03	4.22E+02	3.95E+02	4.36E+02
f29	**4.38E+02**	4.81E+02	7.25E+02	7.38E+02	5.77E+02	7.75E+02	6.22E+02	1.54E+03	7.65E+02	4.75E+02	6.40E+02
f30	2.03E+03	5.06E+03	6.76E+03	6.53E+03	7.28E+03	2.79E+04	5.59E+03	3.51E+07	9.61E+03	**1.16E+03**	6.49E+03

Tablo 5.9. (Devam) *CEC'17 30 boyut PSO varyantları ile karşılaştırma sonuçları*

	TempPSO-ls	TempPSO	GLPSO	GGLPSOD	ELPSO	FIPS	CLPSO	PSO	PSOTD	FrankPSO	PSOk
rank Temp-PSO vs. pso varyantlar		**2.07**	5.40	6.17	6.73	6.13	4.27	9.87	6.97	2.97	4.43
(win, lost, draw) Temp-PSO vs. pso varyantlar			30, 0, 0	30, 0, 0	29, 1, 0	26, 4, 0	26, 4, 0	30, 0, 0	26, 4, 0	20, 10, 0	21, 9, 0
(*p*-value) Temp-PSO vs. pso varyantlar			0.000002	0.000002	0.000003	0.000008	0.005320	0.000002	0.000222	0.054463	0.002585
rank Temp-PSO-ls vs. pso varyantlar	**1.23**		5.40	6.17	6.77	6.23	4.40	9.87	7.10	3.10	4.70
(win, lost, draw) Temp-PSO-ls vs. pso varyantlar			30, 0, 0	30, 0, 0	30, 0, 0	29, 0, 1	30, 0, 0	30, 0, 0	30, 0, 0	24, 6, 0	29, 1, 0
(*p*-value) Temp-PSO-ls vs. pso varyantlar			0.000002	0.000002	0.000002	0.000003	0.000002	0.000002	0.000002	0.000283	0.000003
rank tüm algoritmalar	**2.63**	3.4	6.37	7.77	7.5	5.07	3.53	9.53	6.3	5.9	6.63

Tablo 5.10'de, TempPSO algoritmaları ile PSO varyantlarının CEC'17 ölçüt seti 50 boyut için ortalama hata değerleri yer almaktadır. Tablonun tamamı incelendiğinde TempPSO-ls, 30 ölçüt fonksiyonunun büyük çoğunluğunda diğer algoritmalara karşı üstün geldiği söylenebilir. f11'de, TempPSO ilk sırada yer almıştır. Bununla birlikte, kompozit fonksiyonlar olan f25, f27, f28 ile f30'da FrankPSO'nun yine en iyi algoritma olduğu görülmüştür. Tablo 5.10'e göre, sadece PSO varyantları ile TempPSO (TempPSO-ls hariç) karşılaştırma sonuçlarında TempPSO, 2.13'lük sıralama değeri ile ilk sırada yer almıştır. *p*-value değerine göre ise FrankPSO algoritması hariç diğerlerinden daha iyi sonuçlar aldığı söylenebilir. Buna ek olarak, TempPSO-ls (TempPSO hariç) ile PSO varyantlarının karşılaştırma sonuçları incelendiğinde TempPSO-ls, 1.23 sıralama değeri ile ilk sırayı aldığı ve istatiksel teste göre de bütün algoritmalara üstün geldiği söylenebilir. Böylelikle yerel arama teknikleri ile melezleştirmenin performansa önemli ölçüde katkı sağladığı görülmüştür. Son olarak bütün algoritmaların sıralamasına bakıldığında TempPSO-ls, 1.40'lık sıralama değeri ile birinci, 3.07 sıralama değeriyle de TempPSO ikinci olmuştur. Buradan; TempPSO çatısı ile üretilen iki algoritmanın, diğer geliştirilmiş PSO varyantı algoritmalara göre üstün performans sergilemesi, üretilen algoritma çatısının gücünü ve etkinliğini göstermektedir.

Tablo 5.10. *CEC'17 50 boyut PSO varyantları ile karşılaştırma sonuçları*

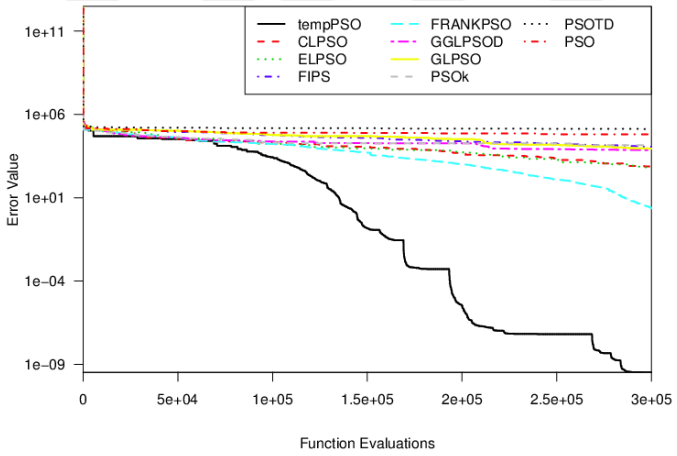
Fonk.	TempPSO-ls	TempPSO	GLPSO	GGLPSOD	ELPSO	FIPS	CLPSO	PSO	PSOTD	FrankPSO	PSOk
f1	**1.00E-08**	3.92E+03	6.68E+03	9.66E+03	3.96E+03	1.93E+03	1.58E+03	3.76E+10	3.20E+03	3.51E+03	2.81E+04
f2	**5.71E-06**	1.99E-04	3.85E+02	1.53E+19	1.36E+09	6.93E+01	2.54E+14	3.30E+71	6.57E+24	7.99E+17	1.95E+29
f3	**1.00E-08**	2.18E+03	6.54E+04	3.03E+04	1.84E+04	5.10E+04	5.37E+03	1.66E+05	2.63E+05	1.78E+03	6.87E+04
f4	**3.47E+01**	1.24E+02	8.58E+01	1.17E+02	1.45E+02	7.63E+01	7.94E+01	5.41E+03	1.10E+02	4.58E+01	1.53E+02
f5	**7.11E+00**	5.26E+01	1.16E+02	1.29E+02	3.64E+02	2.87E+02	1.58E+02	5.00E+02	2.16E+02	5.28E+01	4.37E+01
f6	**1.00E-08**	4.52E-05	3.87E-02	2.40E-01	3.19E-02	1.07E-04	1.02E-06	7.26E+01	2.82E-04	5.82E-03	3.02E-01
f7	**6.01E+01**	1.07E+02	1.98E+02	2.01E+02	4.20E+02	4.01E+02	2.28E+02	1.10E+03	3.03E+02	1.82E+02	7.35E+01
f8	**7.20E+00**	5.29E+01	1.15E+02	1.34E+02	3.69E+02	2.78E+02	1.63E+02	5.09E+02	2.17E+02	5.97E+01	4.75E+01
f9	**1.00E-08**	1.15E+00	4.76E+02	1.71E+03	1.39E+01	3.96E+03	3.03E+03	2.44E+04	1.18E+04	8.95E+00	5.42E+01
f10	**2.21E+02**	5.54E+03	5.45E+03	4.79E+03	1.27E+04	1.19E+04	6.82E+03	9.91E+03	6.08E+03	8.00E+03	2.78E+03
f11	1.41E+02	**3.63E+01**	1.05E+02	1.11E+02	4.25E+02	1.76E+02	1.16E+02	6.15E+03	1.77E+03	7.71E+01	1.86E+02
f12	**2.89E+03**	6.34E+04	5.91E+05	1.73E+06	8.98E+05	1.99E+06	1.09E+06	1.46E+10	1.10E+07	8.98E+05	2.51E+07
f13	**1.83E+03**	2.80E+03	9.33E+03	1.38E+04	3.72E+03	1.75E+04	2.50E+03	6.26E+09	1.20E+04	2.68E+03	1.21E+04
f14	**2.10E+02**	6.98E+03	5.27E+04	1.01E+05	4.99E+04	6.08E+04	1.82E+05	2.13E+06	1.59E+06	2.92E+04	4.08E+04
f15	**4.47E+02**	3.25E+03	8.80E+03	1.01E+04	5.99E+03	9.98E+03	3.39E+03	2.71E+07	9.32E+03	9.27E+03	5.01E+03
f16	**1.94E+02**	6.44E+02	1.53E+03	1.57E+03	1.76E+03	1.71E+03	1.27E+03	3.11E+03	1.62E+03	9.44E+02	7.58E+02
f17	**3.18E+02**	5.32E+02	1.06E+03	1.07E+03	1.48E+03	1.21E+03	8.25E+02	2.12E+03	1.14E+03	8.87E+02	6.42E+02
f18	**2.06E+02**	4.30E+04	1.56E+05	3.66E+05	2.63E+06	9.88E+05	5.12E+05	5.48E+06	3.07E+06	3.89E+05	2.87E+05
f19	**9.46E+01**	1.30E+04	1.61E+04	1.67E+04	1.43E+04	1.12E+04	5.12E+03	8.59E+07	4.66E+03	1.55E+04	1.43E+04
f20	**7.14E+01**	3.92E+02	8.24E+02	8.49E+02	1.13E+03	1.07E+03	7.17E+02	1.33E+03	9.74E+02	4.88E+02	2.49E+02
f21	**2.09E+02**	2.53E+02	3.17E+02	3.33E+02	5.61E+02	4.80E+02	3.57E+02	8.00E+02	4.28E+02	2.52E+02	2.68E+02
f22	**4.14E+02**	5.18E+03	6.02E+03	5.56E+03	1.15E+04	1.08E+04	6.65E+03	1.01E+04	6.42E+03	2.62E+03	1.49E+03
f23	**4.26E+02**	4.79E+02	5.76E+02	5.96E+02	7.92E+02	7.40E+02	5.99E+02	1.71E+03	6.60E+02	4.93E+02	6.52E+02
f24	**4.98E+02**	5.53E+02	6.39E+02	7.01E+02	8.66E+02	8.69E+02	7.53E+02	1.94E+03	8.46E+02	6.06E+02	7.95E+02
f25	4.81E+02	5.24E+02	5.36E+02	5.43E+02	5.90E+02	5.52E+02	5.70E+02	3.36E+03	5.35E+02	**4.34E+02**	5.53E+02
f26	**8.96E+02**	1.62E+03	2.70E+03	2.74E+03	3.47E+03	3.94E+03	2.93E+03	9.33E+03	3.52E+03	1.29E+03	2.90E+03
f27	5.70E+02	5.58E+02	6.75E+02	6.79E+02	6.71E+02	6.35E+02	6.72E+02	1.66E+03	6.89E+02	**4.71E+02**	7.65E+02
f28	4.59E+02	4.74E+02	4.96E+02	4.90E+02	5.63E+02	4.98E+02	5.56E+02	3.67E+03	5.62E+02	**4.45E+02**	5.36E+02
f29	**4.60E+02**	4.61E+02	1.14E+03	1.03E+03	6.44E+02	1.18E+03	9.08E+02	4.15E+03	1.11E+03	5.64E+02	1.19E+03
f30	8.43E+05	9.38E+05	9.47E+05	1.00E+06	1.74E+06	1.83E+06	8.41E+05	2.89E+08	8.23E+05	**2.43E+03**	9.39E+05

Tablo 5.10. (Devam) *CEC'17 50 boyut PSO varyantları ile karşılaştırma sonuçları*

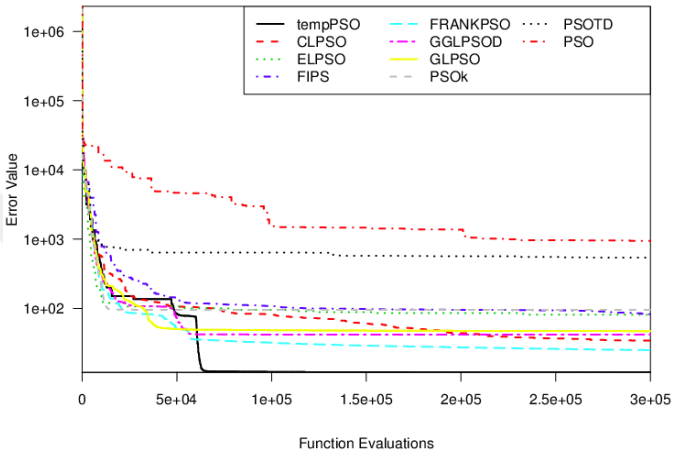
	TempPSO-ls	TempPSO	GLPSO	GGLPSOD	ELPSO	FIPS	CLPSO	PSO	PSOTD	FrankPSO	PSOk
rank Temp-PSO vs. pso varyantlar		**2.13**	4.60	5.67	6.97	6.67	4.67	9.83	6.57	2.87	5.03
(win, lost, draw) Temp-PSO vs. pso varyantlar			28, 2, 0	28, 2, 0	30, 0, 0	27, 3, 0	24, 6, 0	30, 0, 0	26, 4, 0	19, 11, 0	24, 6, 0
(p-value) Temp-PSO vs. pso varyantlar			0.000008	0.000011	0.000002	0.000082	0.003609	0.000002	0.000616	0.228880	0.000771
rank Temp-PSO-ls vs. pso varyantlar	**1.33**		4.63	5.70	6.97	6.77	4.80	9.83	6.67	3.07	5.23
(win, lost, draw) Temp-PSO-ls vs. pso varyantlar			29, 1, 0	29, 1, 0	30, 0, 0	30, 0, 0	28, 2, 0	30, 0, 0	29, 1, 0	25, 5, 0	30, 0, 0
(p-value) Temp-PSO-ls vs. pso varyantlar			0.000002	0.000002	0.000002	0.000002	0.000015	0.000002	0.000020	0.000359	0.000002
rank tüm algoritmalar	**1.4**	3.07	5.57	6.63	7.97	7.67	5.6	10.83	7.53	3.7	6.03

CEC'17 sonuçlarından 30 boyut ve 50 boyutta her bir problem türünden birer fonksiyon seçilmiştir ve bunların yakınsama grafiklerine yer verilmiştir. 30 boyutta unimodal için f3, multimodal için f11, hibrit için f18 ve kompozit içinse f24 seçilmiştir. Grafiklerde algoritmaların her bir fonksiyon için 51 defa çalıştırma sonucunda elde edilen medyan sonuçlarının yakınsama davranışları gösterilmiştir. 30 boyut yakınsama grafikleri Şekil 5.2'de yer almaktadır. Dört fonksiyonda da TempPSO, diğer algoritmalardan daha iyi hata değerine en kısa sürede ulaşmıştır. f3'de 2E+05 FEs değerinde TempPSO optimum değere ulaştığı görülmektedir. Buna karşın bu FEs değerinde diğer algoritmalar arama işlemine devam etmektedirler. f11 fonksiyonunda TempPSO, 6E+04 FEs değerinde aramasını tamamlayarak en iyi değere ulaşmıştır. f18 fonksiyonunda önerilen algoritma, 2E+05 FEs'de en hızlı ve en iyi sonucu elde eden algoritma olmuştur. Son olarak, f24 fonksiyonunda FrankPSO ile TempPSO en iyi değerleri almıştır. TempPSO 5E+04 FEs'de aramasını tamamlarken, FrankPSO bu değere 2.5E+05 FEs'de ancak ulaştığı görülmektedir.

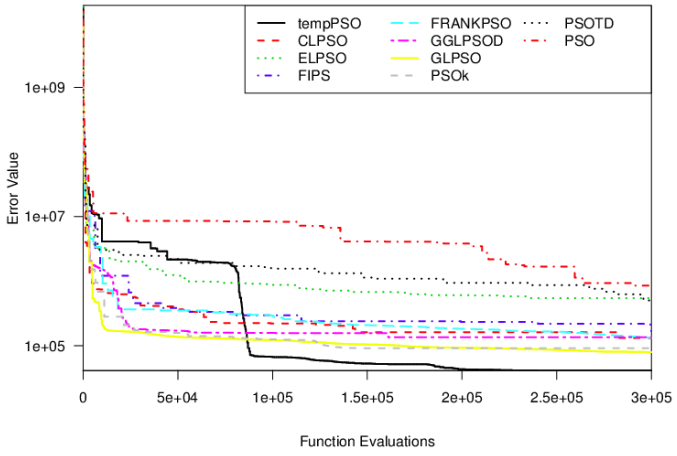
50 boyutta unimodal için f2, multimodal için f12, hibrit için f17 ve kompozit için ise f29 fonksiyonlarının yakınsama grafiklerinin çizdirilmesi için seçilmişlerdir. 50 boyut yakınsama grafikleri Şekil 5.3'de yer almaktadır. Seçilen fonksiyonların hepsinde en hızlı yakınsama eğrisine sahip olan algoritmanın TempPSO olduğu görülmüştür. f2 fonksiyonunda bütün algoritmalar benzer davranış göstererek, 1E+05 FEs değerine kadar aynı değere aynı yakınsama hızında ulaşmışlardır. 2.5E+05 FEs değerinde TempPSO en iyi değere ulaşmıştır. f12'de yine 1E+05 FEs değerine kadar bütün algoritmalar aynı değere aynı yakınsama hızında varmışlardır. Buna ek olarak, 4E+05 FEs'de TempPSO aramayı tamamlamıştır. Diğerlerinin ise bu FEs değerinde hala daha aramaya devam ettiği görülmektedir. Hibrit fonksiyon olan f17'de bütün algoritmalar yakın sonuçlar elde etmekle birlikte, en iyi sonuca TempPSO 2E+05 FEs değerinde ulaşmıştır. Kompozit fonksiyon f29'da FrankPSO ve TempPSO yine birbirine yakın değerlere ulaşmışlardır. TempPSO bu değere, FrankPSO'dan daha hızlı yakınsayarak 1E+05 FEs'te ulaşmıştır.



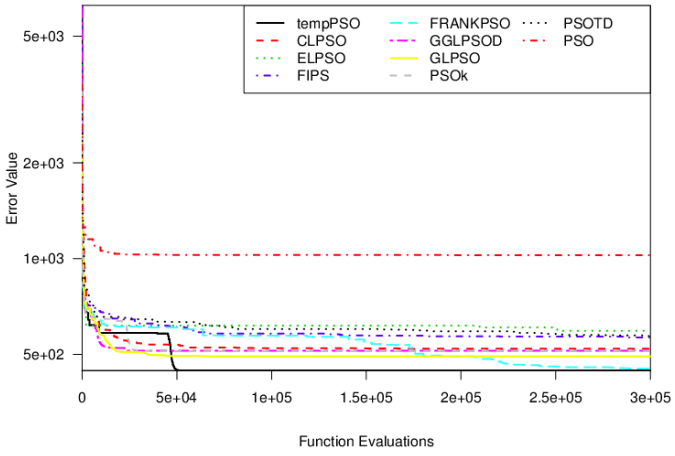
(f3)



(f11)

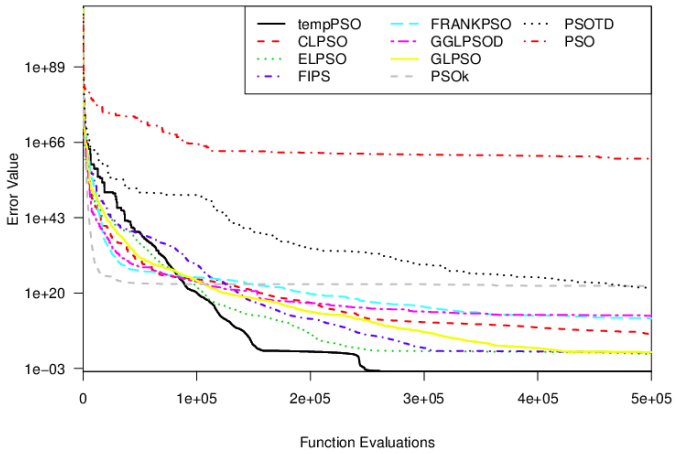


(f18)

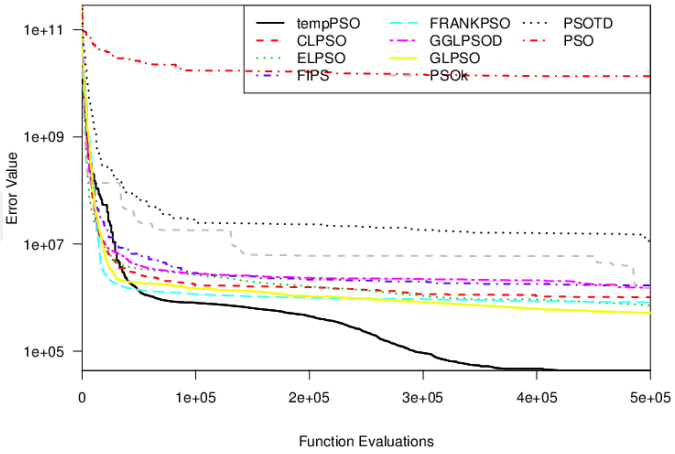


(f24)

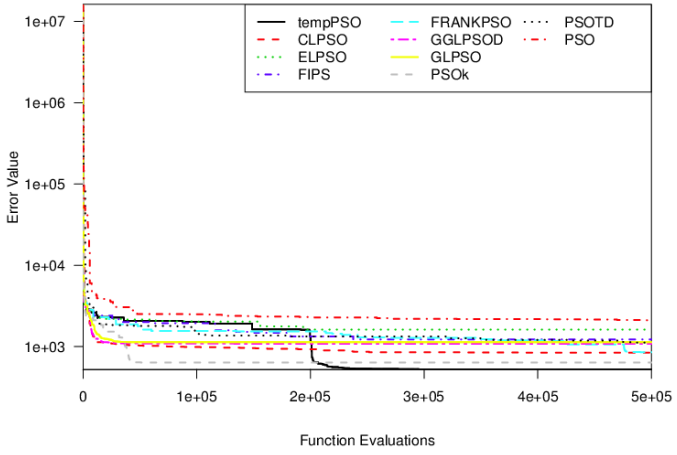
Şekil 5.2. *Unimodal fonksiyon olan 3, multimodal fonksiyon 11, hibrit fonksiyon 18 ve kompozit fonksiyon 24 için 30 boyutta yakınsama grafikleri*



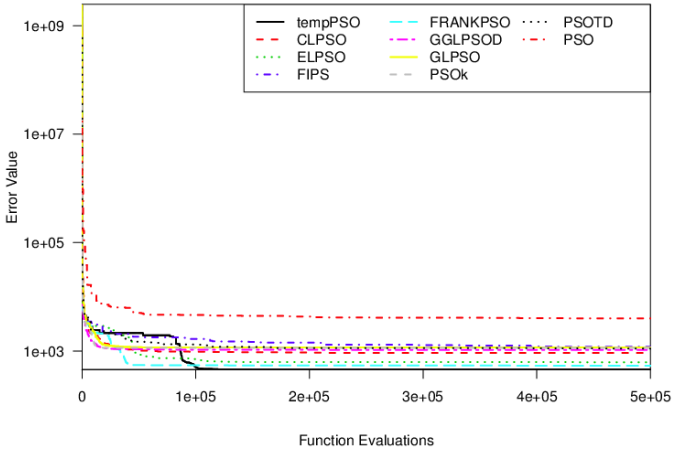
(f2)



(f12)



(f17)



(f29)

Şekil 5.3. *Unimodal fonksiyon olan 2, multimodal fonksiyon 12, hibrit fonksiyon 17 ve kompozit fonksiyon 29'un 50 boyutta yakınsama grafikleri*

5.2.1.4. *CEC'17 yarışmacıları ile karşılaştırma*

TempPSO-ls'nin performansı, CEC'17 yarışmasına katılan algoritmalar ile karşılaştırılmıştır. Testlerde yer alan CEC'17 yarışmacıları jSO (Brest vd., 2017), LSHADE-cnEpsin (Awad vd., 2017), LSHADE_SPACMA (Mohamed vd., 2017), MM_OED (Sallam vd., 2017), IDEbestNsize (Bujok ve Tvrdik, 2017), RB-IPOP-CMA-ES (Biedrzycki, 2017), DES (Jagodzinski ve Arabas, 2017), DYYPO (Maharana vd., 2017), TLBO-FL (Kommadath ve Kotecha, 2017), PPSO (Tangherloni vd., 2017) ve MOS-SOCO2011 (LaTorre ve Pena, 2017)'dir.

TempPSO-ls ile CEC'17 yarışmacılarının ortalama hata değerleri 30 boyut ve 50 boyut için karşılaştırılmıştır. 30 boyuttaki karşılaştırma sonuçları Tablo 5.11'te verilmiştir. 50-boyut karşılaştırma sonuçları ise Tablo 5.12'da görülmektedir. CEC'17 yarışmacı algoritmalarının ortalama hata değerleri kendi makalelerinden temin edilmiştir. 30 boyut sonuçları yani Tablo 5.11 incelendiğinde, JSO 13 adet, LSHADE_SPACMA 10 adet, MM_OED 9 adet, LSHADE-cnEpSin 8 adet, TempPSO-ls ve RB-IPOP-CMA-ES 7 adet, IDEbestNsize 5 adet, DES 4 adet, DYYPO, TLBO-FL, PPSO ve MOS-SOCO 1 adet fonksiyonda ilk sırayı elde ettiği görülmüştür. Ortalama sıralama değerlerine göre, 2.9'luk sıralama ile LSHADE_SPACMA birinci, 3 sıralama değeri ile jSO ikinci, 3.3 sıralama değeri ile LSHADE-cnEpSin üçüncü, 3.73 sıralama değeri ile MM_OED dördüncü olmuştur. Son olarak, önerilen TempPSO-ls algoritması da 4.13'lük değeri ile beşinci olmuştur.

50 boyut sonuçları Tablo 5.12'da yer almaktadır. Tabloda yer alan ortalama hata değerlerine göre 2.6'lık sıralama değerine sahip LSHADE_SPACMA en iyi algoritma olmuştur. Bunula birlikte, 3.10'luk sıralama değeri ile jSO ikinci 4.03'lük değer ile de TempPSO-ls üçüncü olmuştur. Buradan elde edilen sonuca göre, boyut büyüdükçe TempPSO-ls algoritmasının sonuçlarının iyileşmesi algoritmanın ölçeklenebilirlik yeteneğini göstermektedir. 50 boyutta, önerilen algoritma 30 boyuta göre daha iyi sonuçlar alarak üçüncü olmuştur. Bununla birlikte, wilcoxon sıralama toplam testi 0.05 anlamlılık seviyesinde TempPSO-ls, LSHADE_SPACMA ve jSO algoritmalarının sonuçlarına benzer sonuçlar elde etmiştir.

Tablo 5.11. *CEC'17 30 boyut CEC'17 yarışmacıları karşılaştırma sonuçları*

Fonk.	TempPSO-ls	jSO	LSHADE-cnEpSin	LSHADE_SPACMA	MM_OED	IDEbestNsize	RB IPOP CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS-SOCO2011
f1	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	3.154E-08	3.851E-08	3.700E+03	3.500E+03	7.480E+02	2.190E+03
f2	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	1.590E-04	**1.000E-08**	**1.000E-08**	3.200E+09	8.500E+16	5.320E+01	2.110E-02
f3	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	4.052E+00	**1.000E-08**	**1.000E-08**	5.300E+02	3.000E+03	1.130E+00	1.720E+03
f4	5.856E+01	5.867E+01	4.228E+01	**1.000E-08**	1.167E+01	2.417E+00	5.528E+01	5.856E+01	9.100E+01	9.000E+01	4.390E+01	3.890E+01
f5	3.545E+00	8.557E+00	1.225E+01	3.690E+00	4.234E+00	2.269E+01	**1.649E+00**	6.945E+00	9.100E+01	4.000E+01	1.120E+02	5.580E+01
f6	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	1.208E-07	3.826E-07	8.600E-01	4.900E-01	2.030E+01	**1.000E-08**
f7	3.571E+01	3.893E+01	4.330E+01	**3.380E+01**	3.439E+01	5.154E+01	3.433E+01	3.551E+01	1.400E+02	1.400E+02	1.350E+02	8.670E+01
f8	3.901E+00	9.092E+00	1.293E+01	3.590E+00	4.565E+00	2.363E+01	**1.756E+00**	6.575E+00	9.600E+01	3.700E+01	8.100E+01	6.360E+01
f9	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	6.500E+02	3.400E+01	1.360E+03	2.830E+02
f10	2.058E+02	1.528E+03	1.388E+03	1.440E+03	2.045E+03	2.017E+03	1.443E+03	**1.722E+02**	2.800E+03	6.700E+03	3.130E+03	2.530E+03
f11	4.156E+01	**3.038E+00**	1.354E+01	3.790E+00	1.226E+01	6.439E+00	4.107E+01	2.560E+01	1.200E+02	8.200E+01	8.430E+01	7.500E+01
f12	1.088E+03	**1.704E+02**	3.725E+02	5.540E+02	1.046E+03	3.448E+03	1.093E+03	1.280E+03	1.500E+06	5.700E+04	2.770E+04	3.730E+04
f13	2.155E+01	**1.484E+01**	1.726E+01	1.520E+01	1.795E+01	3.093E+01	1.187E+02	4.346E+04	9.600E+03	2.000E+04	3.210E+03	1.330E+04
f14	1.125E+02	2.183E+01	**2.157E+01**	2.280E+01	2.308E+01	2.334E+01	9.083E+01	2.188E+01	2.100E+03	7.100E+03	2.320E+03	1.990E+04
f15	2.607E+01	**1.088E+00**	3.240E+00	5.370E+00	5.422E+00	7.579E+00	2.177E+02	1.717E+01	1.100E+04	2.200E+04	2.130E+03	1.090E+04

Tablo 5.11. (Devam) *CEC'17 30 boyut CEC'17 yarışmacıları karşılaştırma sonuçları*

Fonk.	TempPSO-ls	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbestNsize	RB IPOP CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS-SOCO 2011
f16	**1.754E+01**	7.892E+01	2.288E+01	3.770E+01	1.343E+02	1.793E+02	5.016E+02	3.415E+01	6.700E+02	4.900E+02	8.460E+02	7.320E+02
f17	5.011E+01	3.293E+01	**2.860E+01**	3.170E+01	4.668E+01	4.137E+01	1.323E+02	7.496E+01	2.500E+02	1.400E+02	3.310E+02	2.660E+02
f18	8.940E+01	**2.041E+01**	2.109E+01	2.340E+01	2.333E+01	3.215E+01	1.601E+02	2.663E+01	1.200E+05	3.700E+05	6.990E+04	1.360E+05
f19	8.978E+01	**4.503E+00**	5.828E+00	9.450E+00	7.148E+00	9.178E+00	1.146E+02	1.888E+04	1.400E+04	1.100E+04	1.710E+03	9.940E+03
f20	3.636E+01	**2.937E+01**	3.035E+01	8.160E+01	4.546E+01	4.053E+01	2.965E+02	5.757E+01	2.500E+02	2.200E+02	3.480E+02	2.640E+02
f21	2.055E+02	2.093E+02	2.121E+02	2.090E+02	**1.311E+02**	2.248E+02	2.086E+02	2.091E+02	3.000E+02	2.300E+02	3.050E+02	2.620E+02
f22	1.049E+02	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	6.716E+02	1.317E+02	**1.000E+02**	**1.000E+02**	**1.000E+02**	1.400E+03
f23	3.488E+02	3.508E+02	3.562E+02	3.570E+02	3.574E+02	3.678E+02	**3.394E+02**	3.585E+02	4.500E+02	4.000E+02	6.810E+02	4.180E+02
f24	4.221E+02	4.265E+02	4.285E+02	4.290E+02	**3.939E+02**	4.371E+02	4.194E+02	4.296E+02	5.600E+02	4.700E+02	7.390E+02	5.390E+02
f25	3.867E+02	3.867E+02	3.867E+02	**3.780E+02**	3.867E+02	3.869E+02	3.867E+02	3.867E+02	3.900E+02	4.000E+02	3.850E+02	3.860E+02
f26	7.826E+02	9.202E+02	9.486E+02	9.430E+02	9.426E+02	1.054E+03	**3.939E+02**	8.328E+02	2.200E+03	1.400E+03	2.040E+03	1.710E+03
f27	5.134E+02	4.974E+02	5.042E+02	4.970E+02	5.076E+02	**4.962E+02**	5.118E+02	5.103E+02	5.400E+02	5.300E+02	7.080E+02	5.150E+02
f28	**3.041E+02**	3.087E+02	3.152E+02	3.340E+02	3.287E+02	3.166E+02	3.089E+02	3.517E+02	3.900E+02	4.300E+02	3.270E+02	3.390E+02
f29	4.380E+02	4.337E+02	4.346E+02	**3.950E+02**	4.388E+02	4.553E+02	4.933E+02	4.307E+02	7.500E+02	6.200E+02	7.800E+02	6.470E+02
f30	2.033E+03	1.971E+03	1.977E+03	**4.010E+02**	1.999E+03	2.301E+03	2.844E+03	2.414E+06	3.300E+04	2.600E+04	3.320E+03	9.910E+03
rank	4.13	3	3.3	**2.9**	3.73	5.6	5.47	5.93	10.5	9.93	9.63	9.27
Our-win		12	10	9	10	18	17	18	29	29	27	27
Our-lost		13	15	16	15	9	10	9	1	1	3	2
draw		5	5	5	5	3	3	3	0	0	0	1
P-value		0.19854	0.30368	0.15000	0.20870	0.19451	0.03394	0.07357	0.00000	0.00000	0.00000	0.00001

Tablo 5.12. *CEC'17 50 boyut CEC'17 yarışmacıları karşılaştırma sonuçları*

Fonk.	TempPSO-ls	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbestNsize	RB IPOP CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS-SOCO2011
f1	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	2.630E-02	1.131E-07	1.253E-07	6.600E+03	6.100E+05	3.890E+02	1.490E+04
f2	5.706E-06	**1.000E-08**	1.569E+00	**1.000E-08**	**1.000E-08**	9.560E-04	2.768E+05	2.353E-01	5.700E+15	1.600E+39	1.250E+12	1.810E+03
f3	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	2.205E+02	**1.000E-08**	**1.000E-08**	4.700E+01	2.600E+04	8.650E+02	3.360E+04
f4	3.472E+01	2.851E+01	5.140E+01	**5.680E-01**	3.749E+01	4.107E+01	2.959E+01	6.273E+01	1.400E+02	1.900E+02	9.130E+01	2.810E+01
f5	7.106E+00	1.620E+01	2.517E+01	6.220E+00	1.116E+01	5.903E+01	**2.790E+00**	8.701E+00	1.900E+02	9.700E+01	2.010E+02	1.340E+02
f6	**1.000E-08**	3.107E-07	9.157E-07	**1.000E-08**	7.435E-08	9.610E-08	1.632E-07	5.717E-07	3.800E+00	4.500E+00	3.180E+01	**1.000E-08**
f7	6.010E+01	6.664E+01	7.664E+01	5.710E+01	5.885E+01	1.021E+02	**5.663E+01**	5.842E+01	2.600E+02	1.700E+02	2.780E+02	1.810E+02
f8	7.203E+00	1.697E+01	2.632E+01	5.850E+00	1.032E+01	6.008E+01	**2.575E+00**	9.086E+00	1.900E+02	9.300E+01	1.990E+02	1.380E+02
f9	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	1.756E-03	7.776E-02	**1.000E-08**	**1.000E-08**	3.500E+03	1.300E+03	6.060E+03	1.210E+03
f10	**2.211E+02**	3.232E+03	3.200E+03	3.610E+03	3.824E+03	4.216E+03	1.726E+03	7.936E+02	4.800E+03	1.300E+04	5.200E+03	4.730E+03
f11	1.410E+02	2.852E+01	2.139E+01	**9.170E+00**	4.039E+01	3.774E+01	1.832E+02	3.009E+01	1.900E+02	1.700E+02	1.270E+02	1.950E+02
f12	2.889E+03	1.653E+03	**1.475E+03**	1.580E+03	2.139E+03	1.576E+04	2.438E+06	1.992E+03	7.800E+06	9.200E+05	5.520E+05	1.450E+05
f13	1.829E+03	3.726E+01	6.943E+01	**3.530E+01**	4.639E+01	1.622E+02	1.652E+03	1.553E+05	7.600E+03	8.000E+03	8.470E+02	1.090E+04
f14	2.096E+02	2.435E+01	2.652E+01	2.890E+01	3.708E+01	6.272E+01	2.416E+02	**2.278E+01**	2.900E+04	8.600E+04	1.950E+04	3.780E+04
f15	4.470E+02	2.342E+01	2.560E+01	**1.570E+01**	4.086E+01	4.102E+01	5.290E+02	1.244E+06	8.200E+03	6.900E+03	1.190E+03	1.220E+04

Tablo 5.12. (Devam) *CEC'17 50 boyut CEC'17 yarışmacıları karşılaştırma sonuçları*

Fonk.	TempPSO-ls	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbestNsize	RB IPOP CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS-SOCO 2011
f16	1.938E+02	4.773E+02	2.745E+02	4.090E+02	6.774E+02	6.322E+02	8.899E+02	**1.059E+02**	1.300E+03	8.500E+02	1.240E+03	1.400E+03
f17	3.182E+02	2.599E+02	2.071E+02	2.940E+02	4.806E+02	4.625E+02	3.978E+02	**9.648E+01**	8.900E+02	8.500E+02	1.030E+03	9.490E+02
f18	2.055E+02	**2.387E+01**	2.433E+01	3.320E+01	3.859E+01	2.035E+03	3.574E+02	2.621E+01	1.800E+05	1.100E+06	2.090E+05	1.880E+05
f19	9.464E+01	**1.386E+01**	1.741E+01	2.180E+01	4.118E+01	2.331E+01	1.394E+02	1.361E+06	9.600E+03	1.400E+04	8.670E+03	2.420E+04
f20	**7.136E+01**	1.133E+02	1.141E+02	1.630E+02	3.020E+02	2.179E+02	5.468E+02	1.312E+02	6.500E+02	1.000E+03	7.700E+02	7.980E+02
f21	2.091E+02	2.192E+02	2.268E+02	2.150E+02	2.122E+02	2.548E+02	**2.062E+02**	2.104E+02	4.000E+02	2.800E+02	4.330E+02	3.460E+02
f22	4.142E+02	**1.000E+02**	1.595E+03	8.170E+02	6.803E+02	2.837E+03	2.051E+03	5.511E+02	4.800E+03	6.600E+03	5.970E+03	5.070E+03
f23	4.260E+02	4.298E+02	4.393E+02	4.390E+02	4.449E+02	4.787E+02	**4.227E+02**	4.261E+02	6.500E+02	5.700E+02	1.060E+03	5.990E+02
f24	4.984E+02	5.073E+02	5.128E+02	5.130E+02	5.166E+02	5.409E+02	**4.912E+02**	5.090E+02	7.300E+02	6.700E+02	1.080E+03	8.220E+02
f25	4.812E+02	4.802E+02	4.803E+02	**4.630E+02**	4.819E+02	5.407E+02	4.807E+02	4.803E+02	5.200E+02	6.200E+02	5.410E+02	5.120E+02
f26	8.957E+02	1.133E+03	1.203E+03	1.140E+03	1.244E+03	1.651E+03	**6.548E+02**	1.080E+03	3.400E+03	2.900E+03	5.450E+03	2.810E+03
f27	5.701E+02	5.125E+02	5.254E+02	**5.000E+02**	5.401E+02	5.258E+02	6.081E+02	5.354E+02	6.700E+02	8.700E+02	1.460E+03	6.760E+02
f28	4.588E+02	4.589E+02	4.591E+02	**4.400E+02**	4.818E+02	4.870E+02	4.700E+02	4.588E+02	4.800E+02	6.100E+02	4.890E+02	4.860E+02
f29	4.602E+02	3.636E+02	3.529E+02	**2.750E+02**	3.617E+02	4.111E+02	6.690E+02	3.705E+02	9.800E+02	1.000E+03	1.520E+03	9.650E+02
f30	8.429E+05	5.905E+05	6.575E+05	**6.500E+02**	6.635E+05	6.198E+05	6.460E+06	2.065E+06	1.500E+06	1.200E+06	7.810E+05	8.220E+05
rank	4.03	3.10	4.17	**2.60**	5.13	6.87	5.43	4.70	10.03	10.40	10.37	9.43
Our-win		12	15	8	16	22	18	17	30	30	27	27
Our-lost		15	12	18	12	8	10	11	0	0	3	2
draw		3	3	4	2	0	2	2	0	0	0	1
P-value		0.17850	0.48598	0.13734	0.98183	0.08972	0.01148	0.37404	0.00000	0.00000	0.00013	0.00003

5.2.2. SOCO deney sonuçları ve analizleri

Büyük ölçekli optimizasyonda TempPSO algoritmasının performansını ölçmek için, Soft Computing (SOCO) dergisinin özel sayısının ("Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems") ölçüt seti kullanılmıştır (Lozano vd., 2010). SOCO, 19 adet fonksiyondan oluşmaktadır. Bunlardan yedi tanesi unimodal, on iki tanesi ise multimodal'dir. Aynı zamanda, bu fonksiyonların dört fonksiyonu ayrılabilir (separable), geri kalanları ise ayrılamazdır (nonseparable). Tablo 5.13'de kısaca SOCO ölçüt fonksiyonlarının özelliklerini sunmaktadır.

Tablo 5.13. *SOCO ölçüt setinin özellikleri*

id	Fonksiyon ismi	Aralık	Separable	Uni/Multi
f1	Shifted Sphere Function	[-100, 100]	Y	U
f2	Shifted Schwefels Problem 2.21	[-100, 100]	N	U
f3	Shifted Rosenbrocks Function	[-100, 100]	N	M
f4	Shifted Rastrigins Function	[-5, 5]	Y	M
f5	Shifted Griewanks Function	[-600, 600]	N	M
f6	Shifted Ackleys Function	[-32, 32]	Y	M
f7	Schwefels Problem 2.22	[-10, 10]	Y	U
f8	Schwefels Problem 1.2	[-65.536, 65.536]	N	U
f9	Extended f10	[-100, 100]	N	U
f10	Bohachevsky	[-15, 15]	N	U
f11	Schaffer	[-100, 100]	N	U
f12	Hybrid f9 & f1 (25%, 75%)	[-100, 100]	N	M
f13	Hybrid f9 & f3 (25%, 75%)	[-100, 100]	N	M
f14	Hybrid f9 & f4 (25%, 75%)	[-5, 5]	N	M
f15	Hybrid f10 & f7 (25%, 75%)	[-10, 10]	N	M
f16	Hybrid f9 & f1 (50%, 50%)	[-100, 100]	N	M
f17	Hybrid f9 & f3 (75%, 25%)	[-100, 100]	N	M
f18	Hybrid f9 & f4 (75%, 25%)	[-5, 5]	N	M
f19	Hybrid f10 & f7 (75%, 25%)	[-10, 10]	N	M

Deneylerde, TempPSO ile karşılaştırılan tüm PSO algoritmaları her fonksiyon için bağımsız olarak 25 defa çalıştırılmıştır. Maksimum fonksiyon çalıştırma sayısı, $D \times 5000$ olarak alınır; burada *D*, bir fonksiyonun boyutunu göstermektedir. Algoritma, sonlandırma kriterini yerine getirdiğinde hata değeri kaydedilir. Hata değeri $f(x) - f(x^*)$ olarak hesaplanır, burada x aday çözüm ve x^* optimum çözümdür. 10^{-14}'den küçük test sonuçları 10^{-14} olarak kabul edilmiştir. Ayrıca, TempPSO ile karşılaştırılan algoritmaların arasındaki performansın önemini değerlendirmek ve

istatistiksel olarak sağlam sonuçlara varmak için 0.05 anlamlılık düzeyinde iki taraflı Wilcoxon sıra toplamı testi kullanılmıştır. Tüm deneyler 8 GB RAM ve Intel Xeon E5-2620 2 GHz işlemciye sahip bir bilgisayarda yapılmıştır.

5.2.2.1. *Parametre ayarları*

TempPSO ile deneylerde yer alan gelişmiş PSO varyantlarının kontrol parametreleri Bölüm 4.2.1'de anlatılmış olan otomatik algoritma yapılandırma aracı irace ile belirlenmiştir. Otomatik yapılandırma işlemi, SOCO'nun 19 probleminin 50 boyutta çalıştırılması ile yapılmıştır. irace aracı varsayılan parametre değerleri ile çalıştırılmıştır. Algoritmaların ayarlanmış parametre değerleri Tablo 5.14'de listelenmiştir. Ayrıca TempPSO parametrelerinin belirlenmesi sonucunda, ortaya çıkan PSO varyantının sözde kodu Algoritma 5.6'de verilmiştir.

Tablo 5.14. *Algoritmaların, SOCO ölçüt seti için ayarlanmış parametre değerleri*

Algoritmalar	Parametre değerleri
TempPSO	lsID = 0 nparticles = 24 iStr = 1 velinitstr = 3 velmaxstr = 1 velclampstr = 2 vu1term = 0 vu2term = 11 vu3term = 7 pm = 0.001 lf1 = 3.1785 lf2 = 1.8172 isc1 = 0 isc2 = 0 lf1upstr = 2 lf2upstr = 2 isconst = 1 posboundstr = 7 pu1term = 2 pu2term = 3 pu3term = 2 isref = 1 refgap = 9 refgapstr = 2
CLPSO	nparticles = 78 iw = 2.8305 min_omega = 0.2403 max_omega = 0.5317 lf1 = 3.8574 refgap = 9
ELPSO	nparticles = 83 iw = 3.4411 refgap = 4 lf1 = 1.2412 lf2 = 3.0496
FrankPSO	nparticles = 16 iw = 2.2442 min_omega = 0.1582 max_omega = 0.7715 lf1 = 2.28 lf2 = 0.3574 iwschedule = 4 tschedule = 5 topostr = 1
GGLPSOD	nparticles = 74 iw = 0.9544 min_omega = 0.3359 max_omega = 0.7121 refgap = 1 pm = 0.005 min_lf1 = 0.7347 max_lf1 = 1.7952 min_lf2 = 1.0672 max_lf2 = 0.1135
GLPSO	nparticles = 22 iw = 1.9323 min_omega = 0.4774 max_omega = 0.6903 lf1 = 2.414 refgap = 8 pm = 0
PSO	nparticles = 70 min_omega = 0.1764 max_omega = 0.8709 lf1 = 1.7743 lf2 = 0.3653
PSOk	nparticles = 22 lf1 = 0.2133 lf2 = 3.9224 topostr = 3 rns = 20

5.2.2.2. *PSO varyantları ile karşılaştırma sonuçları*

TempPSO yüksek boyutlu problemlerdeki davranışının test edilmesi için SOCO ölçüt seti tercih edilmiştir. Deneyler 500 boyut ve 1000 boyut için gerçekleştirilmiştir. TempPSO ile yedi PSO varyantı karşılaştırılmıştır. Deneylerde yer alan algoritmalar, CLPSO (Liang vd., 2006), ELPSO (Huang vd., 2012), FrankPSO (Montes de Oca vd., 2009), GLPSO (Gong vd., 2016), GGLPSOD (Lin vd., 2018) PSO ve PSOk (García-Nieto ve Alba, 2014) algoritmalarıdır.

Algoritma 5.6. *irace aracı ile SOCO için üretilen ve yerel arama içermeyen TempPSO varyantının sözde kodu*

1: Pozisyon değerlerini uniform dağılım kullanarak ata
2: Hız değerlerini "smallrandom" stratejisi kullanarak ata
3: c2 = 1.485
4: **while** sonlandırma kriteri sağlanıncaya kadar **do**
5: **for** her bir parçacık i **do**
6: $V_{i+1} = \chi(rand(0,1)(e_{GGLPSO} - L_i) + 1.8172 * rand(0,1)(gbest - pbest_R))$
7: Hız değerlerini "random" stratejisi ile sınırlandır
8: $P_{i+1} = 2V_i + pbest_i$
9: Pozisyon değerlerini "no action" stratejisi ile sınırlandır
10: CLPSO yenilenme stratejisini uygula
11: Kişisel en iyileri güncelle
12: Global en iyiyi güncelle
13: iteration = iteration + 1

Tablo 5.15'de, SOCO ölçüt setinin 500 boyut için 25 bağımsız çalıştırma sonucunda algoritmaların elde ettiği medyan değerleri yer almaktadır. TempPSO algoritması, 19 problemin tamamında CLPSO, ELPSO, GLPSO ve PSO algoritmalarına karşı en iyi değerleri elde etmiştir. Bununla birlikte, FrankPSO'yı 18, PSOk'yı 16 ve GGLPSOD algoritmasını ise 13 problemde geçmiştir. İstatistiksel testlere göre de GGLPSOD algoritması hariç diğer algoritmalara karşı elde ettiği sonuçların anlamlı olduğu sonucu çıkarılabilir. Fonksiyonların bütününde elde ettikleri sıralama değerlerinin ortalamasına göre algoritmalar incelendiğinde, TempPSO'nun 1.53'lük sıralama değeri ile ilk sırada yer aldığı görülmüştür. Onu 1.79'luk sıralama değerine sahip GGLPSOD algoritması takip etmektedir.

SOCO 1000 boyut sonuçları Tablo 5.16'de yer almaktadır. 500 boyutta olduğu gibi TempPSO; CLPSO, ELPSO, GLPSO ve PSO algoritmalarına karşı bütün problemlerde galip gelmiştir. Buna ek olarak TempPSO; FrankPSO, PSOk ve GGLPSOD algoritmalarından sırasıyla 18, 14 ve 12 problemlerinde daha iyi sonuçlar elde etmiştir. TempPSO'nun, GGLPSOD ve PSOk algoritmaları hariç diğer algoritmalara karşı istatistiksel olarak daha iyi sonuçlar aldığı söylenebilir. Ayrıca sıralama değerleri incelendiğinde 1.68'lik sıralama değeri ile TempPSO birinci, 1.74'lük sıralama değeri ile ise GGLPSOD algoritması ikinci sırada yer almıştır. Buradan, problem boyutu yükselse bile algoritmanın gösterdiği performansın değişmediği sonucu çıkarılabilir. Böylelikle algoritmanın ölçeklenebilirlik özelliğinin iyi olduğu söylenebilir.

Tablo 5.15. *PSO varyantlarının SOCO 500 boyut için medyan karşılaştırma sonuçları*

Fonk.	TempPSO	CLPSO	ELPSO	FrankPSO	GGLPSOD	GLPSO	PSO	PSOk
f1	**2.325E-04**	2.066E+05	2.244E+06	1.354E+05	1.569E+01	2.354E+06	2.024E+06	2.016E+03
f2	**1.756E+01**	1.094E+02	1.647E+02	9.968E+01	3.584E+01	1.608E+02	1.648E+02	1.211E+02
f3	**9.385E+02**	4.587E+10	2.065E+12	3.208E+10	3.429E+04	1.910E+12	1.402E+12	7.233E+07
f4	2.224E+03	4.742E+03	1.037E+04	6.289E+03	**2.229E+01**	1.046E+04	9.885E+03	1.545E+03
f5	**2.828E-05**	1.903E+03	1.894E+04	1.356E+03	1.096E+00	1.946E+04	1.667E+04	1.067E+01
f6	**4.641E-05**	1.692E+01	2.134E+01	2.039E+01	5.928E-01	2.131E+01	2.130E+01	8.056E+00
f7	4.511E-04	8.965E+02	1.509E+224	6.854E+02	2.994E+00	1.194E+200	1.653E+03	**1.000E-08**
f8	5.449E+05	1.594E+06	3.838E+06	4.427E+05	**2.500E+05**	7.719E+06	6.944E+06	7.239E+05
f9	1.320E+03	4.150E+03	5.875E+03	3.570E+03	**5.255E+02**	5.622E+03	4.982E+03	1.649E+03
f10	**2.858E-07**	1.826E+04	9.388E+04	7.581E+03	1.559E+01	8.099E+04	4.429E+04	8.650E+01
f11	1.305E+03	4.142E+03	5.875E+03	3.616E+03	**5.282E+02**	5.608E+03	5.039E+03	1.595E+03
f12	**7.054E+00**	1.079E+05	1.570E+06	6.565E+04	1.183E+02	1.681E+06	1.572E+06	1.934E+03
f13	**8.122E+02**	1.785E+10	1.332E+12	1.142E+10	1.318E+04	1.341E+12	9.523E+11	3.526E+03
f14	1.798E+03	3.480E+03	7.778E+03	4.450E+03	**4.257E+01**	7.871E+03	7.662E+03	1.222E+03
f15	**1.308E-04**	3.325E+03	4.465E+160	1.438E+03	3.499E+00	1.786E+144	7.022E+03	4.501E+01
f16	**1.662E+02**	4.257E+04	9.299E+05	1.520E+04	2.592E+02	1.033E+06	9.859E+05	2.921E+03
f17	**1.094E+03**	1.162E+08	1.849E+11	1.157E+07	1.893E+03	2.494E+11	2.494E+11	4.617E+03
f18	7.648E+02	1.464E+03	3.110E+03	1.648E+03	**8.897E+01**	3.242E+03	3.085E+03	7.833E+02
f19	**3.010E-06**	7.049E+03	9.805E+39	2.925E+03	5.245E+00	1.056E+34	2.139E+04	6.618E+01
rank	**1.53**	4.74	7.21	4.05	1.79	7.42	6.37	2.89
Our-win		19	19	18	13	19	19	16
Our-lost		0	0	1	6	0	0	3
draw		0	0	0	0	0	0	0
P-value		0.000132	0.000131	0.001285	0.717218	0.000132	0.000132	0.004274

Tablo 5.16. *PSO varyantlarının SOCO 1000 boyut için medyan karşılaştırma sonuçları*

Fonk.	TempPSO	CLPSO	ELPSO	FrankPSO	GGLPSOD	GLPSO	PSO	PSOk
f1	**1.773E+00**	6.494E+05	4.931E+06	6.491E+05	8.042E+02	4.876E+06	4.203E+06	2.063E+04
f2	**4.076E+01**	1.270E+02	1.745E+02	1.101E+02	5.806E+01	1.689E+02	1.742E+02	1.414E+02
f3	**2.828E+03**	2.311E+11	5.028E+12	2.734E+11	2.562E+06	4.562E+12	2.658E+12	1.896E+09
f4	6.126E+03	1.131E+04	2.212E+04	1.401E+04	**3.652E+02**	2.168E+04	1.979E+04	4.790E+03
f5	**2.081E-01**	5.889E+03	4.414E+04	5.660E+03	7.882E+00	4.356E+04	3.488E+04	1.912E+02
f6	**1.075E-02**	1.842E+01	2.142E+01	2.071E+01	2.603E+00	2.139E+01	2.134E+01	1.542E+01
f7	**2.428E-01**	1.423E+51	inf	2.058E+03	3.546E+01	inf	3.225E+03	1.194E+01
f8	2.327E+06	5.998E+06	1.381E+07	1.714E+06	**1.131E+06**	2.553E+07	2.887E+07	2.715E+06
f9	5.796E+03	9.170E+03	1.214E+04	7.782E+03	**2.077E+03**	1.158E+04	1.006E+04	5.006E+03
f10	**2.990E-02**	5.331E+04	2.062E+05	2.856E+04	2.563E+02	1.723E+05	9.211E+04	1.488E+02
f11	5.777E+03	9.153E+03	1.214E+04	7.933E+03	**2.101E+03**	1.155E+04	1.008E+04	5.055E+03
f12	**7.710E+02**	3.762E+05	3.594E+06	3.508E+05	9.188E+02	3.636E+06	3.100E+06	1.161E+04
f13	**3.911E+03**	1.020E+11	3.626E+12	1.147E+11	1.354E+05	3.294E+12	1.746E+12	3.315E+08
f14	4.483E+03	8.367E+03	1.678E+04	1.039E+04	**3.563E+02**	1.663E+04	1.515E+04	3.535E+03
f15	**1.470E-01**	2.468E+04	inf	6.220E+03	7.299E+01	inf	1.574E+04	2.460E+02
f16	2.505E+03	1.657E+05	2.229E+06	1.269E+05	**1.343E+03**	2.328E+06	2.062E+06	1.378E+04
f17	**5.239E+03**	2.273E+09	6.469E+11	1.266E+09	5.829E+03	7.312E+11	4.929E+11	1.022E+04
f18	2.125E+03	3.504E+03	6.735E+03	3.769E+03	**4.104E+02**	6.813E+03	5.915E+03	1.894E+03
f19	**3.484E-02**	2.356E+04	3.833E+96	1.299E+04	1.568E+02	2.133E+85	4.297E+04	3.127E+02
rank	**1.68**	4.74	7.58	4.16	1.74	7.16	6.05	2.79
Our-win		19	19	18	12	19	19	14
Our-lost		0	0	1	7	0	0	5
draw		0	0	0	0	0	0	0
P-value		0.000132	0.000131	0.001285	0.872118	0.000131	0.000132	0.058573

6. GENELLEŞTİRİLMİŞ YAPAY ARI KOLONİSİ ÇATISI

6.1. Algoritma Tanımı

ABC algoritmasının ortaya çıkmasından bu yana birçok ABC varyantı ileri sürülmüştür. Bu algoritmalarının temel amaçları, yakınsama hızının artırılması ve elde edilen çözümün kalitesinin geliştirilmesidir. Önerilen ABC varyantları, orijinal ABC algoritmasından küçük değişiklikler ile farklılaşmaktadırlar. Bu farklılaşmalara örnek olarak; çözümlerin ilklendirilmesine önerilen yöntemler, işçi arı ve gözcü arı aşamalarında yer alan arama denklemlerine önerilen denklemler ve son olarak gözcü arı adımındaki seçme olasılığının hesaplanması için önerilen yöntemler verilebilir. Bu tez çalışmasında, genelleştirilmiş ABC çatısı (ABC-X) önerilmiştir. Bu çatı, literatürde yer alan ABC varyantları gibi davranabilmektedir. Buna ek olarak; henüz literatürde yer almayan ABC varyantlarını oluşturma gücüne sahiptir. Çalışmada önerilen, ABC varyantlarını elle (manual) araştırmaktan ziyade otomatik algoritma yapılandırma aracı kullanılarak ABC algoritma tasarım uzayını araştırmaktır.

Algoritma 6.1. *Genelleştirilmiş ABC algoritmasının sözde kodu*

```
İlklendirme stratejisini uygula                          # Bileşen 1: İlklendirme
for her bir işçi arı do                                  # Bileşen 2: İşçi arı adımı
        for m = 1 to M_e do
                j boyutunu rastgele seç
                İşçi arı adımının arama denklemini uygula (Denk. (6.1))
        if x̂_i < x_i then
                x_i = x̂_i, trial_i = 0,
        else
                trial_i = trial_i + 1
Seçme olasılığını hesapla                                # Bileşen 3: Olasılık hesaplama
for her bir gözcü arı do                                 # Bileşen 4: Gözcü arı adımı
        for m = 1 to M_o do
                j boyutunu rastgele seç
                Gözcü arı adımının arama denklemini uygula (Denk. (6.1))
        if x̂_i < x_i then
                x_i = x̂_i, trial_i = 0,
        else
                trial_i = trial_i + 1
Kâşif arı stratejisini uygula                            # Bileşen 5: Kâşif arı adımı
Popülasyon boyutunu güncelle                             # Bileşen 6: Popülasyon boyut
```

ABC-X'e ait sözde kod Algoritma 6.1'de gösterilmektedir. ABC-X kodu incelendiğinde, genel yapısı orijinal ABC algoritmasına benzemektedir. Buna karşın, gözcü arı adımındaki aday çözümlerin seçim olasılığını hesaplamak için algoritma

bileşenleri ve popülasyon boyutunun sabit mi değişken mi olduğunu belirleyen yeni algoritma bileşenleri yer almaktadır. Farklı ABC varyantlarına ait özellikler altı adet algoritma bileşenlerinde toplanmış ve karmaşıklıklar bu bileşenlerin arkasına gizlenmiş ve ABC-X'e ait sözde kod Algoritma 6.1'de verilmiştir. Bundan sonra, ilk olarak ABC-X'de yer alan altı algoritma bileşenlerinin her biri için farklı seçenekler sunulmuştur. Daha sonra çeşitli ABC varyantlarının, ABC-X vasıtasıyla üretilmesi anlatılmıştır.

6.1.1. Bileşen 1, ilklendirme

Arama ilklendirmesi için, dört tane strateji gerçekleştirilmiştir. Bunlar; varsayılan (default), kaotik (chaotic), karşıtlık (opposition) ve mix (mix)'dir.

- *varsayılan (default)*: Orijinal ABC algoritması ile gelen ilklendirme yöntemidir. Bu stratejiye göre; aday çözümler, arama uzayında düzgün dağılımlı (uniform) olarak rastgele konumlandırılarak üretilmektedir.
- *kaotik (chaotic)*: Rastgele sayı üreteci olarak kaotik haritalar kullanılmaktadır. Çeşitli çalışmalarda kullanılan yedi adet kaotik harita ABC-X altında tanımlanmıştır (Alatas, 2010; Lu vd., 2014; Xiang vd., 2014; Xiang ve An, 2013).
- *karşıtlık (opposition)*: Karşıtlık-tabanlı öğrenme yöntemini ABC algoritmasına uyarlamaktadır. Bu yönteme göre; rastgele üretilen her bir aday çözüm ile birlikte, çözüm uzayının merkezine göre yansıması da üretilir. Elde edilen $2\times SN$ çözüm içinden *SN* tane kötü olan çözüm elenir ve *SN* tane iyi çözüm kalır. Böylelikle başlangıç popülasyonu elde edilmiş olur. (Bao ve Zeng, 2011; El-Abd, 2011; Sharma ve Pant, 2013).
- *mix (mix)*: Çeşitli ABC varyantlarında yer alan bu strateji (W. Gao vd., 2012; Gao ve Liu, 2011) karşıtlık-tabanlı öğrenme ile kaotik rastgele üreticisini bir araya getirmektedir.

İlklendirme adımı ile alakalı iki adet parametre daha mevcuttur. Bunlardan ilki, tam sayı olan popülasyon boyutu *SN* ve ikincisi ise rastgele sayı üretiminde kullanılması düşünülen kaotik haritasının seçimini belirten değerdir.

6.1.2. Bileşen 2 ve 4: İşçi arı ve gözcü arı adımları

İşçi arı ve gözcü arı adımlarında genelleştirilmiş arama denklemi gerçekleştirilmiştir. Bunun sayesinde; orijinal ABC, daha önce tanımlanmış arama denklemleri ve daha önce keşfedilmemiş arama denklemlerini üretilebilmektedir. Denklemin genel yapısı şu şekildedir:

$$\hat{x}_{i,j} = terim1 + terim2 + terim3 + terim4 \tag{6.1}$$

Orijinal ABC algoritmasında yer alan, işçi ve gözcü arı standart arama denklemi birinci $x_{i,j}$ ve ikincisi $\phi_{i,j}(x_{i,j} - x_{r1,j})$ olmak üzere iki tane terim içermektedir. Denklem (6.1), bu yapıyı genelleştirerek dört terimli genel bir yapıya kavuşturmuştur. Standart arama denkleminde olduğu gibi genelleştirilmiş arama denkleminde de tek bir anda tek bir değişken değiştirilebilir. Bununla birlikte, genelleştirilmiş arama denklemi sayesinde dört terim için farklı seçenekler düşünülerek çok çeşitli arama denklemleri üretilebilmektedir.

- *terim1*: Bu terim için *j* değişkini kullanılarak üç seçenek vardır. (i) referans aday çözüm, $x_{i,j}$, (ii) global-en iyi aday çözüm ($x_{G,j}$), veya (iii) düzgün dağılımlı olarak rastgele seçilen aday çözüm $x_{r1,j}$.
- *terim2*: Yoğunlaştırmanın (intensification) veya çeşitlendirmenin (diversification) farklı seviyeleri, çözümün yönlendirilme kaynağına göre değişkenlik göstermektedir. Bu kaynaklar; rastgele bir komşu, global en iyi aday çözüm veya en-yakın komşu aday çözüm olabilmektedir. Aynı zamanda orijinal ABC arama denkleminde yer alan ikinci terim de kullanılabilir.
- *terim3 ve terim4*: ABC arama denkleminde yapılan değişikliklerin bazıları popülasyon tabanlı teknikler olan, parçacık sürü optimizasyonu (PSO) ve diferansiyel gelişim (DE)'den alınmıştırlar (Gao and Liu 2011; Zhu and Kwong 2010; Bin and Qian 2011; Akay and Karaboga 2012; Gao et al. 2014; Imanian et al. 2014). Genellikle, yeni terimlerin kullanılması sonucunda arama denkleminde global en iyi (x_G), rastgele (x_r), en-yakın komşu (x_{GD}) veya referans aday çözümü (x_i) değişikliklerini ortaya çıkarmıştır. Ayrıca üçüncü ve dördüncü terimde "kullanma" seçeneği de yer almaktadır. Böylece, Denklem (6.1) tarafından belirli bir problem için

üretilen uygun arama denklemleri aşağıda açıklandığı gibi bir, iki, üç veya dört terimden oluşabilir.

Arama denklemine ekstra parametreler eklenmiştir. Varsayılan olarak, $\phi_{i,j}$, seçilen her boyutu değiştirmek için bağımsız olarak [-1,1] aralığında düzgün dağılım kullanılarak üretilen rastgele bir sayıdır. Ancak, $\phi_{i,j}$'nin aralığı $[-a,a]$, a>0 olacak şekilde ayarlanabilir (Akay ve Karaboga 2012). Genelleştirilmiş arama denkleminde $\phi_{i,j}$, $[-a,a]$ veya [-1,1] aralığından rastgele seçilebilen genelleştirilmiş bir parametre haline getirilmiştir. Orijinal ABC algoritmasının uygulanmasında, arama denklemi tarafından yalnızca bir boyut değiştirilmektedir bu da algoritmanın düşük yakınsama hızının sebebi olarak görülmektedir. Multimodal ve kompozit fonksiyonlar için, bazı ABC varyantları (Akay ve Karaboga, 2012; Gao ve Liu, 2011), birden fazla boyutu değiştirdiğinde daha iyi bir performans elde etmiştirler. Bunun için arama denkleminde değiştirilecek boyut sayısını temsil eden M ($1 \le M \le D$) parametresi kullanılmıştır. ABC-X algoritmasında tek bir M değeri yerine işçi ve gözcü arı adımları için iki farklı parametre kullanılmıştır. M_e ve M_o değerleri sırasıyla işçi arı ve gözcü arı adımlarında değiştirilmesi istenen boyut sayısını göstermektedir.

Genelleştirilmiş arama denkleminin her bir terimi için olası alternatifleri Tablo 6.1' de listelenmiştir. Tablo 6.1'de $\phi 1_{i,j}$, $\phi 2_{i,j}$ ve $\phi 3_{i,j}$ $[-1,1]$ veya $[-a,a]$ aralıkların herhangi birinden rasgele üretilen sayılardır. n ve k ($1 \le n \le k \le D$) pozitif tamsayı sayılarıdır. x_G global-en iyi aday çözüm, x_{r1} ve x_{r2} rastgele seçilmiş aday çözümleri ve x_{GD} aşağıdaki denklem kullanılarak hesaplanan olasılığa göre seçilen "yakın"-komşu aday çözümdür.

$$p_{GD} = \frac{\frac{1}{dist(x_i, x_{GD})}}{\sum_{n=1, n\neq i}^{SN} \frac{1}{dist(x_i, x_n)}} \tag{6.2}$$

p_{GD}, komşu x_{GD}'nin olasılığını ve iki aday çözümün, x ve y, arasındaki mesafesi $dist(x,y)$ ile gösterilir (Diwold vd., 2011). Denklem (6.2) arama uzayındaki mevcut çözüme yakın olan çözümlerin seçilmesi tercih edilir. İşçi arı ve gözcü arı adımlarındaki arama denklemleri bağımsız olarak kullanılır. Bu nedenle, arama denklemlerinin

yapılandırılmasından sonra, işçi arı ve gözcü arı adımlarında iki farklı arama denklemi kullanabilir.

Tablo 6.1. *Önerilen arama stratejisinin her bir bileşeni için olası alternatifler*

M_e, M_o	terim1	terim2	terim3	terim4
1	$x_{i,j}$	$\phi 1_{i,j}(x_{i,j}-x_{G,j})$	$\phi 2_{i,j}(x_{i,j}-x_{G,j})$	$\phi 3_{i,j}(x_{i,j}-x_{G,j})$
n	$x_{G,j}$	$\phi 1_{i,j}(x_{i,j}-x_{r1,j})$	$\phi 2_{i,j}(x_{i,j}-x_{r1,j})$	$\phi 3_{i,j}(x_{i,j}-x_{r1,j})$
[n,k]	$x_{r1,j}$	$\phi 1_{i,j}(x_{G,j}-x_{r1,j})$	$\phi 2_{i,j}(x_{G,j}-x_{r1,j})$	$\phi 3_{i,j}(x_{G,j}-x_{r1,j})$
		$\phi 1_{i,j}(x_{r1,j}-x_{r2,j})$	$\phi 2_{i,j}(x_{r1,j}-x_{r2,j})$	$\phi 3_{i,j}(x_{r1,j}-x_{r2,j})$
		$\phi 1_{i,j}(x_{i,j}-x_{GD,j})$	$\phi 2_{i,j}(x_{i,j}-x_{GD,j})$	$\phi 3_{i,j}(x_{i,j}-x_{GD,j})$
		kullanma	kullanma	kullanma

6.1.3. Bileşen 3: olasılık hesaplama

Gözcü arı adımında, referans aday çözümler keşif için olasılıkla seçilirler; orijinal ABC algoritmasında, bu olasılıklı seçim Denklem (3.7) ile yapılır. Bununla birlikte, referans çözümler birbirine çok yakın uygunluk değerlere sahip ise bu denklem ile ayırt edilemez. Öte yandan, iyi ve kötü çözümlerin uygunluk değerleri arasında önemli bir fark varsa, bazı aday referans çözümlerinin neredeyse sıfır seçilme olasılığı olur. Alternatif olarak, ağırlıklı-toplam (Aydın ve Özyön, 2013) ve sıralama-tabanlı yaklaşımları (Kang vd., 2011) önerilmiştir. Ağırlıklı toplam yönteminde, en kötü çözümün ağırlığı en azından $1-w$ olarak ayarlanır ve seçilecek bir çözümün olasılığı, $1-w$'den 1'e değişecek şekilde aşağıdaki denklem ile seçilir.

$$p_i = (1-w) + (w \times \frac{f(x_i)}{f(x_G)}) \tag{6.3}$$

w, genellikle 0.9 olarak ayarlanmış bir parametredir. Sıralama-tabanlı yaklaşımda (Kang vd., 2011), sıralama-tabanlı uygunluk uyarlama yöntemi şu şekilde tanımlanmıştır:

$$f_i = 2 - SP + \frac{2(SP-1)(r_i-1)}{SN-1}, \tag{6.4}$$

$SP \in [1.0, 2.0]$ seçim baskısı ve r_i referans aday çözüm x_i'nin uygunluk durumuna göre popülasyondaki sıralamasıdır.

ABC-X'te, seçim olasılıklarının hesaplanması için yukarıda belirtilen üç alternatif dâhil edilmiştir. Bunlar; "varsayılan", "ağırlıklı" ve "sıralamaya-tabanlı" olasılık hesaplama stratejileridir.

6.1.4. Kâşif arı adımı

Kâşif arı adımının amacı, yeni arama uzay alanları keşfederek yerel optimumdan kaçmaktır. Bu adım için beş tasarım seçeneği tanımlanmıştır:

- *Varsayılan (default)*: Orijinal ABC'de yer alan varsayılan stratejidir.
- *IABCSBStrategy* : Artan ABC (Incremental ABC, IABC)'de ki (Aydın vd., 2012, 2014; Aydın ve Özyön, 2013) kaşif arı adımında yer alan Denklem (3.5)'i aşağıdaki denklem ile değiştirmiştir:

$$\hat{x}_{i,j} = x_{G,j} + Rf\,(x_{G,j} - x_{new,j}) \tag{6.5}$$

Burada $\hat{x}_i$, yeni çözüm ve *Rf* ise arama bölgesini global-en iyi çözüm, x_G, etrafında ayarlayan yer değiştirme faktörüdür. x_{new}, Denklem (3.5)'e göre rastgele düzgün dağılımlı olarak üretilen yeni bir çözümdür.

- *BABCSBStrategy*: Şimdiye Kadarki En İyi Seçim ABC (Best-so-far Selection ABC, BABC) (Banharnsakun vd., 2011) algoritmasında kullanılan ve kâşif arının arama bölgesini uyarlanabilir şekilde ayarlayan denklem şu şekildedir:

$$\hat{x}_{i,j} = x_{i,j} + x_{i,j}\phi_{i,j}(w_{max} - (itr_{current} / itr_{max})(w_{max} - w_{min})) \tag{6.6}$$

w_{min} ve w_{max} kaşif arının pozisyon ayarlamasının sırasıyla minimum ve maksimum yüzdesidir; $itr_{current}$ geçerli iterasyon sayısını ve itr_{max}ise maksimum iterasyon sayısını gösterir (Banharnsakun vd., 2011).

- *chaoticSearch1*: Kaotik ABC (Chaotic ABC, CABC) algoritmasında (Alatas, 2010) kâşif arı adımı için önerilmiş kaotik arama yöntemidir.
- *chaoticSearch2*: Verimli ve Sağlam ABC (efficient and robust ABC, ERABC) algoritmasında (Xiang ve An, 2013) kullanılan bir diğer kaotik arama yöntemidir.

6.1.5. Popülasyon boyutu

Orijinal ABC algoritmasında, popülasyon büyüklüğü sabit bir değerdir. Aydın vd. (Aydın vd., 2012; Aydın ve Özyön, 2013) artan sosyal öğrenme mekanizmasını (Liao, de Oca, vd., 2011) kullanarak, artan popülasyon boyut stratejisini önermişlerdir. Bu yaklaşımda, algoritma küçük bir popülasyonla başlar ve daha sonra, her *g* iterasyonda bir popülasyon büyüklüğü SN_{Max}'a eşit oluncaya kadar popülasyona yeni bir çözüm eklenmeye devam edilir. "sosyal öğrenme" bileşeni; yeni çözümün, en iyi çözüme daha yakın olacak şekilde aşağıdaki denklem kullanılarak üretilmesi sağlanır.

$$\acute{x}_{new,j} = x_{new,j} + \varphi_{i,j}(x_{gbest,j} - x_{new,j}) \tag{6.7}$$

$x_{new,j}$, Denklem (3.5)'e göre üretilen yeni bir çözümü ve $\acute{x}_{new,j}$ yeni çözümün güncellenmiş yeridir.

ABC-X çatısının parametreleri ve mevcut seçeneklerinin özet hali Tablo 6.2'de verilmiştir.

Tablo 6.2. *ABC-X'de yer alan parametreler*

İsim	Tür	Değerler	Açıklama
Bileşen 1: İlklendirme			
SN	i	[3, 100]	Popülasyon boyut
chaoticMap	c	7 kaotik harita	Kaotik rastgele sayı üreteci için kaotik harita
initStrategy	c	default, chaotic, opposition, mix	İlklendirme adımı stratejisi
Bileşen 2: İşçi arı adımı			
M_e	c	one, fix, n_e var, rand (n_e, k_e)	İşçi arı adımında değiştirilecek değişkenlerin sayısı
$terim1_e$	c	3 seçenek (Tablo 6.1)	İşçi arı adımı arama denkleminin ilk terimi
$terim2_e$	c	6 seçenek (Tablo 6.1)	İşçi arı adımı arama denkleminin ikinci, üçüncü ve dördüncü terimi
$terim3_e$			
$terim4_e$			
a1, a2, a3	c	1 Reel (0, 4)	İşçi arı adımındaki ϕ_1, ϕ_2 ve ϕ_3'ün sınır değerleri
Bileşen 3: Olasılık hesaplama			
selectProb Stratejisi	c	default, weighted , rank-based	Seçim olasılık hesaplama stratejileri
SP	r	Reel (1, 2)	Sıralama-tabanlı olasılık seçiminde yer alan seçim baskısı
Bileşen 4: Gözcü arı adımı			
M_o	c	one, fix, n_o var, rand(n_o, k_o)	Gözcü arı adımında değiştirilecek değişkenlerin sayısı
$terim1_o$	c	3 seçenek (Tablo 6.1)	Gözcü arı adımı arama denkleminin ilk terimi
$terim2_o$	c	6 seçenek (Tablo 6.1)	Gözcü arı adımı arama denkleminin ikinci, üçüncü ve dördüncü terimi
$terim3_o$			
$terim4_o$			
b1,b2, b3	c	1 Reel (0, 4)	Gözcü arı adımındaki ϕ_1, ϕ_2 ve ϕ_3'ün sınır değerleri
Bileşen 5: Kaşif arı adımı			
scoutBee Stratejisi	c	Default IABCSB-Strategy BABCSB-Strategy chaoticSearch1 chaoticSearch2	Kâşif arı adımı stratejisi
limitF	r	(0, 4)	Limit faktörü
w_{min}	r	(0, 0.5)	Kâşif arı ayarlama minimum yüzdesi
w_{max}	r	(0.5, 1)	Kâşif arı ayarlama maksimum yüzdesi

Tablo 6.2. (Devam) *ABC-X'de yer alan parametreler*

İsim	Tür	Değerler	Açıklama
Bileşen 6: Popülasyon boyut stratejisi			
popSize	c	fix, incremental	İşlem sırasındaki popülasyon boyutunun değişimi
Rf	r	$1^{\text{Reel}(-14,0)}$	IABCstrategy yer alan yer değiştirme faktörü
SN_{max}	i	[20, 100]	Artan popülasyon için maksimum popülasyon boyutu
g	i	[1, 10]	Popülasyon boyutunun büyüğüme periyotu

Tablo 6.3'de, literatürde yer alan ABC algoritmalarının ABC-X algoritması kullanılarak ile nasıl oluşturuldukları listelenmiştir. Tabloda verilmiş olan ABC varyantları: ABC (Karaboga ve Basturk, 2007), Şimdiye Kadarki En İyi Seçim ABC (Best-so-far Selection ABC, BABC) (Banharnsakun vd., 2011), Kaotik ABC (Chaotic ABC, CABC) (Alatas, 2010), Değiştirilmiş ABC (Modified ABC, MABC) (Akay ve Karaboga, 2012), Global-En İyi ABC (Global-best ABC, GABC) (W. Gao vd., 2012), Global-En İyi Mesafe rehberli ABC (Global-best Distance guided ABC, GDABC) (Diwold vd., 2011), Geliştirilmiş ABC (Enhanced ABC, EABC) (Gao vd., 2014), Artan ABC (Incremental ABC, IABC) (Aydın vd., 2011), Verimli ve Sağlam ABC (efficient and robust ABC, ERABC) (Xiang ve An, 2013) ve Karşıtlık-Tabanlı ABC (opposition-based ABC, OABC) (El-Abd, 2011) .

ABC-X algoritması, ABC varyantlarını basit komut satırı yapılandırılmaları ile üretebilmektedir. Komut satırı parametreleri vasıtasıyla algoritma içerisinden uygun yöntemi seçmektedir. ABC-X algoritması henüz daha keşfedilmemiş ABC varyantlarını üretebilme gücüne sahiptir. Aslında, ABC-X dokuz milyondan fazla ABC varyantını oluşturmaya olanak sağlar. Ayrıca bunların büyük çoğunluğu literatürde daha henüz keşfedilmemiştir. Bu varyant sayısı, tamsayı ve reel parametreleri dikkate alınmadan sadece kategorik değişkenlerin mümkün olan değerlerinin hesaplanması ile ortaya çıkmaktadır.

Tablo 6.3. *Öne çıkan ABC varyantlarının ABC-X çatısı kullanılarak nasıl üretilebileceğini göstermektedir.*

Adımlar	Parametreler	ABC	BABC	CABC	MABC	GABC
İlklendirme	Strateji	Default	Default	Chaotic	Default	Default
İşçi arı	M_e	1	1	1	n_e	1
	a1	1	1	1	Reel (0,4)	1
	a2	–	Reel (0,4)	–	–	1
	a3	–	–	–	–	–
	$terim1_e$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$
	$terim2_e$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$
	$terim3_e$	–	$(x_{G,j}-x_{i,j})$	–	–	$(x_{G,j}-x_{i,j})$
	$terim4_e$	–	–	–	–	–
Olasılık hesaplama	Strateji	Default	Default	Default	Default	Default
Gözcü arı	M_o	1	1	1	n_o	1
	b1	1	1	1	Reel (0,4)	1
	b2	–	Reel (0,4)	–	–	1
	b3	–	–	–	–	–
	$terim1_o$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$
	$terim2_o$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$	$(x_{i,j}-x_{r1,j})$
	$terim3_o$	–	$(x_{G,j}-x_{i,j})$	–	–	$(x_{G,j}-x_{i,j})$
	$terim4_o$	–	–	–	–	–
Kâşif arı	Strateji	Default	BABC	chaoticSearch1	Default	Default
Pop. boyut	popSize	Fix	Fix	Fix	Fix	Fix

Tablo 6.3. (Devam) *Tablo, bazı öne çıkan ABC varyantlarının ABC-X çatısı kullanılarak nasıl üretilebileceğini göstermektedir***.**

Adımlar	Parametreler	GDABC	EABC	IABC	OABC	ERABC
İlklendirme	Strateji	Default	Default	Default	Opp.-based	Chaotic
İşçi arı	M_e	1	1	1	1	1
	a1	1	Reel (0,4)	1	1	1
	a2	1	Reel (0,4)	–	–	Reel (0,4)
	a3	–	–	–	–	–
	$terim1_e$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$
	$terim2_e$	$(x_{G,j} - x_{i,j})$	$(x_{G,j} - x_{r1,j})$	$(x_{G,j} - x_{i,j})$	$(x_{i,j} - x_{r1,j})$	$(x_{i,j} - x_{r1,j})$
	$terim3_e$	$(x_{G,j} - x_{i,j})$	$(x_{r1,j} - x_{r2,j})$	–	–	$(x_{G,j} - x_{i,j})$
	$terim4_e$	–	–	–	–	–
Olasılık hesaplama	Strateji	Default	Default	Weighted	Default	Default
Gözcü arı	M_o	1	1	1	1	1
	b1	1	Reel (0,4)	1	1	1
	b2	1	Reel (0,4)	–	–	Reel (0,4)
	b3	–	–	–	–	–
	$terim1_o$	$x_{i,j}$	$x_{r1,j}$	$x_{i,j}$	$x_{i,j}$	$x_{i,j}$
	$terim2_o$	$(x_{i,j} - x_{GD,j})$	$(x_{G,j} - x_{r1,j})$	$(x_{G,j} - x_{i,j})$	$(x_{i,j} - x_{r1,j})$	$(x_{i,j} - x_{r1,j})$
	$terim3_o$	$(x_{G,j} - x_{i,j})$	$(x_{G,j} - x_{i,j})$	–	–	$(x_{G,j} - x_{i,j})$
	$terim4_o$	–	–	–	–	–
Kâşif arılar	Strateji	Default	Default	IABC	Default	chaoticSearch2
Pop. boyut	popSize	Fix	Fix	Incremental	Fix	Fix

6.2. Deneysel Sonuçları

6.2.1. Karma ölçüt seti ile analizler

6.2.1.1. *Karma ölçüt fonksiyonları*

ABC-X'in etkinliği ve esnekliğinin ölçülmesi için 50 fonksiyon içeren bir karma ölçüt seti kullanılmıştır. Bu fonksiyonlar; CEC'05, CEC'14 ve SOCO yarışmalarından alınmıştır (Liang vd., 2013; Lozano vd., 2010; Suganthan vd., 2005). Bu çalışmada, fonksiyonlar türlerine göre unimodal, multimodal ve hibrit adlı üç gruba ayrılmıştır. Bu fonksiyonlar; ayrılamayan (non-separability), ötelenmiş (shift), döndürülmüş (rotation), uygunluk değerinde gürültü bulunan ve çok sayıda yerel optimuma sahip olma gibi çeşitli özelliklere sahiptirler. Üçüncü grup, birinci ve ikinci grupta yer alan çeşitli ölçüt fonksiyonlarının melezleştirilmesinden veya birleştirmesinden oluşan 14 hibrit fonksiyon ve 10 kompozit fonksiyon içerir. Fonksiyonların kısa tanımları Tablo 6.4'de yer almaktadır. Bu fonksiyonların ayrıntılı bir açıklamasına (Liang vd., 2013; Lozano vd., 2010; Suganthan vd., 2005)'dan ulaşılabilir.

Tablo 6.4. *Karma ölçüt setinde yer alan fonksiyonların genel özeti*

Id	İsim	Aralık	Yarışma
	UNIMODAL FONKSİYONLAR		
f1	Shifted Sphere	[-100,100]	SOCO
f2	Shifted Rotated High Conditioned Elliptic	[-100,100]	CEC'14
f3	Sifted Rotated Bent Cigar	[-100,100]	CEC'14
f4	Shifted Rotated Discus	[-100,100]	CEC'14
f5	Shifted Schwefel 22.1	[-100,100]	SOCO
f6	Shifted Rotated Schwefel 1.2	[-65.536,65.536]	SOCO
f7	Shifted Scfewels12 noise in fittness	[-100,100]	CEC'05
f8	Sifted Schwefel 2.22	[-10,10]	SOCO
f9	Shifted Extended f10	[-100,100]	SOCO
f10	Shifted Bohachevsky	[-100,100]	SOCO
f11	Shifted Schaffer	[-100,100]	CEC'05
f12	Schwefel 2.6 Global Optimum on Bounds	[-100,100]	CEC'05
	MULTIMODAL FONKSİYONLAR		
f13	Shifted Ackley	[-32,32]	SOCO
f14	Shifted Rotated Ackley	[-100,100]	CEC'14
f15	Shifted Rosenbrock	[-100,100]	SOCO
f16	Shifted Rotated Rosenbrock	[-100,100]	CEC'14
f17	Shifted Griewank	[-600,600]	SOCO
f18	Shifted Rotated Griewank	[-100,100]	CEC'14
f19	Shifted Rastrigin	[-100,100]	SOCO
f20	Shifted Rotated Rastrigin	[-100,100]	CEC'14
f21	Shifted Schwefel	[-100,100]	SOCO
f22	Shifted Rotated Schwefel	[-100,100]	CEC'14
f23	Shifted Rotated WeierStrass	[-100,100]	CEC'14
f24	Shifted Rotated Katsuura	[-100,100]	CEC'14
f25	Sifted Rotated HappyCat	[-100,100]	CEC'14
f26	Shifted Rotated HGBat	[-100,100]	CEC'14

Tablo 6.4. (Devam) *Karma ölçüt setinde yer alan fonksiyonların genel özeti*

HİBRİT FONKSİYONLAR			
f27	Hybrid Function 1 (N=2)	[-100,100]	SOCO
f28	Hybrid Function 2 (N=2)	[-100,100]	SOCO
f29	Hybrid Function 3 (N=2)	[-5,5]	SOCO
f30	Hybrid Function 4 (N=2)	[-10,10]	SOCO
f31	Hybrid Function 7 (N=2)	[-100,100]	SOCO
f32	Hybrid Function 8 (N=2)	[-100,100]	SOCO
f33	Hybrid Function 9 (N=2)	[-5,5]	SOCO
f34	Hybrid Function 10 (N=2)	[-10,10]	SOCO
f35	Hybrid Function 1 (N=3)	[-100,100]	CEC'14
f36	Hybrid Function 2 (N=3)	[-100,100]	CEC'14
f37	Hybrid Function 3 (N=4)	[-100,100]	CEC'14
f38	Hybrid Function 4 (N=4)	[-100,100]	CEC'14
f39	Hybrid Function 5 (N=5)	[-100,100]	CEC'14
f40	Hybrid Function 6 (N=5)	[-100,100]	CEC'14
f41	Hybrid Composition Function	[-5,5]	CEC'05
f42	Rotated Hybrid Composition Function	[-5,5]	CEC'05
f43	Rotated H. Composition F. with Noise in Fitness	[-5,5]	CEC'05
f44	Rotated Hybrid Composition F.	[-5,5]	CEC'05
f45	Rotated H. Comp. F. with a Narrow Basin for the	[-5,5]	CEC'05
f46	Rotated H. Comp. F. with the Global Opt. On the	[-5,5]	CEC'05
f47	Rotated Hybrid Composition Function	[-5,5]	CEC'05
f48	Rotated H. Comp. F. with High Condition Number	[-5,5]	CEC'05
f49	Non-Continuous Rotated Hybrid Composition	[-5,5]	CEC'05
f50	Rotated Hybrid Composition Function	[-5,5]	CEC'05

Yapılan deneyler 10, 30 ve 50 boyut için gerçekleştirilmiştir. Sonuçların istatistiksel olarak anlamlı olabilmesi için, her algoritma ve her test fonksiyonu için 50 defa bağımsız olarak çalıştırılmıştır. Çalıştırmaların bitme kriteri, D boyut olmak üzere $10000 \times D$ fonksiyon çağrım sayısı kadardır. 50 bağımsız çalıştırma içinde bulunan en iyi sonuçların medyan hata değerleri rapor edilmiştir. Algoritmanın bir çalıştırılmasında bulunan en iyi çözüm x'in hata değeri şöyle hesaplanır: $f(x) - f(x^*)$ $f(x^*)$, f fonksiyonunun bilinen optimum değeridir. Hata değerleri 10^{-8} değerinden az olanlar sıfır olarak kabul edilmiştir. Deney sonuçlarında 0.05 anlamlılık düzeyinde $(p < 0.05)$ wilcoxon ikili testleri uygulanmıştır. Tüm deneyler 8 GB RAM ve Intel Xeon E5-2620 2 GHz işlemciye sahip bir bilgisayarda yapılmıştır.

ABC-X algoritmasının parametrelerine, üç problem türünün her biri için otomatik yapılandırma işlemi uygulanmıştır. Her fonksiyon türünde eğitim örneği olarak unimodal, multimodal ve hibrit fonksiyonların 10 boyuttaki halleri kullanılmıştır. Ayrıca her bir fonksiyon türünde ABC-X ile deneylerde yer alan ve özellikleri Tablo 6.3'de verilen ABC varyantları beş farklı yapılandırma elde etmek için irace ile beş defa çalıştırılmıştır. Bunu yaparak, ABC-X çatısının otomatik parametre yapılandırma yaklaşımının sağlamlığı analiz edilmiştir. İlerleyen bölümlerde, ABC-X'in beş defa

yapılandırılmasından elde edilen sonuçlar sunulmuştur. Bir sonraki adımda, rastgele bir tane yapılandırma (varsayılan olarak ilk ayarda elde edilen) ve eğitim setlerindeki en iyi sıralamayı elde eden iki ABC-X yapılandırması seçilmiştir. Daha sonra bu iki yapılandırma eğitim setine göre, ABC-X'ten elde edilen algoritmaların avantajını incelemek için her ABC algoritmasının en iyi sıralamaya sahip olan yapılandırması ile karşılaştırılmıştır.

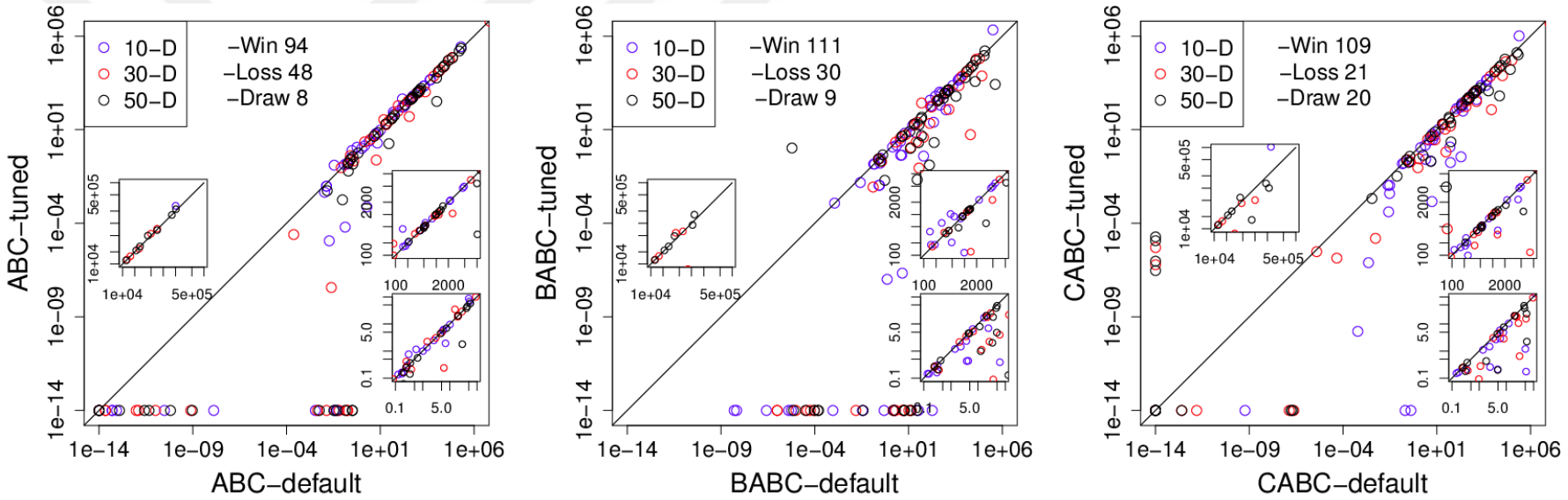
Ölçüt seti; unimodal, multimodal ve hibrit fonksiyonlarına bölünmüştür. Bu bölünmede fonksiyonların, farklı gruplara ayrılarak türleri hakkında bilgi sahibi olunma durumunun, olunmama durumundan daha iyi sonuçlar elde edip etmediği test edilmiştir. Ayrıca belirli bir örnek sınıfına odaklanarak daha iyi sonuçlar elde edilip edilemeyeceğini incelemek için, örnek sınıflar arasında ayrım yapmadan 50 fonksiyonun tümü için bir ABC-X varyantı da ayarlanmıştır. İlerleyen kısımlarda, ABC-X algoritmasının problem sınıfına özgü ayarları ile ABC-X^{all} olarak adlandırdığımız bu son yapılandırma birbirleri ile karşılaştırılmıştır.

6.2.1.2. *Varsayılan ve yapılandırılmış ABC algoritmaların karşılaştırılması*

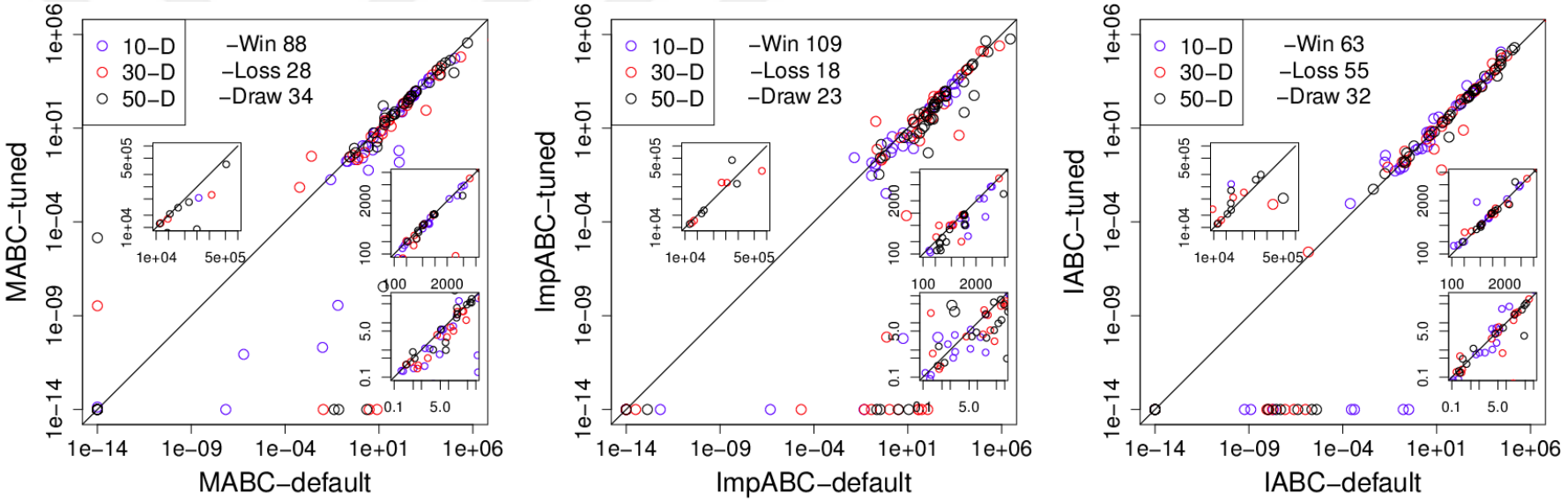
Bu çalışmanın amacı; literatürde bugüne kadar önerilen birçok varyanttan daha iyi performans gösteren ABC algoritmalarının, otomatik olarak yapılandırılabileceğini göstermektir. Dolayısıyla ABC-X, literatürde yer alan çeşitli ABC varyantları ile karşılaştırılmıştır. Orijinal ABC (ABC) (Karaboga ve Basturk, 2007), BABC (Banharnsakun vd., 2011), CABC (Alatas, 2010), MABC (Akay ve Karaboga, 2012), IABC (Aydın vd., 2012; Aydın ve Özyön, 2013), ImpABC (Gao ve Liu, 2011), GABC (Diwold vd., 2011; W. Gao vd., 2012), GDABC (Diwold vd., 2011) ve EABC (Gao vd., 2014) karşılaştırmada yer almışlardır. Tüm bu algoritmalar doğrudan ABC-X tarafından üretilebilirler. Yapılandırılmış ABC-X varyantları ile ABC varyantlarının karşılaştırmasının adil olması ve elde edilen farklılıkların iyileştirilmiş sayısal parametre ayarlarından dolayı olmaması için bütün ABC varyantların parametreleri ABC-X algoritmasındaki gibi yapılandırılmıştır. Algoritmalar hazırlanan ölçüt seti ile test edilmişlerdir. Bu prosedürün bir diğer yönü, çeşitli ABC varyantlarının, diğer ölçüt fonksiyon setlerine göre parametreleri yapılandırılmış olabilmektedir. Bu yüzden ilgili yazarlar tarafından önerilen varsayılan parametre ayarlarını bu şekilde kullanmak yazarlara karşı haksızlık olacaktır. ABC-X ile diğer ABC algoritmaların yapılandırmaları arasındaki son karşılaştırmaya, 10 boyutlu eğitim setindeki en iyi olan algoritma

yapılandırmaları dâhil edilmiştir. Ek olarak, sonuçları derinlemesine incelemek için rastgele seçilmiş olan bir yapılandırmanın sonuçları da sunulmuştur. Bu rastgele yapılandırmadaki sonuçlar, bağımsız olarak eğitilmiş beş yapılandırmadan ilkinin sonuçlarıdır.

Parametre ayarlamanın, ABC algoritmasının performansı üzerindeki etkisini göstermek için Şekil 6.1'de görülen korelasyon grafiklerini kullanarak ayarlanmış parametreli algoritmalar ile varsayılan parametreli algoritmaların sonuçları karşılaştırılmıştır. Her bir şekil 10, 30 ve 50 boyutlu fonksiyonlardaki bir ABC varyantına ait varsayılan yapılandırma ile ayarlanmış yapılandırmanın medyan sonuçlarını karşılaştırmaktadır. Varsayılan parametre ayarlarıyla gözlemlenen medyan sonuçlar x ekseninde ve ayarlanmış parametre ile elde edilen medyan sonuçlar y ekseninde verilmiştir. İçteki şekiller, yer alan farklılıkların daha net görülebilmesi için belirli alanlara yakınlaştırılarak sonuçları sunmaktadır. Ayrıca ayarlanmış parametre ayarının, varsayılan parametre ayarından daha iyi (win), eşit (draw) veya daha kötü (lost) olma durumunun sayısı verilmiştir. (Her bir boyut için her fonksiyonda istatistiksel olarak anlamlı farklılıkların ortaya çıkıp çıkmadığını belirlemek için Wilcoxon testi kullanılmıştır. İstatistiksel olarak anlamlı bir fark gözlenmezse, bu durum berabere olarak rapor edilmiştir.) Şekil 6.1'de görüldüğü gibi, ABC, BABC, CABC, MABC ve ImpABC'de parametre ayarlaması ile her boyutta önemli ve büyük gelişmeler gözlemlenmiştir. Ayrıca, IABC, GDABC ve EABC'nin ayarlanmış sürümleri, varsayılan parametreli haline göre daha iyi olmuştur. Ancak bu çok kuvvetli değildir. Buna karşın GABC'nin varsayılan ve ayarlanmış parametrelerinin sonuçları istisnai olarak çok benzer olduğu görülmüştür.

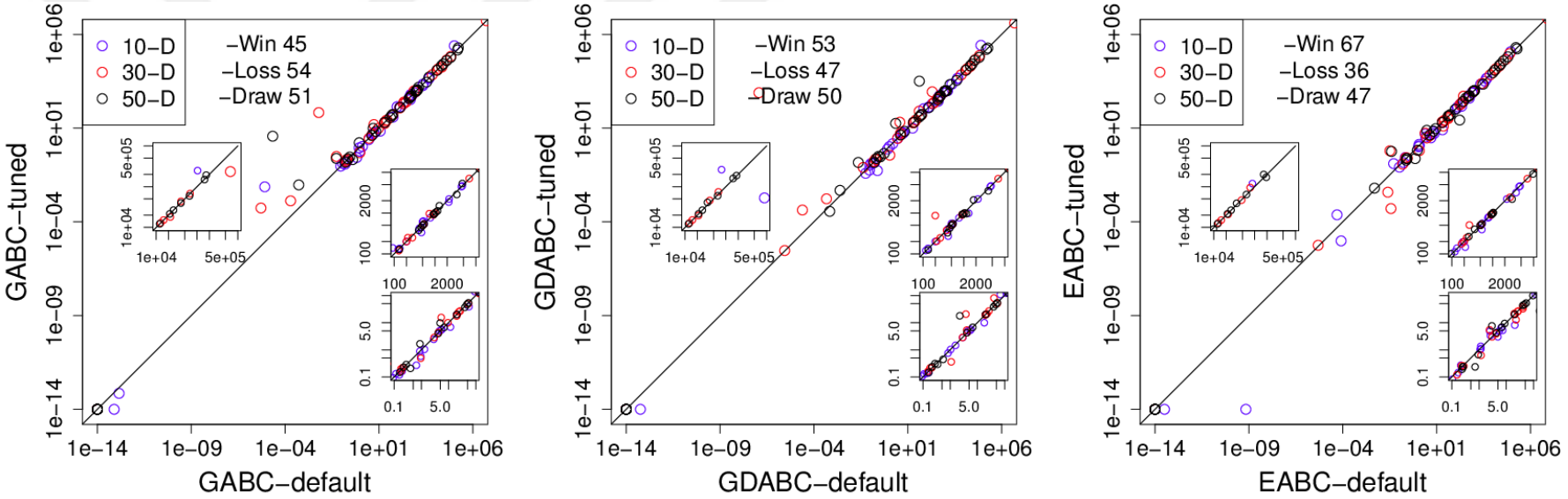


Şekil 6.1. *10, 30 ve 50 boyutlu ölçüt fonksiyonlarında ABC algoritmalarının varsayılan (x ekseni) ve ayarlanmış (y ekseni) yapılandırılmaları ile elde edilen medyan amaç fonksiyon değerlerinin korelasyon grafiği. Her bir nokta, sırasıyla ayarlanmış ve varsayılan yapılandırmalardaki algoritma ile elde edilen medyan amaç değerini temsil etmektedir. Üst üçgen üzerindeki bir nokta, varsayılan ayarın ayarlanmıştan daha iyi olduğunu göstermektedir; sağ alt üçgenin üzerindeki bir nokta, ayarlanmış ayarın varsayılan ayardan üstün olduğunu göstermektedir. Algoritmanın y eksenindeki (yani, ayarlanmış sürüm) karşılaştırma sonuçları sırasıyla -Win, -Loss ve -Draw sayısı biçiminde sunulmuştur.*



Şekil 6.1. (Devam) *10, 30 ve 50 boyutlu ölçüt fonksiyonlarında ABC algoritmalarının varsayılan (x ekseni) ve ayarlanmış (y ekseni) yapılandırılmaları ile elde edilen medyan amaç fonksiyon değerlerinin korelasyon grafiği. Her bir nokta, sırasıyla ayarlanmış ve varsayılan yapılandırmalardaki algoritma ile elde edilen medyan amaç değerini temsil etmektedir. Üst üçgen üzerindeki bir nokta, varsayılan ayarın ayarlanmıştan daha iyi olduğunu göstermektedir; sağ alt üçgenin üzerindeki bir nokta, ayarlanmış ayarın varsayılan ayardan üstün olduğunu göstermektedir. Algoritmanın y eksenindeki (yani, ayarlanmış sürüm) karşılaştırma sonuçları sırasıyla -Win, -Loss ve -Draw sayısı biçiminde sunulmuştur.*

.



Şekil 6.1. (Devam) *10, 30 ve 50 boyutlu ölçüt fonksiyonlarında ABC algoritmalarının varsayılan (x ekseni) ve ayarlanmış (y ekseni) yapılandırılmaları ile elde edilen medyan amaç fonksiyon değerlerinin korelasyon grafiği. Her bir nokta, sırasıyla ayarlanmış ve varsayılan yapılandırmalardaki algoritma ile elde edilen medyan amaç değerini temsil etmektedir. Üst üçgen üzerindeki bir nokta, varsayılan ayarın ayarlanmıştan daha iyi olduğunu göstermektedir; sağ alt üçgenin üzerindeki bir nokta, ayarlanmış ayarın varsayılan ayardan üstün olduğunu göstermektedir. Algoritmanın y eksenindeki (yani, ayarlanmış sürüm) karşılaştırma sonuçları sırasıyla -Win, -Loss ve -Draw sayısı biçiminde sunulmuştur.*

6.2.1.3. *Unimodal fonksiyon deneyleri*

Bahsedildiği gibi, ABC-X beş kez yapılandırılmıştır. Böylece 10, 30 ve 50 boyutlu unimodal fonksiyonlar üzerinde test edilmiş beş farklı ABC algoritması oluşturulmuştur. Bu beş ABC-X algoritmasının medyan sonuçları Tablo 6.5'de verilmiştir. f1 ve f8-f11 arasında, tüm yapılandırmalar her boyutta en iyi çözümü elde etmişlerdir. Dahası, tüm yapılandırmalar f5 ile f7 arasındaki on boyutlu fonksiyonlarda optimum eşiğe ulaşmışlardır. İkinci ABC-X yapılandırması ABC-X_2^u, birinci sırayı elde etmiştir ve Wilcoxon'un testi ile Holm'un çoklu test düzeltmesi dikkate alındığında en az bir diğer yapılandırmaya göre istatistiksel olarak daha iyi performans göstermiştir.

Algoritma 6.2 ve Algoritma 6.3, sırasıyla ABC-X çatısının ilk yapılandırması ABC-X_1^u'e ve en iyi yapılandırması ABC-X_2^u'ye ait sözde kodları göstermektedirler. Sözde kodlar incelendiğinde, bu algoritmaların ikisinin de daha önce literatürde sunulmadığı anlaşılmaktadır. Örneğin, işçi ve gözcü arı adımlarındaki arama denklemleri birbirinden farklıdır, bununla birlikte her ikisi de global en iyi aday çözümüne karşı yönelmektedir. Ayrıca arama denklemlerindeki terimlerin sayısı, çoğu ABC algoritmasından farklı olarak üçtür. Bununla birlikte, her adımda değiştirilen değişkenlerin sayısı birden fazladır ve belirli bir aralıktan rastgele seçilirler. Sonucunda, bu yapılandırmalar ABC algoritmasının yeni varyantlarına karşılık gelmektedir.

Tablo 6.6, ABC-X çatısının iki yapılandırması ile ayarlanmış ABC varyantlarının 10, 30 ve 50 boyut için unimodal fonksiyonlarda yapılan karşılaştırmaların sonuçlarını sunmaktadır. Sonuçlar, ABC-X_1^u ve ABC-X_2^u'in en üst sıradaki algoritmalar olduğunu göstermektedir. ABC-X'in en iyi yapılandırmasının, ImpABC algoritması haricindeki diğer tüm algoritmalardan önemli ölçüde daha iyi olduğu görülmektedir.

Tablo 6.5. *Unimodal fonksiyonlarda irace tarafından elde edilen beş ABC-X yapılandırmasının medyan sonuçları ($ABC-X_1^u$'den $ABC-X_5^u$'e gösterilir) ve tüm fonksiyon tiplerine göre ayarlanmış $ABC-X^{all}$'a ait sonuçlar yer almaktadır. Sıralama, tüm fonksiyonlar arasında ortalama sıralamayı verir. Wilcoxon T. satırı bir algoritma yapılandırmasının, en iyi olandan istatistiksel olarak anlamlı derecede daha kötü olduğunu (+ ile gösterilir) veya olmadığını ("≈" ile gösterilir) gösterir.*

Boy	Fonk.	$ABC-X_1^u$	$ABC-X_2^u$	$ABC-X_3^u$	$ABC-X_4^u$	$ABC-X_5^u$	$ABC-X^{all}$
10	f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f2	3.42E+04	**1.71E+03**	1.76E+04	2.82E+04	7.08E+03	1.40E+04
	f3	1.92E+03	2.03E+03	2.41E+03	1.91E+03	1.20E+03	**9.77E+02**
	f4	3.16E-01	**4.17E-02**	4.70E-01	4.31E+00	1.28E+01	2.38E+00
	f5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f12	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
30	f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f2	**3.67E+06**	3.82E+06	4.91E+06	4.71E+06	7.87E+06	6.20E+06
	f3	**6.38E-01**	1.53E+00	4.34E+02	1.93E+01	7.73E+00	4.03E+02
	f4	1.42E-02	3.93E+00	2.11E-01	1.89E+00	5.63E+00	**3.64E-06**
	f5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**3.24E-06**	**0.00E+00**	**0.00E+00**
	f6	1.95E+00	**3.81E-03**	3.96E+01	2.86E+01	3.89E+00	1.32E+02
	f7	1.31E+03	**1.52E+02**	7.52E+03	1.53E+03	1.03E+03	4.04E+03
	f8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f12	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
50	f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f2	5.30E+06	**5.06E+06**	1.03E+07	8.63E+06	1.04E+07	1.45E+07
	f3	2.96E+03	4.79E+03	**2.93E+03**	5.69E+03	3.20E+03	2.96E+03
	f4	1.04E+04	9.69E+03	1.28E+04	1.24E+04	1.33E+04	**2.94E+02**
	f5	5.77E-04	**3.11E-08**	2.77E-06	7.59E-03	1.41E-04	1.02E-03
	f6	**1.32E+02**	2.76E+02	2.44E+03	1.55E+03	1.74E+03	7.60E+03
	f7	3.20E+04	1.68E+04	3.62E+04	**1.25E+04**	2.03E+04	4.98E+04
	f8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f12	1.22E-05	0.00E+00	1.98E-08	1.15E-03	2.17E-06	**0.00E+00**
Sıralama		1.75	**1.47**	2.31	2.42	2.22	2.17
Wilcox. T.		≈		+	≈	≈	≈

Algoritma 6.2. *Unimodal fonksiyonlar için elde edilen ABC-X_1^{II}'e ait sözde kod*

SN = 67, limitF = 0.3381
Kaotik tend harita kullanarak ilklendirme uygula
for her bir işçi arı x_i **do**
$n_e = 5,\ k_e = 9$
for m = 1 to $rand(n_e, k_e)$ **do**
j boyutunu rastgele seç
$\phi_1 = rand(-0.7, 0.7),\ \phi_2 = rand(-1,1)$
$\hat{x}_{i,j} = x_{G,j} + \phi 1(x_{i,j} - x_{G,j}) + \phi 2(x_{i,j} - x_{GD,j})$
if $\hat{x}_i$ < xi **then** $x_i = \hat{x}_i,\ trial_i = 0$ **else** $trial_i = trial_i + 1$
Seçme olasılığını ağırlıklı strateji ile hesapla
for her bir gözcü arı x_i **do**
$n_o = 5$
for m = 1 to n_o **do**
j boyutunu rastgele seç
$\phi_1 = rand(-1,1)$
$\hat{x}_{i,j} = x_{G,j} + \phi 1(x_{i,j} - x_{G,j})$
if $\hat{x}_i$ < xi **then** $x_i = \hat{x}_i,\ trial_i = 0$ **else** $trial_i = trial_i + 1$
Varsayılan kaşif arı stratejisini uygula

Algoritma 6.3. *Unimodal fonksiyonlar için elde edilen ABC-X_2^{II}'ye ait sözde kod*

SN=58, limitF=0.54
Varsayılan ilklendirme stratejisini uygula
for her bir işçi arı x_i **do**
$n_e = 4, k_e = 8$
for m = 1 to $rand(n_e, k_e)$ **do**
j boyutunu rastgele seç
$\phi_1 = rand(-0.55, 0.55), \phi_2 = rand(-0.46, 0.46)$
$v_{i,j} = x_{G,j} + \phi 1(x_{i,j} - x_{G,j}) + \phi 2(x_{i,j} - x_{r1,j})$
if $\hat{x}_i$ < xi **then** $x_i = \hat{x}_i,\ trial_i = 0$ **else** $trial_i = trial_i + 1$
Seçme olasılığını varsayılan strateji ile hesapla
for her bir gözcü arı x_i **do**
$n_o = 4, k_o = 8$
for m = 1 to $rand(n_o, k_o)$ **do**
j boyutunu rastgele seç
$\phi_1 = rand(-3.34, 3.34), \phi_2 = rand(-0.05, 0.05)$
$\hat{x}_{i,j} = x_{G,j} + \phi 1(x_{i,j} - x_{G,j}) + \phi 2(x_{G,j} - x_{r1,j})$
if $\hat{x}_i$ < xi **then** $x_i = \hat{x}_i,\ trial_i = 0$ **else** $trial_i = trial_i + 1$
chaoticSearch1 kaşif arı stratejisini Tend harita ile uygula

Tablo 6.6. *ABC-X'in; en iyi yapılandırması ($ABC-X_2^u$), ilk yapılandırması ($ABC-X_1^u$) ve ABC varyantlarının Unimodal fonksiyonlarındaki medyan hata değerleri*

Boy.	Fonk.	$ABC-X_1^u$	$ABC-X_2^u$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
10	f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f2	3.42E+04	1.71E+03	2.69E+05	2.15E+06	1.19E+05	2.48E+05	2.61E+05	1.16E+05	2.28E+03	5.34E+04	1.03E+06
	f3	1.92E+03	2.03E+03	6.78E+01	1.16E+02	1.46E+02	3.90E+01	9.97E+00	6.53E+00	8.14E+02	7.27E+01	9.88E+01
	f4	3.16E-01	4.17E-02	4.42E+02	6.20E+02	1.95E+02	1.39E+02	1.09E+02	3.26E+01	3.32E-03	8.02E+01	2.65E+02
	f5	**0.00E+00**	**0.00E+00**	6.25E-05	2.16E-07	**0.00E+00**	7.11E-14	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.71E-12	1.45E-03
	f6	**0.00E+00**	**0.00E+00**	1.93E+00	3.88E-01	5.23E+00	5.85E-01	5.43E-02	1.84E+00	**0.00E+00**	3.60E-09	1.20E+00
	f7	**0.00E+00**	**0.00E+00**	6.23E+02	2.51E+01	3.95E+02	3.33E+02	2.72E+02	3.99E+01	**0.00E+00**	1.53E-01	4.07E+02
	f8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f12	**0.00E+00**	**0.00E+00**	7.53E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.56E-01
30	f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f2	3.67E+06	3.82E+06	6.23E+06	7.88E+06	6.90E+06	5.39E+06	4.05E+06	8.70E+06	**2.45E+05**	2.11E+07	4.92E+06
	f3	6.38E-01	1.53E+00	1.92E+02	7.47E+03	5.08E+02	1.20E+02	8.07E+01	7.87E+00	**0.00E+00**	2.98E+00	1.73E+02
	f4	1.42E-02	3.93E+00	3.46E+01	2.67E+02	3.26E+01	1.56E+01	5.32E+00	7.71E-01	**0.00E+00**	2.73E-01	1.56E+01
	f5	**0.00E+00**	**0.00E+00**	2.40E-01	2.24E+00	3.83E-03	1.32E-03	4.28E-04	5.97E-02	2.17E-04	7.04E-03	6.50E-01
	f6	1.95E+00	3.81E-03	1.06E+03	1.22E+02	2.43E+03	9.69E+02	3.78E+02	1.32E+03	**0.00E+00**	8.99E+01	3.21E+02
	f7	1.31E+03	1.52E+02	2.37E+04	2.28E+03	2.09E+04	1.80E+04	1.95E+04	2.82E+04	**4.19E+00**	6.47E+03	7.69E+03
	f8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.32E+01	**0.00E+00**	**0.00E+00**
	f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.29E+01	**0.00E+00**	**0.00E+00**
	f12	**0.00E+00**	**0.00E+00**	5.06E+01	5.44E+00	5.12E-04	6.77E+01	7.76E+02	1.71E+03	7.28E-11	3.19E-01	1.17E+02

Tablo 6.6 (Devam) *ABC-X'in; en iyi yapılandırması ($ABC-X_2^u$), ilk yapılandırması ($ABC-X_1^u$) ve ABC varyantlarının Unimodal fonksiyonlarındaki medyan hata değerleri*

Boy	Fonk.	$ABC-X_1^u$	$ABC-X_2^u$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
50	f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f2	5.30E+06	5.06E+06	9.58E+06	7.87E+06	1.26E+07	9.43E+06	8.96E+06	1.34E+07	**5.73E+05**	5.41E+07	9.28E+06
	f3	2.96E+03	4.79E+03	**1.61E+03**	2.66E+03	2.20E+03	2.85E+03	2.10E+03	1.63E+03	2.96E+03	2.68E+03	1.70E+03
	f4	1.04E+04	9.69E+03	4.96E+04	4.86E+04	3.89E+04	3.95E+04	4.09E+04	3.92E+04	**6.93E+01**	9.15E+03	5.87E+04
	f5	5.77E-04	**3.11E-08**	1.80E+00	1.80E+01	2.37E-01	2.41E-01	1.48E-01	3.35E+00	2.82E+00	8.48E-01	2.25E+00
	f6	1.32E+02	2.76E+02	5.80E+03	6.08E+02	9.07E+03	5.31E+03	3.52E+03	5.83E+03	**1.66E-09**	7.46E+03	1.19E+03
	f7	3.20E+04	1.68E+04	6.78E+04	6.04E+03	6.41E+04	5.92E+04	5.97E+04	8.13E+04	**5.57E+02**	4.19E+04	1.66E+04
	f8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f9	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.15E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.80E+02	**0.00E+00**	**0.00E+00**
	f10	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.05E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.15E+00	**0.00E+00**	**0.00E+00**
	f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.89E+02	**0.00E+00**	**0.00E+00**
	f12	1.22E-05	**0.00E+00**	3.25E+02	3.76E+03	2.69E+01	1.51E+03	3.23E+03	5.96E+03	3.63E-09	1.55E+02	6.40E+02
Sıralama		2.14	**2.08**	5.28	5.17	4.52	4.22	3.69	4.44	3.25	3.67	4.86
Wilcoxon T.		≈		+	+	+	+	+	+	≈	+	+

6.2.1.4. *Multimodal fonksiyon deneyleri*

Multimodal fonksiyonlar için ABC-X'in ayarlanması, benzer sonuçlar üreten beş yapılandırma elde edildi ve bunlar arasında Tablo 6.7'de gösterildiği gibi istatistiksel olarak anlamlı bir fark gözlenmedi. Daha detaylı olarak, tüm yapılandırmalar f13, f17 ve f19 fonksiyonları için optimum eşiğe ulaşmışlardır. Ayrıca f14, f16 ve f21 fonksiyonlarında, beş yapılandırma tüm boyutlar için yaklaşık olarak aynı değerleri bulmuşlardır. Bu davranış; bir taraftan uygun şekilde ayarlanmış ABC-X yapılandırmalardan elde edilebilecek sağlam performansı, diğer taraftan ise irace ile sürekli olarak yüksek performanslı yapılandırmalar elde edildiğini göstermektedir.

Bu beş yapılandırmanın karşılaştırılmasına göre, beşinci yapılandırma (ABC-X_5^m) 10 boyutlu eğitim verilerinde en iyi sıralamayı elde etti. Bu nedenle bir sonraki adımda ilk sıradaki yapılandırma ABC-X_1^m, en iyi sıralamaya sahip yapılandırma ABC-X_5^m ve parametreleri ayarlanmış ABC algoritmaların elde ettiğin medyan hata değerleri Tablo 6.8'de listelenmiştir. ABC-X'in en iyi yapılandırması; ABC, BABC, GABC, GDABC, MABC ve CABC algoritmalarından çok daha iyi performans göstermiştir. ABC-X_5^m, genel olarak en düşük ortalama sıralama değerini elde etmiştir. Ayrıca EABC, IABC ve ImpABC; ABC-X_5^m'den istatistiksel olarak daha kötü olmamışlardır.

Sırasıyla Algoritma 6.4 ve Algoritma 6.5'de verilen ABC-X'in ilk ve en iyi yapılandırma arasındaki benzerlikleri incelendiğinde, her ikisi de artan popülasyon boyut stratejisini ve kâşif arı adımı için ise *IABCSBStrategy* seçeneğini kullanmaları dikkat çekicidir. Unimodal fonksiyonların ayarlarına benzer olarak, arama denklemlerinde üç veya daha fazla terim kullanılmış ve global en iyi aday çözümüne doğru yönelmiştir. Bununla birlikte, unimodal durumundan farklı olarak, her iki algoritmadaki arama denklemlerinde yalnızca bir boyut değiştirilmektedir. Bu sonuç, iki algoritmadaki parametre ayarlarından çıkarılabilir. Burada sadece Algoritma 6.4 içindeki gözcü arı adımında beş değişken her defasında değiştirilirken, diğer durumlarda işçi ve gözcü arı adımlarında her defasında yalnızca bir değişken değiştirilir.

Tablo 6.7. *Multimodal fonksiyonlarda irace tarafından elde edilen beş ABC-X yapılandırmasının medyan sonuçları ($ABC-X_1^m$ 'den $ABC-X_5^m$ 'e gösterilir) ve tüm fonksiyon tiplerine göre ayarlanmış $ABC-X^{all}$ 'a ait sonuçlar yer almaktadır. Detaylı açıklama Tablo 6.5'de verilmiştir.*

Boy.	Fonk	$ABC-X_1^m$	$ABC-X_2^m$	$ABC-X_3^m$	$ABC-X_4^m$	$ABC-X_5^m$	$ABC-X^{all}$
10	f13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f14	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	2.01E+01
	f15	3.95E+00	2.03E-01	**6.88E-04**	3.44E-01	5.22E-02	4.46E+00
	f16	**5.63E+00**	**5.63E+00**	**5.63E+00**	**5.63E+00**	**5.63E+00**	**5.63E+00**
	f17	9.86E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f18	2.48E-02	1.29E-02	7.40E-03	8.79E-03	**1.87E-04**	1.48E-02
	f19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f20	3.98E+00	4.97E+00	6.96E+00	3.98E+00	4.55E+00	**2.98E+00**
	f21	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**
	f22	**4.45E+03**	4.46E+03	4.81E+03	4.58E+03	4.69E+03	4.47E+03
	f23	**2.77E+00**	4.48E+00	3.81E+00	4.18E+00	4.97E+00	2.94E+00
	f24	8.26E-02	**3.94E-02**	4.78E-02	1.08E-01	1.38E-01	1.73E-01
	f25	8.56E-02	1.17E-01	1.02E-01	8.97E-02	**8.39E-02**	1.03E-01
	f26	1.12E-01	1.22E-01	**4.85E-02**	7.96E-02	1.02E-01	6.39E-02
30	f13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f14	**2.00E+01**	**2.00E+01**	**2.00E+01**	2.01E+01	**2.00E+01**	2.02E+01
	f15	7.76E+01	2.56E+01	4.99E+00	2.13E+01	**1.63E+00**	1.56E+01
	f16	2.52E+01	2.35E+01	2.35E+01	2.50E+01	2.25E+01	**2.05E+01**
	f17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f18	**0.00E+00**	1.79E-07	1.40E-07	8.78E-07	1.04E-06	**0.00E+00**
	f19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f20	4.65E+01	4.89E+01	5.37E+01	4.19E+01	**4.08E+01**	4.69E+01
	f21	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	1.30E+04
	f22	1.52E+04	**1.51E+04**	**1.51E+04**	1.52E+04	**1.51E+04**	1.52E+04
	f23	**2.09E+01**	2.41E+01	2.31E+01	2.35E+01	2.62E+01	2.74E+01
	f24	1.71E-01	**7.81E-02**	1.06E-01	1.87E-01	1.27E-01	2.10E-01
	f25	2.34E-01	2.95E-01	2.45E-01	**2.31E-01**	2.52E-01	2.44E-01
	f26	2.41E-01	2.38E-01	1.64E-01	2.04E-01	1.93E-01	**1.37E-01**
50	f13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f14	2.03E+01	**2.00E+01**	**2.00E+01**	2.03E+01	**2.00E+01**	2.02E+01
	f15	9.36E+01	8.16E+01	2.10E+00	6.96E+01	**1.32E+00**	8.20E+01
	f16	**4.70E+01**	**4.70E+01**	**4.70E+01**	**4.70E+01**	**4.70E+01**	**4.70E+01**
	f17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f18	1.35E-05	7.92E-04	4.18E-04	2.51E-03	5.41E-03	**0.00E+00**
	f19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f20	1.20E+02	1.15E+02	1.19E+02	**1.01E+02**	1.03E+02	1.23E+02
	f21	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	2.17E+04
	f22	2.75E+04	2.62E+04	**2.61E+04**	2.68E+04	2.64E+04	2.67E+04
	f23	**4.49E+01**	5.01E+01	5.02E+01	5.07E+01	5.24E+01	5.49E+01
	f24	3.19E-01	**9.78E-02**	1.14E-01	2.61E-01	1.37E-01	2.05E-01
	f25	3.52E-01	4.03E-01	3.54E-01	**3.34E-01**	3.59E-01	3.47E-01
	f26	2.64E-01	2.81E-01	**2.00E-01**	2.35E-01	2.13E-01	6.28E-01
Sıralama		2.79	2.62	**2.07**	2.67	2.36	3.17
Wilcoxon T.		$\approx$	$\approx$		$\approx$	$\approx$	$\approx$

Tablo 6.8. *ABC-X'in; en iyi yapılandırması (ABC- X_5^m), ilk yapılandırması (ABC- X_1^m) ve ABC varyantlarının Multimodal fonksiyonlarındaki medyan hata değerleri*

Boy.	Fonk.	$ABC-X_1^m$	$ABC-X_5^m$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
10	f13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f14	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	2.01E+01	**2.00E+01**	2.01E+01
	f15	3.95E+00	**5.22E-02**	1.16E+00	6.97E+00	4.86E+00	2.70E-01	1.36E-01	1.56E-01	2.69E-01	6.93E-01	4.85E+00
	f16	**5.63E+00**	**5.63E+00**	5.64E+00	6.54E+00	6.41E+00	**5.63E+00**	**5.63E+00**	5.75E+00	**5.63E+00**	5.64E+00	5.84E+00
	f17	9.86E-03	**0.00E+00**	**0.00E+00**	1.20E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.04E-11	1.68E-10
	f18	2.48E-02	1.87E-04	9.54E-03	1.51E-02	**9.66E-06**	7.40E-03	1.90E-03	9.62E-04	2.95E-02	1.80E-02	1.07E-02
	f19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f20	3.98E+00	4.55E+00	9.52E+00	1.70E+01	3.03E+00	5.00E+00	5.00E+00	5.97E+00	**2.98E+00**	7.56E+00	1.05E+01
	f21	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**	**4.45E+03**
	f22	**4.45E+03**	4.69E+03	4.86E+03	4.91E+03	4.60E+03	4.63E+03	4.63E+03	4.79E+03	4.48E+03	4.81E+03	4.84E+03
	f23	2.77E+00	4.97E+00	6.19E+00	**3.00E-01**	5.01E+00	5.42E+00	5.20E+00	4.12E+00	1.11E+00	5.66E+00	2.94E+00
	f24	**8.26E-02**	1.38E-01	1.46E-01	1.62E-01	2.70E-01	1.64E-01	1.45E-01	8.45E-02	1.44E-01	2.10E-01	2.12E-01
	f25	8.56E-02	8.39E-02	1.36E-01	1.46E-01	8.29E-02	9.43E-02	1.07E-01	**7.87E-02**	1.20E-01	1.67E-01	1.53E-01
	f26	1.12E-01	1.02E-01	1.64E-01	1.46E-01	1.46E-01	1.32E-01	1.30E-01	**5.26E-02**	8.72E-02	1.74E-01	1.66E-01
30	f13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f14	**2.00E+01**	**2.00E+01**	2.01E+01	2.01E+01	2.02E+01	2.01E+01	2.01E+01	**2.00E+01**	2.06E+01	2.04E+01	2.01E+01
	f15	7.76E+01	**1.63E+00**	2.25E+00	2.18E+01	1.28E+01	9.96E+00	2.06E+01	7.03E+00	2.30E+01	1.26E+01	2.85E+00
	f16	2.52E+01	2.25E+01	2.03E+01	2.50E+01	2.56E+01	2.50E+01	2.13E+01	2.29E+01	**1.94E+01**	2.56E+01	2.00E+01
	f17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f18	**0.00E+00**	1.04E-06	2.50E-05	8.64E-03	5.64E-06	5.48E-04	2.95E-06	2.50E-06	0.00E+00	3.40E-09	1.37E-06
	f19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f20	4.65E+01	4.08E+01	9.93E+01	1.62E+02	**3.83E+01**	4.99E+01	5.10E+01	7.49E+01	4.20E+01	7.88E+01	9.27E+01
	f21	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	**1.29E+04**	1.30E+04	**1.29E+04**	**1.29E+04**
	f22	1.52E+04	1.51E+04	1.55E+04	1.57E+04	1.51E+04	1.52E+04	1.53E+04	1.52E+04	**1.48E+04**	1.63E+04	1.55E+04
	f23	2.09E+01	2.62E+01	2.88E+01	**2.16E+00**	2.66E+01	2.70E+01	2.67E+01	2.47E+01	5.32E+00	2.66E+01	1.07E+01
	f24	1.71E-01	**1.27E-01**	1.65E-01	2.01E-01	2.61E-01	1.66E-01	1.68E-01	1.47E-01	7.44E-01	5.06E-01	1.92E-01
	f25	2.34E-01	2.52E-01	2.52E-01	2.78E-01	**1.20E-01**	2.00E-01	2.16E-01	1.94E-01	2.51E-01	3.63E-01	2.53E-01
	f26	2.41E-01	1.93E-01	2.60E-01	2.57E-01	2.52E-01	2.23E-01	2.21E-01	**1.56E-01**	1.99E-01	2.87E-01	2.39E-01

Tablo 6.8. (Devam) *ABC-X'in; en iyi yapılandırması (ABC-* X_5^m *), ilk yapılandırması (ABC-* X_1^m *) ve ABC varyantlarının Multimodal fonksiyonlarındaki medyan hata değerleri*

Boy.	Fonk.	$ABC-X_1^m$	$ABC-X_5^m$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
50	f13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f14	2.03E+01	**2.00E+01**	2.02E+01	2.02E+01	2.03E+01	2.01E+01	2.01E+01	2.01E+01	2.09E+01	2.08E+01	2.01E+01
	f15	9.36E+01	1.32E+00	**1.43E-01**	2.06E+00	9.34E+00	9.73E+00	1.79E+01	4.73E-01	7.15E+01	7.62E+01	4.30E-01
	f16	4.70E+01	4.70E+01	**4.43E+01**	4.70E+01	4.70E+01	4.70E+01	4.70E+01	4.44E+01	4.70E+01	4.70E+01	4.70E+01
	f17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.44E-02	**0.00E+00**	**0.00E+00**
	f18	1.35E-05	5.41E-03	5.50E-03	1.97E-02	6.30E-03	9.38E-03	3.65E-04	5.59E-03	**0.00E+00**	1.44E-05	2.11E-03
	f19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.08E+01	3.54E-08	**0.00E+00**
	f20	1.20E+02	1.03E+02	2.17E+02	3.39E+02	**9.99E+01**	1.28E+02	1.25E+02	1.96E+02	1.21E+02	2.35E+02	2.07E+02
	f21	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	**2.16E+04**	2.21E+04	**2.16E+04**	**2.16E+04**
	f22	2.75E+04	**2.64E+04**	2.70E+04	2.71E+04	**2.64E+04**	2.66E+04	2.65E+04	2.69E+04	**2.64E+04**	3.07E+04	2.72E+04
	f23	4.49E+01	5.24E+01	5.76E+01	**4.52E+00**	5.29E+01	5.40E+01	5.42E+01	5.29E+01	9.55E+00	5.51E+01	2.39E+01
	f24	3.19E-01	**1.37E-01**	1.63E-01	2.09E-01	2.38E-01	1.51E-01	1.46E-01	1.44E-01	1.36E+00	1.02E+00	1.80E-01
	f25	3.52E-01	3.59E-01	3.65E-01	4.50E-01	**1.94E-01**	2.86E-01	3.11E-01	3.07E-01	3.42E-01	4.95E-01	3.77E-01
	f26	2.64E-01	2.13E-01	2.49E-01	2.71E-01	2.37E-01	2.29E-01	2.38E-01	**1.72E-01**	2.65E-01	2.95E-01	2.42E-01
Sıralama		4.74	**3**	5.62	6.98	4.93	4.26	4.26	3.36	4.93	7.57	5.48
Wilcoxon T.		≈		+	+	≈	+	+	≈	≈	+	+

Algoritma 6.4. *Multimodal fonksiyonlar için elde edilen ABC-X_1^m 'e ait sözde kod*

$SN = 31$, $limitF = 1.06$
Varsayılan ilklendirme stratejisini uygula
for her bir işçi arı x_i **do**
    $n_e = 1$ (default)
    for m = 1 to n_e **do**
        j boyutunu rastgele seç
        $\phi_1 = rand(-2.04, 2.04)$, $\phi_2 = rand(-1,1)$
        $\hat{x}_{i,j} = x_{r1,j} + \phi 1(x_{G,j} - x_{r2,j}) + \phi 2(x_{i,j} - x_{GD,j})$
    if $\hat{x}_i$ < x_i **then** $x_i = \hat{x}_i$, $trial_i = 0$ **else** $trial_i = trial_i + 1$
Seçme olasılığını ağırlıklı strateji ile hesapla
for her bir işçi arı x_i **do**
    $n_o = 5$
    for m = 1 to n_o **do**
        j boyutunu rastgele seç
        $\phi_1 = rand(-0.65, 0.65)$, $\phi_2 = rand(-0.17, 0.17)$
        $\hat{x}_{i,j} = x_{G,j} + \phi 1(x_{i,j} - x_{GD,j}) + \phi 2(x_{i,j} - x_{r1,j})$
    if $\hat{x}_i$ < x_i **then** $x_i = \hat{x}_i$, $trial_i = 0$ **else** $trial_i = trial_i + 1$
IABCSBStrategy kaşif arı stratejisini $Rf = 10^{-10}$ olarak uygula
$g = 15, SN_{max} = 66$ olarak popülasyon boyutunu güncelle

Algoritma 6.5. *Multimodal fonksiyonlar için elde edilen ABC-X_5^m 'e ait sözde kod*

$SN = 14, limitF = 0.83$
Karşıtlık tabanlı ilklendirme stratejisini uygula
for her bir işçi arı x_i **do**
    $n_e = 1$ (default)
    for m = 1 to n_e **do**
        j boyutunu rastgele seç
        $\phi_1 = rand(-0.55, 0.55), \phi_2 = rand(-1,1)$
        $\hat{x}_{i,j} = x_{G,j} + \phi 1(x_{G,j} - x_{i,j}) + \phi 2(x_{i,j} - x_{GD,j})$
    if $\hat{x}_i$ < x_i **then** $x_i = \hat{x}_i$, $trial_i = 0$ **else** $trial_i = trial_i + 1$
Seçme olasılığını SP=1.23 olmak üzere rank-tabanlı stratejisi ile hesapla
for her bir işçi arı x_i **do**
    $n_o = 5$
    for m = 1 to n_o **do**
        j boyutunu rastgele seç
        $\phi_1 = rand(-1,1), \phi_2 = rand(-1,1), \phi_3 = rand(-1,1)$
        $\hat{x}_{i,j} = x_{G,j} + \phi 1(x_{G,j} - x_{i,j}) + \phi 2(x_{i,j} - x_{GD,j}) + \phi 3(x_{i,j} - x_{r1,j})$
    if $\hat{x}_i$ < x_i **then** $x_i = \hat{x}_i$, $trial_i = 0$ **else** $trial_i = trial_i + 1$
IABCSBStrategy kaşif arı strajesini $Rf = 10^{-5.1}$ olarak uygula
$g = 17, SN_{max} = 78$ olarak popülasyon boyutunu güncelle

6.2.1.5. *Hibrit fonksiyon deneyleri*

Bu bölüm, hibrit fonksiyon sonuçlarının analizine odaklanmıştır. Sekiz ötelenmiş SOCO fonksiyonu, altı ötelenmiş-döndürülmüş (shifted-rotated) CEC'14 ve on adet hibrit kompozit (hybrid composition) CEC'05 fonksiyonlarından oluşmuştur. Bunlar, sürekli optimize ediciler için zor problemler olması amaçlanarak tasarlanmıştır. Bu, özellikle CEC'05 ve CEC'14 ölçüt fonksiyonları için geçerlidir.

Beş ABC-X^h yapılandırması için elde edilen sonuçlar Tablo 6.9'da verilmiştir. Bu sonuçlara göre, ABC-X_3^h en iyi performansı vermiştir, onu ABC-X_1^h izlemektedir. Ayrıca, ABC-X_3^h diğer iki ABC-X^h yapılandırmasından istatistiksel olarak anlamlı derecede daha iyi bir performans sergilemiştir. Hibrit fonksiyonlarda, ABC-X ile elde edilen sonuçlar göz önüne alındığında, SOCO ölçüt seti ile CEC'14 veya CEC'05 ölçüt setindeki fonksiyonlar arasında zorluk açısından büyük bir fark olduğu görülmektedir. Aslında, SOCO fonksiyonları için (bunlar f27-f32 arası fonksiyonlar) sadece f28 ve f34 fonksiyonlarında sıfır (optimum) değeri bulunamamıştır. CEC fonksiyonlarında tüm ABC-X yapılandırmaları, sadece 10 boyuttaki f41 fonksiyonun da medyan değerini sıfır olarak bulmuşlardır.

Tablo 6.9. *Hibrit fonksiyonlarda irace tarafından elde edilen beş ABC-X yapılandırmasının medyan sonuçları ($ABC-X_1^h$ 'den $ABC-X_5^h$ 'e gösterilir) ve tüm fonksiyon tiplerine göre ayarlanmış $ABC-X^{all}$ 'a ait sonuçlar yer almaktadır. Detaylı açıklama Tablo 6.5'de verilmiştir.*

Boy.	Fonk.	$ABC-X_1^h$	$ABC-X_2^h$	$ABC-X_3^h$	$ABC-X_4^h$	$ABC-X_5^h$	$ABC-X^{all}$
10	f27	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f28	**1.51E-02**	1.60E-01	1.36E-01	9.31E-01	9.92E-01	4.93E-01
	f29	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f31	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f32	4.93E-07	1.06E-06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f33	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f34	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f35	**2.21E+03**	**2.21E+03**	**2.21E+03**	**2.21E+03**	**2.21E+03**	2.27E+03
	f36	8.20E-01	7.60E+00	**7.90E-01**	5.13E+00	9.65E-01	8.87E-01
	f37	3.55E+00	2.76E+00	**1.58E+00**	2.40E+00	2.94E+00	3.18E+00
	f38	5.43E+02	1.19E+03	**3.25E+02**	2.38E+03	3.74E+03	2.78E+03
	f39	**6.58E-01**	1.04E+00	8.86E-01	8.94E-01	1.04E+00	7.33E-01
	f40	8.01E-01	5.25E-01	9.06E-01	**4.58E-01**	7.76E-01	6.82E-01
	f41	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f42	1.07E+02	1.19E+02	1.09E+02	1.09E+02	1.10E+02	**1.03E+02**
	f43	1.25E+02	1.48E+02	1.17E+02	1.16E+02	1.30E+02	**1.15E+02**
	f44	**3.96E+02**	6.79E+02	8.00E+02	7.61E+02	8.00E+02	8.00E+02
	f45	**3.00E+02**	6.75E+02	**3.00E+02**	5.87E+02	8.00E+02	8.00E+02
	f46	**3.05E+02**	6.30E+02	8.00E+02	8.00E+02	8.00E+02	8.00E+02
	f47	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**
	f48	7.57E+02	7.69E+02	**7.54E+02**	7.59E+02	7.63E+02	7.60E+02
	f49	**5.59E+02**	**5.59E+02**	**5.59E+02**	**5.59E+02**	**5.59E+02**	6.40E+02
	f50	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**	6.77E+02	8.52E+02
30	f27	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.10E+00
	f28	**1.24E+00**	4.43E+00	1.06E+01	5.63E+00	1.74E+01	5.49E+01
	f29	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.95E-01
	f30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f31	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.05E+01
	f32	1.33E-01	2.21E-02	**4.11E-03**	8.70E-02	7.07E-02	7.88E+01
	f33	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f34	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f35	**6.90E+03**	6.98E+03	6.98E+03	7.05E+03	6.94E+03	7.08E+03
	f36	1.21E+05	1.32E+05	1.26E+05	**1.03E+05**	1.17E+05	1.28E+05
	f37	2.38E+01	**1.60E+01**	1.82E+01	1.96E+01	1.97E+01	1.99E+01
	f38	4.29E+04	4.92E+04	**3.35E+04**	5.13E+04	5.91E+04	7.05E+04
	f39	2.73E+00	3.00E+00	2.92E+00	2.74E+00	2.84E+00	**2.44E+00**
	f40	3.65E+00	4.52E+00	3.55E+00	4.16E+00	3.70E+00	**2.73E+00**
	f41	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.70E-07	1.91E+02	3.00E+02
	f42	1.57E+02	1.50E+02	**1.34E+02**	1.39E+02	1.60E+02	1.71E+02
	f43	2.20E+02	2.08E+02	**1.52E+02**	1.72E+02	2.02E+02	1.74E+02
	f44	9.10E+02	9.09E+02	**9.07E+02**	**9.07E+02**	**9.07E+02**	9.08E+02
	f45	9.10E+02	9.08E+02	**9.07E+02**	**9.07E+02**	**9.07E+02**	9.08E+02
	f46	9.10E+02	9.08E+02	**9.06E+02**	9.07E+02	9.07E+02	9.08E+02
	f47	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**
	f48	9.34E+02	9.31E+02	8.97E+02	9.24E+02	9.00E+02	**8.80E+02**
	f49	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**
	f50	**2.00E+02**	8.65E+02	8.52E+02	8.55E+02	8.55E+02	8.49E+02

Tablo 6.9. (Devam) *Hibrit fonksiyonlarda irace tarafından elde edilen beş ABC-X yapılandırmasının medyan sonuçları ($ABC-X_1^h$ 'den $ABC-X_5^h$ 'e gösterilir) ve tüm fonksiyon tiplerine göre ayarlanmış $ABC-X^{all}$ 'a ait sonuçlar yer almaktadır. Detaylı açıklama Tablo 6.5'de verilmiştir.*

Boy.	Fonk.	$ABC-X_1^h$	$ABC-X_2^h$	$ABC-X_3^h$	$ABC-X_4^h$	$ABC-X_5^h$	$ABC-X^{all}$
50	f27	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.81E+01
	f28	**5.99E-01**	7.93E+00	2.77E+01	3.33E+01	3.40E+01	1.52E+02
	f29	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.95E-01
	f30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f31	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.13E+02
	f32	1.47E-01	2.22E-01	**6.67E-02**	4.14E-01	2.23E-01	2.06E+02
	f33	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.50E+00
	f34	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f35	**1.21E+04**	**1.21E+04**	**1.21E+04**	1.22E+04	**1.21E+04**	1.22E+04
	f36	2.31E+05	2.23E+05	2.44E+05	**2.15E+05**	2.46E+05	2.36E+05
	f37	4.76E+01	**3.65E+01**	4.09E+01	4.14E+01	3.95E+01	3.93E+01
	f38	1.55E+05	1.66E+05	**1.16E+05**	1.87E+05	2.09E+05	2.57E+05
	f39	3.97E+00	4.54E+00	4.25E+00	3.97E+00	4.31E+00	**3.80E+00**
	f40	5.52E+00	7.34E+00	5.64E+00	6.29E+00	6.64E+00	**4.76E+00**
	f41	2.19E-05	**0.00E+00**	3.51E+01	7.16E-06	2.00E+02	3.29E+02
	f42	2.33E+02	2.11E+02	1.93E+02	2.12E+02	2.24E+02	**1.63E+02**
	f43	3.33E+02	2.92E+02	2.29E+02	3.00E+02	3.03E+02	**2.05E+02**
	f44	9.30E+02	9.30E+02	**9.25E+02**	9.28E+02	**9.25E+02**	9.26E+02
	f45	9.30E+02	9.29E+02	**9.25E+02**	9.31E+02	9.26E+02	9.26E+02
	f46	9.29E+02	9.31E+02	9.24E+02	9.29E+02	**9.23E+02**	9.26E+02
	f47	**5.00E+02**	8.00E+02	1.01E+03	1.02E+03	1.02E+03	1.01E+03
	f48	9.67E+02	9.53E+02	9.29E+02	9.41E+02	9.17E+02	**9.07E+02**
	f49	**5.39E+02**	1.02E+03	1.01E+03	1.02E+03	1.01E+03	1.02E+03
	f50	9.47E+02	**9.39E+00**	8.98E+02	8.95E+02	8.84E+02	8.75E+02
Sıralama		2.49	2.81	**1.88**	2.57	2.85	3.32
Wilcoxon T.		≈	≈		+	+	+

Bir sonraki adımda, ABC-X_1^h ve ABC-X_3^h yapılandırmaları ABC algoritmaları ile karşılaştırılmıştır ve sonuçları Tablo 6.9'da verilmiştir. Sonuçlar göstermiştir ki, iki yapılandırma bilinen diğer ABC algoritmalarına göre üstün performans sergilemiştir. Özellikle, ABC-X_1^h yapılandırması, ABC-X_3^h ve CABC algoritmaları haricinde, genel olarak en iyi sıralamadaki ABC varyantı olmuştur. Ayrıca diğer ABC varyantlarından da istatistiksel olarak anlamlı derecede daha iyidir. Diğer bir gerçek ise, CABC'nin bazı fonksiyonlarda en kötü performans gösteren varyant olmasından dolayı yüksek bir ortalama sıralama değerine sahip olmuştur. Buna karşın, f44-f46 gibi bazı fonksiyonlarda, en iyi performans gösteren varyantlar arasında olması bu durumu açıklamaktadır. Özetlemek gerekirse, test setindeki en zor fonksiyonlar arasında yer alan hibrit fonksiyonlarda ABC-X varyantlarının performansı oldukça iyidir.

Tablo 6.10. *ABC-X'in, en iyi yapılandırması (ABC-* X_3^h *), ilk yapılandırması (ABC-* X_1^h *) ve ABC varyantlarının Hibrit fonksiyonlarındaki medyan hata değerleri***.**

Boy.	Fonk.	$ABC-X_1^h$	$ABC-X_3^h$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
10	f27	**0.00E+00**	**0.00E+00**	9.62E-03	1.07E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.07E-02
	f28	**1.51E-02**	1.36E-01	2.98E-01	4.30E-01	1.35E+00	1.62E-01	5.81E-02	6.55E-02	8.13E-01	2.56E-01	2.59E-01
	f29	**0.00E+00**	**0.00E+00**	4.53E-03	1.07E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.34E-14	3.93E-03
	f30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f31	**0.00E+00**	**0.00E+00**	1.16E-05	1.02E-07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.10E-04
	f32	4.93E-07	**0.00E+00**	7.65E-01	4.40E-01	1.28E-01	1.20E-01	3.88E-02	3.54E-01	6.12E-11	5.78E-02	2.12E-01
	f33	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.86E-07
	f34	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f35	**2.21E+03**	**2.21E+03**	2.24E+03	2.29E+03	2.22E+03	2.22E+03	**2.21E+03**	2.22E+03	2.27E+03	2.23E+03	2.28E+03
	f36	8.20E-01	7.90E-01	8.70E+01	3.79E+02	8.15E+00	6.92E+00	1.47E+00	**6.84E-01**	1.65E+00	6.41E+01	1.36E+02
	f37	3.55E+00	1.58E+00	2.48E+00	2.35E+00	3.37E+00	2.44E+00	2.85E+00	**1.05E+00**	3.58E+00	3.19E+00	2.68E+00
	f38	5.43E+02	3.25E+02	2.12E+03	5.90E+03	3.18E+03	1.86E+03	9.89E+02	1.87E+03	**2.74E+02**	2.70E+03	2.33E+03
	f39	**6.58E-01**	8.86E-01	1.00E+00	1.19E+00	1.02E+00	8.86E-01	7.61E-01	7.99E-01	8.54E-01	1.16E+00	1.11E+00
	f40	8.01E-01	9.06E-01	1.37E+00	1.45E+00	1.12E+00	1.05E+00	1.03E+00	8.01E-01	**5.17E-01**	1.12E+00	1.47E+00
	f41	**0.00E+00**	**0.00E+00**	1.27E-01	**0.00E+00**	2.38E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.09E-11	6.38E-01	1.76E-01
	f42	**1.07E+02**	1.09E+02	1.61E+02	1.80E+02	1.12E+02	1.30E+02	1.21E+02	1.59E+02	1.09E+02	1.37E+02	1.56E+02
	f43	1.25E+02	**1.17E+02**	1.68E+02	1.81E+02	1.37E+02	1.35E+02	1.36E+02	1.65E+02	1.21E+02	1.63E+02	1.70E+02
	f44	**3.96E+02**	8.00E+02	5.03E+02	9.00E+02	7.90E+02	6.36E+02	5.57E+02	7.47E+02	9.00E+02	5.87E+02	5.02E+02
	f45	**3.00E+02**	**3.00E+02**	5.11E+02	9.00E+02	7.90E+02	6.12E+02	5.08E+02	7.45E+02	9.00E+02	5.40E+02	5.01E+02
	f46	**3.05E+02**	8.00E+02	5.05E+02	9.00E+02	8.00E+02	6.12E+02	5.00E+02	8.00E+02	9.00E+02	5.51E+02	5.01E+02
	f47	5.00E+02	5.00E+02	4.11E+02	1.03E+03	5.00E+02	4.23E+02	**4.10E+02**	5.00E+02	5.00E+02	4.19E+02	4.12E+02
	f48	7.57E+02	**7.54E+02**	7.92E+02	7.99E+02	7.66E+02	7.65E+02	7.63E+02	7.73E+02	7.58E+02	7.86E+02	7.99E+02
	f49	5.59E+02	5.59E+02	5.49E+02	8.93E+02	5.59E+02	**5.48E+02**	5.54E+02	5.59E+02	7.21E+02	5.54E+02	5.50E+02
	f50	**2.00E+02**	**2.00E+02**	**2.00E+02**	2.06E+02	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**	5.00E+02	**2.00E+02**	**2.00E+02**

Tablo 6.10. (Devam) *ABC-X'in, en iyi yapılandırması (ABC-* X_3^h *), ilk yapılandırması (ABC-* X_1^h *) ve ABC varyantlarının Hibrit fonksiyonlarındaki medyan hata değerleri.*

Boy.	Fonk.	$ABC-X_1^h$	$ABC-X_3^h$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
30	f27	**0.00E+00**	**0.00E+00**	3.70E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.56E-05
	f28	1.24E+00	1.06E+01	4.15E-01	3.98E+00	5.43E+00	4.93E-01	3.62E-01	5.96E-01	4.17E+01	6.88E+00	**2.80E-01**
	f29	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.11E+00	**0.00E+00**	3.03E-06
	f30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.00E+00
	f31	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.09E-06
	f32	1.33E-01	**4.11E-03**	2.11E-01	1.13E+00	6.30E-01	3.08E-01	7.95E-02	8.33E-02	3.52E+01	2.09E-01	9.32E-02
	f33	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.09E-07
	f34	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f35	**6.90E+03**	6.92E+03	7.09E+03	7.28E+03	6.98E+03	7.04E+03	6.98E+03	7.04E+03	7.14E+03	7.14E+03	7.06E+03
	f36	1.21E+05	1.26E+05	7.36E+04	6.33E+04	9.49E+04	6.76E+04	7.45E+04	7.27E+04	1.26E+05	5.09E+05	**5.07E+04**
	f37	2.38E+01	1.82E+01	1.84E+01	1.77E+01	2.18E+01	1.76E+01	1.91E+01	**1.25E+01**	2.96E+01	2.67E+01	1.84E+01
	f38	4.29E+04	**3.35E+04**	5.55E+04	5.84E+04	4.67E+04	4.35E+04	4.58E+04	5.45E+04	1.27E+05	6.31E+04	4.27E+04
	f39	2.73E+00	2.92E+00	3.07E+00	3.49E+00	3.02E+00	2.86E+00	2.86E+00	**2.63E+00**	3.13E+00	3.59E+00	3.04E+00
	f40	3.65E+00	**3.55E+00**	4.78E+00	5.30E+00	3.93E+00	4.39E+00	4.31E+00	4.44E+00	3.77E+00	5.06E+00	4.79E+00
	f41	**0.00E+00**	**0.00E+00**	9.28E-02	9.00E-02	6.17E-01	4.80E-02	1.70E-03	8.70E-04	5.00E+02	1.62E+02	5.02E-05
	f42	1.57E+02	**1.34E+02**	3.34E+02	3.59E+02	1.76E+02	2.48E+02	2.04E+02	3.39E+02	5.00E+02	2.35E+02	3.24E+02
	f43	2.20E+02	**1.52E+02**	3.98E+02	4.03E+02	2.50E+02	2.49E+02	2.46E+02	3.61E+02	2.59E+02	2.84E+02	3.67E+02
	f44	9.10E+02	9.07E+02	9.24E+02	**9.00E+02**	9.11E+02	9.15E+02	9.12E+02	9.21E+02	**9.00E+02**	9.09E+02	**9.00E+02**
	f45	9.10E+02	9.07E+02	9.22E+02	**9.00E+02**	9.11E+02	9.15E+02	9.12E+02	9.22E+02	**9.00E+02**	9.09E+02	**9.00E+02**
	f46	9.10E+02	9.06E+02	9.20E+02	**9.00E+02**	9.12E+02	9.14E+02	9.11E+02	9.22E+02	**9.00E+02**	9.10E+02	**9.00E+02**
	f47	**5.00E+02**	**5.00E+02**	**5.00E+02**	1.29E+03	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**
	f48	9.34E+02	**8.97E+02**	1.10E+03	1.14E+03	9.79E+02	1.00E+03	9.66E+02	1.06E+03	9.07E+02	9.27E+02	1.11E+03
	f49	**5.34E+02**	**5.34E+02**	**5.34E+02**	1.26E+03	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**
	f50	2.00E+02	8.52E+02	9.92E+02	1.32E+03	**2.00E+02**	**2.00E+02**	8.67E+02	9.48E+02	**2.00E+02**	2.09E+02	1.12E+03

Tablo 6.10. (Devam) *ABC-X'in, en iyi yapılandırması (ABC-* X_3^h *), ilk yapılandırması (ABC-* X_1^h *) ve ABC varyantlarının Hibrit fonksiyonlarındaki medyan hata değerleri.*

Boy.	Fonk.	$ABC-X_1^h$	$ABC-X_3^h$	ABC	BABC	EABC	GABC	GDABC	IABC	ImpABC	MABC	CABC
50	f27	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.85E-01	**0.00E+00**	1.17E-05
	f28	5.99E-01	2.77E+01	6.15E-02	1.30E-01	6.67E-01	4.39E-01	2.08E-01	3.76E-02	1.21E+02	3.54E+01	**3.29E-02**
	f29	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.95E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.58E+01	9.95E-01	2.99E-07
	f30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f31	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.00E+01	**0.00E+00**	1.87E-05
	f32	1.47E-01	**6.67E-02**	5.61E-01	4.80E+00	7.64E+00	1.19E+00	1.67E+00	1.14E+00	1.96E+02	1.90E+00	2.02E-01
	f33	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.28E+01	**0.00E+00**	9.39E-07
	f34	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	f35	**1.21E+04**	**1.21E+04**	1.22E+04	1.25E+04	1.22E+04	1.22E+04	1.22E+04	1.22E+04	1.22E+04	1.25E+04	1.23E+04
	f36	2.31E+05	2.44E+05	1.60E+05	**9.07E+04**	1.99E+05	1.44E+05	1.49E+05	1.49E+05	1.19E+05	1.83E+07	1.01E+05
	f37	4.76E+01	4.09E+01	3.78E+01	3.87E+01	4.40E+01	3.89E+01	3.69E+01	**3.04E+01**	5.76E+01	5.75E+01	3.85E+01
	f38	1.55E+05	**1.16E+05**	2.22E+05	1.63E+05	1.70E+05	1.78E+05	1.92E+05	1.98E+05	4.49E+05	3.55E+05	1.31E+05
	f39	3.97E+00	4.25E+00	4.44E+00	4.52E+00	4.30E+00	4.16E+00	4.02E+00	**3.96E+00**	4.47E+00	5.68E+00	4.31E+00
	f40	5.52E+00	5.64E+00	6.97E+00	8.00E+00	6.92E+00	6.81E+00	6.72E+00	6.54E+00	**5.36E+00**	1.54E+01	7.64E+00
	f41	**2.19E-05**	3.51E+01	1.79E-03	2.03E+00	5.72E-01	2.94E-01	3.71E+00	1.85E-03	4.00E+02	2.00E+02	8.67E-03
	f42	2.33E+02	1.93E+02	4.00E+02	4.00E+02	2.65E+02	3.53E+02	3.80E+02	4.00E+02	**1.40E+02**	3.40E+02	4.01E+02
	f43	3.33E+02	**2.29E+02**	4.64E+02	4.74E+02	4.00E+02	4.51E+02	4.42E+02	4.57E+02	2.50E+02	3.74E+02	4.69E+02
	f44	9.30E+02	9.25E+02	1.00E+03	**9.00E+02**	9.33E+02	9.36E+02	9.42E+02	9.96E+02	**9.00E+02**	9.27E+02	**9.00E+02**
	f45	9.30E+02	9.25E+02	9.87E+02	**9.00E+02**	9.33E+02	9.40E+02	9.40E+02	9.95E+02	**9.00E+02**	9.27E+02	**9.00E+02**
	f46	9.29E+02	9.24E+02	1.00E+03	**9.00E+02**	9.32E+02	9.37E+02	9.42E+02	9.92E+02	**9.00E+02**	9.27E+02	**9.00E+02**
	f47	**5.00E+02**	1.01E+03	**5.00E+02**	1.33E+03	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**
	f48	9.67E+02	**9.29E+02**	1.19E+03	1.25E+03	9.98E+02	1.05E+03	1.05E+03	1.14E+03	9.48E+02	9.53E+02	1.18E+03
	f49	**5.39E+02**	1.01E+03	**5.39E+02**	1.31E+03	**5.39E+02**	**5.39E+02**	**5.39E+02**	**5.39E+02**	5.44E+02	**5.39E+02**	**5.39E+02**
	f50	9.47E+02	8.98E+02	1.30E+03	1.38E+03	1.03E+03	1.13E+03	1.13E+03	1.25E+03	**8.83E+02**	9.34E+02	1.30E+03
Sıralama		**2.78**	2.99	5.79	6.78	4.78	3.97	3.58	4.26	5.31	5.51	5.76
Wilcoxon T.			≈	+	+	+	+	+	+	+	+	≈

6.2.1.6. *ABC-X^{all} ile karşılaştırılma*

Son olarak, bir problem türü için yapılandırılmış ABC-X varyantları ile tüm problem türlerinde iyi performans gösterecek şekilde yapılandırılmış bir ABC-X varyantının sonuçları karşılaştırılmıştır. Bu son ABC-X^{all} varyantının unimodal, multimodal ve hibrit fonksiyonlar için sonuçları sırasıyla Tablo 6.5, Tablo 6.7 ve Tablo 6.9'da gösterilmiştir. Multimodal ve hibrit fonksiyonları için, ABC-X^{all}, karşılaştırıldığı tüm altı ABC-X varyantı içinde en kötüsü olmuştur. Unimodal fonksiyonlarda ise, diğer varyantlarla aynı düzeyde performans elde etmiştir. Öte yandan ABC-X^{all} istatistiksel olarak en iyi varyanttan daha kötü değildir. Bununla birlikte, unimodal fonksiyonlar 50 boyutlu problem türleri içinde genel olarak en kolay olanlarıdır. Yapılandırmaların her biri on iki fonksiyondan beşi veya altısında en iyi çözümü bulmuştur. Daha zor olan multilmodal ve hibrit fonksiyonlarda ABC-X^{all}'un daha kötü performans göstermesi, belirli bir fonksiyon türü için yapılandırma yapılmasının bazı avantajlar sağladığını göstermektedir.

Deneysel sonuçlar, tüm problem sınıflarında ABC-X algoritmalarıyla rekabet edebilecek başka bir ABC varyantının olmadığını göstermektedir. Problem türüne göre kategorilendirme fonksiyonların sadece bir özelliğine dayanmasına rağmen, sağlamış olduğu esneklik ile ABC-X, algoritmaları çeşitli problem türlerine iyi adapte ederek güzelce kullanılmıştır.

6.2.2. Valf nokta etkili konveks olmayan ekonomik güç dağıtım probleminin çözümü

6.2.2.1. *Problem tanımı*

Bugünlerde elektriğe olan ihtiyacın artmasıyla, ekonomik güç dağıtım konusunun, enerji sistemlerinin işletilmesindeki en önemli konulardan biri olmasına sebep olmuştur. Elektrik üretiminde kullanılan yakıt maliyetinin, üretim maliyetlerinde önemli bir miktara ulaşması, yakıt üreten firmaların yakıtları daha verimli kullanmasına neden olmaktadır. Literatürde ekonomik güç dağıtım problemi (Economic Power Dispatch Problem, EPDP), üretim birimlerinin aktif güç çıkışının sistemdeki mevcut yükü karşılayacak şekilde ayarlanması (sistem kısıtlamaları ve minimum yakıt maliyeti altında) olarak tanımlanmaktadır. Bu tür problemlerde, üretim birimlerinin yakıt maliyet fonksiyon eğrileri, toplam yakıt maliyetini hesaplamak için kullanılır (Aydın ve Özyön, 2013).

Ekonomik güç dağıtım probleminin çözümü, toplam yakıt maliyetinin sistem sınırları dâhilinde en aza indirilmesiyle bulunmaktadır. Standart EPDP'nin amaç fonksiyonu, aşağıdaki gibi tanımlanmıştır (Coelho ve Mariani, 2006; Yaşar ve Özyön, 2011):

$$\min F_{total} = \min \sum_{n=1}^{N} F_n(Pow_{G,n}) \tag{6.8}$$

$F_n(Pow_{G,n})$, n'inci üretim biriminin yakıt maliyeti fonksiyonunu R/h olarak gösterir;

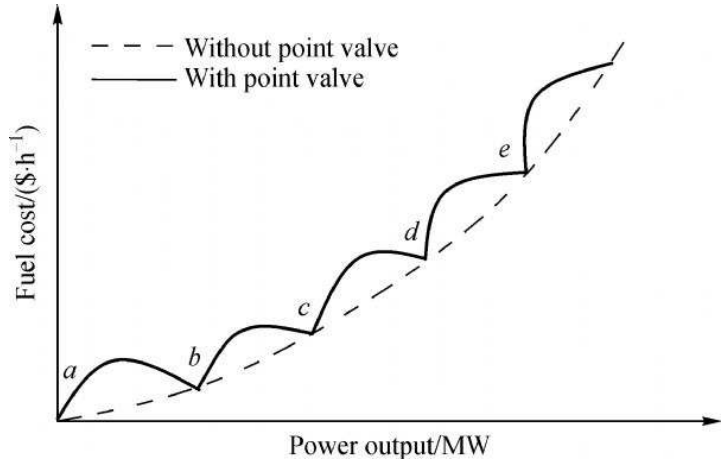
$$F_n(Pow_{G,n}) = a_n + b_n Pow_{G,n} + c_n Pow_{G,n}^2, (R/h). \tag{6.9}$$

$Pow_{G,n}$, *n.* üretim birimin MW cinsinden çıkış gücünü gösterir ve a_n, b_n, c_n sabitlerdir.

Üretim birimlerinin yakıt maliyet fonksiyonu, Şekil 6.2'de şematik bir şekilde gösterilmiştir. Şekil 6.2'deki kesikli çizgiyle gösterilen konveks olmayan fonksiyonu, valf noktası etkisi içermeyen Denklem (6.9) kullanıldığında yakıt maliyet fonksiyonunu gösterir. Aslında, çok-valfli buhar tribünlü üretim birimlerinin giriş-çıkış eğrisi, Denklem (6.9)'de yer alan eşitlikten daha farklıdır. Valf noktası etkisinin, üretim biriminin yakıt maliyetine dâhil edilmesiyle, bu maliyete ait gösterimini daha uygun duruma getirir. Yakıt maliyet fonksiyonu, valf noktası dalgalanmalarla sonlandırıldığında, daha yüksek doğrusal olmayan seriler barındırır. Bu nedenle, valf noktası etkilerini göz önünde bulundurarak yapılan çalışmada, Denklem (6.10)'de Şekil 6.2'deki sürekli çizgiyi izleyerek değişen bir fonksiyon kullanılmıştır.

$$F_n(Pow_{G,n}) = a_n + b_n Pow_{G,n} + c_n Pow_{G,n}^2 + |d_n.sin(e_n(Pow_{G,n}^{min} - Pow_{G,n}))| \tag{6.10}$$

Denklem (6.10)'de, d_n ve e_n valf noktası etkilerini göstermektedir.



Şekil 6.2. *Valf nokta etki üretim biriminin giriş-çıkış karakteristiği* (Aydın, Yavuz, Özyön, vd., 2017)

Problem kısıtları

Valf nokta etkilerine sahip EPDP, problemle uğraşırken yerine getirilmesi gereken iki kısıta sahiptir. Bunlar aşağıdaki gibi tanımlanan, kapasite kısıtları ve güç dengesi kısıtıdır:

Üretme kapasitesi kısıtları: n. birimden üretilen güç $Pow_{G,n}$, minimum $Pow_{G,n}^{min}$ ve maksimum $Pow_{G,n}^{max}$ güç sınırlarını aşmamalıdır. Bu kısıt aşağıdaki gibi tanımlanmıştır:

$$Pow_{G,n}^{min} \le Pow_{G,n} \le Pow_{G,n}^{max}, (n \in N) \tag{6.11}$$

Güç dengesi kısıtı: Enerji üretim sistemlerinin optimum çalışması sırasında planlama yapılırken, önce AC güç akışı analizi yapılması gerekir. Enerji sisteminin AC güç akışı analizinde, tüm jeneratörlerin net aktif ve reaktif güçleri belirlenmedir. Ayrıca hepsinin genlikleri ve faz açıları bulunmalıdır. Daha sonra bu bilgi ile sistemdeki iletim hatlarında meydana gelen aktif ve reaktif güç akışları sayesinde sistem kayıpları hesaplanır. Optimum güç akışı problemi, eşitlik ve eşitsizlik kısıtlamalarının sağlanmasıyla sistemin amaç fonksiyonu olan enerji yakıt maliyetinin optimal değerini bulmaktır. Bu optimizasyon probleminin çözümü incelendiğinde, Pow_{load} güç yükü, sabit olarak alınır. Aslında, literatürde elektrik yükünün bir yere ait enerji talebi olduğu ve günde 2 ila 10 dakika boyunca sabit kaldığı varsayılmaktadır. Bu nedenle yakıt maliyetinin optimizasyonu çok hızlı bir şekilde yapılmalıdır. Aksi takdirde, uygun çözümleri bulmak neredeyse imkânsız olacaktır. Bununla birlikte, AC güç akışı analizi

yinelemeli bir ölçümdür ve üretilen güç değerleri değiştiğinde tekrardan hesaplanması gerekir. Bu da zaman ve belleğin tüketilmesine yol açar. Bu nedenle, eğer bir algoritma AC güç akışı analizi ile ilgili bir problemi çözmeye çalışırsa, bu analiz her amaç fonksiyonu bulunurken hesaplanmalıdır. Bu da zaman alıcı bir işlemdir. Bu yüzden, önerilen yeni bir algoritmanın ekonomik dağıtım problemleri üzerindeki performansın test edilmesi amaçlanıyorsa, iletim hattı kayıpları göz ardı edilebilir veya yaklaşık B-kayıp matrisleri kullanılarak tek bir denklem ile hesaplanabilir. B-kayıp matrisleri iletim hattı kayıplarının, üretilen güçler için AC yük akışından daha hızlı ele alınmasını sağlar. Sistem kayıpları göz ardı edildiğinde güç dengesi aşağıdaki gibi hesaplanır:

$$Pow_{load} - \sum_{n=1}^{D_{jer}} F_n(Pow_{G,n}) = \Delta Pow = 0 \tag{6.12}$$

D_{jer}, toplam jeneratör sayısıdır (ayrıca problem boyutunu gösterir). Güç kayıpları göz önünde bulundurulduğunda, güç dengesi kısıtlaması Denklem (6.13)'deki gibi alınır:

$$Pow_{load} - Pow_{loss} - \sum_{n=1}^{D_{jer}} F_n(Pow_{G,n}) = \Delta Pow = 0 \tag{6.13}$$

Burada Pow_{loss}, aşağıdaki denklem (Yaşar ve Özyön, 2011) kullanılarak B-kayıp matrisi ile hesaplanan güç kayıplarıdır :

$$Pow_{loss} = \sum_{n=1}^{D_{jer}} \sum_{j=1}^{D_{jer}} Pow_{G,n}.B_{nj}.Pow_{G,j} + \sum_{n=1}^{D_{jer}} B_{0n}.Pow_{G,n} + B_{00} \tag{6.14}$$

Literatürde hem kayıplı hem de kayıpsız güç dağıtım problem örnekleri vardır. Kayıplı sistemlerin çoğunda, kayıplar *B*-kayıp matrisleri ile hesaplanır. Bu bölümde, algoritma, kayıplı ve kayıpsız problem örnekleri üzerinden test edilmiştir. Algoritmanın literatürdeki diğer algoritmalarla karşılaştırmak için *B*-kayıp matrislerinin kayıplı sistemlerde kullanılması tercih edilmiştir.

6.2.2.2. ABC-X çatısı ile uygun ABC algoritmasının elde edilmesi

ABC-X'in parametre değerleri Bölüm 4.2.1'de anlatılmış olan otomatik parametre ayarlama aracı irace ile belirlenmiştir. Böylece ABC-X kullanılarak probleme uygun bir ABC algoritması üretilmiştir. irace aracı varsayılan parametre değerleriyle çalıştırılmıştır. Eğitim örnekleri olarak; 3, 5, 6, 13 ve 40 jeneratör birim sistemlerinden problem örnekleri

seçilmiştir. Bunlardan 3,5 ve 6 jeneratörleri kayıplı, diğerleri ise kayıpsız sistemlerdir. Parametre yapılandırma aşamasında eğitim örneklerinin test durumundan farklı olması için problem örnekleri üretilmiştir. Bunlar güç yük talepleri açısından literatürdeki problem örneklerinin her birinden farklı üretilmişlerdir. Böylelikle ABC-X'in parametre değerleri, 30 örnek (her bir sistem için 6 örnek) kullanılarak irace vasıtasıyla ayarlanmıştır.

ABC-X çatısı ile elde edilen algoritmanın sözde kodu Algoritma 6.6'de verilmiştir. Sözde kod incelendiğinde, işçi arı ve gözcü arı adımlarında farklı arama denklemleri kullanılmıştır. Ayrıca kâşif arı adımında scoutBABC bileşeni seçilmiş ve yerel arama algoritma ile melezleştirilmiştir. Buna göre, daha önce literatürde yer almayan yeni bir ABC algoritmasının elde edildiği görülmüştür.

Algoritma 6.6. *ABC-X çatısı vasıtasıyla elde edilen ABC varyantının sözde kodu*

SN = 22, limit = 2.7*SN
Henon kaotik harita kullanılarak ilklendirme uygula

for her bir işçi arı x_i **do**
    for t=1 to m=5 **do**
        j boyutunu rastgele seç
        $\hat{x}_{i,j} = x_{r1,j} + \phi_1(x_{i,j} - x_{r1,j}) + \phi_2(x_{i,j} - x_{G,j})$
    if $\hat{x}_i \prec x_i$ **then**
        $x_i = \hat{x}_i, trial_i = 0$
    else
        $trial_i = trial_i + 1$
Seçme olasılığını rankBased stratejisini SP=1.95 olarak hesapla
for her bir gözcü arı x_i **do**
    for t=3 to m=4 **do**
        j boyutunu rastgele seç
        $\hat{x}_{i,j} = x_{G,j} + \phi_1(x_{i,j} - x_{G,j}) + +\phi_1(x_{r1,j} - x_{r2,j}) + +\phi_2(x_{i,j} - x_{GD,j})$.
    if $\hat{x}_i \prec x_i$ **then**
        $x_i = \hat{x}_i, trial_i = 0$
    else
        $trial_i = trial_i + 1$
scoutBABC kaşif stratejisini $w_{Min} = 0.24$ and $w_{Max} = 0.72$ parametreleri ile uygula
MTSLS1 ve CMAES kullanarak rekabetçi yerel arama stratejisini uygula
Sabit boyut popülasyonu uygula

6.2.2.3. Bir aday çözümün yapısı

D jeneratörlü bir EPDP problemi için, bir aday çözüm X, aşağıdaki gibi ifade edilen bir vektördür:

$$X = \left[Pow_{G,1}, \ldots, Pow_{G,n}, \ldots, Pow_{G,D} \right] \tag{6.15}$$

Her bir eleman $Pow_{G,n}$, jeneratör birimi *n* tarafından üretilen güce karşılık gelir ve bu da problem kısıtlarını sağlamalıdır.

6.2.2.4. Problem kısıtlamalarını ele almak

EPDP problemini ABC-X çatısıyla çözerken, kısıtlama yönetim mekanizmalarının tanımlanması gerekmektedir. Üretim kapasitesi kısıtlamaları için, olası iki ele alma yöntemi vardır. Bunlardan ilki, uygunluk fonksiyonunda penaltı kısıtlama ihlallerini kullanmaktadır. Diğer yöntem ise, uygulanabilir olmayan çözümleri onarmak ve uygulanabilir çözümler oluşturarak optimizasyon işlemi sırasında yalnızca tamir edilmiş çözümlerle çalışmaktır (Park vd., 2005). Bu tezde ikinci yaklaşım kullanılmıştır. Tezde uygulanan tamir işleminin amacı, Denklem (6.12) ve (6.13)'da tanımlanan eşitlik kısıtlarının ihlalini önlemektir. Bu onarım yöntemi Algoritma 6.7'de verilmiştir.

Algoritma 6.7. *Güç dengesi kısıtlama ele alma yöntemi* ε *güç farkı hassasiyetini göstermektedir.*

while $|\Delta Pow| \leq \varepsilon$ **do**
  Çözümden rastgele bir jenaratör seç
  if $\Delta Pow < 0$ **then**
    $Pow_{G,n}^{max}$ ($Pow_{G,i} = min(Pow_{G,i} + \Delta Pow, Pow_{G,n}^{max})$) ihlal etmeyen
    ΔPow 'u $Pow_{G,i}$ 'ye ekle.
  else if $\Delta Pow > 0$ **then**
    $Pow_{G,n}^{min}$ (such as $Pow_{G,i} = min(Pow_{G,i} - \Delta Pow, Pow_{G,n}^{min})$) ihlal etmeyen
    $Pow_{G,i}$ 'den ΔPow 'i çıkar

Tablo 6.11. *(Durum I) Üretim birimlerinin maliyet fonksiyonu katsayıları ve aktif güç üretim limitleri*

Bus no.	1	2	5	8	11	13
A	150	25	0	0	0	0
B	2	2.5	1	3.25	3	3
C	0.0016	0.01	0.0625	0.00834	0.025	0.025
E	50	40	0	0	0	0
F	0.063	0.098	0	0	0	0
$Pow_{min}(MW)$	50	20	15	10	10	12
$Pow_{max}(MW)$	200	80	50	35	30	40

6.2.2.5. Deneysel sonuçlar

ABC-X çatısının etkinliğini ve sağlamlığını değerlendirmek için, valf noktası etkisi olan ekonomik dağıtım iki test durumu kullanılmıştır. İlki, iletim kaybı olan altı jeneratör (Durum I) sistemidir. Diğeri ise, iletim kaybı olmayan 40 jeneratör birimine sahip gerçekçi Taipower sistemidir (Durum II).

Tüm testler 2 x 6 MB L2 önbellek ve 8 GB RAM ile 2.33 GHz hızında çalışan bir Intel Xeon E5410 dört çekirdekli işlemcide gerçekleştirilmiştir. Her durumun çalışmasındaki sonuçlar $D\times1000$ ve $D\times10000$ (D, jeneratör sayısıdır) fonksiyon çağırım sayısı sonrası sonlandırılmış 30 bağımsız çalışmadan elde edilmiştir. Programın bu FEs değerleri ile çalıştırılmasının nedenleri şunlardır: (i) literatürde hiçbir standart belirtilmemiştir ve geçmişteki birçok yöntem farklı FEs'lerle çalıştırılmıştır (ii) algoritmanın yakınsama hızını test edebilmektir. Amaç fonksiyonun değeri hesaplanmasında uygun çözümler elde etmek için ϵ değeri, 10^{-6} olarak ayarlanmıştır.

Durum I: 6 jeneratörlü birim sistemi

İlk durum, altı jeneratör birimden oluşan kayıplı bir sistemdir ve beklenen güç talebi, Pow_{load}, 283.4 MW'dır. Maliyet fonksiyonu katsayıları ve problemle ilgili *B*-kayıp matrisleri sırasıyla Tablo 6.11 ve Tablo 6.12'de verilmiştir.

İlk durum için, $D\times1000$ ve $D\times10000$ FEs'de elde edilen en iyi ve en kötü toplam yakıt maliyetleri Tablo 6.13'te verilmiştir.

Tablo 6.13'te görüldüğü üzere, ABC-X'in düşük ve yüksek FEs değerlerinde en iyi ve en kötü sonuçları benzerdir. Bu da sonuçların düşük bir değişkenliğini ve her çalışmada ABC-X'in hızlı bir şekilde yakınsadığını göstermektedir.

Tablo 6.14'de ABC-X'in elde ettiği sonuçlar, daha önce literatürde aynı problem örneğine uygulanmış çeşitli algoritmalar ile karşılaştırılmıştır. Karşılaştırılan algoritmaların çoğu, en iyi ve en kötü maliyetleri elde etmek için çok fazla fonksiyon çağırım sayısı kullanmasına rağmen ABC-X'den elde edilen algoritma onları geride bırakmıştır.

Tablo 6.12. *(Durum I) B-kayıp matris değerleri*

$$[B] = \begin{vmatrix} 0.0224 & 0.0103 & 0.0016 & -0.0053 & 0.0009 & -0.0013 \\ 0.0103 & 0.0158 & 0.001 & -0.0074 & 0.0007 & 0.0024 \\ 0.0016 & 0.001 & 0.0474 & -0.0687 & -0.0060 & -0.0350 \\ -0.0053 & -0.0074 & -0.0687 & 0.3464 & 0.0105 & 0.0534 \\ 0.0009 & 0.0007 & -0.0060 & 0.0105 & 0.0119 & 0.0007 \\ -0.0013 & 0.0024 & -0.0350 & 0.0534 & 0.0007 & 0.2353 \end{vmatrix}$$

$$[B_0] = [-0.0005 \quad 0.0016 \quad -0.0029 \quad 0.006 \quad 0.0014 \quad 0.0015]$$

$$[B_0] = [0.0011]$$

Tablo 6.13. *(Durum I) 30 denemede elde edilen sonuçlar*

	$D \times 1000$ FEs		$D \times 10000$ FEs	
Jeneratör	Çıkış gücü	Çıkış gücü	Çıkış gücü	Çıkış gücü
Pow_1(MW)	199.60	199.60	199.6	199.6
Pow_2(MW)	20.00	20.01	20.00	20.00
Pow_3(MW)	24.01	25.24	23.96	23.96
Pow_4(MW)	18.68	17.51	18.86	19.54
Pow_5(MW)	18.53	17.28	18.17	18.17
Pow_6(MW)	13.54	14.69	13.82	13.47
Pow_{loss}(MW)	11.22	9.24	11.18	10.77
$F_{total}(R/h)$	925.0 (En iyi maliyet)	925.2 (En kötü maliyet)	925.0 (En iyi maliyet)	925.0 (En kötü maliyet)

Tablo 6.14. *(Durum I) ABC-X tarafından üretilen ABC algoritması ile literatürdeki diğer algoritmalar arasındaki karşılaştırma sonuçları*

Algoritmalar	En iyi maliyet (R/h)	En kötü maliyet (R/h)	FEs
MSG-HS (Yaşar ve Özyön, 2011)	925.60	928.60	1000000
GA (Malik vd., 2010)	996.00	111.70	6120
DE (Özyön, Yaşar, ve Temurtaş, 2011)	963.00	-	10000
ABC (Özyön, Yaşar, Özcan, vd., 2011)	928.40	-	10000
EP (Ongsakul ve Tantimaporn, 2006)	955.50	959.40	-
IEP (Ongsakul ve Tantimaporn, 2006)	953.60	958.30	-
SADE ALM (Thitithamrongchai ve Eua-Arporn, 2007)	944.00	964.80	20000
TS-SA (Ongsakul ve Tantimaporn, 2006)	959.60	966.00	-
ITS (Ongsakul ve Tantimaporn, 2006)	969.10	985.50	-
ABC-X	**925.00**	**925.20**	6000
ABC-X	**925.00**	**925.00**	60000

Durum II: 40 jeneratörlü birim sistemi

Bu durum çalışmasında, valf nokta etkisi ile birlikte ikinci dereceden maliyet fonksiyonuna sahip 40 jeneratör bulunmaktadır. Bu durumda, iletim kayıpları göz ardı edilir ve tüm kırk üretim birimlerinin karşılaması gereken güç talebi 10500 MW'tır. Sistem için birim veriler (maliyet fonksiyon katsayıları) literatürde mevcuttur ve bundan dolayı doğrudan (Sinha vd., 2003)'den alınmıştır. Taipower sistemi, EPDP ile ilgili literatürde valf noktası etkisi olan en büyük problemlerden biridir. $D\times1000$ ve $D\times10000$ FEs'lik iki durum çalışması için elde edilen en iyi ve en kötü toplam yakıt maliyetleri Tablo 6.15 ve Tablo 6.16'da listelenmiştir.

Bu, doğrusal olmayan elemanlara sahip büyük bir sistem olduğundan, aynı zamanda daha fazla yerel minimuma sahiptir ve bu nedenle çözülmesi de zordur. Bu durum, Tablo 6.15 ve Tablo 6.16'daki sonuçlara da yansımıştır. İlk durum çalışmasından farklı olarak, $D\times1000$ FEs ile elde edilen sonuçların, $D\times10000$ FEs ile elde edilen sonuçlardan daha kötü olduğu görülmüştür. Ayrıca elde edilen sonuçların değişkenliği daha yüksektir.

Tablo 6.17'de, ABC-X sonuçlarının literatürden elde edilen birkaç algoritmanın sonuçları ile karşılaştırılması verilmiştir. Tablo 6.17'deki karşılaştırma sonuçları incelendiğinde, her algoritmanın farklı FEs'ler ile çalışmasına rağmen, geliştirilen algoritmanın diğer algoritmalara göre çok iyi sonuçlar verdiği görülmüştür. Ayrıca $D\times1000$ FEs'te ABC-X için bildirilen en iyi maliyetler, NPSO-LRS (Selvakumar ve Thanushkodi, 2008), ST-HDE (Wang vd., 2007) ve DEC(2)-SQP(1) (Coelho ve Mariani, 2006) dışındaki diğer tüm algoritmalardan daha iyidir. Öte yandan, ABC-X'in en kötü sonuçları, bütün durumlarda, diğer algoritmaların herhangi biri tarafından elde edilen en kötü sonuçlarından, hatta birçok durumda bu algoritmaların en iyi sonuçlarından da daha iyi olmuştur. $D\times10000$ FEs'te elde edilen ABC-X sonuçları, ilave hesaplama zamanından önemli ölçüde yararlanabileceğini göstermektedir. Bu nedenle, genel olarak elde edilen sonuçlar göstermektedir ki, belirli bir endüstriyel tarz problem için özel bir ABC-X algoritmasının otomatik yapılandırma yaklaşımıyla oluşturmak için önemli bir algoritma mühendisliği çabası gerektirmediğini göstermektedir.

Tablo 6.15. *(Durum II) $D\times1000$ FEs ile 30 denemede elde edilen en iyi ve en kötü sonuçlar*

	En iyi sonuçlar				En kötü sonuçlar		
Jeneratör	Çıkış gücü	Jeneratör	Çıkış gücü	Jeneratör	Çıkış gücü	Jeneratör	Çıkış gücü
Pow_1(MW)	112.0	Pow_{21}(MW)	523.2	Pow_1(MW)	114.4	Pow_{21}(MW)	523.3
Pow_2(MW)	111.0	Pow_{22}(MW)	523.2	Pow_2(MW)	113.9	Pow_{22}(MW)	523.7
Pow_3(MW)	97.4	Pow_{23}(MW)	523.3	Pow_3(MW)	120.2	Pow_{23}(MW)	523.6
Pow_4(MW)	179.7	Pow_{24}(MW)	523.2	Pow_4(MW)	179.7	Pow_{24}(MW)	549.8
Pow_5(MW)	88.3	Pow_{25}(MW)	523.2	Pow_5(MW)	97.0	Pow_{25}(MW)	523.6
Pow_6(MW)	140.0	Pow_{26}(MW)	523.2	Pow_6(MW)	140.6	Pow_{26}(MW)	523.3
Pow_7(MW)	300.0	Pow_{27}(MW)	10.0	Pow_7(MW)	300.0	Pow_{27}(MW)	10.0
Pow_8(MW)	284.7	Pow_{28}(MW)	10.0	Pow_8(MW)	300.0	Pow_{28}(MW)	10.0
Pow_9(MW)	284.6	Pow_{29}(MW)	10.0	Pow_9(MW)	290.5	Pow_{29}(MW)	10.0
Pow_{10}(MW)	130.0	Pow_{30}(MW)	94.0	Pow_{10}(MW)	130.0	Pow_{30}(MW)	97.0
Pow_{11}(MW)	94.0	Pow_{31}(MW)	190.0	Pow_{11}(MW)	94.0	Pow_{31}(MW)	190.0
Pow_{12}(MW)	94.0	Pow_{32}(MW)	190.0	Pow_{12}(MW)	94.0	Pow_{32}(MW)	190.0
Pow_{13}(MW)	214.7	Pow_{33}(MW)	189.9	Pow_{13}(MW)	125.0	Pow_{33}(MW)	190.0
Pow_{14}(MW)	304.5	Pow_{34}(MW)	199.9	Pow_{14}(MW)	394.2	Pow_{34}(MW)	199.9
Pow_{15}(MW)	394.2	Pow_{35}(MW)	199.9	Pow_{15}(MW)	304.6	Pow_{35}(MW)	200.0
Pow_{16}(MW)	394.2	Pow_{36}(MW)	199.9	Pow_{16}(MW)	394.2	Pow_{36}(MW)	200.0
Pow_{17}(MW)	489.2	Pow_{37}(MW)	109.9	Pow_{17}(MW)	489.2	Pow_{37}(MW)	110.0
Pow_{18}(MW)	489.2	Pow_{38}(MW)	109.9	Pow_{18}(MW)	489.2	Pow_{38}(MW)	110.0
Pow_{19}(MW)	511.2	Pow_{39}(MW)	109.9	Pow_{19}(MW)	511.2	Pow_{39}(MW)	110.0
Pow_{20}(MW)	511.2	Pow_{40}(MW)	511.2	Pow_{20}(MW)	511.2	Pow_{40}(MW)	511.3
$F_{total}(R/h)$	121465 (En iyi maliyet)			$F_{total}(R/h)$	121751 (En kötü maliyet)		

Tablo 6.16. *(Durum II) $D\times10000$ FEs ile 30 denemede elde edilen en iyi ve en kötü sonuçlar*

	En iyi sonuçlar				En kötü sonuçlar		
Jeneratör	Çıkış gücü	Jeneratör	Çıkış gücü	Jeneratör	Çıkış gücü	Jeneratör	Çıkış gücü
Pow_1(MW)	114.0	Pow_{21}(MW)	523.0	Pow_1(MW)	114.0	Pow_{21}(MW)	523.5
Pow_2(MW)	114.0	Pow_{22}(MW)	524.1	Pow_2(MW)	114.0	Pow_{22}(MW)	550.0
Pow_3(MW)	102.3	Pow_{23}(MW)	523.1	Pow_3(MW)	120.0	Pow_{23}(MW)	523.4
Pow_4(MW)	190.0	Pow_{24}(MW)	523.1	Pow_4(MW)	190.0	Pow_{24}(MW)	550.0
Pow_5(MW)	97.0	Pow_{25}(MW)	523.0	Pow_5(MW)	97.0	Pow_{25}(MW)	526.3
Pow_6(MW)	140.0	Pow_{26}(MW)	525.6	Pow_6(MW)	140.0	Pow_{26}(MW)	550.0
Pow_7(MW)	269.9	Pow_{27}(MW)	10.0	Pow_7(MW)	300.0	Pow_{27}(MW)	10.0
Pow_8(MW)	285.7	Pow_{28}(MW)	10.0	Pow_8(MW)	300.0	Pow_{28}(MW)	10.0
Pow_9(MW)	285.8	Pow_{29}(MW)	10.0	Pow_9(MW)	300.0	Pow_{29}(MW)	10.0
Pow_{10}(MW)	130.0	Pow_{30}(MW)	97.0	Pow_{10}(MW)	130.0	Pow_{30}(MW)	97.0
Pow_{11}(MW)	169.3	Pow_{31}(MW)	190.0	Pow_{11}(MW)	94.0	Pow_{31}(MW)	190.0
Pow_{12}(MW)	94.0	Pow_{32}(MW)	190.0	Pow_{12}(MW)	94.0	Pow_{32}(MW)	190.0
Pow_{13}(MW)	125.0	Pow_{33}(MW)	190.0	Pow_{13}(MW)	214.7	Pow_{33}(MW)	190.0
Pow_{14}(MW)	304.6	Pow_{34}(MW)	200.0	Pow_{14}(MW)	215.5	Pow_{34}(MW)	200.0
Pow_{15}(MW)	391.9	Pow_{35}(MW)	200.0	Pow_{15}(MW)	304.6	Pow_{35}(MW)	200.0
Pow_{16}(MW)	394.7	Pow_{36}(MW)	200.0	Pow_{16}(MW)	394.3	Pow_{36}(MW)	200.0
Pow_{17}(MW)	489.5	Pow_{37}(MW)	110.0	Pow_{17}(MW)	500.0	Pow_{37}(MW)	110.0
Pow_{18}(MW)	491.3	Pow_{38}(MW)	110.0	Pow_{18}(MW)	489.8	Pow_{38}(MW)	110.0
Pow_{19}(MW)	517.5	Pow_{39}(MW)	110.0	Pow_{19}(MW)	511.6	Pow_{39}(MW)	110.0
Pow_{20}(MW)	511.4	Pow_{40}(MW)	512.2	Pow_{20}(MW)	513.3	Pow_{40}(MW)	512.4
$F_{total}(R/h)$	121770 (En iyi maliyet)			$F_{total}(R/h)$	122356 (En kötü maliyet)		

Tablo 6.17. *(Durum II) Literatürdeki ve önerilen ABC-X tarafından elde edilen sonuçlar*

Algoritmalar	En iyi maliyet (R/h)	En kötü maliyet (R/h)	FEs
HGPSO (Ling vd., 2008)	124797	-	100000
SPSO (Ling vd., 2008)	124350	-	100000
CEP (Sinha vd., 2003)	123488	126903	60000
HGAPSO (Ling vd., 2008)	122780	-	100000
FEP (Sinha vd., 2003)	122680	127246	60000
MFEP (Sinha vd., 2003)	122648	124356	60000
IFEP (Sinha vd., 2003)	122624	125741	60000
HPSOM (Ling vd., 2008)	122112	-	100000
PSO-LRS (Selvakumar ve Thanushkodi, 2007)	122036	123462	20000
Improved GA (Ling ve Leung, 2007)	121916	123334	100000
HPSOWM (Ling vd., 2008)	121915	-	100000
IGAMU (Chiang, 2007)	121819	-	450000
HDE (Wang vd., 2007)	121813	-	42500
PPSO (Chen ve Yeh, 2006)	121788	124998	20000
DEC(2)-SQP(1) (Coelho ve Mariani, 2006)	121742	122839	18000
ST-HDE (Wang vd., 2007)	121699	-	42500
NPSO-LRS (Selvakumar ve Thanushkodi, 2007)	121664	122982	40000
ABC-X	**121770**	122356	40000
ABC-X	**121465**	121751	400000

7. KENDİNDEN UYARLAMALI ARAMA DENKLEMİ-TABANLI YAPAY ARI KOLONİSİ ALGORİTMASI

7.1. Algoritma Tanımı

Önerilen Kendinden Uyarlamalı Arama Denklemi-Tabanlı Yapay Arı Kolonisi (Self-adaptive Search Equation-Based Artificial Bee Colony (SSEABC)) algoritmasının sözde kodu Algoritma 7.1'de sunulmuştur. SSEABC, orijinal ABC algoritmasına üç yeni strateji eklemiştir. Bunlar; "kendinden uyarlamalı arama denklemi belirleme stratejisi", "rekabetçi yerel arama seçim stratejisi" ve "artan popülasyon boyut stratejisi" dir. Bu stratejilerin ayrıntılı açıklaması aşağıdaki alt bölümlerde verilmiştir.

Algoritma 7.1. *SSEABC algoritmasının sözde kodu*

```
1: function SelfAdaptiveSearchEquationBasedABC
2:     Initialization()
3:     S denklem havuzunu rastgele üretilmiş arama denklemleri ile doldur
4:     itr = 0
5:     isCompetitionNeeded = TRUE
6:     while termination condition is not met do
7:         EmployedBeeStep(S[itr mod ps])
8:         OnlookerBeeStep(S[itr mod ps])
9:         ScoutBeesStep()
10:        ApplyLocalSearchStrategy()
11:        UpdatePopulationSize()
12:        UpdateEquationsPoolSize()
13:        itr = itr + 1
14:    return the best solution
15:
16: function ApplyLocalSearchStrategy()
17:    Xgbest = getBestFoodSource()
18:    if currentFES ≥ tr*MAXFES & isCompetitionNeeded == TRUE then
19:        selectedLS = selectLocalSearchWithCompetition(Xgbest)
20:    if selectedLS ≠ NULL then
21:        selectedLS de seçilmiş yerel aramayı Xgbest'e uygula
22:    if Xgbest is not improved then
23:        isCompetitionNeeded = TRUE
24:
25: function selectLocalSearchWithCompetition(X)
26:    selectedLS = NULL
27:    XMTSLS = applyMTSLS(X, tr *0.5*MAXFES)
28:    XCMAES = applyCMAES(X, tr *0.5*MAXFES)
29:    if X = XMTSLS = XCMAES then
30:        Herhangi bir yerel arama kullanma
31:    else if objectiveValue(XMTSLS) < objectiveValue(XCMAES) then
32:        selectedLS = MTSLS
33:    else
```

Algoritma 7.1 (Devam) *SSEABC algoritmasının sözde kodu*

```
34:             selectedLS = CMAES
35:     if selectedLS ≠ NULL then
36:             isCompetitionNeeded = FALSE
37:     return selectedLS
39: function UpdatePopulationSize()
40:     if SN < SN_max and itr (mod g) == 0 then
41:             Denklem (3.5) kullanılarak popülasyona yeni bir çözüm ekle
42:
43: function UpdateEquationsPoolSize()
44:     if all search equations are used then
45:             ps havuz boyutunu Denklem (7.1) kullanılarak küçült
```

Tablo 7.1. *Algoritma 7.2'deki genelleştirilmiş arama denkleminin her bir bileşeni için alternatif seçenekler yer almaktadır.* $\phi 1$, $\phi 2$ *ve* $\phi 3$ *iki farklı aralık içinde değer alabilmektedir: bunlar [-1,-1] ve* $[-SF, SF]$ *Burada SF, rastgele seçilen pozitif reel sayıdır. Yeni bir arama denklemi oluşturulurken bu aralıklara rastgele değerler verilir.* $x_{G,j}$ *ve* $x_{GD,j}$ *global-en iyi çözümü ile en yakın çözümü temsil etmektedir.* $x_{r1,j}$ *ve* $x_{r2,j}$ *rastgele seçilmiş olan çözümü göstermektedir.* $x_{sc,j}$ *ve* $x_{WO,j}$ *sırasıyla en iyi ikinci çözüm ile en kötü çözümdür.* $x_{MD,j}$*, ortanca değere sahip olan çözümdür. Son olarak,* $x_{AVE,j}$*, bütün çözümlerin ortalamasını göstermektedir.*

m	terim1	terim2	terim3	terim4
1	$x_{i,j}$	$\phi 1(x_{i,j} - x_{G,j})$	$\phi 2(x_{i,j} - x_{G,j})$	$\phi 3(x_{i,j} - x_{G,j})$
$k\ 1 \le k \le D$	$x_{G,j}$	$\phi 1(x_{i,j} - x_{r1,j})$	$\phi 2(x_{i,j} - x_{r1,j})$	$\phi 3(x_{i,j} - x_{r1,j})$
$[t,k]\ 1 \le t < k \le D$	$x_{r1,j}$	$\phi 1(x_{G,j} - x_{r1,j})$	$\phi 2(x_{G,j} - x_{r1,j})$	$\phi 3(x_{G,j} - x_{r1,j})$
		$\phi 1(x_{r1,j} - x_{r2,j})$	$\phi 2(x_{r1,j} - x_{r2,j})$	$\phi 3(x_{r1,j} - x_{r2,j})$
		$\phi 1(x_{i,j} - x_{GD,j})$	$\phi 2(x_{i,j} - x_{GD,j})$	$\phi 3(x_{i,j} - x_{GD,j})$
		$\phi 1(x_{i,j} - x_{SC,j})$	$\phi 2(x_{i,j} - x_{SC,j})$	$\phi 3(x_{i,j} - x_{SC,j})$
		$\phi 1(x_{i,j} - x_{MD,j})$	$\phi 2(x_{i,j} - x_{MD,j})$	$\phi 3(x_{i,j} - x_{MD,j})$
		$\phi 1(x_{i,j} - x_{WO,j})$	$\phi 2(x_{i,j} - x_{WO,j})$	$\phi 3(x_{i,j} - x_{WO,j})$
		$\phi 1(x_{SC,j} - x_{MD,j})$	$\phi 2(x_{SC,j} - x_{MD,j})$	$\phi 3(x_{SC,j} - x_{MD,j})$
		$\phi 1(x_{MD,j} - x_{WO,j})$	$\phi 2(x_{MD,j} - x_{WO,j})$	$\phi 3(x_{MD,j} - x_{WO,j})$
		$\phi 1(x_{G,j} - x_{WO,j})$	$\phi 2(x_{G,j} - x_{WO,j})$	$\phi 3(x_{G,j} - x_{WO,j})$
		$\phi 1(x_{r1,j} - x_{MD,j})$	$\phi 2(x_{r1,j} - x_{MD,j})$	$\phi 3(x_{r1,j} - x_{MD,j})$
		$\phi 1(x_{G,j} - x_{MD,j})$	$\phi 2(x_{G,j} - x_{MD,j})$	$\phi 3(x_{G,j} - x_{MD,j})$
		$\phi 1(x_{r1,j} - x_{WO,j})$	$\phi 2(x_{r1,j} - x_{WO,j})$	$\phi 3(x_{r1,j} - x_{WO,j})$
		$\phi 1(x_{SC,j} - x_{r1,j})$	$\phi 2(x_{SC,j} - x_{r1,j})$	$\phi 3(x_{SC,j} - x_{r1,j})$
		$\phi 1(x_{i,j} - x_{AVE,j})$	$\phi 2(x_{i,j} - x_{AVE,j})$	$\phi 3(x_{i,j} - x_{AVE,j})$
		$\phi 1(x_{r1,j} - x_{AVE,j})$	$\phi 2(x_{r1,j} - x_{AVE,j})$	$\phi 3(x_{r1,j} - x_{AVE,j})$
		$\phi 1(x_{G,j} - x_{AVE,j})$	$\phi 2(x_{G,j} - x_{AVE,j})$	$\phi 3(x_{G,j} - x_{AVE,j})$
		kullanma	kullanma	kullanma

7.1.1. Kendinden uyarlamalı arama denklemi belirleme stratejisi

Sayısal bir optimizasyon problemini çözerken, bir ABC algoritmasının etkinliği, işçi arı ve gözcü arı adımlarında kullanılan arama denklemlerine bağlıdır. Ancak, her bir problem için uygun bir arama denklemi farklı olabilir ve bu nedenle iyi sonuçlar elde etmek için arama denklemi dikkatlice seçilmelidir (Aydın, 2015). Bununla birlikte, algoritma performansını etkileyen bir diğer önemli unsur, seçilen arama denkleminde (Liao vd., 2013) değiştirilecek olan değişkenlerin sayısıdır. Bu tezde her bir problem için gerekli arama denklemleri ve değişkenlik oranını kendinden uyarlamalı olarak belirleyebilecek bir strateji önerilmiştir. Bunun için bir genelleştirilmiş arama denklemi kalıbı tanımlanmış bu denklemdeki her bir terimin uygun bileşenlerinin, birkaç aday içerisinden seçilebilmesi kendinden uyarlamalı bir arama denklemi seçim stratejisi ile gerçekleştirilmiştir.

Algoritma 7.2'de dört terimden oluşan bir genelleştirilmiş arama denklem yapısı ve değiştirilecek değişken miktarı (m) tanımlanmıştır. Her bir terim ve m parametresi, Tablo 7.1'de listelenen çeşitli alternatif bileşenlere sahiptir. Böylelikle, önerilen genelleştirilmiş arama denklemi kullanılarak çeşitli arama denklemleri üretilebilmektedir. SSEABC'de, bir arama denklemleri havuzu mevcuttur. Bu havuz, her bir terimin bileşenleri Tablo 7.1'de verilen listeden rastgele seçilerek oluşturulmuş arama denklemleri ile doldurulur.

Algoritma 7.2. *SSEABC algoritmasında yer alan arama denkleminin genel yapısı*

1: **for** t = 1 to m **do**
2: rastgele bir boyut seç j $(1 \le j \le D)$
3: $\hat{x}_{i,j} = terim_1 + terim_2 + terim_3 + terim_4$

Rasgele oluşturulmuş bir arama denklemi, her iterasyonda denklem havuzundan seçilerek sırasıyla işçi arı ve gözcü arı adımında kullanılır. Her arama denkleminin başarı oranları, başarılı olmuş çözüm güncellemeleri sayılarak hesaplanır. Havuzda yer alan arama denklemlerinin tamamı kullanıldıktan sonra, arama denklemleri başarı oranlarına göre sıralanır. Başarısız olan arama denklemleri ise, aşağıdaki denklemin yardımıyla havuz büyüklüğü, ps, azaltılarak havuzdan çıkarılır.

$$ps = \frac{ps^2}{itr_{MAX}} \tag{7.1}$$

itr_{max}, hesaplanan maksimum iterasyon sayısının yaklaşık değeridir.

$$itr_{max} = \frac{MAXFES}{2 \times SN} \tag{7.2}$$

MAXFES, bir çalıştırma için maksimum fonksiyon çağırım sayısı ve $2 \times SN$ her iterasyondaki fonksiyon çağrımı sayısıdır. itr_{max}, yaklaşık bir değerdir. Çünkü artan popülasyon büyüklüğü stratejisi *SN*'nin değerini zaman içinde değiştirmektedir.

Ayrıca, yerel optimumdan kaçmak için, her bir havuz boyutu küçültme işleminden sonra rastgele üretilmiş bir arama denklemi de havuza dâhil edilir. Bu yaklaşım ile algoritma çalışırken havuzun boyutu doğrusal olarak azaltılır. Öte yandan, hesaplama bütçesinin yarısına ulaştıktan sonra çok az sayıda arama denklemi havuzda kalır. Bu nedenle, algoritma geri kalan iterasyonlarda kendinden uyarlamalı olarak seçilen iyi arama denklemleriyle çalışır.

7.1.2. Rekabetçi yerel arama seçim stratejisi

Son yıllarda metasezgisel algoritmalar, doğruluk eksikliklerini geliştirmek için yerel arama yöntemler ile melezleştirilmişlerdir. Literatürde yer alan birçok melezleştirme yöntemi, verilen problem seti için araştırma ve faydalanma davranış dengesini optimize etmeyi amaçlanmaktadır (Rego ve Glover, 2007; Talbi, 2002). Metasezgisel algoritmalar, çözülmesi düşünülen problemin türü göz önüne alınarak araştırmacılar tarafından belirlenen tek bir yerel arama yöntemi ile melezleştirilirler. Bu melezleştirme işlemi problem türü ve araştırmacıların tecrübeleri ile sınırlı kalmaktadır. Öte yandan gerçek dünya problemlerinin yapıları çok çeşitli olabilmektedir. Metasezgisel algoritmaların melezleştirildiği yerel arama yöntemleri farklı bir problem türü ile karşılaştığında işe yaramaz hale gelebilmektedir. Bu da yerel arama yönteminin algoritmanın çalıştırma bütçesini gereksiz yere tüketmesine sebep olur. Bu sıkıntının önüne geçilebilmesi için rekabetçi yerel arama seçim stratejisi önerilmiş ve SSEABC algoritmasına dâhil edilmiştir.

SSEABC, iki aşamayı içeren rekabetçi bir yerel arama seçim stratejisini kullanmaktadır: rekabet (competition) ve dağıtım (deployment). Rekabet aşamasında, Çoklu Yörünge Araması (Multiple Trajectory Search, MTSLS1) (Tseng ve Chen, 2008) ve Kovaryans Matrisi Adaptasyon Gelişim Stratejisi (Covariance Matrix Adaptation Evolution Strategy, CMAES) (Auger ve Hansen, 2005) yerel arama olarak kabul edilmiştir. SSEABC yerel arama kullanmadan sabit bir fonksiyon çağırım sayısı

(*CompBudget*) kadar çalıştırıldıktan sonra MTSLS1 ve CMAES yerel arama metotları da bu bütçe kadar çalıştırılır. Dağıtım aşamasında, yerel arama metotlarından birinin (CMAES ya da MTSLS1'den herhangi biri) rekabeti kazanması durumunda, her iterasyonda SSEABC o yerel arama metodunu başlatır. Aksi durumda, SSEABC bir yerel arama yapmadan çalışmaya devam eder. Algoritma bazı iterasyonlar sonrası durağanlığa (stagnation) girerse, daha sonraki iterasyonlarda hangi yerel aramanın uygun olduğunu belirlemek için rekabet aşamasına geri döner. Önerilen rekabetçi yerel arama stratejisinin sözde kodu Algoritma 7.1 içinde verilmiştir.

7.1.3. Artan popülasyon boyutu stratejisi

Artan popülasyon boyut stratejisi, "artan sosyal öğrenme" (Incremental Social Learning, ISL) çatısına dayanır (Liao, Oca, vd., 2011) (Montes de Oca vd., 2011). Popülasyon temelli optimizasyon algoritmalarındaki ISL'nin temel fikri, popülasyondaki iyi çözümler tarafından zaman içerisinde etkilenen yeni çözümleri popülasyona eklemektir (Liao, Oca, vd., 2011) (Montes de Oca vd., 2011). ISL, büyüme periyodu (growth period) ismindeki *g* parametresini kullanmaktadır. Her *g* iterasyonu sonunda popülasyona, maksimum popülasyon büyüklüğüne ulaşıncaya kadar yeni bir çözümün eklenir (Aydın ve Özyön, 2013) (Aydın ve Stützle, 2015; Liao, Oca, vd., 2011). SSEABC'de optimizasyon işlemi küçük bir popülasyon ile başlar. Daha sonra, her *g* iterasyonda, şimdiye kadarki en iyi çözümün (best-so-far solution) bilgisi kullanılarak yönlendirilen yeni bir çözüm (x_{gbest}) popülasyona aşağıdaki gibi eklenir:

$$x_{new,j} = x_{new,j} + \varphi_{i,j}(x_{gbest,j} - x_{new,j}) \tag{7.3}$$

x_{new}, Denklem (7.3)'e göre üretilen bir çözümdür ve x_{new} yeni çözümün son başlangıç pozisyonudur. Daha önceki çalışmalar (Liao, Oca, vd., 2011) (Montes de Oca vd., 2011), bu stratejinin, algoritmanın arama davranışını etkilediğini göstermiştir. Küçük *g* değerleri keşfetmeyi teşvik ederken, büyük *g* değerleri faydalanma yeteneğine yardımcı olur. Bu nedenle *g* parametresi, algoritma performansını artırmak için dikkatlice ayarlanmalıdır.

7.2. Deneysel Sonuçlar

7.2.1. CEC'17 ölçüt seti deneyleri

Önerilen SSEABC algoritmasının performans değerlendirmesi ve yerel arama seçim stratejisinin başarımını doğrulamak için, CEC'17 ölçüt seti (Awad vd., 2016) ile deneysel testler gerçekleştirilmiştir. İlk olarak, SSEABC algoritması literatürdeki ABC varyantları ile karşılaştırılmıştır. İkinci olarak, çoklu arama denklemine sahip ABC algoritmalarıyla SSEABC algoritması karşılaştırılmıştır. Üçüncü olarak ise, rekabetçi yerel arama seçim stratejisinin katkısı, önerilen stratejiye sahip SSEABC ile diğer varyantları karşılaştırarak test edilmiştir (tek yerel arama içeren SSEABC ve yerel arama olmadan SSEABC). Son olarak, CEC'17 yarışmasına katılan yarışmacı algoritmalar ile karşılaştırma yapılmıştır.

7.2.1.1. *Ölçüt seti ve algoritma yapılandırmaları*

Tüm deneysel çalışmalar, 3 unimodal, 13 multimodal, 6 hibrit ve 8 kompozit fonksiyonlardan oluşan ve toplamda 30 fonksiyon içeren CEC'17 ölçüt seti kullanılarak gerçekleşmiştir. CEC'17 ölçüt setindeki test fonksiyonlarının adları ve bazı özellikleri Tablo 5.6'de listelenmiştir.

CEC'17 ölçüt seti için sonlandırma kriterleri, aşağıda belirtilen şartlara göre uygulanmıştır (Awad vd., 2016). Algoritma sonlandırma kriteri olarak Maksimum fonksiyon çalıştırma sayısı (MAXFES), $10000 \times D$ olarak alınır; burada D, bir fonksiyonun boyutunu gösterir. Algoritma MAXFES ile sonlandırıldığın hata değeri kaydedilir. Hata değeri $f(x) - f(x^*)$ olarak hesaplanır, burada x bulunan en iyi aday çözümdür ve x^* optimum çözümdür. CEC'17'deki ölçüt tanımlarına göre, optimum eşik değeri 10^{-8}'den daha düşük hata değerleri 10^{-8} olarak kabul edilir. Her fonksiyon için, tüm algoritmalar her biri farklı bir başlangıç popülasyonuna sahip olmak üzere üst üste 51 defa çalıştırılır. Ölçüt fonksiyonların her bir boyutunda bir algoritma ile elde edilen ortalama hata hesaplanır. CEC'17'deki tüm test fonksiyonlarındaki bu ortalama hatalar daha sonra algoritmaların performansını karşılaştırmak için kullanılır.

SSEABC algoritmasının ve karşılaştırmalarda kullanılan diğer tüm ABC varyantlarının parametreleri, Bölüm 4.2.1'de detayları verilmiş olan irace aracı ile (López-Ibáñez vd., 2016) ayarlanmıştır.

irace, rastgele aday yapılandırmaları üzerinden bir sonuç verdiğinden, her bağımsız çalışma için farklı parametre ayarları önerir. Bu nedenle bu çalışmada, her ABC algoritması için uygun parametre yapılandırması beş bağımsız irace aracının çalıştırılmasıyla elde edilmiştir. Her bir çalışma için irace, varsayılan parametre değerleri ile 10000 deneme bütçesi kullanılarak çalıştırılmıştır. 10 boyutlu CEC'17 ölçüt fonksiyonları problem örnekleri olarak seçilmiştir. irace aracı kullanılarak SSEABC algoritması için önerilen parametre değerleri, Tablo 7.2'de verilmiştir. Karşılaştırmada kullanılan diğer ABC varyantları için önerilen parametre değerleri Tablo 7.3'de verilmiştir.

Tablo 7.2. *SSEABC algoritmasının parametreleri ile ayarlanmış değerleri*

İsim	Tür	Aralık	Ayarlanmış değerler	Açıklama
SN	Tam Sayı	[3,100]	66	Başlangıç popülasyon boyutu
limitF	Reel	[0,4]	0.6873	Limit faktörü
SN_{max}	Tam Sayı	[20,100]	81	Maksimum popülasyon boyutu
g	Tam Sayı	[1,20]	1	Popülasyon boyutu büyüme periyodu
a	Reel	[0,10]	8.5569	CMAES'ye ait bir kontrol parametresi
b	Reel	[0,5]	2.3072	CMAES'ye ait bir kontrol parametresi
c	Reel	[0.1,1]	0.1759	CMAES'ye ait bir kontrol parametresi
d	Reel	[1,4]	2.6384	CMAES'ye ait bir kontrol parametresi
e	Reel	[-20,-6]	-17.13	CMAES'ye ait bir kontrol parametresi
f	Reel	[-20,-6]	-12.72	CMAES'ye ait bir kontrol parametresi
g	Reel	[-20,-6]	-15.8	CMAES'ye ait bir kontrol parametresi
$MTSLS_{itr}$	Tam Sayı	[3,100]	97	MTSLS1 iterasyon sayısı
$CMAES_{FES}$	Kategorik	(0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3)* MAXFES	0.3	CMAES için fonksiyon çağrım sayısı
CompBudget	Kategorik	(0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3)* MAXFES	0.01	Rekabet aşaması için fonksiyon çağrım sayısı
ps	Kategorik	(50, 100, 500, 1000, 1500, 2000)	50	Denklem havuzunun boyutu

Tablo 7.3. *ABC algoritmalarının parametreleri ile ayarlanmış değerleri*

Algoritmalar	Parametreler
ABC	limitF = 2.4076 SN = 64
BABC	limitF = 0.2991 SN = 20 wMin = 0.0197 wMax = 0.9188
CABC	limitF = 2.2208 SN = 47 chaoticmap = 1
EABC	limitF = 0.908 SN = 33
GABC	limitF = 0.6545 SN = 18 C = 1.9024
GDABC	limitF = 0.3145 SN = 57 C = 1.3083
IABC	limitF = 0.9022 SN = 8 shakingfactor = -0.9973 maxfoodsize = 46 growthperiod = 15
MABC	limitF = 1.3815 SN = 95 MR = 0.2886 adaptiveSF = 1 SF = 0.214
OPIABC	limitF = 2.5898 SN = 72

7.2.1.2. *ABC varyantları ile karşılaştırma*

Bu alt bölümde SSEABC, orijinal ABC ile ABC'nin sekiz gelişmiş varyantı ile karşılaştırılmıştır. Bu algoritmalar; Şimdiye Kadarki En İyi Seçim ABC (Best-so-far Selection ABC, BABC) (Banharnsakun vd., 2011), Kaotik ABC (Chaotic ABC, CABC) (Alatas, 2010), Geliştirilmiş ABC (Enhanced ABC, EABC) (Gao vd., 2014), Global-En İyi ABC (Global-best ABC, GABC) (W. Gao vd., 2012), Global-En İyi Mesafe rehberli ABC (Global-best Distance guided, ABC, GDABC) (Diwold vd., 2011), Artan ABC (Incremental ABC, IABC) (Aydın vd., 2011), Değiştirilmiş ABC (Modified ABC, MABC) (Akay ve Karaboga, 2012) ve Tek-Nokta Kalıtım ABC (One-Position Inheritance ABC, OPIABC) (Zhang ve Yuen, 2013). SSEABC ve ABC varyantlarının parametrelerini ayarlamak için 10 boyutlu ölçüt fonksiyon seti kullanıldığından, 30 ve 50 boyutlu fonksiyonlar kullanılarak karşılaştırma sonuçları elde edilmiştir. 30 ve 50 boyutları için karşılaştırma sonuçları Tablo 7.4 ve Tablo 7.5'de verilmiştir.

Tablo 7.4. *SSEABC ve ABC varyantlarının 30 boyut için karşılaştırma sonuçları*

Fonksiyonlar	SSEABC	ABC	BABC	CABC	EABC	GABC	GDABC	IABC	MABC	OPIABC
f1	**1.000E-08**	4.244E+02	2.077E+02	2.184E+02	7.244E+02	3.411E+02	4.130E+02	1.189E+02	2.619E+03	3.136E+02
f2	**1.000E-08**	1.333E+10	2.201E+11	8.591E+07	5.425E+09	6.677E+09	4.657E+08	1.375E+10	3.526E+04	1.268E+11
f3	**1.000E-08**	1.226E+05	6.676E+04	6.837E+04	1.246E+05	1.095E+05	1.041E+05	5.283E+04	7.383E+04	8.269E+04
f4	5.114E+01	2.893E+01	6.697E+01	2.558E+01	5.273E+01	4.897E+01	4.091E+01	**2.165E+01**	4.735E+01	3.278E+01
f5	**3.570E+00**	9.165E+01	1.630E+02	8.827E+01	4.133E+01	4.644E+01	5.177E+01	6.560E+01	1.754E+02	5.604E+01
f6	**1.000E-08**	1.122E-04	**1.000E-08**	1.221E-06	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	4.715E+01	**1.000E-08**
f7	**3.516E+01**	1.025E+02	1.794E+02	1.044E+02	6.939E+01	6.665E+01	7.513E+01	9.809E+01	5.745E+02	8.517E+01
f8	**3.609E+00**	9.950E+01	1.031E+02	9.485E+01	4.568E+01	5.984E+01	5.468E+01	6.008E+01	1.655E+02	6.130E+01
f9	**1.000E-08**	1.011E+03	1.873E+03	5.343E+02	3.201E+01	1.044E+02	9.627E+01	2.671E+02	6.488E+03	2.156E+02
f10	**1.191E+03**	2.249E+03	2.493E+03	2.324E+03	1.935E+03	2.150E+03	2.023E+03	1.989E+03	3.294E+03	2.241E+03
f11	**3.236E+01**	4.167E+02	2.432E+02	2.417E+02	5.290E+02	8.478E+02	1.523E+02	1.732E+02	1.208E+02	3.060E+02
f12	**1.292E+03**	4.652E+05	9.328E+05	5.592E+05	7.225E+05	2.195E+05	4.794E+05	2.007E+05	1.025E+05	5.267E+05
f13	**9.665E+01**	7.651E+03	2.453E+03	6.487E+03	1.314E+04	3.020E+04	1.916E+04	1.029E+04	7.485E+03	8.512E+03
f14	**1.097E+02**	7.235E+04	8.908E+04	1.231E+05	1.280E+05	2.087E+05	5.775E+04	2.796E+04	4.875E+03	6.412E+04
f15	**1.083E+02**	2.979E+03	3.268E+02	6.155E+02	5.022E+03	1.495E+04	7.209E+03	2.915E+03	9.316E+02	2.421E+03

Tablo 7.4. (Devam) *SSEABC ve ABC varyantlarının 30 boyut için karşılaştırma sonuçları*

Fonksiyonlar	SSEABC	ABC	BABC	CABC	EABC	GABC	GDABC	IABC	MABC	OPIABC
f16	**1.508E+02**	6.037E+02	9.164E+02	6.843E+02	6.350E+02	7.659E+02	5.629E+02	5.004E+02	9.209E+02	5.462E+02
f17	**7.404E+01**	1.992E+02	2.645E+02	1.922E+02	1.280E+02	9.639E+01	1.347E+02	1.447E+02	3.745E+02	1.270E+02
f18	**1.013E+02**	2.006E+05	2.224E+05	1.806E+05	2.410E+05	2.208E+05	1.523E+05	9.856E+04	1.054E+05	1.763E+05
f19	**5.252E+01**	2.350E+03	1.602E+03	1.593E+03	6.577E+03	8.811E+03	6.205E+03	5.018E+03	6.843E+02	2.392E+03
f20	**8.060E+01**	2.684E+02	3.349E+02	2.705E+02	1.870E+02	1.339E+02	1.769E+02	2.027E+02	4.207E+02	1.980E+02
f21	**1.928E+02**	2.379E+02	2.637E+02	2.027E+02	2.450E+02	2.557E+02	2.540E+02	2.502E+02	3.249E+02	2.366E+02
f22	1.284E+02	4.197E+02	1.010E+02	1.006E+02	2.614E+02	**1.001E+02**	3.276E+02	9.106E+02	3.070E+03	1.590E+02
f23	**3.226E+02**	4.179E+02	4.558E+02	4.402E+02	4.010E+02	4.070E+02	4.040E+02	3.963E+02	5.091E+02	4.087E+02
f24	4.182E+02	**3.950E+02**	5.779E+02	5.433E+02	5.108E+02	5.260E+02	4.710E+02	4.854E+02	5.711E+02	4.782E+02
f25	3.865E+02	3.847E+02	3.836E+02	3.835E+02	3.868E+02	3.851E+02	3.856E+02	**3.786E+02**	3.861E+02	3.857E+02
f26	3.932E+02	3.246E+02	2.328E+02	**2.099E+02**	1.333E+03	1.750E+03	7.815E+02	7.969E+02	2.191E+03	3.595E+02
f27	5.036E+02	5.154E+02	5.200E+02	5.159E+02	5.113E+02	5.055E+02	5.098E+02	**5.022E+02**	5.322E+02	5.085E+02
f28	**3.021E+02**	4.078E+02	4.013E+02	4.014E+02	4.110E+02	4.141E+02	4.034E+02	3.987E+02	4.189E+02	4.186E+02
f29	**4.732E+02**	6.323E+02	6.951E+02	6.077E+02	5.512E+02	5.282E+02	5.566E+02	5.301E+02	1.131E+03	5.500E+02
f30	**2.161E+03**	6.072E+03	6.536E+03	6.460E+03	7.253E+03	1.346E+04	5.765E+03	5.963E+03	1.124E+04	6.484E+03
p-value		0.00001	0.00002	0.00003	0.00000	0.00001	0.00000	0.00001	0.00000	0.00001
rank	**1.80**	6.30	6.77	5.43	6.17	5.93	4.93	4.71	7.60	5.20
SSEABC-win		26	26	26	29	26	27	26	28	26
SSEABC-lost		4	3	4	0	3	2	3	2	3
draw		0	1	0	1	1	1	1	0	1

Tablo 7.5 *SSEABC ve ABC varyantlarının 50 boyut için karşılaştırma sonuçları*

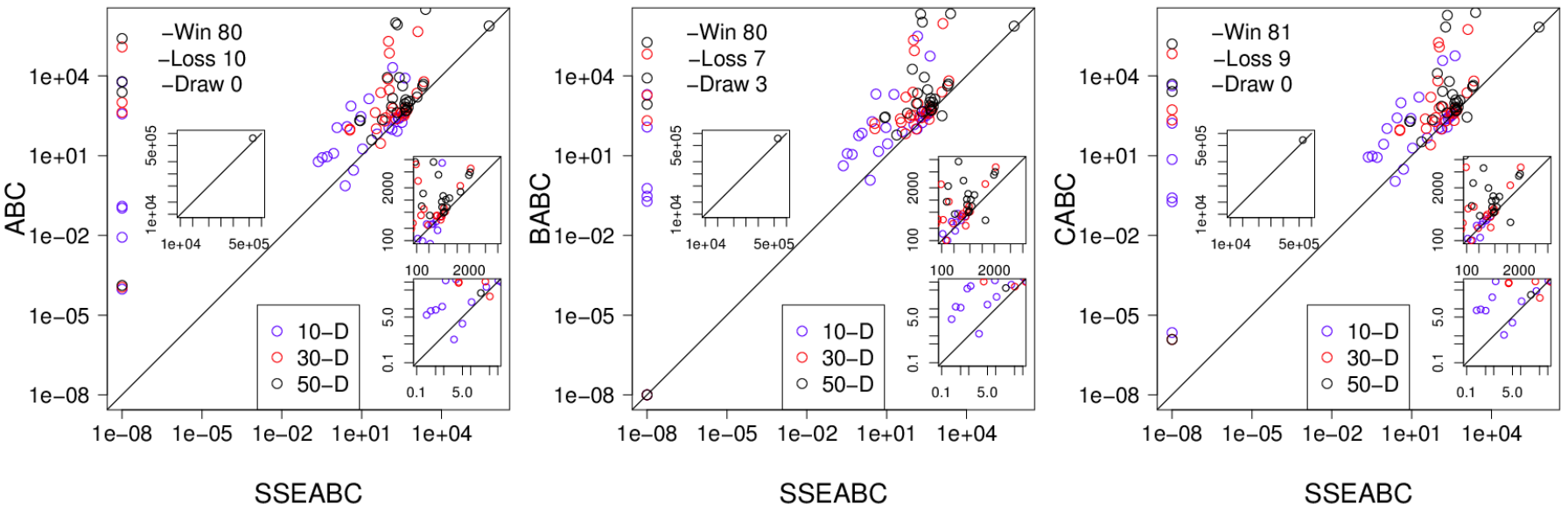
Fonksiyonlar	SSEABC	ABC	BABC	CABC	EABC	GABC	GDABC	IABC	MABC	OPIABC
f1	**1.000E-08**	2.444E+03	8.492E+02	2.583E+03	2.791E+03	4.508E+03	4.779E+03	1.997E+03	3.502E+03	2.381E+03
f2	**1.000E-08**	6.963E+24	1.426E+21	7.707E+14	1.671E+16	1.796E+15	4.028E+12	4.389E+18	1.585E+18	4.115E+24
f3	**1.000E-08**	2.569E+05	1.836E+05	1.599E+05	2.510E+05	2.169E+05	2.415E+05	1.344E+05	2.124E+05	2.041E+05
f4	**2.376E+01**	3.879E+01	5.965E+01	3.312E+01	5.029E+01	1.050E+02	4.282E+01	3.928E+01	6.484E+01	4.739E+01
f5	**8.857E+00**	2.050E+02	2.871E+02	1.936E+02	1.035E+02	1.297E+02	1.257E+02	1.640E+02	4.264E+02	1.389E+02
f6	**1.000E-08**	1.310E-04	**1.000E-08**	1.211E-06	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	6.263E+01	**1.000E-08**
f7	**6.005E+01**	2.248E+02	3.996E+02	2.228E+02	1.390E+02	1.646E+02	1.566E+02	2.211E+02	1.107E+03	1.803E+02
f8	**8.369E+00**	2.134E+02	2.696E+02	1.958E+02	1.033E+02	1.190E+02	1.236E+02	1.652E+02	4.095E+02	1.418E+02
f9	**1.000E-08**	6.144E+03	8.314E+03	4.946E+03	2.821E+02	4.016E+02	7.509E+02	2.976E+03	1.773E+04	2.054E+03
f10	**1.947E+03**	4.136E+03	4.188E+03	3.948E+03	3.689E+03	3.527E+03	3.719E+03	3.764E+03	6.025E+03	4.388E+03
f11	**1.444E+02**	1.456E+03	2.217E+03	2.630E+03	2.526E+03	4.805E+03	1.330E+03	1.044E+03	2.525E+02	1.240E+03
f12	**2.513E+03**	3.169E+06	2.204E+06	2.436E+06	3.758E+06	4.218E+06	2.846E+06	2.877E+06	9.109E+05	3.337E+06
f13	**3.455E+02**	4.224E+03	3.033E+03	1.989E+03	3.863E+03	2.246E+03	9.108E+03	4.119E+03	2.354E+04	2.774E+03
f14	**2.190E+02**	8.472E+05	1.059E+06	1.919E+06	1.236E+06	4.160E+05	7.311E+05	3.609E+05	2.971E+04	6.988E+05
f15	**2.603E+02**	8.723E+03	8.956E+03	6.207E+03	7.177E+03	1.122E+04	9.415E+03	6.490E+03	6.434E+03	7.885E+03

Tablo 7.5. (Devam) *SSEABC ve ABC varyantlarının 50 boyut için karşılaştırma sonuçları*

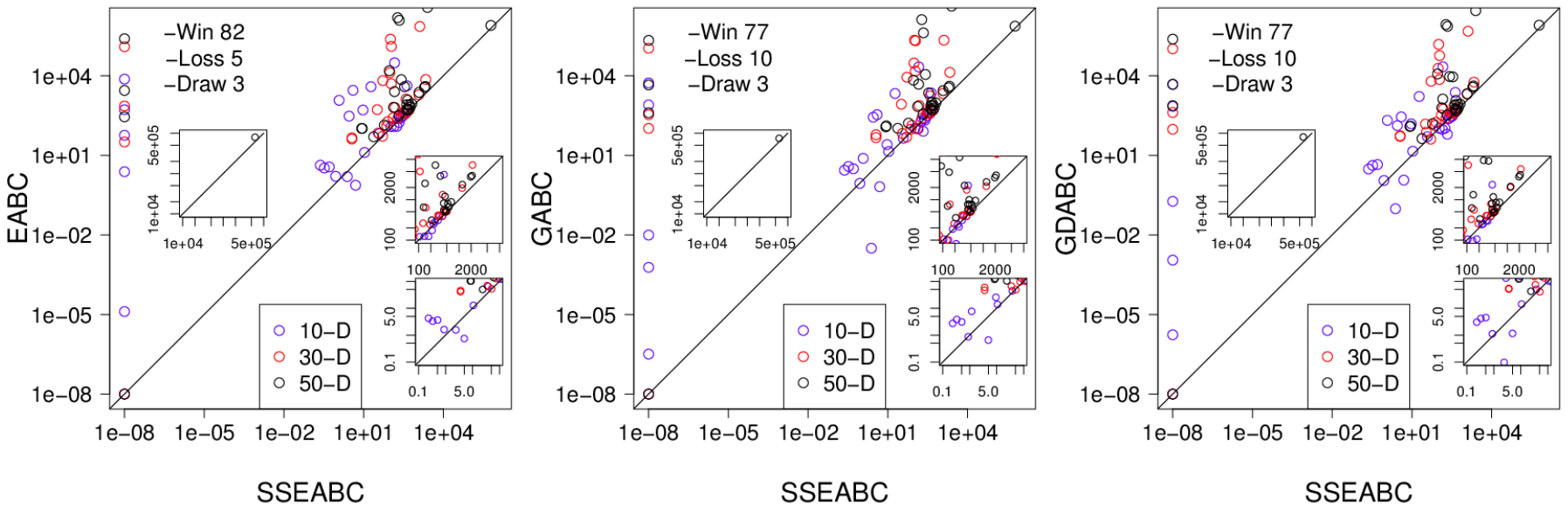
Fonksiyonlar	SSEABC	ABC	BABC	CABC	EABC	GABC	GDABC	IABC	MABC	OPIABC
f16	**4.514E+02**	1.267E+03	1.434E+03	1.256E+03	1.209E+03	9.943E+02	1.177E+03	1.013E+03	2.065E+03	1.183E+03
f17	**4.178E+02**	9.040E+02	1.047E+03	9.522E+02	8.146E+02	7.625E+02	8.073E+02	7.614E+02	1.363E+03	7.793E+02
f18	**1.889E+02**	1.001E+06	2.130E+06	6.928E+05	1.555E+06	1.309E+06	8.545E+05	4.713E+05	1.916E+05	9.119E+05
f19	**9.209E+01**	8.767E+03	1.801E+04	1.245E+04	1.556E+04	6.804E+03	1.176E+04	1.031E+04	5.094E+03	1.020E+04
f20	**1.328E+02**	7.072E+02	9.115E+02	7.208E+02	6.504E+02	6.957E+02	6.213E+02	4.928E+02	1.162E+03	6.062E+02
f21	**2.106E+02**	4.179E+02	4.525E+02	4.094E+02	3.113E+02	3.576E+02	3.334E+02	3.658E+02	5.626E+02	3.487E+02
f22	**2.092E+03**	4.895E+03	4.983E+03	4.424E+03	3.945E+03	4.039E+03	4.136E+03	4.481E+03	6.712E+03	4.631E+03
f23	**4.243E+02**	6.703E+02	7.658E+02	6.648E+02	5.584E+02	5.783E+02	5.787E+02	6.031E+02	8.527E+02	5.946E+02
f24	**5.267E+02**	1.027E+03	1.028E+03	9.992E+02	7.967E+02	7.614E+02	8.452E+02	8.015E+02	8.493E+02	8.240E+02
f25	**4.789E+02**	5.149E+02	5.439E+02	5.162E+02	5.238E+02	4.992E+02	5.030E+02	5.092E+02	5.177E+02	5.203E+02
f26	1.204E+03	1.598E+03	3.168E+02	**2.818E+02**	2.369E+03	2.755E+03	2.118E+03	2.331E+03	5.392E+03	2.070E+03
f27	**5.577E+02**	6.588E+02	7.066E+02	6.697E+02	6.523E+02	6.164E+02	6.285E+02	5.787E+02	9.365E+02	6.209E+02
f28	**4.598E+02**	4.838E+02	4.947E+02	4.836E+02	4.984E+02	7.699E+02	4.729E+02	4.820E+02	5.111E+02	4.923E+02
f29	**6.436E+02**	1.121E+03	1.341E+03	1.217E+03	8.825E+02	9.522E+02	8.615E+02	8.549E+02	2.106E+03	8.599E+02
f30	**6.128E+05**	7.643E+05	7.508E+05	6.865E+05	7.927E+05	7.438E+05	8.053E+05	7.315E+05	6.603E+06	7.534E+05
p-value		0.00000	0.00001	0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
rank	**1.07**	7.07	7.47	5.80	5.50	5.10	4.97	4.27	7.83	5.23
SSEABC-win		30	28	29	29	29	29	29	30	29
SSEABC-lost		0	1	1	0	0	0	0	0	0
draw		0	1	0	1	1	1	1	0	1

CEC'17 ölçüt setinde 30 ve 50 boyutundaki bütün fonksiyonlar göz önünde bulundurulduğunda, SSEABC tüm ABC algoritmalarından belirgin biçimde daha iyi performans göstermiştir. Unimodal fonksiyonlarda, SSEABC algoritması her durumda optimum eşik değerinin altında dikkate değer bir sonuç elde etmiştir. Bununla birlikte, ABC varyantlarının herhangi bir optimum değere ulaşamadığı açıkça görülmektedir. Sonuç olarak SSEABC'nin, karşılaştırılan tüm ABC algoritmalarından çok daha iyi olduğu söylenebilir. Birden fazla yerel optimum içeren multimodal fonksiyonların sonuçları incelendiğinde, f4 ve f6 hariç tüm multimodal fonksiyonlarda, SSEABC algoritmasının karşılaştırılan diğer ABC varyantlarından daha iyi sonuçlar ürettiği görülmüştür. f6 fonksiyonunda SSEABC, BABC, EABC, GABC, GDABC, IABC ve OPIABC tarafından elde edilen hata değerleri optimum eşiğin altındadır. Yalnızca 30 boyut için f4 fonksiyonunda, IABC, SSEABC'den daha küçük hata değerine sahip olmuştur. Önerilen algoritma, unimodal ve multimodal test fonksiyonlarından çok daha zor olan hibrit ve kompozit fonksiyonlarından da daha iyi bir performans göstermiştir. Buraya dikkat edilmelidir ki, SSEABC'nin 30 boyutlu hibrit fonksiyonlar da elde ettiği hata değerleri, bazı fonksiyonlar için diğer tüm ABC varyantlarından daha iyi değildir. Buna karşın, bu durum problem boyutunun 50'ye yükselmesiyle değişmiştir. Sadece bir fonksiyonda CABC, SSEABC algoritmasından daha iyi sonuç vermiştir. Diğerlerinde ise SSABC'nin daha iyi sonuçlar elde ettiği görülmektedir. Ölçeklenebilirlik performansı yönünden SSEABC'nin diğer ABC varyantlarından daha iyi olduğunu gösterir.

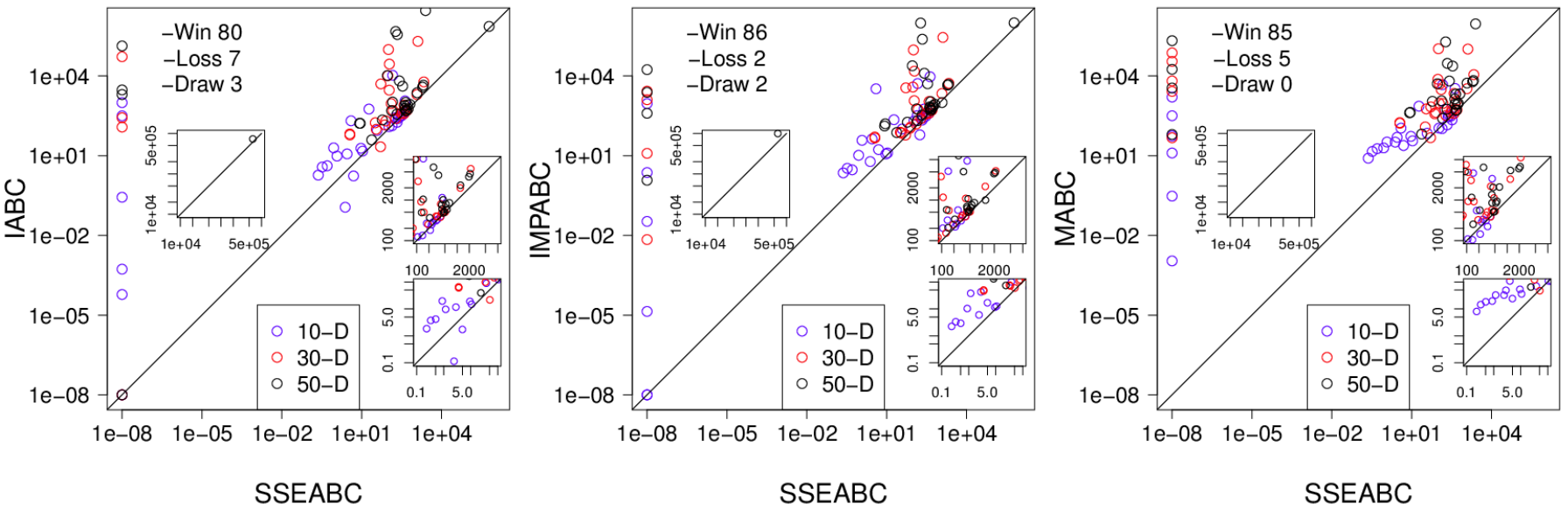
SSEABC algoritmasının; 10, 30 ve 50 boyutlu CEC'17 fonksiyonlarındaki ABC varyantlarına karşı kazanma (win), kaybetme (lost) ve beraberlik (draw) durumları Şekil 7.1'de gösterilmiştir. SSEABC ve ABC varyantları birebir bazda karşılaştırılmıştır. SSEABC'nin ortalama hataları x ekseninde, ABC algoritmalarının sonuçları da y ekseninde yer almaktadır. Buna ek olarak, her bir şekildeki ana diyagonal üzerindeki noktalar, SSEABC algoritmasının kazanmasını temsil eder. Buna karşın, ana diyagonalin altındaki noktalar, ABC varyantının SSEABC algoritmasından daha iyi olduğunu gösterir. Örnek olarak, SSEABC'nin yüzdelik olarak GDABC ve GABC algoritmalarından en az yüzde 85,5 daha iyi olduğu söylenebilir.



Şekil 7.1. *10, 30 ve 50 boyutlu ölçüt fonksiyonlarındaki SSEABC (x-ekseni) ve ABC varyantları (y-ekseni) konfigürasyonları ile elde edilen ortalama amaç fonksiyonu değerlerinin korelasyon grafiği. Her bir nokta algoritmalar tarafından elde edilen ortalama amaç fonksiyonu değerini göstermektedir.*



Şekil 7.1. (Devam) *10, 30 ve 50 boyutlu ölçüt fonksiyonlarındaki SSEABC (x-ekseni) ve ABC varyantları (y-ekseni) konfigürasyonları ile elde edilen ortalama amaç fonksiyonu değerlerinin korelasyon grafiği. Her bir nokta algoritmalar tarafından elde edilen ortalama amaç fonksiyonu değerini göstermektedir.*



Şekil 7.1. (Devam) *10, 30 ve 50 boyutlu ölçüt fonksiyonlarındaki SSEABC (x-ekseni) ve ABC varyantları (y-ekseni) konfigürasyonları ile elde edilen ortalama amaç fonksiyonu değerlerinin korelasyon grafiği. Her bir nokta algoritmalar tarafından elde edilen ortalama amaç fonksiyonu değerini göstermektedir.*

7.2.1.3. *Çoklu arama denklemi tabanlı ABC varyantları ile karşılaştırma*

Önceki alt bölümde SSEABC, sadece bir arama denklemi kullanan ABC varyantları ile karşılaştırılmıştır. Bu alt bölümde ise, önerilen stratejilerinin algoritmaya katkısını incelemek için çoklu arama denklemleri kullanan ABC algoritmaları ile testler yapılmıştır. Testte katılan bu algoritmalar, Geliştirilmiş ABC (Improved ABC, IMPABC) (Gao ve Liu, 2011), Çok-Strateji Topluluğu ABC (Multi-strategy Ensemble ABC, MEABC) (Wang vd., 2014) ve bir diğer Geliştirilmiş ABC (Improved ABC, IMABC) (Ghambari ve Rahati, 2018). 30 ve 50 boyutlu CEC'17 fonksiyonlarından elde edilen karşılaştırma sonuçları, sırasıyla Tablo 7.6 ve Tablo 7.7'de verilmiştir.

Tablolar incelendiğinde, SSEABC'nin diğer karşılaştırılan algoritmalardan daha üstün olduğu açıktır. Sadece 30 boyuttaki f4, f22, f25 ve f26 fonksiyonlarında, SSEABC ilk sırada yer alamamıştır. Problem boyutu 50 olduğunda, algoritmada yer alan strateji SSEABC'nin performansını daha da artırmakta ve SSEABC, f26 hariç tüm fonksiyonlarda ilk sırada yer almaktadır. Bu da gösteriyor ki, maksimum fonksiyon çağırım sayısı (MAXFES) ile birlikte boyutun artması, en doğru arama denklemleri ile yerel arama stratejisini seçme olasılığına önemli ölçüde katkı sağladığını göstermektedir. Sonuç olarak, önerilen strateji yalnızca algoritmanın performansını iyileştirmekle kalmaz, aynı zamanda boyut ve MAXFES artırıldığında bunların performansa katkılarını da arttırır.

Tablo 7.6. *SSEABC ve çoklu-arama denklemi tabanlı ABC varyantlarının 30 boyut için karşılaştırma sonuçları*

Fonksiyonlar	SSEABC	IMPABC	MEABC	IMABC
f1	**1.000E-08**	2.679E+03	5.661E+02	1.076E+02
f2	**1.000E-08**	9.372E+16	1.774E+11	1.060E+06
f3	**1.000E-08**	1.272E+03	1.161E+05	6.087E+04
f4	5.114E+01	5.943E+01	**2.865E+01**	4.872E+01
f5	**3.570E+00**	5.034E+01	5.167E+01	9.508E+01
f6	**1.000E-08**	6.956E-03	**1.000E-08**	**1.000E-08**
f7	**3.516E+01**	7.077E+01	7.819E+01	1.085E+02
f8	**3.609E+00**	4.620E+01	5.997E+01	9.704E+01
f9	**1.000E-08**	1.223E+01	1.825E+02	8.344E+02
f10	**1.191E+03**	2.175E+03	2.024E+03	2.211E+03
f11	**3.236E+01**	7.633E+01	6.099E+02	1.022E+02
f12	**1.292E+03**	2.816E+05	7.223E+05	6.628E+05
f13	**9.665E+01**	3.824E+03	1.721E+04	9.962E+02
f14	**1.097E+02**	1.514E+04	1.484E+05	1.323E+05
f15	**1.083E+02**	1.169E+03	6.281E+03	3.362E+02
f16	**1.508E+02**	5.538E+02	5.688E+02	7.415E+02
f17	**7.404E+01**	1.062E+02	1.612E+02	2.339E+02
f18	**1.013E+02**	9.502E+04	2.452E+05	1.920E+05
f19	**5.252E+01**	3.563E+03	4.464E+03	9.485E+02
f20	**8.060E+01**	1.206E+02	1.870E+02	2.993E+02
f21	**1.928E+02**	2.473E+02	2.460E+02	2.450E+02
f22	1.284E+02	1.531E+02	2.011E+02	**1.072E+02**
f23	**3.226E+02**	4.123E+02	4.033E+02	4.378E+02
f24	**4.182E+02**	5.115E+02	5.019E+02	5.175E+02
f25	3.865E+02	3.916E+02	3.853E+02	**3.837E+02**
f26	3.932E+02	1.096E+03	5.565E+02	**2.271E+02**
f27	**5.036E+02**	5.150E+02	5.106E+02	5.157E+02
f28	**3.021E+02**	3.651E+02	4.045E+02	3.584E+02
f29	**4.732E+02**	5.302E+02	5.696E+02	6.291E+02
f30	**2.161E+03**	5.274E+03	9.007E+03	4.870E+03
p-value		0.00000	0.00000	0.00003
rank	**1.20**	2.80	3.07	2.83
SSEABC-win		30	27	25
SSEABC-lost		0	2	4
draw		0	1	1

Tablo 7.7. *SSEABC ve çoklu-arama denklemi tabanlı ABC varyantlarının 50 boyut için karşılaştırma sonuçları*

Fonksiyonlar	SSEABC	IMPABC	MEABC	IMABC
f1	**1.000E-08**	2.409E+03	5.880E+03	1.713E+03
f2	**1.000E-08**	6.091E+32	1.842E+24	1.385E+13
f3	**1.000E-08**	1.728E+04	2.598E+05	1.500E+05
f4	**2.376E+01**	7.411E+01	4.391E+01	3.754E+01
f5	**8.857E+00**	1.586E+02	1.295E+02	2.174E+02
f6	**1.000E-08**	1.189E+00	**1.000E-08**	**1.000E-08**
f7	**6.005E+01**	1.820E+02	1.618E+02	2.346E+02
f8	**8.369E+00**	1.305E+02	1.325E+02	2.233E+02
f9	**1.000E-08**	3.934E+02	1.274E+03	4.738E+03
f10	**1.947E+03**	4.517E+03	3.857E+03	4.050E+03
f11	**1.444E+02**	2.071E+02	2.447E+03	1.215E+03
f12	**2.513E+03**	6.658E+06	3.872E+06	2.105E+06
f13	**3.455E+02**	5.059E+03	6.196E+03	1.144E+03
f14	**2.190E+02**	2.413E+05	8.862E+05	1.463E+06
f15	**2.603E+02**	1.260E+04	8.956E+03	2.449E+03
f16	**4.514E+02**	1.077E+03	1.135E+03	1.497E+03
f17	**4.178E+02**	6.870E+02	8.381E+02	9.670E+02
f18	**1.889E+02**	9.799E+05	1.440E+06	1.318E+06
f19	**9.209E+01**	2.413E+04	1.118E+04	3.931E+03
f20	**1.328E+02**	5.758E+02	5.843E+02	7.514E+02
f21	**2.106E+02**	3.357E+02	3.479E+02	4.297E+02
f22	**2.092E+03**	4.795E+03	4.611E+03	4.369E+03
f23	**4.243E+02**	6.227E+02	5.857E+02	6.806E+02
f24	**5.267E+02**	6.533E+02	8.477E+02	1.029E+03
f25	**4.789E+02**	5.736E+02	5.050E+02	5.067E+02
f26	1.204E+03	9.976E+02	2.211E+03	**4.808E+02**
f27	**5.577E+02**	7.590E+02	6.388E+02	6.799E+02
f28	**4.598E+02**	5.070E+02	4.797E+02	4.764E+02
f29	**6.436E+02**	8.576E+02	8.958E+02	1.202E+03
f30	**6.128E+05**	9.955E+05	7.461E+05	7.077E+05
p-value		0.00001	0.00000	0.00001
rank	**1.07**	2.97	2.93	2.93
SSEABC-win		29	29	28
SSEABC-lost		1	0	1
draw		0	1	1

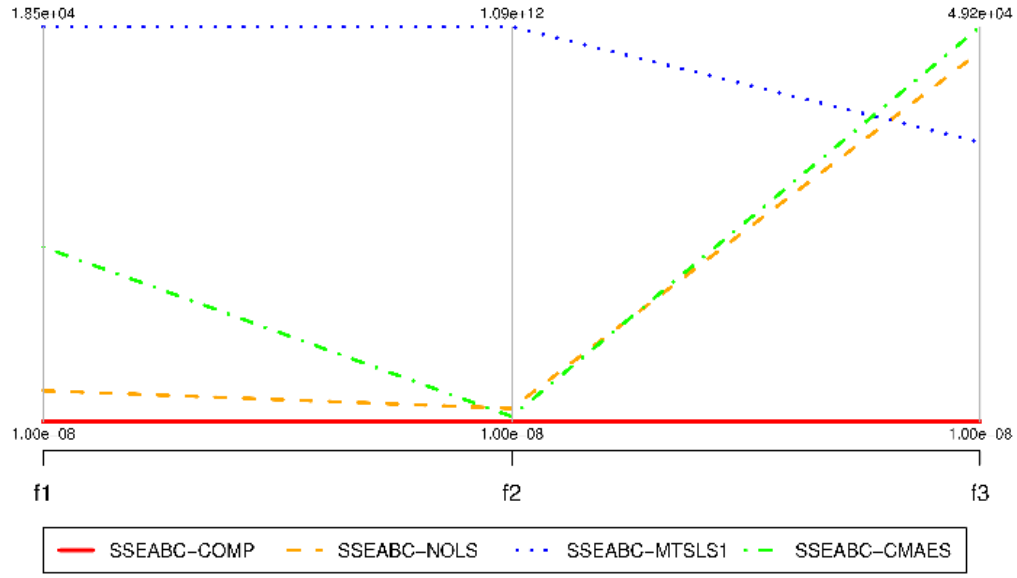
7.2.1.4. *Rekabetçi yerel aramanın SSEABC algoritması üzerindeki etkisinin analizi*

Bu alt bölümde, önerilen yerel arama ile melezleştirme yönteminin, yani rekabetçi yerel arama stratejisinin, SSEABC algoritmasının arama davranışına katkısı araştırılmıştır. Bunu yapmak için önerilen strateji, (sadece bu alt bölümde SSEABC-COMP ile gösterilen) CMAES ve MTSLS1 ile melezleştirilmiş diğer iki SSEABC

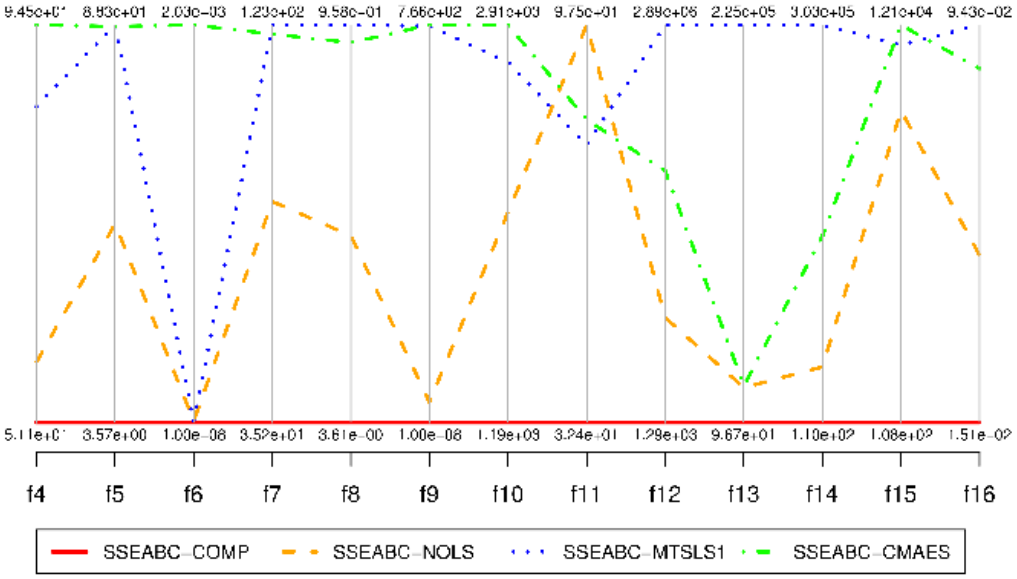
algoritması ile karşılaştırılmıştır. Bu iki algoritma, sırasıyla SSEABC-CMAES ve SSEABC-MTSLS1 olarak adlandırılmıştır. Bunlara ek olarak, yerel arama tekniklerinin etkilerini görmek için karşılaştırmaya herhangi bir yerel arama tekniğine sahip olmayan SSEABC (SSEABC-NOLS) de dâhil edilmiştir.

Karşılaştırma 30 ve 50 boyutları için gerçekleştirilmiştir. 30 ve 50 boyutlu fonksiyonlarda karşılaştırma sonuçları problem türlerine göre sırasıyla Şekil 7.2 ve Şekil 7.3'de sunulmuştur. Şekil 7.2'de gösterilen paralel koordinasyon grafikleri incelendiğinde, SSEABC-COMP'un, tek bir yerel arama içeren ve yerel arama tekniğine sahip olmayan SSEABC varyantlarına göre unimodal test fonksiyonlarında daha iyi olduğu söylenebilir. Multimodal fonksiyonlarında ise, SSEABC-MTSLS, SSEABC-CMAES ve SSEABC-NOLS algoritmaları f4, f10 ve f11 fonksiyonlarda rekabetçi yerel arama stratejisini içeren algoritmaya yakın sonuçlar vermişlerdir. Kalan fonksiyonlar f5, f7, f8, f9, f12, f13, f14, f15 ve f16'da SSEABC-COMP en düşük hata değerini elde etmiştir. Multimodal problemler dikkate alındığında, bir metasezgisel algoritmanın yerel arama olmadan kullanılması veya algoritmanın tek bir yerel arama ile melezleştirilmesinin problemlerin çözümünde iyi sonuçlar üretemediği sonucu çıkarılabilir. Hibrit fonksiyonlarda, SSEABC-COMP'un daha iyi sonuçlar elde etmiş olduğu söylenebilir. Kompozit fonksiyon problem türünde SSEABC-COMP; f23, f24, f26, f28, f29 ve f30 fonksiyonlarında diğer stratejilerden daha iyi sonuçlar üretmiştir. f25 ve f27'de sonuçların birbirine çok benzer olduğu görülmektedir. Bundan dolayı MTSLS1 ve CMAES tekniklerinin, SSEABC algoritması performansı üzerinde önemli bir etki yapmadığı ifade edilebilir.

Şekil 7.3'de verilen paralel koordinasyon grafikleri, 50 boyutlu fonksiyonlar için SSEABC üzerindeki yerel arama etkilerini göstermektedir. Unimodal fonksiyonlarda SSEABC-COMP, 30 boyutlu fonksiyonlarda olduğu gibi diğer stratejilerden daha iyi performans göstermiştir. SSEABC-COMP algoritması, hibrit fonksiyonlarda SSEABC-MTSLS1, SSEABC-CMAES ve SSEABC-NOLS'den daha düşük hata değerleri elde etmiştir. Son olarak kompozit fonksiyonları incelendiğinde, SSEABC varyantlarının tümü f25 ve f28 test fonksiyonlarında benzer sonuçlar elde etmişlerdir. Kalan fonksiyonlarda SSEABC-COMP, bütün algoritmalar arasında ilk sırada yer almıştır. Sonuç olarak, önerilen rekabetçi yerel arama stratejisi, SSEABC algoritmasının performansını problem türünden ve problem boyutundan bağımsız olarak önemli ölçüde geliştirdiği açıkça görülmektedir.

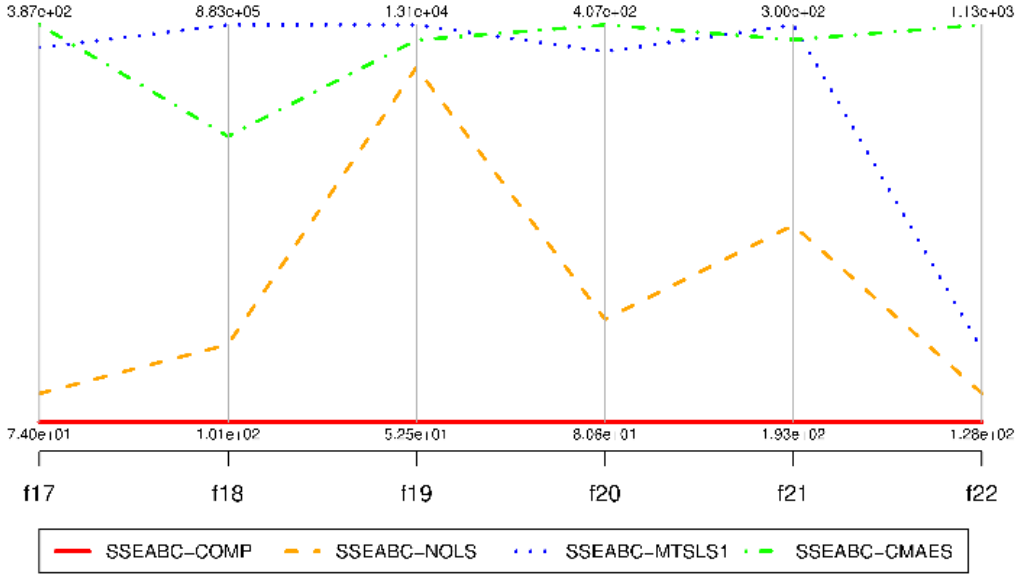


Unimodal

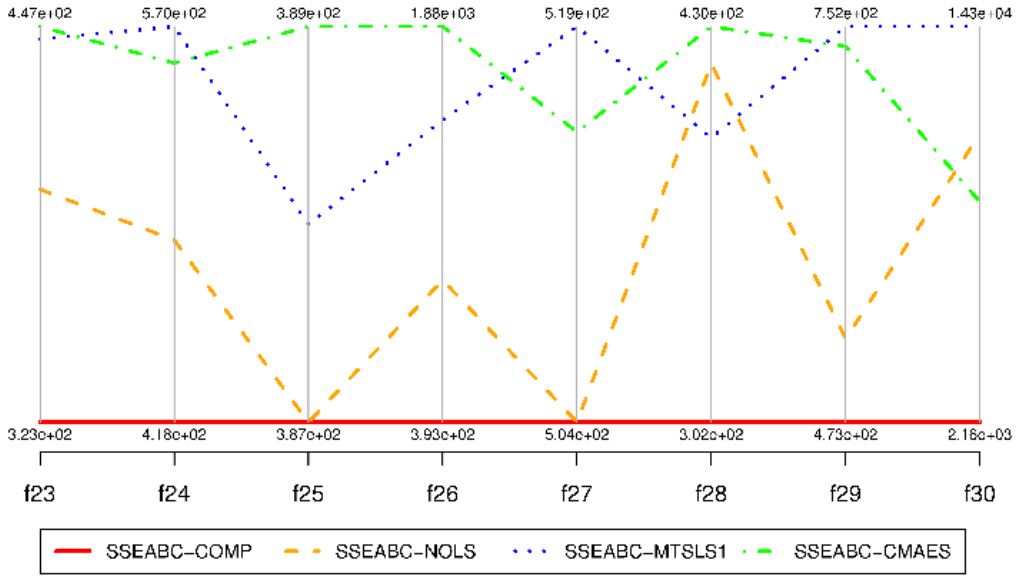


Multimodal

Şekil 7.2. *30 boyut için yerel arama yöntemleri ile melezleştirmenin katkısının analizi. Karşılaştırma problem türlerine göre gruplandırılmıştır.*

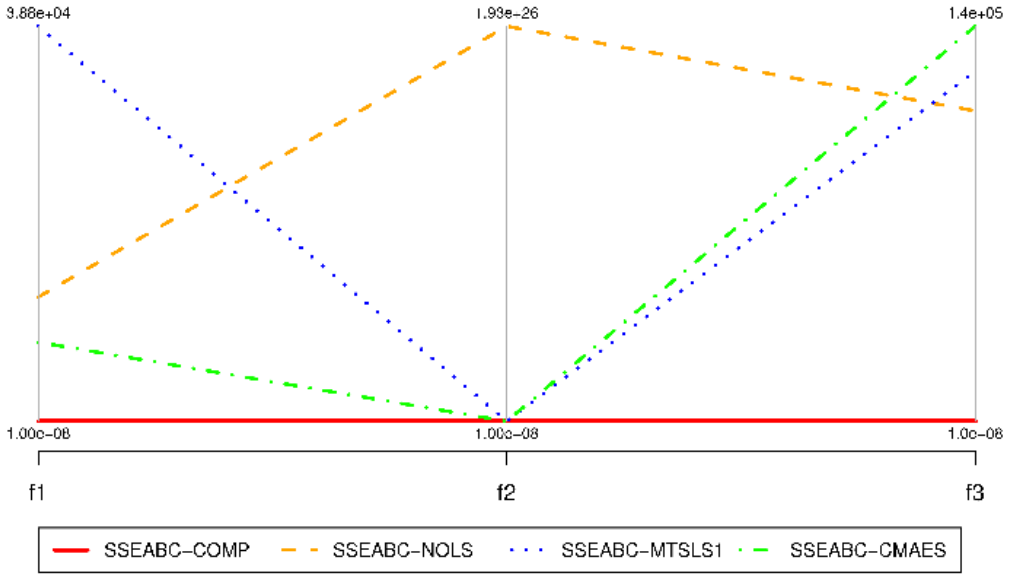


Hibrit

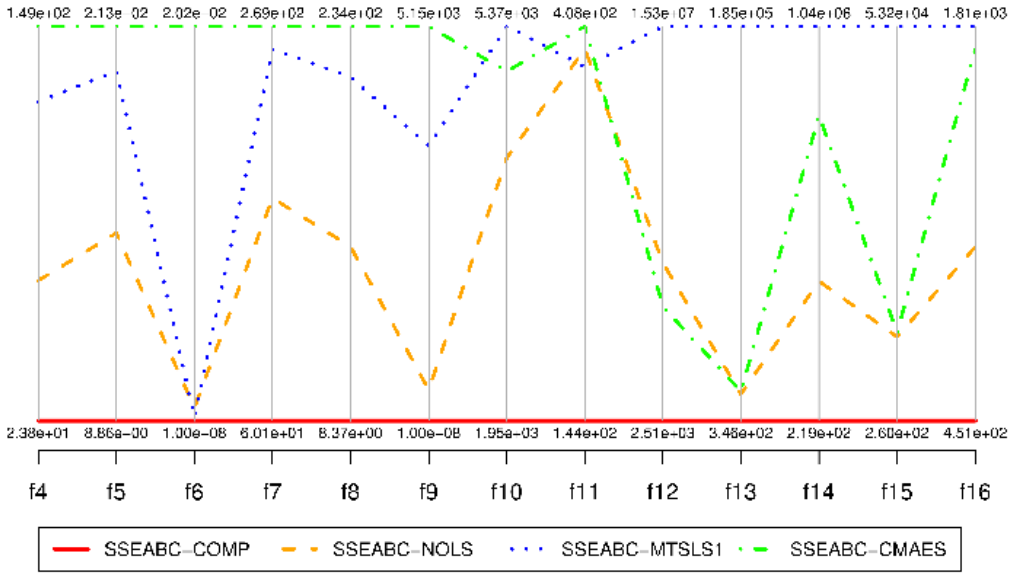


Kompozit

Şekil 7.2. (Devam) *30 boyut için yerel arama yöntemleri ile melezleştirmenin katkısının analizi. Karşılaştırma problem türlerine göre gruplandırılmıştır.*

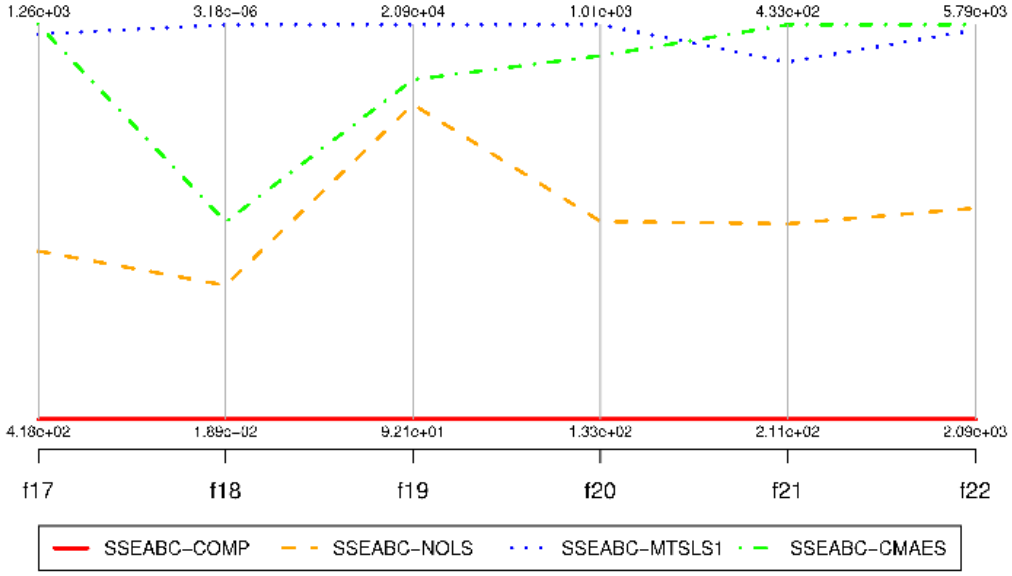


Unimodal

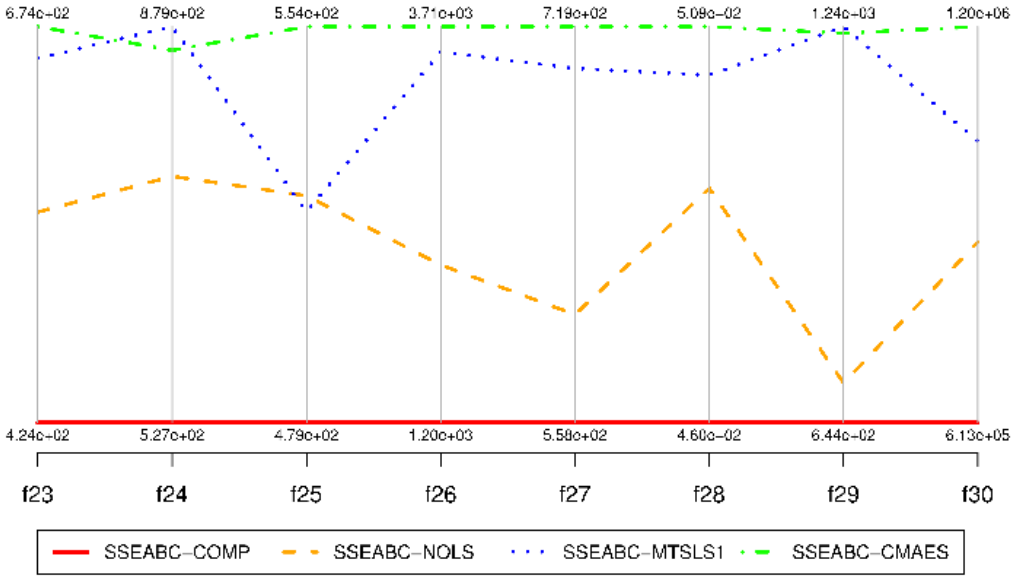


Multimodal

Şekil 7.3. *50 boyut için yerel arama yöntemleri ile melezleştirmenin katkısının analizi. Karşılaştırma problem türlerine göre gruplandırılmıştır.*



Hibrit



Kompozit

Şekil 7.3. (Devam) *50 boyut için yerel arama yöntemleri ile melezleştirmenin katkısının analizi. Karşılaştırma problem türlerine göre gruplandırılmıştır.*

7.2.1.5. *CEC'17 yarışmasına katılan algoritmalar ile karşılaştırma*

SSEABC algoritması, CEC'17 yarışmasına katılan 11 optimizasyon algoritması ile karşılaştırılmıştır. Karşılaştırmada yer alan tekniklerin sonuçları orijinal makalelerinden alınmıştır. Kullanılan bu algoritmalar jSO (Brest vd., 2017), LSHADE-cnEpsin (Awad vd., 2017), LSHADE_SPACMA (Mohamed vd., 2017), MM_OED (Sallam vd., 2017), IDEbestNsize (Bujok ve Tvrdik, 2017), RB-IPOP-CMA-ES (Biedrzycki, 2017), DES (Jagodzinski ve Arabas, 2017), DYYPO (Maharana vd., 2017), TLBO-FL (Kommadath ve Kotecha, 2017), PPSO (Tangherloni vd., 2017) ve MOS-SOCO2011 (LaTorre ve Pena, 2017). Tablo 7.8'de 30 boyut ve Tablo 7.9'da 50 boyut karşılaştırma sonuçları verilmiştir.

Tablo 7.8'deki sonuçlar incelendiğinde, önerilen algoritma (SSEABC), JSO, LSHADE_SPACMA, MM_OED ve LSHADE_cnEpSi, unimodal fonksiyonlarda optimum eşik değeri 10^{-8}'e ulaştığı görülür. Bununla birlikte, RB-IPOP-CMA-ES ve DES, f2, f3 fonksiyonlarında ve IDEbestNsize ise sadece f1 fonksiyonunda optimum değeri elde etmişlerdir.

Multimodal ölçüt fonksiyonlarındaki hata değerleri göz önüne alındığında jSO'nun en iyi algoritma olduğu görülmektedir. Bununla birlikte; SSEABC, jSO, LSHADE-cnEpsin, LSHADE_SPACMA, MM_OED ve IDEbestNsize algoritmalarının sonuçları, f6 ve f9 fonksiyonlar için optimum eşiğin altındadır. Ayrıca, LSHADE-cnEpSin f16 ve f14 fonksiyonlarında en düşük hata değerine sahip olmuştur. LSHADE_SPACMA algoritması, f4 ve f7 arasında ise ilk sırada yer almıştır. RB-IPOP-CMA-ES, f5 ve f8 fonksiyonlarında en küçük hata değerine sahip olmuştur. DES algoritması da f10 fonksiyonunda diğerlerinden daha iyi sonuçlar elde etmiştir.

Ayrıca önerilen algoritma hibrit fonksiyonlarında ilk sırayı alamamıştır (f17-f22). f22 test fonksiyonunda, jSO, LSHADE-cnEpSin, LSHADE_SPACMA, MM_OED, IDEbestNsize, DYYPO, TLBO-FL ve PPSO en iyi algoritmalar olmuşlardır. Ayrıca, f17'de LSHADE-cnEpSin ve f21'de MM_OED en iyi hata değerlerine sahip olmuşlardır. jSO ise f18, f19 ve f20 fonksiyonlarında en küçük sonuçları üretmiştir.

Hibrit ölçüt fonksiyonlarından farklı olarak, SSEABC algoritması f23, f26 ve f28 kompozit fonksiyonlarında birinci olmuştur. MM_OED, f24'de IDEbestNsize ise f27 fonksiyonunda en iyi optimizasyon algoritması olmuştur. Son olarak LSHADE_SPACMA; f25, f29 ve f30 fonksiyonlarında en iyi olan tekniktir. Genel olarak,

SSEABC'nin ortalama sıralaması 4.17'dir. Bu da SSEABC algoritmasının, 30-boyutlu ölçüt fonksiyonlarında yarışmacı algoritmalar ile rekabet ettiğini göstermektedir.

Tablo 7.9'daki sonuçlar incelendiğinde, SSEABC, jSO, LSHADE_SPACMA ve MM_OED unimodal fonksiyonlarda en düşük sonuçları üretmişlerdir. Ayrıca, LSHADE-cnEpSin, RB-IPOP-CMA-ES ve DES algoritmaları f3 fonksiyonunda en iyi değeri elde etmişlerdir. Multimodal fonksiyonlarda, SSEABC, LSHADE_SPACMA ve MOS-SOCO2011, f6'da en iyi teknikler olmuşlardır. Ayrıca SSEABC, jSO, LSHADE-cnEpSin, LSHADE_SPACMA, RB-IPOP-CMA-ES ve DES f9 için optimum eşik değerinin altında sonuç elde etmişlerdir. LSHADE _SPACMA; f4, f6, f11, f13, f15 fonksiyonlarında en düşük hata değerine sahip olmuştur. RB-IPOP-CMA-ES; f5, f7, f8, f9 ve DES ise f10, f14, f16 fonksiyonlarında ilk sırada yer almıştır. Ek olarak, LSHADE-cnEpsin, f13 test fonksiyonunda diğer algoritmalardan daha iyi sonuç elde etmiştir. Hibrit fonksiyonlarındaki ortalama hatalar analiz edildiğinde, RB-IPOP-CMA-ES, f21'de en iyi sonucu vermiştir. Öte yandan, DES f17'de optimum eşiğin altında sonuç elde etmiştir. Ayrıca, jSO algoritması f18, f19, f20 ve f22 test fonksiyonlarında birinci olmuştur. Kompozit fonksiyonlarında, LSHADE_SPACMA algoritması f25, f27, f28, f29, f30 test fonksiyonlarında ve RB-IPOP-CMA-ES algoritması da f23, f24, f26 fonksiyonlarında en küçük hata değerine ulaşmıştır. Sonuç olarak; jSO, LSHADE_SPACMA ve MM_OED algoritmaları, SSEABC'den istatistiksel olarak daha iyi sonuçlar elde etmişlerdir. Buna karşın SSEABC, 50 boyutlu CEC'17 ölçüt fonksiyonlarında karşılaştırılan diğer sekiz katılımcı algoritmadan daha üstün veya rekabetçi olduğu söylenebilir.

Tablo 7.8. *CEC'17 yarışmacılarının 30 boyut karşılaştırılması*

Fonk.	SSEABC	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbest Nsize	RB-IPOP-CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS SOCO2011
f1	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	3.154E-08	3.851E-08	3.700E+03	3.500E+03	7.480E+02	2.190E+03
f2	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	1.590E-04	**1.000E-08**	**1.000E-08**	3.200E+09	8.500E+16	5.320E+01	2.110E-02
f3	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	4.052E+00	**1.000E-08**	**1.000E-08**	5.300E+02	3.000E+03	1.130E+00	1.720E+03
f4	5.114E+01	5.867E+01	4.228E+01	**1.000E-08**	1.167E+01	2.417E+00	5.528E+01	5.856E+01	9.100E+01	9.000E+01	4.390E+01	3.890E+01
f5	3.570E+00	8.557E+00	1.225E+01	3.690E+00	4.234E+00	2.269E+01	**1.649E+00**	6.945E+00	9.100E+01	4.000E+01	1.120E+02	5.580E+01
f6	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	1.208E-07	3.826E-07	8.600E-01	4.900E-01	2.030E+01	**1.000E-08**
f7	3.516E+01	3.893E+01	4.330E+01	**3.380E+01**	3.439E+01	5.154E+01	3.433E+01	3.551E+01	1.400E+02	1.400E+02	1.350E+02	8.670E+01
f8	3.609E+00	9.092E+00	1.293E+01	3.590E+00	4.565E+00	2.363E+01	**1.756E+00**	6.575E+00	9.600E+01	3.700E+01	8.100E+01	6.360E+01
f9	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	6.500E+02	3.400E+01	1.360E+03	2.830E+02
f10	1.191E+03	1.528E+03	1.388E+03	1.440E+03	2.045E+03	2.017E+03	1.443E+03	**1.722E+02**	2.800E+03	6.700E+03	3.130E+03	2.530E+03
f11	3.236E+01	**3.038E+00**	1.354E+01	3.790E+00	1.226E+01	6.439E+00	4.107E+01	2.560E+01	1.200E+02	8.200E+01	8.430E+01	7.500E+01
f12	1.292E+03	**1.704E+02**	3.725E+02	5.540E+02	1.046E+03	3.448E+03	1.093E+03	1.280E+03	1.500E+06	5.700E+04	2.770E+04	3.730E+04
f13	9.665E+01	**1.484E+01**	1.726E+01	1.520E+01	1.795E+01	3.093E+01	1.187E+02	4.346E+04	9.600E+03	2.000E+04	3.210E+03	1.330E+04
f14	1.097E+02	2.183E+01	**2.157E+01**	2.280E+01	2.308E+01	2.334E+01	9.083E+01	2.188E+01	2.100E+03	7.100E+03	2.320E+03	1.990E+04
f15	1.083E+02	**1.088E+00**	3.240E+00	5.370E+00	5.422E+00	7.579E+00	2.177E+02	1.717E+01	1.100E+04	2.200E+04	2.130E+03	1.090E+04
f16	1.508E+02	7.892E+01	**2.288E+01**	3.770E+01	1.343E+02	1.793E+02	5.016E+02	3.415E+01	6.700E+02	4.900E+02	8.460E+02	7.320E+02
f17	7.404E+01	3.293E+01	**2.860E+01**	3.170E+01	4.668E+01	4.137E+01	1.323E+02	7.496E+01	2.500E+02	1.400E+02	3.310E+02	2.660E+02
f18	1.013E+02	**2.041E+01**	2.109E+01	2.340E+01	2.333E+01	3.215E+01	1.601E+02	2.663E+01	1.200E+05	3.700E+05	6.990E+04	1.360E+05
f19	5.252E+01	**4.503E+00**	5.828E+00	9.450E+00	7.148E+00	9.178E+00	1.146E+02	1.888E+04	1.400E+04	1.100E+04	1.710E+03	9.940E+03

Tablo 7.8. (Devam) *CEC'17 yarışmacılarının 30 boyut karşılaştırılması*

Fonk.	SSEABC	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbest Nsize	RB-IPOP-CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS SOCO2011
f20	8.060E+01	**2.937E+01**	3.035E+01	8.160E+01	4.546E+01	4.053E+01	2.965E+02	5.757E+01	2.500E+02	2.200E+02	3.480E+02	2.640E+02
f21	1.928E+02	2.093E+02	2.121E+02	2.090E+02	**1.311E+02**	2.248E+02	2.086E+02	2.091E+02	3.000E+02	2.300E+02	3.050E+02	2.620E+02
f22	1.284E+02	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	6.716E+02	1.317E+02	**1.000E+02**	**1.000E+02**	**1.000E+02**	1.400E+03
f23	**3.226E+02**	3.508E+02	3.562E+02	3.570E+02	3.574E+02	3.678E+02	3.394E+02	3.585E+02	4.500E+02	4.000E+02	6.810E+02	4.180E+02
f24	4.182E+02	4.265E+02	4.285E+02	4.290E+02	**3.939E+02**	4.371E+02	4.194E+02	4.296E+02	5.600E+02	4.700E+02	7.390E+02	5.390E+02
f25	3.865E+02	3.867E+02	3.867E+02	**3.780E+02**	3.867E+02	3.869E+02	3.867E+02	3.867E+02	3.900E+02	4.000E+02	3.850E+02	3.860E+02
f26	**3.932E+02**	9.202E+02	9.486E+02	9.430E+02	9.426E+02	1.054E+03	3.939E+02	8.328E+02	2.200E+03	1.400E+03	2.040E+03	1.710E+03
f27	5.036E+02	4.974E+02	5.042E+02	4.970E+02	5.076E+02	**4.962E+02**	5.118E+02	5.103E+02	5.400E+02	5.300E+02	7.080E+02	5.150E+02
f28	**3.021E+02**	3.087E+02	3.152E+02	3.340E+02	3.287E+02	3.166E+02	3.089E+02	3.517E+02	3.900E+02	4.300E+02	3.270E+02	3.390E+02
f29	4.732E+02	4.337E+02	4.346E+02	**3.950E+02**	4.388E+02	4.553E+02	4.933E+02	4.307E+02	7.500E+02	6.200E+02	7.800E+02	6.470E+02
f30	2.161E+03	1.971E+03	1.977E+03	**4.010E+02**	1.999E+03	2.301E+03	2.844E+03	2.414E+06	3.300E+04	2.600E+04	3.320E+03	9.910E+03
p-value		0.07800	0.09263	0.04501	0.03243	0.96168	0.00152	0.50114	0.00000	0.00000	0.00001	0.00001
rank	4.17	2.97	3.33	**2.87**	3.67	5.50	5.63	5.93	10.50	9.93	9.63	9.27
SSEABC-win		11	18	8	8	15	22	18	29	29	27	27
SSEABC-lost		14	6	17	17	12	5	9	1	1	3	2
draw		5	6	5	5	3	3	3	0	0	0	1

Tablo 7.9. *CEC'17 yarışmacılarının 50 boyut karşılaştırılması*

Fonk.	SSEABC	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbest Nsize	RB-IPOP-CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS-SOCO2011
f1	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	2.630E-02	1.131E-07	1.253E-07	6.600E+03	6.100E+05	3.890E+02	1.490E+04
f2	**1.000E-08**	**1.000E-08**	1.569E+00	**1.000E-08**	**1.000E-08**	9.560E-04	2.768E+05	2.353E-01	5.700E+15	1.600E+39	1.250E+12	1.810E+03
f3	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	2.205E+02	**1.000E-08**	**1.000E-08**	4.700E+01	2.600E+04	8.650E+02	3.360E+04
f4	2.376E+01	2.851E+01	5.140E+01	**5.680E-01**	3.749E+01	4.107E+01	2.959E+01	6.273E+01	1.400E+02	1.900E+02	9.130E+01	2.810E+01
f5	8.857E+00	1.620E+01	2.517E+01	6.220E+00	1.116E+01	5.903E+01	**2.790E+00**	8.701E+00	1.900E+02	9.700E+01	2.010E+02	1.340E+02
f6	**1.000E-08**	3.107E-07	9.157E-07	**1.000E-08**	7.435E-08	9.610E-08	1.632E-07	5.717E-07	3.800E+00	4.500E+00	3.180E+01	**1.000E-08**
f7	6.005E+01	6.664E+01	7.664E+01	5.710E+01	5.885E+01	1.021E+02	**5.663E+01**	5.842E+01	2.600E+02	1.700E+02	2.780E+02	1.810E+02
f8	8.369E+00	1.697E+01	2.632E+01	5.850E+00	1.032E+01	6.008E+01	**2.575E+00**	9.086E+00	1.900E+02	9.300E+01	1.990E+02	1.380E+02
f9	**1.000E-08**	**1.000E-08**	**1.000E-08**	**1.000E-08**	1.756E-03	7.776E-02	**1.000E-08**	**1.000E-08**	3.500E+03	1.300E+03	6.060E+03	1.210E+03
f10	1.947E+03	3.232E+03	3.200E+03	3.610E+03	3.824E+03	4.216E+03	1.726E+03	**7.936E+02**	4.800E+03	1.300E+04	5.200E+03	4.730E+03
f11	1.444E+02	2.852E+01	2.139E+01	**9.170E+00**	4.039E+01	3.774E+01	1.832E+02	3.009E+01	1.900E+02	1.700E+02	1.270E+02	1.950E+02
f12	2.513E+03	1.653E+03	**1.475E+03**	1.580E+03	2.139E+03	1.576E+04	2.438E+06	1.992E+03	7.800E+06	9.200E+05	5.520E+05	1.450E+05
f13	3.455E+02	3.726E+01	6.943E+01	**3.530E+01**	4.639E+01	1.622E+02	1.652E+03	1.553E+05	7.600E+03	8.000E+03	8.470E+02	1.090E+04
f14	2.190E+02	2.435E+01	2.652E+01	2.890E+01	3.708E+01	6.272E+01	2.416E+02	2.278E+01	2.900E+04	8.600E+04	1.950E+04	3.780E+04
f15	2.603E+02	2.342E+01	2.560E+01	**1.570E+01**	4.086E+01	4.102E+01	5.290E+02	1.244E+06	8.200E+03	6.900E+03	1.190E+03	1.220E+04

Tablo 7.9. (Devam) *CEC'17 yarışmacılarının 50 boyut karşılaştırılması*

Fonk.	SSEABC	jSO	LSHADE-cnEpSin	LSHADE SPACMA	MM_OED	IDEbest Nsize	RB-IPOP-CMA-ES	DES	DYYPO	TLBO-FL	PPSO	MOS SOCO2011
f16	4.514E+02	4.773E+02	2.745E+02	4.090E+02	6.774E+02	6.322E+02	8.899E+02	**1.059E+02**	1.300E+03	8.500E+02	1.240E+03	1.400E+03
f17	4.178E+02	2.599E+02	2.071E+02	2.940E+02	4.806E+02	4.625E+02	3.978E+02	**9.648E+01**	8.900E+02	8.500E+02	1.030E+03	9.490E+02
f18	1.889E+02	**2.387E+01**	2.433E+01	3.320E+01	3.859E+01	2.035E+03	3.574E+02	2.621E+01	1.800E+05	1.100E+06	2.090E+05	1.880E+05
f19	9.209E+01	**1.386E+01**	1.741E+01	2.180E+01	4.118E+01	2.331E+01	1.394E+02	1.361E+06	9.600E+03	1.400E+04	8.670E+03	2.420E+04
f20	1.328E+02	**1.133E+02**	1.141E+02	1.630E+02	3.020E+02	2.179E+02	5.468E+02	1.312E+02	6.500E+02	1.000E+03	7.700E+02	7.980E+02
f21	2.106E+02	2.192E+02	2.268E+02	2.150E+02	2.122E+02	2.548E+02	2.062E+02	2.104E+02	4.000E+02	2.800E+02	4.330E+02	3.460E+02
f22	2.092E+03	**1.000E+02**	1.595E+03	8.170E+02	6.803E+02	2.837E+03	2.051E+03	5.511E+02	4.800E+03	6.600E+03	5.970E+03	5.070E+03
f23	4.243E+02	4.298E+02	4.393E+02	4.390E+02	4.449E+02	4.787E+02	**4.227E+02**	4.261E+02	6.500E+02	5.700E+02	1.060E+03	5.990E+02
f24	5.267E+02	5.073E+02	5.128E+02	5.130E+02	5.166E+02	5.409E+02	**4.912E+02**	5.090E+02	7.300E+02	6.700E+02	1.080E+03	8.220E+02
f25	4.789E+02	4.802E+02	4.803E+02	**4.630E+02**	4.819E+02	5.407E+02	4.807E+02	4.803E+02	5.200E+02	6.200E+02	5.410E+02	5.120E+02
f26	1.204E+03	1.133E+03	1.203E+03	1.140E+03	1.244E+03	1.651E+03	**6.548E+02**	1.080E+03	3.400E+03	2.900E+03	5.450E+03	2.810E+03
f27	5.577E+02	5.125E+02	5.254E+02	**5.000E+02**	5.401E+02	5.258E+02	6.081E+02	5.354E+02	6.700E+02	8.700E+02	1.460E+03	6.760E+02
f28	4.598E+02	4.589E+02	4.591E+02	**4.400E+02**	4.818E+02	4.870E+02	4.700E+02	4.588E+02	4.800E+02	6.100E+02	4.890E+02	4.860E+02
f29	6.436E+02	3.636E+02	3.529E+02	**2.750E+02**	3.617E+02	4.111E+02	6.690E+02	3.705E+02	9.800E+02	1.000E+03	1.520E+03	9.650E+02
f30	6.128E+05	5.905E+05	6.575E+05	**6.500E+02**	6.635E+05	6.198E+05	6.460E+06	2.065E+06	1.500E+06	1.200E+06	7.810E+05	8.220E+05
p-value		0.01425	0.10232	0.00143	0.77312	0.03160	0.04508	0.25488	0.00000	0.00000	0.00000	0.00000
rank	4.33	3.03	4.03	**2.47**	5.10	6.90	5.43	4.50	10.03	10.40	10.43	9.50
SSEABC-win		10	21	4	15	23	18	11	30	30	29	29
SSEABC-lost		16	6	21	12	7	10	17	0	0	1	0
draw		4	3	5	3	0	2	2	0	0	0	1

7.2.2. Soft computing büyük ölçekli ölçüt seti deneyleri

7.2.2.1. *Parametre ayarları*

Büyük ölçekli optimizasyonda SSEABC algoritmasının performansını ölçmek için, Soft Computing (SOCO) dergisinin özel sayısının ("Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems") ölçüt seti kullanılmıştır (Lozano vd., 2010). SOCO, 19 adet fonksiyondan oluşmaktadır. Bunlardan yedi tanesi unimodal, on iki tanesi ise multimodal'dir. Aynı zamanda, bu fonksiyonların dört fonksiyonu ayrılabilir (separable), geri kalanları ise ayrılamazdır (nonseparable). Tablo 5.13'de SOCO ölçüt fonksiyonlarının özellikleri sunulmuştur.

Deneylerde, SSEABC ile karşılaştırılan tüm ABC algoritmaları her fonksiyon için bağımsız olarak 25 defa çalıştırılmıştır. Maksimum fonksiyon çalıştırma sayısı, $D\times5000$ olarak alınır; burada D, bir fonksiyonun boyutunu göstermektedir. Algoritma, sonlandırma kriterini yerine getirdiğinde hata değeri kaydedilir. Hata değeri $f(x)-f(x^*)$ olarak hesaplanır, burada x algoritmanın bulduğu en iyi aday çözüm ve x^* optimum çözümdür. 10^{-14}'den küçük test sonuçları 10^{-14} olarak kabul edilmiştir. Tüm ABC algoritmalar, orijinal makalelerinde belirtilen "varsayılan parametreler (default parameters)" ile çalıştırılmıştır. SSEABC ve ABC algoritmalarının varsayılan parametre değerleri Tablo 7.10'da listelenmiştir. Ayrıca, SSEABC ile karşılaştırılan algoritmaların arasındaki performansın önemini değerlendirmek ve istatistiksel olarak sağlam sonuçlara varmak için 0.05 anlamlılık düzeyinde iki taraflı Wilcoxon sıra toplamı testi kullanılmıştır. Tüm deneyler 8 GB RAM ve Intel Xeon E5-2620 2 GHz işlemciye sahip bir bilgisayarda yapılmıştır.

Tablo 7.10. *Deneylerde kullanılan parametreler*

Algoritmalar	Parametreler
SSEABC	nfoods = 10 limitf = 1 asize = 1000
ABC	nfoods = 62 limitf = 1
BABC	nfoods = 100 limitf = 0.1 wmin = 0.2 wmax = 1
MABC	nfoods = 10 limitf = 1 MR = 0.4 SF = 1
ImpABC	nfoods = 25 limitf = 1 MR = 1 p = 0.25
OPIABC	nfoods = 62 limitf = 1

7.2.2.2. *SSEABC'nin ABC varyantları ile karşılaştırılması*

SSEABC'nin 1000 boyutlu SOCO fonksiyonlarındaki performansı beş ABC varyantı ile karşılaştırılmıştır. Bu ABC algoritmaları Orijinal ABC (Artificial Bee Colony, ABC) (Karaboga ve Basturk, 2007), Değiştirilmiş ABC (Modified ABC, MABC) (Akay ve Karaboga, 2012), Gelişmiş ABC (Improved ABC, ImpABC) (Gao ve Liu, 2011), Şimdiye Kadarki En İyi Seçim ABC (Best-so-far selection ABC, BABC) (Banharnsakun vd., 2011) ve Tek-Nokta Kalıtım ABC (One-Point Inheritance ABC, OPIABC) (Zhang ve Yuen, 2013).

1000 boyut SOCO fonksiyonlarındaki medyan karşılaştırma sonuçları, Tablo 7.11'de sunulmuştur. Tablo 7.11 incelendiğinde SSEABC, 17 fonksiyonda ABC, BABC ve ImpABC'yi, 19 fonksiyonda MABC'yi ve 9 fonksiyonda ise OPIABC'yi geçmiştir. Sonuç olarak; SSEABC, ilk sırada yer almış ve OPIABC hariç tüm ABC varyantlarından daha iyi bir performans göstermiştir.

Tablo 7.11. *SSEABC algoritmasının SOCO ölçüt setinin 1000 boyutta çalıştırılması ile elde edilen medyan değerleri ve ABC varyantları ile karşılaştırılması. Tablonun "Win (Kazanma)", "Lost (Kaybetme)" ve "Draw (Berabere)" kısmı, SSEABC'nin diğer algoritmadan daha iyi, aynı veya daha kötü olduğunu göstermektedir.*

Fonk.	SSEABC	ABC	BABC	MABC	ImpABC	OPIABC
f1	**1.000E-14**	2.111E-06	2.411E+04	1.477E+00	1.545E+04	**1.000E-14**
f2	8.576E+01	1.565E+02	**4.965E+01**	1.411E+02	4.986E+01	1.417E+02
f3	1.026E+03	7.671E+02	1.667E+09	9.041E+05	1.730E+09	**2.414E+00**
f4	**1.000E-14**	1.481E+02	2.933E+03	4.121E+03	8.460E+03	**1.000E-14**
f5	**1.000E-14**	6.179E-07	2.121E+02	3.020E-01	1.197E+02	**1.000E-14**
f6	**1.000E-14**	2.006E-02	1.450E+01	1.982E+01	1.944E+01	9.499E-13
f7	**1.000E-14**	9.965E-04	1.375E+01	2.018E-02	4.253E+02	4.519E-14
f8	1.071E+06	2.514E+06	8.178E+05	6.756E+06	**4.785E+05**	2.297E+06
f9	**1.000E-14**	3.228E+02	4.351E+03	7.223E+03	6.388E+03	2.856E-02
f10	**1.000E-14**	7.370E-06	4.840E+02	6.399E+01	1.130E+03	**1.000E-14**
f11	**4.334E-07**	3.240E+02	4.195E+03	7.097E+03	6.489E+03	2.938E-02
f12	**1.308E-11**	4.967E+01	1.705E+04	2.146E+03	1.319E+04	3.769E-05
f13	8.083E+02	6.742E+02	1.159E+09	6.139E+03	5.692E+08	**6.150E+00**
f14	9.950E-01	1.263E+02	2.517E+03	3.262E+03	6.422E+03	**3.993E-06**
f15	**1.000E-14**	7.010E-04	4.725E+01	1.267E+01	1.137E+03	**1.000E-14**
f16	**7.462E-08**	1.170E+02	9.658E+03	4.174E+03	7.461E+03	2.826E-04
f17	2.135E+02	3.152E+02	8.625E+07	7.413E+03	7.105E+07	**2.219E-01**
f18	**8.937E-08**	8.753E+01	1.242E+03	1.944E+03	2.657E+03	1.516E-04
f19	**1.000E-14**	2.169E-04	9.801E+01	4.418E+01	8.553E+02	**1.000E-14**
rank	1	3	5	4	6	2
win		17	17	19	17	9
loss		2	2	0	2	4
draw		0	0	0	0	6
P-value		0.0088	0.00222	0.00014	0.00194	0.75656

7.2.2.3. *SSEABC'nin SOCO yarışmacıları ile karşılaştırılması*

SSEABC, SOCO özel sayısındaki 15 katılımcı algoritmaları ile de karşılaştırılmıştır. Karşılaştırmada kullanılan algoritmalar: CHC (Eshelman, 1991), DE (Storn ve Price, 1997), DEDM (García-Martínez vd., 2011), EM323 (Gardeux vd., 2011), EvoPROpt (Duarte vd., 2011), GaDE (Yang vd., 2011), GODE (Wang vd., 2011), jDElscop (Brest ve Maučec, 2011), MASSW (Molina vd., 2011), MOS (LaTorre, Muelas ve Pena, 2011), RPSOmv (Garcia-Nieto ve Alba, 2011), SaDEMMTS (Zhao vd., 2011), SOUPDE (Weber vd., 2011), IPSOLS (Montes de Oca vd., 2011), VXQR (Neumaier vd., 2011).

Bu algoritmaların medyan sonuçları orijinal makalelerinden alınmıştır. Karşılaştırma Tablo 7.12'de verilmiştir. Tablo 7.12'de görüldüğü gibi SSEABC algoritması; CHC, EvoPROpt, JDElscop ve MASSW'den önemli ölçüde daha iyi sonuçlar elde etmiştir. Sadece yarışmanın galibi olan MOS algoritması, SSEABC'yi geride bırakmıştır. Bu karşılaştırma sonuçlarının ışığında, SSEABC'nin SOCO ölçüt fonksiyonlarında çok rekabetçi sonuçlar elde ettiği açıkça söylenebilir.

Tablo 7.12. *SSEABC algoritmasının SOCO ölçüt setinin 1000 boyutta çalıştırılması ile elde edilen medyan değerleri ve SOCO yarışmacıları ile karşılaştırılması. Tablonun "Win (Kazanma)", "Lost (Kaybetme)" ve "Draw (Berabere)" kısmı SSEABC'nin diğer algoritmadan daha iyi, aynı veya daha kötü olduğunu göstermektedir.*

Algoritmalar	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12
SSEABC	**1.000E-14**	8.576E+01	1.026E+03	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.071E+06	**1.000E-14**	**1.000E-14**	4.334E-07	1.308E-11
CHC	1.296E-11	1.442E+02	1.493E+03	4.731E+03	1.705E-13	1.399E+01	1.876E-03	3.134E+05	6.109E+03	3.166E+02	4.832E+03	1.041E+03
DE	**1.000E-14**	8.444E+01	9.685E+02	1.315E+00	**1.000E-14**	3.297E-12	**1.000E-14**	2.459E+05	5.130E+03	**1.000E-14**	1.351E-03	1.698E-08
DEDM	**1.000E-14**	7.056E+01	9.465E+02	9.950E-01	**1.000E-14**	1.535E-12	**1.000E-14**	9.713E+10	**1.000E-14**	**1.000E-14**	**1.000E-14**	4.601E-12
EM323	1.114E-11	2.294E+01	4.906E+02	1.359E-11	6.651E-12	1.299E-11	1.000E+200	1.541E+06	2.546E-08	**1.000E-14**	3.256E-07	2.206E-06
EvoPROpt	**1.000E-14**	3.163E+01	1.087E+03	1.214E+03	1.972E-02	2.447E+00	**1.000E-14**	2.057E+05	3.656E+02	4.245E+02	3.503E+02	3.056E+01
GaDE	**1.000E-14**	8.920E+01	9.450E+01	**1.000E-14**	**1.000E-14**	1.420E-14	**1.000E-14**	1.730E+04	**1.000E-14**	**1.000E-14**	**1.000E-14**	3.580E-12
GODE	**1.000E-14**	8.910E+01	9.700E+02	1.080E+00	**1.000E-14**	2.880E-13	1.000E+200	1.891E+05	1.707E-04	**1.000E-14**	1.718E-04	1.935E-09
jDElscop	**1.000E-14**	6.210E+01	8.440E+02	**1.000E-14**	**1.000E-14**	2.610E-12	**1.000E-14**	3.150E+04	5.970E-08	**1.000E-14**	1.200E-07	**1.000E-14**
MASSW	**1.000E-14**	1.391E+02	1.212E+03	1.611E+03	**1.000E-14**	1.451E-09	6.173E-13	7.384E+04	5.988E+03	**1.000E-14**	5.270E+01	9.127E-02
MOS	**1.000E-14**	2.917E-01	5.790E+01	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.896E+05	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**
RPSOmv	2.820E-14	4.290E+01	1.370E+02	2.780E-14	1.120E-01	3.750E-12	**1.000E-14**	9.210E+05	9.840E+00	**1.000E-14**	9.610E+00	2.020E-12
SaDEMMTS	**1.000E-14**	4.880E+01	**1.000E-14**	3.210E+01	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.460E+03	9.250E+01	**1.000E-14**	1.850E+02	**1.000E-14**
SOUPDE	**1.000E-14**	9.250E+01	9.620E+02	2.000E-11	**1.000E-14**	3.420E-13	3.570E-13	2.120E+05	7.390E-05	**1.000E-14**	7.440E-05	**1.000E-14**
IPSOLS	**1.000E-14**	**6.679E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	7.134E-11	1.197E+05	9.760E+00	**1.000E-14**	9.750E+00	1.374E+00
VXQR	**1.000E-14**	9.640E+01	6.050E+02	5.684E-14	1.086E-11	1.668E-11	**1.000E-14**	**1.000E-14**	9.994E+01	**1.000E-14**	1.116E+02	2.243E+02

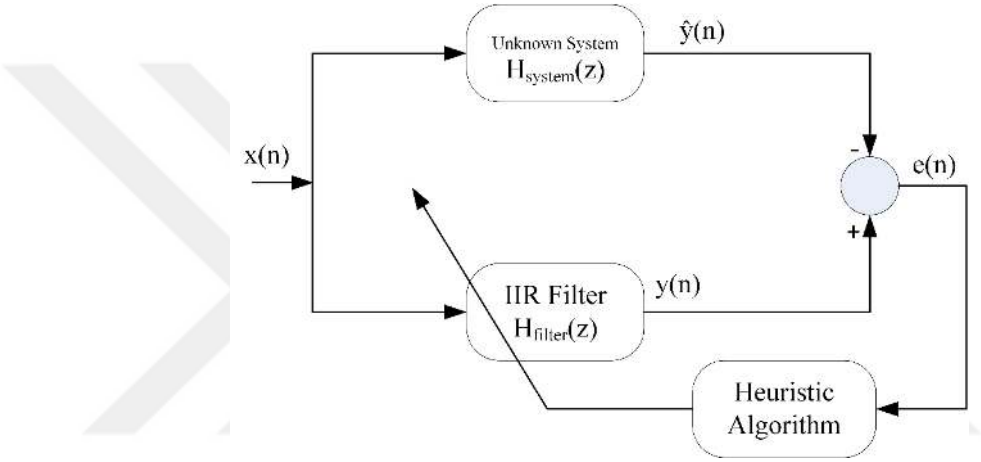
Tablo 7.12. (Devam) *SSEABC algoritmasının SOCO ölçüt setinin 1000 boyutta çalıştırılması ile elde edilen medyan değerleri ve SOCO yarışmacıları ile karşılaştırılması. Tablonun "Win (Kazanma)", "Lost (Kaybetme)" ve "Draw (Berabere)" kısmı SSEABC'nin diğer algoritmadan daha iyi, aynı veya daha kötü olduğunu göstermektedir.*

Algoritmalar	f13	f14	f15	f16	f17	f18	f19	rank	win	loss	draw	P-value
SSEABC	8.083E+02	9.950E-01	**1.000E-14**	7.462E-08	2.135E+02	8.937E-08	**1.000E-14**	6				
CHC	1.803E+03	3.648E+03	7.761E+01	2.299E+03	3.813E+03	1.710E+03	3.446E+03	16	0	0	0	0.0022
DE	7.294E+02	9.950E-01	**1.000E-14**	4.193E-08	2.354E+02	2.367E-03	**1.000E-14**	10	0	0	0	0.8729
DEDM	7.167E+02	8.794E-10	**1.000E-14**	2.259E-11	2.240E+02	3.213E-07	**1.000E-14**	5	0	0	0	0.5823
EM323	1.188E+03	2.182E-06	1.000E+200	5.102E-06	3.185E+02	5.719E-06	**1.000E-14**	13	0	0	0	0.093
EvoPROpt	9.990E+02	5.474E+02	8.119E+01	1.001E+02	7.859E+02	1.754E+02	1.736E+02	15	0	0	0	0.0085
GaDE	7.090E+02	8.240E-11	**1.000E-14**	2.320E-12	2.180E+02	1.180E-07	**1.000E-14**	3	0	0	0	0.215
GODE	7.314E+02	9.950E-01	1.000E+200	4.643E-09	2.358E+02	1.136E-05	**1.000E-14**	11	0	0	0	0.2983
jDElscop	6.410E+02	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.670E+02	2.840E-12	**1.000E-14**	2	0	0	0	0.006
MASSW	9.974E+02	7.895E+02	1.154E-13	7.378E-01	3.157E+02	1.056E+01	4.526E-13	14	0	0	0	0.0071
MOS	7.245E+01	**1.000E-14**	**1.000E-14**	**1.000E-14**	**3.258E+00**	**1.000E-14**	**1.000E-14**	1	0	0	0	0.0051
RPSOmv	2.270E+02	4.070E-08	**1.000E-14**	2.190E-06	4.020E+01	1.330E+00	**1.000E-14**	9	0	0	0	0.3628
SaDEMMTS	6.550E+02	1.400E+02	**1.000E-14**	**1.000E-14**	1.900E+02	8.200E+01	**1.000E-14**	4	0	0	0	0.7566
SOUPDE	7.260E+02	6.370E-12	2.690E-13	**1.000E-14**	2.310E+02	1.360E-12	9.500E-14	7	0	0	0	0.6818
IPSOLS	**1.280E+00**	1.778E+01	1.391E-10	2.408E+00	6.494E+00	2.462E+01	1.009E-10	8	0	0	0	0.6384
VXQR	5.388E+02	5.638E+00	2.695E-02	6.346E+01	2.974E+02	1.572E+01	1.373E-03	12	0	0	0	0.234

7.2.3. Filtre tabanlı sistem tanımlama problemi çözümü

7.2.3.1. *Problem tanımı*

Filtre tabanlı sistem tanımlama probleminin temel amacı, bilinmeyen bir sistemin parametrelerini bir filtreyle tahmin etmektir. Diğer bir ifadeyle; bilinmeyen sistemin transfer fonksiyonunun en uygun filtre katsayılarını belirlemek için filtrenin transfer fonksiyonu tarafından izlenir. Bu işlem, bilinmeyen sistem ile filtreye aynı giriş sinyalinin uygulanmasıyla ortaya çıkan hataların en aza indirilmeye çalışılması bir optimizasyon problemine dönüşür. Filtre tabanlı sistem tanımlama Şekil 7.4 gösterilmektedir.



Şekil 7.4. *IIR filtre tabanlı sistem tanımlama uygulamasının şematik diyagramı*

Genel olarak, bir IIR filtresi aşağıdaki denklem ile temsil edilir:

$$y(n)+\sum_{i=1}^{O} a_i y(n-i)=\sum_{i=0}^{T} b_i x(n-i) \tag{7.4}$$

Denklemdeki O payın derecesini, T ise paydanın derecesini gösterir. Bununla birlikte, $x(n)$ ve $y(n)$, filtrenin giriş ve çıkış değeridir. a_i ve b_i, sırasıyla i derecesindeki filtre çıkış ve giriş katsayılarıdır. IIR filtresinin transfer fonksiyonu şöyle ifade edilir:

$$H_{filter}(Z)=\frac{Y(Z)}{X(Z)}=\frac{\sum_{i=0}^{T} b_i z^{-i}}{1+\sum_{i=1}^{O} a_i z^{-i}} \tag{7.5}$$

IIR filtre temelli tanımlama modelindeki amaç, bilinmeyen sistemin (H_{system}(z)) transfer fonksiyonunu, filtrenin transfer fonksiyonunun (H_{filter}(z)) değerine yaklaştırmaktır. Bu nedenle, bilinmeyen sistemin transfer fonksiyonundan elde edilen çıkışı ile IIR filtre çıkışı arasındaki hata farkı hesaplanır. Ortaya çıkan bu hatayı en aza indirmek için, en iyi çözüm vektörü bulunmaya çalışılır. Bunu yapmak için, amaç fonksiyonu, çıkış değerlerinin ortalama kare hatası (Mean Squared Error, MSE) olarak tanımlanan aşağıdaki formül ile hesaplanır:

$$MSE = \frac{1}{S}\sum_{n=1}^{S} e^2(n) = \frac{1}{S}\sum_{n=1}^{S}\left[y(n) - \hat{y}(n)\right]^2 \tag{7.6}$$

Denklemdeki *S,* giriş örneklerinin toplam sayısıdır. *y(n)* ve $\hat{y}(n)$, sırasıyla n örnek girişi için IIR filtrenin ile bilinmeyen sistemlerin çıkışıdır.

7.2.3.2. *Parametre ayarları*

SSEABC'nin uyarlanabilir IIR filtre tasarımı konusundaki performansını değerlendirmek için, birçok çalışmada yaygın olarak kullanılan üç ölçüt sistem problemi seçilmiştir (Durmuş ve Gün, 2011; Karaboga, 2005, 2009; Kumar vd., 2017; Kumar ve Rawat, 2015; Upadhyay vd., 2014). İlk iki problemde, bilinmeyen sistem ve filtre modeli aynı derecedendir. Kalan problemde ise iki durum dikkate alınmıştır. Sistem tanımlama aynı dereceden ve düşük dereceden IIR filtre modelleri ile gerçekleştirilmiştir.

C++ ve i7 8 GB RAM donanıma sahip olan bir bilgisayarda deneysel çalışmalar yapılmıştır. SSEABC, ABC ve MABC için sonuçlar, her bir örnek için 7500 fonksiyon çağrım sayısı (MAXFEs) kadar bütçeye sahip 100 bağımsız çalışma ile elde edilmiştir. Her bir algoritma çalışması için bilinmeyen sisteme ve IIR filtresine giriş sinyali olarak 100 örnek Gaussian beyaz gürültü (white noise) sinyali uygulanmıştır.

Probleme özgü arama denklemleri, SSEABC algoritmasında yer alan "kendinden uyarlamalı arama denklemi belirleme stratejisi" ile belirlenir. Bununla birlikte, SSEABC'nin performansını önemli ölçüde etkileyen başka parametreleri de vardır. Bu parametrelerin değerleri Bölüm 4.2.1'de anlatılmış olan otomatik parametre ayarlama aracı "irace" vasıtasıyla belirlenmiştir (López-Ibáñez vd., 2016). Buna ek olarak, irace aracı da çeşitli parametrelere sahiptir. İrace, bu parametrelerin varsayılan değerleri kullanılarak çalıştırılmıştır. Parametre ayarlama işleminde kullanılan problem örneklerinin, deneylerde kullanılan problem örneklerinden farklı olması gerekmektedir.

Bunun için IIR sistem tanımlama probleminin diğer örnekleri ile bu tez için üretilen sentetik olarak üretilen problem örnekleri kullanılmıştır.

SSEABC ve karşılaştırmada kullanılan diğer ABC-temelli algoritmaların parametre ayarlaması için irace kullanılmıştır. Deneyde yer alan algoritmaların en iyi parametre değerleri irace aracının beş bağımsız çalışması sonucunda belirlenmiştir. Orijinal ABC, MABC ve SSEABC için elde edilen parametre değerleri Tablo 7.13'de verilmiştir.

Tablo 7.13. *Deneyde yer alan ABC algoritmaları için ayarlanmış parametreler ve değerleri*

Parametreler	ABC	MABC	SSEABC
Başlangıç popülasyon boyutu (SN)	14	8	38
Limit Faktörü (limitF)	2.0456	2637	1.9074
Değişim oranı (MR)	-	0.8512	-
adaptiveSF kullanıldı mı?	Hayır	Evet	Hayır
Ölçeklendirme faktörü (SF)	-	0.7277	-
Maksimum popülasyon boyutu (SN_{max})	-	-	55
Büyüme periyodu (g)	-	-	5
CMAES'ye ait bir kontrol parametresi (a)	-	-	1.1127
CMAES'ye ait bir kontrol parametresi (b)	-	-	2.4823
CMAES'ye ait bir kontrol parametresi (c)	-	-	0.5671
CMAES'ye ait bir kontrol parametresi (d)	-	-	3.3964
CMAES'ye ait bir kontrol parametresi (e)	-	-	-17.8882
CMAES'ye ait bir kontrol parametresi (f)	-	-	-17.9765
CMAES'ye ait bir kontrol parametresi (g)	-	-	-19.2638
MTSLS1 iterasyon sayısı ($MTSLS_{itr}$)	-	-	22
CMAES için fonksiyon çağrım sayısı ($CMAES_{FES}$)	-	-	0.3
Rekabet aşaması için fonksiyon çağrım sayısı (CompBudget)	-	-	0.15
Denklem havuzunun boyutu (ps)	-		2000

7.2.3.3. *Örnek I sonuçları*

Bilinmeyen sistemin transfer fonksiyonları ve (Karaboga, 2005)'den alınan filtre modeli farklı çalışmalarda da kullanılmıştır (Durmuş ve Gün, 2011; Karaboga, 2009; Kumar vd., 2017; Upadhyay vd., 2014). Transfer fonksiyonları aşağıdaki denklemlerde görülmektedir:

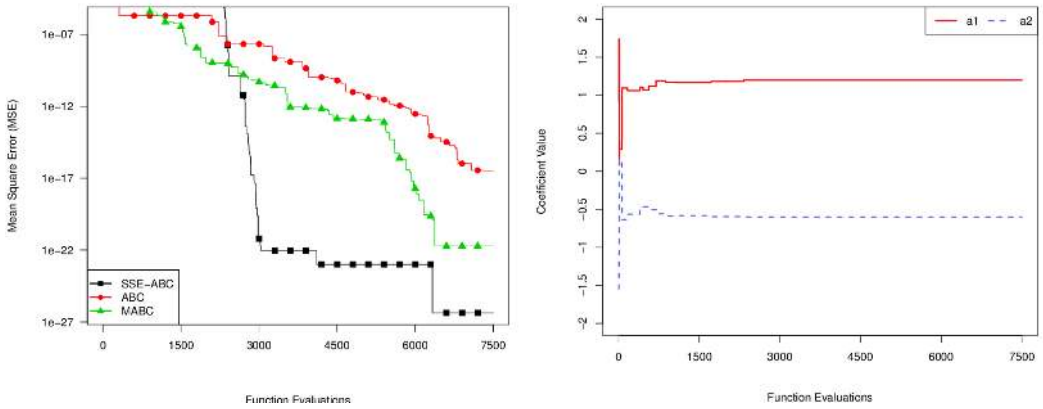
$$H_{system}(z) = \frac{1}{1 - 1.2z^{-1} + 0.6z^{-2}} \tag{7.7}$$

$$H_{filter}(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{7.8}$$

Bu problemin, global optimum değeri $a_1 = 1.2$ ve $a_2 = -0.6$'dır. SSEABC tarafından elde edilen en iyi bireyin yakınsama eğrisi ve evrimsel süreç boyunca katsayı değerlerinin değişimi Şekil 7.5'de gösterilmiştir. Tablo 7.14'de; algoritmaların en iyi, ortalama ve standart sapma sonuçları yer almaktadır. Bu sonuçlar, SSEABC'nin 100 bağımsız çalışmasıyla elde edilmiştir. Bununla birlikte, ABC varyantlarının ve literatürde bulunan diğer algoritmaların sonuçları aynı tabloda listelenmiştir.

Tablo 7.14. *Örnek I için MSE'ye göre istatistiksel karşılaştırma. NR: belirtilmemiş*

Algoritmalar	MSE			MSE (dB)	
	Best	Mean	Std	Best	Mean
SSEABC	4.28E-27	3.35E-24	4.59E-24	-263.689	-234.752
ABC	3.38E-17	8.07E-08	2.97E-07	-164.716	-70.929
MABC	1.90E-22	2.07E-12	9.56E-12	-217.216	-116.837
PSO (Durmuş ve Gün, 2011)	NR	7.94E-14	NR	NR	-131.002
HSA (Durmuş ve Gün, 2011)	NR	2.49E-04	NR	NR	-36.038
HS (Saha vd., 2014)	1.56E-07	NR	NR	-168.062	-160.782
ABC (Karaboga, 2009)	5.14E-16	6.25E-16	NR	-152.89	-152.039
GA (Rashedi vd., 2011)	NR	8.86E-01	NR	NR	-0.526
RGA (Upadhyay vd., 2014)	3.17E-02	5.20E-02	1.52E+00	-14.989	-12.841
PSO (Upadhyay vd., 2014)	2.30E-03	2.98E-03	1.25E+00	-26.383	-25.251
DE (Upadhyay vd., 2014)	3.33E-05	6.11E-05	1.42E+00	-44.772	-42.139
GA (Dai vd., 2010)	1.17E-08	4.72E-06	1.01E-05	-79.318	-53.261



Şekil 7.5. *Örnek I'deki SSEABC, MABC ve ABC'nin yakınsama karakteristikleri ve SSEABC kullanılırken zamanla katsayı değişimi*

Şekil 7.5'de görüldüğü gibi SSEABC algoritması, orijinal ABC ve MABC algoritmalarına göre global optimuma daha hızlı yakınsadığı görülmektedir. SSEABC

algoritması, 4.277E-27 MSE değerine yaklaşık olarak 6300 FEs değerinde ulaşmıştır. Buna karşın, orijinal ABC ve MABC algoritmaları sırasıyla 7000 ve 6300 FEs'de 3.3757E-17 MSE ve 1.8986E-22 MSE'ye takılmışlardır. Şekil 7.5'de SSEABC ile elde edilen tahmini parametre değerlerinin gerçek değerlere erken iterasyonlarda ulaştığı açıkça görülmektedir. Tablo 7.14'deki sonuçlar incelendiğinde algoritmalar arasındaki en küçük MSE değerine sahip olanın SSEABC olduğu görülmektedir.

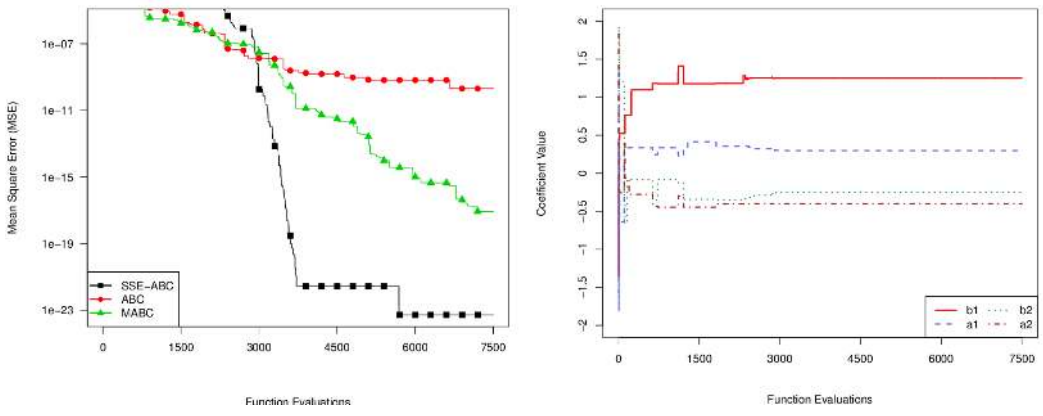
7.2.3.4. *Örnek II sonuçları*

Bu örnekte, ikinci dereceden bilinmeyen bir filtre sistem, ikinci dereceden bir IIR filtresi ile tanımlanır (Dai vd., 2010; Karaboga, 2009). Bilinmeyen sistemin ve IIR filtrenin transfer fonksiyonları sırasıyla Denklem (7.9) ve Denklem (7.10)'de verilmiştir.

$$H_{system}(z) = \frac{1.25z^{-1} - 0.25z^{-2}}{1 - 0.3z^{-1} + 0.4z^{-2}} \tag{7.9}$$

$$H_{filter}(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{7.10}$$

SSEABC tarafından elde edilen en iyi bireyin yakınsama eğrisi ve evrimsel süreç boyunca katsayı değerlerinin değişimi Şekil 7.6'de gösterilmiştir. SSEABC, ABC varyantları ve literatürdeki diğer algoritmaların karşılaştırılması Tablo 7.15'de sunulmuştur.



Şekil 7.6. *Örnek II'deki SSEABC, MABC ve ABC'nin yakınsama grafiği ve SSEABC kullanılırken zamanla katsayı değişimi*

Tablo 7.15. *SSEABC ve ABC varyantlarının Örnek II için sonuçları. NR: belirtilmemiş*

Algoritmalar	MSE			MSE (dB)	
	Best	Mean	Std	Best	Mean
SSEABC	5.25E-24	4.77E-22	4.61E-22	-232.795	-213.215
ABC	2.05E-10	1.46E-06	2.35E-06	-96.882	-58.371
MABC	8.36E-18	9.44E-11	8.81E-10	-170.779	-100.252
PSO (Krusienski ve Jenkins, 2004)	NR	1.26E-04	NR	NR	-39
HS (Saha vd., 2014)	7.36E-10	2.32E-09	2.09E+00	-91.333	-86.349
RGA (Saha vd., 2014)	4.50E-02	6.47E-02	1.28E+00	-13.468	-11.894
DE (Saha vd., 2014)	2.69E-04	4.86E-04	1.51E+00	-35.7	-33.134
PSO (Saha vd., 2014)	1.20E-03	4.47E-03	2.00E+00	-29.208	-23.502
GA (Y. Yang vd., 2017)	3.75E-08	2.38E-05	2.58E-05	-74.26	-46.234
PSO (Y. Yang vd., 2017)	1.46E-10	5.64E-05	1.66E-04	-98.356	-42.487
ICPSO-MS (Y. Yang vd., 2017)	9.20E-17	1.90E-14	2.75E-14	-160.362	-137.212
DE (Y. Yang vd., 2017)	5.23E-11	1.12E-10	2.87E-11	-102.815	-99.508
HSDE (Y. Yang vd., 2017)	1.20E-17	2.53E-16	3.23E-16	-169.208	-155.969
OHCRO (Y. Yang vd., 2017)	6.59E-19	1.54E-15	6.16E-15	-181.811	-148.125
PSO–QI (Luitel ve Venayagamoorthy, 2010)	7.10E-04	7.10E-04	1.15E-07	-31.486	-31.486
DEPSO (Luitel ve Venayagamoorthy, 2010)	7.10E-04	7.28E-04	4.39E-05	-31.486	-31.38

Şekil 7.6'de görüldüğü gibi, SSEABC algoritması, 5.2536E-24 MSE değerine yaklaşık olarak 5700 FEs değerinde ulaşmıştır. Buna karşın, orijinal ABC ve MABC algoritmaları sırasıyla 7000 ve 7400 FEs'de 2.0501E-10 ve 8.3578E-18 MSE'ye takılmışlardır. Şekil 7.6'de SSEABC ile elde edilen tahmini parametre değerlerinin gerçek değerlere erken iterasyonlarda ulaştığı görülmektedir. Bundan dolayı SSEABC'nin yakınsama oranı, rakip algoritmalara göre daha yüksektir denilebilir.Tablo 7.15'deki sonuçlar incelendiğinde, SSEABC'nin diğer bütün algoritmaları en iyi ve ortalama sonuçlarına göre geride bıraktığı söylenebilir.

7.2.3.4. Örnek III sonuçları

Bu örnekte, Denklem (7.11)'de verilen beşinci dereceden bir sistemin transfer fonksiyonu, aynı dereceden (Durum 1) ve düşük-dereceden (Durum 2) IIR filtreler ile tanımlanır (Kumar vd., 2017; Kumar ve Rawat, 2015; Upadhyay vd., 2014).

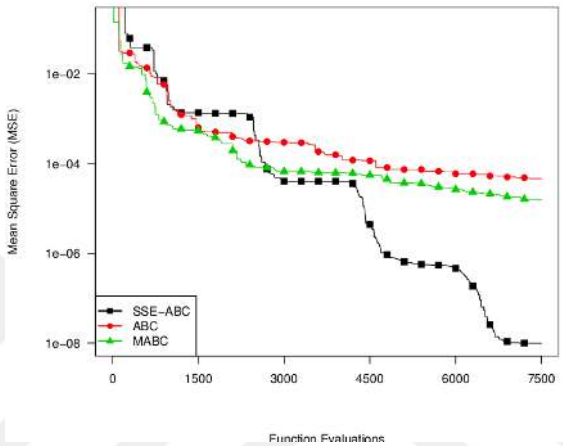
$$H_{system}(z)=\frac{0.1084+0.5419z^{-1}+1.0837z^{-2}+1.0837z^{-3}+0.5419z^{-4}+0.1084z^{-5}}{1+0.9853z^{-1}+0.9738z^{-2}+0.3864z^{-3}+0.1112z^{-4}+0.0133z^{-5}} \quad (7.11)$$

I. Durum

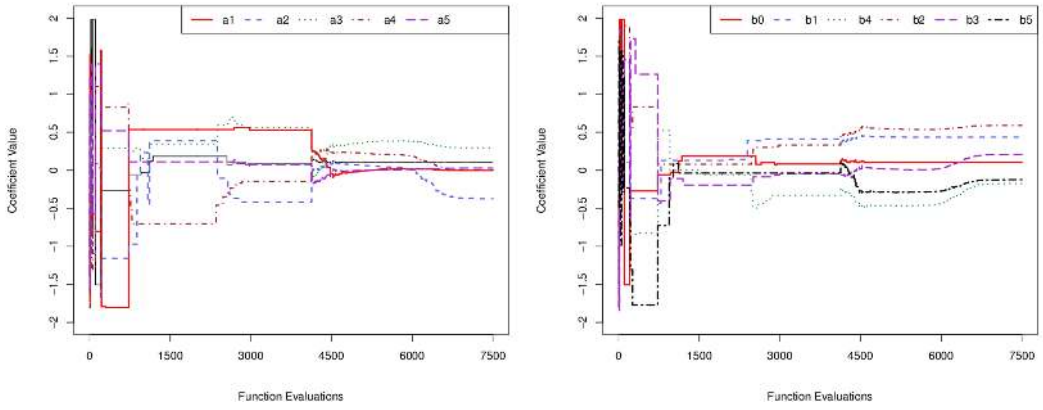
Bu durumda, beşinci dereceden bilinmeyen sistem, yine aynı dereceden bir IIR filtre kullanılarak modellenmiştir. Filtrenin transfer fonksiyonu aşağıdaki gibi tanımlanır:

$$H_{filter}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4} - a_5 z^{-5}} \quad (7.12)$$

ABC algoritmalarının yakınsama davranışları Şekil 7.7'de verilmiştir. SSEABC'nin çalıştırılması sırasındaki IIR filtresinin katsayı değerlerinin değişimi Şekil 7.8'de sunulmuştur. Ayrıca, ABC varyantları ve diğer algoritmaların karşılaştırma sonuçları Tablo 7.16'de listelenmiştir.



Şekil 7.7. *Örnek III'ün I. durumundaki SSEABC, MABC ve ABC'nin yakınsama grafiği*



(a) (b)

Şekil 7.8. *Örnek III'ün I. durumu için SSEABC'nin çalışma sırasındaki "a (sol)" ve "b (sağ)" katsayılarının değişimi*

Tablo 7.16. *Örnek III'ün I. durumundaki SSEABC ve ABC varyantlarının sonuçları*

Algoritmalar	MSE			MSE (dB)	
	Best	Mean	Std	Best	Mean
SSEABC	4.90E-07	2.57E-06	1.48E-06	-63.1	-55.898
ABC	1.54E-05	2.37E-04	1.89E-04	-48.135	-36.251
MABC	7.03E-06	1.35E-04	2.22E-04	-51.53	-38.707
RGA (Saha vd., 2014)	3.07E-02	4.98E-02	1.50E+00	-15.129	-13.031
PSO (Saha vd., 2014)	3.50E-03	1.04E-02	2.06E+00	-24.559	-19.84
DE (Saha vd., 2014)	6.88E-04	1.27E-03	1.50E+00	-31.623	-28.964
HS (Saha vd., 2014)	7.14E-06	1.92E-05	1.91E+00	-51.463	-47.156
CSO (Panda vd., 2011)	1.67E+02	1.68E+02	3.87E-01	22.225	22.247

Şekil 7.7'de görüldüğü gibi, ABC ve MABC algoritmaları 7500 FEs değeri için 1E-05'den daha iyi bir MSE değeri elde edememiştir. Buna rağmen SSEABC algoritması 9.9531E-09 MSE'ye ulaşmıştır. Bu algoritmanın düşünüldüğü gibi daha iyi yakınsama davranışına sahip olduğu söylenebilir. Bu örneğin, tahmin edilecek katsayılarının sayısı fazladır ve bu da MSE değerinin artmasına sebep olur. Bu yüzden, sistem ve filtre modeli arasındaki MSE değerinin daha büyük olması beklenmektedir. Tablo 7.16'de bu durumu açıkça göstermektedir. Problemin zorluğuna rağmen, SSEABC algoritması IIR filtresini diğer algoritmalardan çok daha iyi modelleyebildiği görülmüştür. Tablo 7.16'de de görüldüğü üzere SSEABC çok daha küçük MSE değerleri üretmektedir.

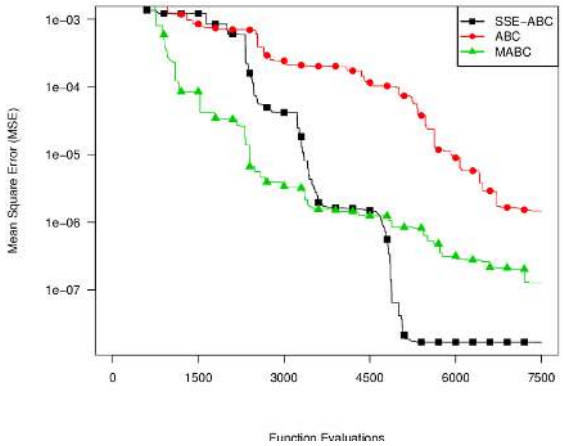
II. Durum

Düşük dereceden IIR filtresinin transfer fonksiyonu aşağıdaki gibidir:

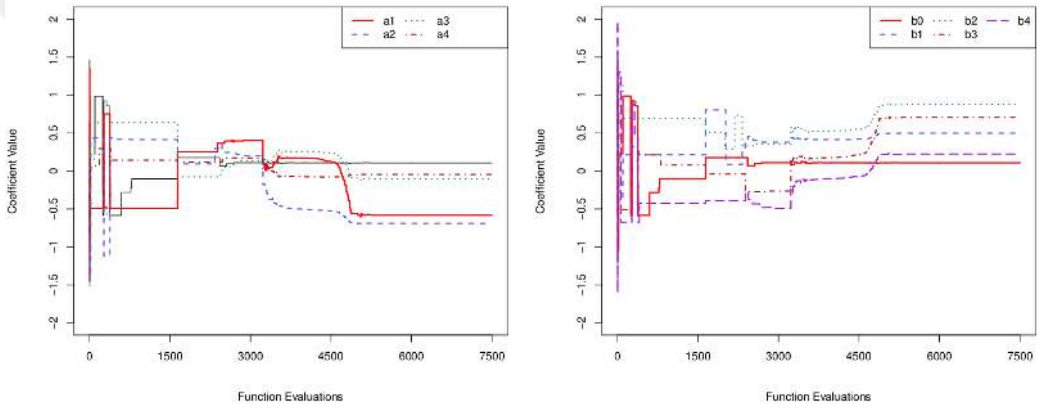
$$H_{filter}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \tag{7.13}$$

Düşük dereceden optimum tanımlama, aynı dereceden yapılan tanımlamaya göre daha zor bir iştir. Şekil 7.9'da ABC tabanlı algoritmalar ile SSEABC'nin yakınsamana eğrileri gösterilmiştir. Şekil 7.10'de ise SSEABC'ye ait en iyi çözümün çalışması sırasındaki katsayı değişiklikleri yer almaktadır. Ayrıca, literatürdeki algoritmaların MSE ile dB sonuçlarının en iyi ve ortalama değerleri Tablo 7.17'de listelenmiştir.

Tablo 7.17'de listelenen sonuçlar incelendiğinde, SSEABC'nin 1.6902E-08 MSE değeri ile ilk sırada yer aldığı görülmektedir. Bununla birlikte, MABC en iyi ikinci ABC ise en iyi üçüncü algoritma olmuştur. Şekil 7.9 ve Tablo 7.17'den elde edilen sonuçlara göre, SSEABC'nin ABC varyantlarından daha iyi yakınsadığı söylenebilir. Ayrıca MSE değerleri karşılaştırılan diğer algoritmalara göre daha iyi performans sergilediği görülmektedir.



Şekil 7.9. *Örnek III'ün II. durumundaki SSEABC, MABC ve ABC'nin yakınsama grafiği*



Şekil 7.10. *Örnek III'ün II. durumu için SSEABC'nin çalışma sırasındaki "a (sol)" ve "b (sağ)" katsayılarının değişimi*

Tablo 7.17. *Örnek III'ün II. durumundaki SSEABC ve ABC varyantlarının sonuçları*

Algoritmalar	MSE			MSE (dB)	
	Best	Mean	Std	Best	Mean
SSEABC	1.69E-08	1.26E-07	6.07E-07	-77.721	-68.988
ABC	1.47E-06	3.05E-05	3.25E-05	-58.331	-45.157
MABC	1.27E-07	3.37E-05	1.72E-04	-68.946	-44.73
RGA (Saha vd., 2014)	1.09E-01	1.59E-01	1.36E+00	-9.638	-7.993
PSO (Saha vd., 2014)	1.27E-02	2.79E-02	1.92E+00	-18.962	-15.54
DE (Saha vd., 2014)	2.70E-03	3.14E-03	1.17E+00	-25.686	-25.03
HS (Saha vd., 2014)	6.12E-06	6.96E-06	1.14E+00	-52.132	-51.572
CSO (Panda vd., 2011)	6.95E-05	7.86E-05	2.50E-05	-41.582	-41.048

8. SONUÇ

Optimizasyon problemlerinin çözümlerinde kullanılan metasezgisellerin, problem çözmede göstermiş olduğu başarılar, algoritma üretme alanında son dönemde popüler olmalarını sağlamıştır. Bu da yüzlerce farklı kaynaklardan esinlenen algoritmaların ortaya çıkmasına sebep olmuştur. Bunların dışında, literatürde farklı yol izleyerek yapılan ve nitelikli olarak kabul gören çalışmaların da yer aldığı görülmektedir. Bunlardan öne çıkan bir tanesi de, birçok algoritma bileşenlerini de içinde barındıran toplu algoritma yapılarıdır.

Bu tez kapsamında, ABC ve PSO için birer toplu algoritma çatısı önerilmiştir. Ayrıca, bu toplu çatılar gerçeklenirken elde edilen deneyimler göstermiştir ki, algoritmaların başarısı büyük oranda algoritmaların bir bileşeni olan arama denklemlerine bağlı olduğudur. Buradan yola çıkılarak arama denklemlerini uyarlamalı olarak belirleyen SSEABC algoritması önerilmiştir.

Çeşitli ABC varyantlarının özeliklerini kendi içinde toplayan ABC-X çatısının performansı ilk olarak farklı ölçüt setlerinin fonksiyonlarını içeren bir karma ölçüt seti ile analiz edilmiştir. Problem türlerine özel ABC-X varyantları üretilerek var olan çeşitli ABC varyantları ile karşılaştırılmıştır. Sonuçlar göstermiştir ki, ABC-X tüm problem türlerinde ABC varyantlarından daha iyi sonuçlar elde etmiştir. Probleme özgü algoritma üretilmesiyle de algoritmanın kendini probleme çözmeye daha iyi adapte ettiği görülmüştür. Boyutlar büyüdükçe algoritmanın üstünlüğünü korumasından dolayı algoritmanın ölçeklenebilirlik özelliğinin diğer ABC varyantlarından iyi olduğu söylenebilir. ABC-X ile yapılan bir başka deney ise gerçek dünya problemi olan valf nokta etkili ekonomik dağıtım problemlerine uygulanmasıdır. Bu probleme özgü üretilen ABC-X varyantı, literatürdeki algoritmalar ile karşılaştırılmıştır. Elde ettiğimiz sonuçlara göre, ABC-X literatürdeki algoritmalara karşı üstünlük sağlamıştır. Burada görülmüştür ki NFL teoreminin de ifade ettiği gibi tek bir algoritma bütün problemlerde üstün olamamaktadır. Bununla birlikte, ABC-X farklı algoritma bileşenlerini içermesi ve parametreler yoluyla şekillendirilebilmesi, ABC-X'e esneklik ve etkinlik katmıştır. ABC-X'in önermiş olduğu toplu algoritma yaklaşımının başarısı deneyler ile doğrulanmıştır.

Tez kapsamı içinde önerilmiş olan bir diğer toplu algoritma çatısı TempPSO'dur. Algoritmanın performansının test edilmesi için ilk olarak CEC'17 ölçüt seti ile deneyler gerçekleştirilmiştir. Literatürde yer alan PSO varyantlarının sonuçları ile karşılaştırıldığında TempPSO'nun diğer PSO varyantlarından daha iyi sonuçlar elde ettiği

görülmüştür. Boyut değiştikçe bu üstünlük değişmemiştir. Bununla birlikte, TempPSO algoritmasının sonuçları CEC'17 yarışmasına katılan algoritmalar ile karşılaştırılmış ve ilk sıralarda yer alan bir algoritma olmuştur. Ayrıca büyük boyut ölçüt seti deneyleri de yapılmıştır. Böylelikle büyük boyutlardaki algoritmanın performansı analiz edilmiştir. Elde edilen sonuçlar PSO varyantları ile karşılaştırılmış ve bu bunlara karşı iyi sonuçlar alınmıştır. PSO için gerçekleştirilen toplu algoritma çatısının farklı problem ve boyutlarda karşılaştırıldığı PSO varyantlarından daha etkin sonuçlar alması tek bir algoritma mantığının problem çözmede yetersizliğini göstermektedir. Yapılan deneyler sonucunda, farklı algoritma bileşenlerini içeren ve problem türlerine göre şekillendirebilen toplu algoritma yapısının, tek algoritma yapısından daha üstün bir yaklaşım olduğu ortaya çıkmaktadır.

PSO ve ABC algoritmaları için önerilen toplu algoritma çatısı geliştirilirken elde edilen deneyimler göstermiştir ki algoritmanın performansına en büyük katkıyı kullanılan arama denkleminin sağladığıdır. Bu düşünce ile yola çıkılarak SSEABC adındaki kendinden uyarlamalı arama denklem yapısını içeren bir ABC varyantı önerilmiştir. SSEABC'nin arama denklemi tek bir arama denklemi ile çalışamaya başlamak yerine çalışma anında dinamik olarak değiştirilerek, farklı algoritmaymış gibi çalışır. Çok sayıda rastgele denklemini bir havuza doldurularak bunu gerçekleştirir. SSEABC algoritmasının performansı CEC'17, SOCO ve gerçek dünya deneyleri ile analiz edilmiştir. Elde edilen sonuçlar ABC varyantları ve CEC'17 yarışmacıları ile karşılaştırılmıştır. Sonuçlar göstermiştir ki SSEABC, hem ABC varyantlarına hem de CEC'17 yarışmacılarına karşı üstün performans sergilemiştir. Bununla birlikte, SOCO büyük boyut deneylerindeki ABC varyantları ile karşılaştırılmış ve ABC varyantlarından daha iyi sonuçlar elde ettiği görülmüştür. Son olarak SSEABC, bir dünya problemi olan IIR filtre tanımlama problemlerine uygulanmıştır. Elde edilen sonuçlar literatürde yer alan diğer algoritmalar ile karşılaştırılmıştır. SSEABC'nin elde ettiği değerler diğer algoritmalardan daha küçük değerlere ulaşmıştır ve sonucunda literatürde yer alan algoritmalardan daha performanslı olduğunu yorumu yapılabilir. Yapılan deneyler göstermiştir ki, farklı problem türlerinin çözümleri için tek bir algoritma ve tek bir arama denkleminin yetersiz kalmaktadır. SSEABC'nin sahip olduğu kendinden uyarlanabilir arama denklemi yapısı mevcut yetersizliklere bir çözüm olmaktadır. Dinamik yapısı ve denklemlerin başarım oranına göre değişimi sayesinde algoritmanın kendisini problem türlerine daha iyi adapte ettiği

söylenebilir. ABC varyantlarına ve farklı metasegizel algoritmalara göre çıkarılan sonuçlar bunu kanıtlamaktadır.

Genel olarak değerlendirilirse, NFL teoreminin de ifade ettiği gibi tek bir algoritma bütün problem türlerine karşı iyi performans sağlamadığı görülmüştür. Dolayısıyla farklı algoritma ve algoritma bileşenleri içeren yaklaşımların tek bir algoritmaya göre daha üstün olduğu açıktır. Geliştirilen toplu algoritma yapıları da bunun doğruluğunu kanıtlamaktadır. Ayrıca algoritmaların arama denklemi yapısının dinamik olarak değiştirilmesi, algoritmanın çalışırken probleme göre özelleşmesini ve farklı bir algoritma gibi davranmasını sağladığı görülmüştür. Bundan sonra yapılabilecek bir diğer çalışma da bir metasegizel algoritmanın varyantlarını baz almak yerine daha genelleşmiş, metaforlardan bağımsız, optimizasyon terminolojisini kullanan ve çok sayıda algoritmanın bileşenlerini üretebilecek bir algoritma çatısının oluşturulmasıdır

KAYNAKÇA

Adenso-Díaz, B., ve Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54 (1), 99-114.

Agarwalla, P., ve Mukhopadhyay, S. (2017). Efficient player selection strategy based diversified particle swarm optimization algorithm for global optimization. *Information Sciences*, 397-398, 1339-1351.

Akay, B., ve Karaboga, D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, 192, 120-142.

Akay, R., Basturk, A., Kalinli, A., ve Yao, X. (2017). Parallel population-based algorithm portfolios: An empirical study. *Neurocomputing*, 247, 115-125.

Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications*, 37(8), 5682-5687.

Alswaitti, M., Albughdadi, M., ve Isa, N. A. M. (2018). Density-based particle swarm optimization algorithm for data clustering. *Expert Systems with Applications*, 91, 170-186.

Ansótegui, C., Sellmann, M., ve Tierney, K. (2009). A gender-based genetic algorithm for the automatic configuration of algorithms. In *International Conference on Principles and Practice of Constraint Programming,* 142-157.

Auger, a., ve Hansen, N. (2005). Performance evaluation of an advanced local search evolutionary algorithm. In *2005 IEEE Congress on Evolutionary Computation*, 2 (2), 1777-1784.

Awad, N. H., Ali, M. Z., Liang, J. J., Qu, B. Y., ve Suganthan, P. N. (2016). *Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization*. Singapore: Nanyang Technological University

Awad, N. H., Ali, M. Z., ve Suganthan, P. N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems. In *2017 IEEE Congress on Evolutionary Computation (CEC),* 372-379.

Aydın, D. (2015). Composite artificial bee colony algorithms: From component-based analysis to high-performing algorithms. *Applied Soft Computing*, 32, 266-285.

Aydın, D., Liao, T., de Oca, M. A. M., ve Stützle, T. (2012). Improving performance via population growth and local search: the case of the artificial bee colony algorithm. *International Conference on Artificial Evolution (Evolution Artificielle),* Springer, Berlin, Heidelberg, 85-96.

Aydın, D., Liao, T., de Oca, M. A. M., ve Stützle, T. (2011). Improving performance via population growth and local search: the case of the artificial bee colony algorithm. *International Conference on Artificial Evolution (Evolution Artificielle)* , Springer, Berlin, Heidelberg, 85-96.

Aydın, D., ve Özyön, S. (2013). Solution to non-convex economic dispatch problem with valve point effects by incremental artificial bee colony with local search. *Applied Soft Computing Journal*, 13 (5), 2456-2466.

Aydın, D., Özyön, S., Yaşar, C., ve Liao, T. (2014). Artificial bee colony algorithm with dynamic population size to combined economic and emission dispatch problem. *International Journal of Electrical Power & Energy Systems*, 54, 144-153.

Aydın, D., ve Stützle, T. (2015). A configurable generalized artificial bee colony algorithm with local search strategies. In *2015 IEEE Congress on Evolutionary Computation (CEC),* IEEE, 1067-1074.

Aydın, D., Yavuz, G., Özyön, S., Yaşar, C., ve Stützle, T. (2017). Artificial bee colony framework to non-convex economic dispatch problem with valve point effects. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, 1311-1318.

Aydın, D., Yavuz, G., ve Stützle, T. (2017). ABC-X: a generalized, automatically configurable artificial bee colony framework. *Swarm Intelligence*, 11 (1), 1-38.

Banharnsakun, A., Achalakul, T., ve Sirinaovakul, B. (2011). The best-so-far selection in Artificial Bee Colony algorithm. *Applied Soft Computing Journal*, 11 (2), 2888-2901.

Bao, L., ve Zeng, J.-C. (2011). A bi-group differential artificial bee colony algorithm. *Control Theory & Applications*, 28 (2), 266-272.

Bartz-Beielstein, T., Lasarczyk, C. W. G., ve Preuss, M. (2005). Sequential parameter optimization. *2005 IEEE Congress on Evolutionary Computation,* IEEE, 773-780

Biedrzycki, R. (2017). A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems. In *2017 IEEE Congress on Evolutionary Computation (CEC),* IEEE, 1489-1494

Birattari, M., Yuan, Z., Balaprakash, P., ve Stützle, T. (2010). F-race and iterated f-race: an overview. *Experimental Methods for the Analysis of Optimization Algorithms,* Berlin, Heidelberg: Springer Berlin Heidelberg, 311-336.

Blum, C., ve Li, X. (2008). Swarm intelligence in optimization. *Swarm Intelligence,* 272, 43-85).

Bonyadi, M. R., ve Michalewicz, Z. (2016). Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary Computation*, 1530, 9304.

Boussaïd, I., Lepagnot, J., ve Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237, 82-117.

Brest, J., Maucec, M. S., ve Boskovic, B. (2017). Single objective real-parameter optimization: Algorithm jSO. In *2017 IEEE Congress on Evolutionary Computation (CEC),* IEEE, 1311-1318.

Brest, J., ve Maučec, M. S. M. (2011). Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, 15 (1997), 2157-2174.

Bujok, P., ve Tvrdik, J. (2017). Enhanced individual-dependent differential evolution with population size adaptation. In *2017 IEEE Congress on Evolutionary Computation (CEC),* IEEE, 1358-1365.

Bullnheimer, B., Hartl, R. F., ve Strauss, C. (1997). A new rank based version of the ant system. A computational study. *Central European Journal of Operations Research,* 7, 25-38

Cahon, S., Melab, N., ve Talbi, E. G. (2004). ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10 (3), 357-380.

Câmara, D. (2015). Swarm intelligence (SI). *Bio-inspired Networking,* In (s. 81-102), ISTE Press

Cazzaniga, P., Nobile, M. S., ve Besozzi, D. (2015). The impact of particles initialization in PSO: Parameter estimation as a case in point. *2015 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), IEEE,* 1-8

Chen, C. H., ve Yeh, S. N. (2006). Particle swarm optimization for economic power dispatch with valve-point effects. *2006 IEEE/PES Transmission Distribution Conference and Exposition: Latin America,* IEEE, 1-5.

Chen, K., Zhou, F., Yin, L., Wang, S., Wang, Y., ve Wan, F. (2017). A hybrid particle swarm optimizer with sine cosine acceleration coefficients. *Information Sciences*, 422, 218-241.

Chen, Y., Li, L., Peng, H., Xiao, J., ve Wu, Q. (2018). Dynamic multi-swarm differential learning particle swarm optimizer. *Swarm and Evolutionary Computation*, 39, 209-221.

Chen, Y., Li, L., Peng, H., Xiao, J., Yang, Y., ve Shi, Y. (2017). Particle swarm optimizer with two differential mutation. *Applied Soft Computing*, 61*,* 314-330.

Chiang, C.-L. (2007). Genetic-based algorithm for power economic load dispatch. *IET generation, transmission & distribution*, 1 (2), 261-269.

Chu, S., Tsai, P., ve Pan, J. (2006). Cat swarm optimization. In *Pacific Rim international conference on artificial intelligence,* Berlin/Heidelberg: Springer-Verlag, 854-858.

Clerc, M., ve Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1), 58-73.

Coelho, L. S., ve Mariani, V. C. (2006). Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect. *IEEE Transactions on power systems*, 21 (2), 989-996.

Dai, C., Chen, W., ve Zhu, Y. (2010). Seeker optimization algorithm for digital IIR filter design. *IEEE transactions on industrial electronics*, 57 (5), 1710-1718.

Diwold, K., Aderhold, A., Scheidler, A., ve Middendorf, M. (2011). Performance evaluation of artificial bee colony optimization and new selection schemes. *Memetic Computing*, 3 (3), 149-162.

Dorigo, M., ve Birattari, M. (2010). *Ant colony optimization*. Springer US.

Dorigo, M., ve Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1 (1), 53-66.

Dorigo, M., Maniezzo, V., ve Colorni, A. (1991). *The ant system: An autocatalytic optimizing process*, Technical Report 91-016 Revised, Italy: Dipartimento di Elettronica, Politecnico di Milano

Duarte, A., Marti, R., ve Gortazar, F. (2011). Path relinking for large-scale global optimization. *Soft Computing*, 15 (11), 2257-2273.

Dubois-Lacoste, J., Lopez-Ibanez, M., ve Stützle, T. (2011). A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Computers and Operations Research*, 38 (8), 1219-1236.

Durmuş, B., ve Gün, A. (2011). Parameter Identification using particle swarm optimization. In *Proceedings, 6th International Advanced Technologies Symposium, Elazığ,* 16-18.

Eberhart, R., ve Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39-43.

Eiben, a. E., Hinterding, R., ve Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3 (2), 124-141.

El-Abd, M. (2011). Opposition-based artificial bee colony algorithm. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM,* 109-116.

Engelbrecht, A. (2012). Particle swarm optimization: Velocity initialization. In *2012 IEEE Congress on Evolutionary Computation,* (2), 10-15.

Ertenlice, O., ve Kalayci, C. B. (2018). A survey of swarm intelligence for portfolio optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 39, 36-52.

Eshelman. (1991). The CHC adaptive search algorithm : How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*, 265-283.

Fister, I., Yang, X. S., Brest, J., ve Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik/Electrotechnical Review*, 80 (3), 116-122.

Fister Jr, I., Mlakar, U., Brest, J., Fister, I., Jr, I. F., Mlakar, U., Brest, J., vd. (2016). A new population-based nature-inspired algorithm every month: is the current era coming to the end. In *Proceedings of the 3rd Student Computer Science Research Conference,* 33-37.

Franzin, A., ve Stützle, T. (2019). Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, 104, 191-206.

Gao, W. feng, Liu, S. yang, ve Huang, L. ling. (2012). Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Communications in Nonlinear Science and Numerical Simulation*, 17 (11), 4316-4327.

Gao, W., ve Liu, S. (2011). Improved artificial bee colony algorithm for global optimization. *Information Processing Letters*, 111 (17), 871-882.

Gao, W., Liu, S., ve Huang, L. (2012). A global best artificial bee colony algorithm for global optimization. *Journal of Computational and Applied Mathematics*, 236 (11), 2741-2753.

Gao, W., Liu, S., ve Huang, L. (2014). Enhancing artificial bee colony algorithm using more information-based search equations. *Information Sciences*, *270*, 112-133. Elsevier.

García-Martínez, C., Rodríguez, F. J., ve Lozano, M. (2011). Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. *Soft Computing*, 15 (11), 2109-2126.

Garcia-Nieto, J., ve Alba, E. (2011). Restart particle swarm optimization with velocity modulation: a scalability test. *Soft Computing*, 15 (11), 2221-2232.

García-Nieto, J., ve Alba, E. (2014). Hybrid PSO6 for hard continuous optimization. *Soft Computing*, 19 (7), 1843-1861.

Gardeux, V., Chelouah, R., Siarry, P., ve Glover, F. (2011). EM323: a line search based algorithm for solving high-dimensional continuous non-linear optimization problems. *Soft Computing*, 15 (11), 2275-2285.

Garnier, S., Gautrais, J., ve Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, 1 (1), 3-31.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, 76 (2), 60-68.

Ghambari, S., ve Rahati, A. (2018). An improved artificial bee colony algorithm and its application to reliability optimization problems. *Applied Soft Computing Journal*, *62*, 736-767.

Glover, F., ve Laguna, M. (1998). Tabu search. *Handbook of combinatorial optimization* Kluwer Academic Publishers, 3, 621-757.

Gomes, C. P., ve Selman, B. (1997). Algorithm portfolio design: Theory vs. practice. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence,* 190-197.

Gong, Y. J., Li, J. J., Zhou, Y., Li, Y., Chung, H. S. H., Shi, Y. H., ve Zhang, J. (2016). Genetic learning particle swarm optimization. *IEEE Transactions on Cybernetics*, 46 (10), 2277-2290.

Gotmare, A., Bhattacharjee, S. S., Patidar, R., ve George, N. V. (2017). Swarm and evolutionary computing algorithms for system identification and filter design: A comprehensive review. *Swarm and Evolutionary Computation*, *32*, 68-84.

Harrison, K. R., Engelbrecht, A. P., ve Ombuki-Berman, B. M. (2016). Inertia weight control strategies for particle swarm optimization. *Swarm Intelligence*, 10 (4), 267-305.

Helwig, S., Branke, J., ve Mostaghim, S. (2013). Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 17 (2), 259-271.

Hiroyasu, T., Miki, M., Ono, Y., ve Minami, Y. (2000). Ant colony for continuous functions. *The Science and Engineering, Doshisha University*, *20*.

Ho, Y. C., ve Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115 (3), 549-570.

Hoos. H. H. (2011). Automated parameter tuning and algorithm configuration. *Autonomous Search*, 37-71

Huang, H., Qin, H., Hao, Z., ve Lim, A. (2012). Example-based learning particle swarm optimization for continuous optimization. *Information Sciences*, 182 (1), 125-138.

Huberman, B. A., Lukose, R. M., ve Hogg, T. (1997). An economics approach to hard computational problems. *Science*, 275 (5296), 51-54.

Hutter, F., Hoos, H. H., ve Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization,* Springer, Berlin, Heidelberg, 507-523.

Hutter, F., Hoos, H. H., Leyton-Brown, K., ve Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36 (1), 267-306.

Hossain, S. I., Akhand, M. A. H., Shuvo, M. I. R., Siddique, N., ve Adeli, H. (2019). Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert Systems with Applications*. 127, 9-24.

Jagodzinski, D., ve Arabas, J. (2017). A differential evolution strategy. In *2017 IEEE Congress on Evolutionary Computation (CEC),* 1872-1876.

Kang, F., Li, J., ve Ma, Z. (2011). Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, 181 (16), 3508-3531.

Karaboga, D., ve Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization*, 39 (3), 459-471.

Karaboga, D., Gorkemli, B., Ozturk, C., ve Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, 42 (1), 21-57.

Karaboga, N. (2005). Digital IIR filter design using differential evolution algorithm. *EURASIP Journal on Applied Signal Processing*, 1269-1276.

Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of the Franklin Institute*, 346 (4), 328-348.

Karafotias, G., Hoogendoorn, M., ve Eiben, A. E. (2013). Why parameter control mechanisms should be benchmarked against random variation. In *2013 IEEE Congress on Evolutionary Computation,* 349-355.

Kashef, S., ve Nezamabadi-pour, H. (2015). An advanced ACO algorithm for feature subset selection. *Neurocomputing*, 147, 271-279.

Kennedy, J. (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 Congress on Evolutionary Computation,* 3, 1931-1938.

Kennedy, J., ve Eberhart, R. (1995). Particle swarm optimization. *In Proceedings of the IEEE international conference on neural networks,4*,1942-1948.

Kommadath, R., ve Kotecha, P. (2017). Teaching learning based optimization with focused learning and its performance on CEC2017 functions. *2017 IEEE Congress on Evolutionary Computation (CEC),* 2397-2403.

Krusienski, D. J., ve Jenkins, W. K. (2004). Particle swarm optimization for adaptive IIR filter structures. *Evolutionary Computation, In Proceedings of the 2004 Congress on Evolutionary Computation,* 965-970.

Kumar, M., ve Rawat, T. K. (2015). Optimal fractional delay-IIR filter design using cuckoo search algorithm. *ISA transactions*, 59, 39-54.

Kumar, M., Rawat, T. K., ve Aggarwal, A. (2017). Adaptive infinite impulse response system identification using modified-interior search algorithm with Lèvy flight. *ISA transactions*, 67, 266-279.

Van Laarhoven, P. J. M., ve Aarts, E. H. L. (1987). Simulated annealing. In *Simulated annealing: Theory and applications,* Springer, Dordrecht, 7-15.

LaTorre, A., Muelas, S., ve Peña, J. M. (2011). A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Computing*, *15*(11), 2187-2199.

LaTorre, A., ve Pena, J.-M. (2017). A comparison of three large-scale global optimizers on the CEC 2017 single objective real parameter numerical optimization benchmark. In *2017 IEEE Congress on Evolutionary Computation (CEC),* 1063-1070.

Leung, S. Y. S., Tang, Y., ve Wong, W. K. (2012). A hybrid particle swarm optimization and its application in neural networks. *Expert Systems with Applications*, 39 (1), 395-405.

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

Liang, J. J., Qu, B. Y., ve Suganthan, P. N. (2013). *Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization.* Singapore: Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University.

Liao, T., Aydın, D., ve Stützle, T. (2013). Artificial bee colonies for continuous optimization: Experimental analysis and improvements. *Swarm Intelligence*, 7 (4), 327-356.

Liao, T., de Oca, M. A., Aydin, D., Stützle, T., ve Dorigo, M. (2011). An incremental ant colony algorithm with local search for continuous optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation,* 125-132.

Liao, T., Oca, M. A. M. De, ve Dorigo, M. (2011). An Incremental ant colony algorithm with local search for continuous optimization. *Soft Computing*, 1-17.

Liao, T., Stützle, T., Montes de Oca, M. A., ve Dorigo, M. (2014). A unified ant colony optimization algorithm for continuous optimization. *European Journal of Operational Research*, 234 (3), 597-609.

Lin, A., Sun, W., Yu, H., Wu, G., ve Tang, H. (2018). Global genetic learning particle swarm optimization with diversity enhancement by ring topology. *Swarm and Evolutionary Computation*, 1-13.

Ling, S.-H., Iu, H. H. C., Chan, K. Y., Lam, H.-K., Yeung, B. C. W., ve Leung, F. H. (2008). Hybrid particle swarm optimization with wavelet mutation and its industrial applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38 (3), 743-763.

Ling, S.-H., ve Leung, F. H. F. (2007). An improved genetic algorithm with average-bound crossover and wavelet mutation operations. *Soft Computing*, 11 (1), 7-31.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., ve Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3, 43-58.

López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., ve Birattari, M. (2011). *The irace package, iterated race for automatic algorithm configuration*. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Lopez-Ibanez, M., ve Stützle, T. (2012). The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16 (6), 861-875.

Lourenço, H. R., Martin, O. C., ve Stützle, T. (2003). Iterated local search. *Handbook of metaheuristics,* 320-353.

Lozano, M., Molina, D., ve Herrera, F. (2010). Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 15 (11), 2085-2087.

Lu, P., Zhou, J., Zhang, H., Zhang, R., ve Wang, C. (2014). Chaotic differential bee colony optimization algorithm for dynamic economic dispatch problem with valve-point effects. *International Journal of Electrical Power & Energy Systems*, 62, 130-143.

Luitel, B., ve Venayagamoorthy, G. K. (2010). Particle swarm optimization with quantum infusion for system identification. *Engineering Applications of Artificial Intelligence*, 23 (5), 635-649.

Lynn, N., ve Suganthan, P. N. (2017). Ensemble particle swarm optimizer. *Applied Soft Computing*, 55, 533-548.

Maharana, D., Kommadath, R., ve Kotecha, P. (2017). Dynamic yin-yang pair optimization and its performance on single objective real parameter problems of CEC 2017. *2017 IEEE Congress on Evolutionary Computation (CEC),* 2390-2396.

Malik, T. N., ul Asar, A., Wyne, M. F., ve Akhtar, S. (2010). A new hybrid approach for the solution of nonconvex economic dispatch problem with valve-point effects. *Electric Power Systems Research*, 80 (9), 1128-1136.

Mendes, R., Kennedy, J., ve Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8 (3), 204-210.

Mohamed, A. W., Hadi, A. A., Fattouh, A. M., ve Jambi, K. M. (2017). LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. *2017 IEEE Congress on Evolutionary Computation (CEC),* 145-152.

Molina, D., Lozano, M., Sánchez, A. M., ve Herrera, F. (2011). Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains. *Soft Computing*, 15 (11), 2201-2220.

Montero, E., Riff, M. C., ve Neveu, B. (2014). A beginner's guide to tuning methods. *Applied Soft Computing Journal*, 17, 39-51.

Montes de Oca, M. A., Aydın, D., ve Stützle, T. (2011). An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms. *Soft Computing*, 15 (11), 2233-2255.

Montes de Oca, M. A., Stützle, T., Birattari, M. ve Dorigo, M. (2009). Frankenstein's PSO: A composite particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 13 (5), 1120-1132.

Montes de Oca, M. A., Stützle, T., Van Den Enden, K., ve Dorigo, M. (2011). Incremental social learning in particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41 (2), 368-384.

Nannen, V., ve Eiben, a. E. (2007). Efficient relevance estimation and value calibration of evolutionary algorithm parameters. *2007 IEEE Congress on Evolutionary Computation,* 103-110.

Neumaier, A., Fendl, H., Schilly, H., ve Leitner, T. (2011). VXQR: derivative-free unconstrained optimization based on QR factorizations. *Soft Computing*, 15 (11), 2287-2298.

Ongsakul, W., ve Tantimaporn, T. (2006). Optimal power flow by improved evolutionary programming. *Electric Power Components and Systems*, 34 (1), 79-95.

Özyön, S., Yaşar, C., Özcan, G., ve Temurtaş, H. (2011). An artificial bee colony algorithm (abc) aproach to nonconvex economic power dispatch problems with valve point effect. In *National Conference on Electrical, Electronics and Computer (FEEB 11),* Elazığ, 294-299.

Özyön, S., Yaşar, C., ve Temurtaş, H. (2011). Differential evolution algorithm approach to nonconvex economic power dispatch problems with valve point effect. In *6th International Advanced Technologies Symposium (IATS11),* Elazığ, 181-186).

Pan, W.-T. (2012). A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems*, 26, 69-74.

Panda, G., Pradhan, P. M., ve Majhi, B. (2011). IIR system identification using cat swarm optimization. *Expert Systems with Applications*, 38 (10), 12671-12683.

Parejo, J. A., Ruiz-Cortés, A., Lozano, S., ve Fernandez, P. (2012). Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Computing*, 16 (3), 527-561.

Park, J.-B., Lee, K.-S., Shin, J.-R., ve Lee, K. Y. (2005). A particle swarm optimization for economic dispatch with nonsmooth cost functions. *IEEE Transactions on Power systems*, 20 (1), 34-42.

Peng, F., Tang, K., Chen, G., ve Yao, X. (2010). Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14 (5), 782-800.

Pérez Cáceres, L., ve Stützle, T. (2017). Exploring variable neighborhood search for automatic algorithm configuration. *Electronic Notes in Discrete Mathematics*, 58, 167-174.

Piotrowski, A. P., Napiorkowski, J. J., ve Rowinski, P. M. (2014). How novel is the "novel" black hole optimization approach? *Information Sciences*, 267, 191-200.

Rashedi, E., Nezamabadi-Pour, H., ve Saryazdi, S. (2011). Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 24 (1), 117-122.

Ratnaweera, A., Halgamuge, S. K., ve Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8 (3), 240-255.

Rego, C., ve Glover, F. (2007). Local search and metaheuristics. *The traveling salesman problem and its variations,* 309-368

Ren, Z., Zhang, A., Wen, C., ve Feng, Z. (2014). A scatter learning particle swarm optimization algorithm for multimodal problems. *IEEE Transactions on Cybernetics*, 44(7), 1127-1140.

Saad, A., Khan, S. A., ve Mahmood, A. (2018). A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. *Swarm and Evolutionary Computation*, 38, 187-201.

Saha, S. K., Kar, R., Mandal, D., ve Ghoshal, S. P. (2014). Harmony search algorithm for infinite impulse response system identification. *Computers & Electrical Engineering*, 40 (4), 1265-1285.

Sallam, K. M., Elsayed, S. M., Sarker, R. A., ve Essam, D. L. (2017). Multi-method based orthogonal experimental design algorithm for solving CEC2017 competition problems. In *2017 IEEE Congress on Evolutionary Computation*, 1350-1357.

Selvakumar, A. I., ve Thanushkodi, K. (2007). A new particle swarm optimization solution to nonconvex economic dispatch problems. *IEEE transactions on power systems*, 22 (1), 42-51.

Selvakumar, A. I., ve Thanushkodi, K. (2008). Anti-predatory particle swarm optimization: solution to nonconvex economic dispatch problems. *Electric Power Systems Research*, 78 (1), 2-10.

Sharma, T. K., ve Pant, M. (2013). Enhancing the food locations in an artificial bee colony algorithm. *Soft Computing*, 17 (10), 1939-1965.

Shi, Y., ve Eberhart, R. (1998). A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence,* 69-73.

Shi, Y., ve Eberhart, R. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1945*-1950

Shi, Y., Liu, H., Gao, L., ve Zhang, G. (2011). Cellular particle swarm optimization. *Information Sciences*, 181 (20), 4460-4493.

Sinha, N., Chakrabarti, R., ve Chattopadhyay, P. K. (2003). Evolutionary programming techniques for economic load dispatch. *IEEE Transactions on evolutionary computation*, 7 (1), 83-94.

Smit, S. K., ve Eiben, A. E. (2010). Using entropy for parameter analysis of evolutionary algorithms. *Experimental Methods for the Analysis of Optimization Algorithms*, 287-310.

Socha, K., ve Dorigo, M. (2008). Ant colony optimization for continuous domains. *European journal of operational research*, 185 (3), 1155-1173.

Sörensen, K. (2015). Metaheuristics-the metaphor exposed. *International Transactions in Operational Research*, 22 (1), 3-18.

Sorensen, K., Sevaux, M., ve Glover, F. (2017). A history of metaheuristics, *Handbook of heuristics* , 1-18.

Storn, R., ve Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 341-359.

Stützle, T., ve Hoos, H. H. (2000). Max-min ant system. *Future Generation Computer Systems*, 16 (8), 889-914.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., ve Tiwari, S. (2005). *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization.* Nanyang Tech. Univ, Singapore and KanGAL: Kanpur Genetic Algorithms Lab, Kanpur, India.

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

Talbi, E.-G. (2009). *Metaheuristics: From design to implementation. Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: John Wiley & Sons.

Tangherloni, A., Rundo, L., ve Nobile, M. S. (2017). Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems. In *2017 IEEE Congress on Evolutionary Computation*, 1940-1947.

Thitithamrongchai, C., ve Eua-Arporn, B. (2007). Self-adaptive differential evolution based optimal power flow for units with non-smooth fuel cost functions. *Journal of Electrical Systems*, 3 (2), 88-99.

Tseng, L.-Y., ve Chen, C. (2008). Multiple trajectory search for large scale global optimization. In *2008 IEEE Congress on Evolutionary Computation,* 3052-3059.

Upadhyay, P., Kar, R., Mandal, D., ve Ghoshal, S. P. (2014). Craziness based particle swarm optimization algorithm for IIR system identification problem. *AEU-International Journal of Electronics and Communications*, 68 (5), 369-378.

Vrugt, J. A., Robinson, B. A., ve Hyman, J. M. (2009). Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Transactions on Evolutionary Computation*, 13 (2), 243-259.

Wang, H., Wu, Z., ve Rahnamayan, S. (2011). Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Computing*, 15 (11), 2127-2140.

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

Wang, M., Tong, X., ve Li, B. (2017). KW-race and fast KW-race: Racing-based frameworks for tuning parameters of evolutionary algorithms on black-box optimization problems. In *Asia-Pacific Conference on Simulated Evolution and Learning*, 617-628.

Wang, S. K., Chiou, J. P., ve Liu, C. W. (2007). Non-smooth/non-convex economic dispatch by a novel hybrid differential evolution algorithm. *IET Generation, Transmission & Distribution*, 1 (5), 793-803.

Weber, M., Neri, F., ve Tirronen, V. (2011). Shuffle or update parallel differential evolution for large-scale optimization. *Soft Computing*, 15 (11), 2089-2107.

Weyland, D. (2015). A critical analysis of the harmony search algorithm-How not to solve sudoku. *Operations Research Perspectives*, 2, 97-105.

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

Wu, G., Shen, X., Li, H., Chen, H., Lin, A., ve Suganthan, P. N. (2018). Ensemble of differential evolution variants. *Information Sciences*, 423, 172-186.

Xiang, W., ve An, M. (2013). An efficient and robust artificial bee colony algorithm for numerical optimization. *Computers & Operations Research*, 40 (5), 1256-1265.

Xiang, W., Ma, S., ve An, M. (2014). HABCDE: A hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution. *Applied Mathematics and Computation*, *238*, 370-386.

Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. United Kingdom: Luniver press.

Yang, X.-S., Deb, S., Zhao, Y.-X., Fong, S., ve He, X. (2017). Swarm intelligence: past, present and future. *Soft Computing*, 22 (18), 5923-5933.

Yang, X. (2010). Firefly algorithm, Lévy flights and global optimization. In *Research and Development in Intelligent Systems XXVI,* London, 209-218.

Yang, X. S. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2 (2), 78.

Yang, Y., Yang, B., ve Niu, M. (2017). Adaptive infinite impulse response system identification using opposition based hybrid coral reefs optimization algorithm. *Applied Intelligence*, 1-18.

Yang, Z., Tang, K., ve Yao, X. (2011). Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing*, 15 (11), 2141-2155.

Yaşar, C., ve Özyön, S. (2011). A new hybrid approach for nonconvex economic dispatch problem with valve-point effect. *Energy*, *36*(10), 5838-5845.

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

Yu, X., ve Gen, M. (2010). *Introduction to evolutionary algorithms*. London : Springer Science & Business Media.

Zhan, Z.-H. H., Zhang, J., Li, Y., ve Shi, Y.-H. H. (2011). Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 15 (6), 832-847.

Zhang, X., ve Yuen, S. Y. (2013). Improving artificial bee colony with one-position inheritance mechanism. *Memetic Computing*, 5 (3), 187-211.

Zhao, S.-Z., Suganthan, P. N., ve Das, S. (2011). Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing*, 15 (11), 2175-2185.

ÖZGEÇMİŞ

Adı Soyadı	: Gürcan YAVUZ
Yabancı Dil	: İngilizce
Doğum Yeri ve Yılı	: Akhisar / 02.05.1985
E-Posta	: gurcan.yavuz@dpu.edu.tr

Eğitim Geçmişi:

- Doktora (2013-) : Eskişehir Teknik Üniversitesi, Fen Bil. Ens., Bilgisayar Mühendisliği A.B.D.
- Yüksek Lisans (2010-2013) : Kütahya Dumlupınar Üniversitesi, Fen Bil. Ens., Elektrik-Elektronik Mühendisliği A.B.D.
- Lisans (2006-2010) : Kütahya Dumlupınar Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü
- Lisans (2005-2009) : Kütahya Dumlupınar Üniversitesi, Bilgisayar Mühendisliği Bölümü
- Lise (1996-2003) : Akhisar Anadolu Lisesi

Meslek Geçmişi:

- Araştırma Görevlisi (2009-) : Kütahya Dumlupınar Üniversitesi, Bilgisayar Mühendisliği Bölümü

Uluslararası bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler:

Tural, A., **Yavuz, G**., & Aydin, D. (2018). Multi-level image thresholding with global-best distance artificial bee colony algorithm. *2018 26th Signal Processing and Communications Applications Conference (SIU),* 1-4.

Aydin, D., & **Yavuz, G. (2017). Self-adaptive search equation-based artificial bee colony algorithm with CMA-ES on the noiseless BBOB testbed. *Proceedings of the Genetic and Evolutionary Computation Conference Companion,* ACM. 1742-1749.

Aydin, D., **Yavuz, G., Özyön, S., Yaşar, C., & Stützle, T. (2017, July). Artificial Bee colony framework to non-convex economic dispatch problem with valve point effects: A case study. *Proceedings of the Genetic and Evolutionary Computation Conference Companion,* ACM, 1311-1318.

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

Aydın, D., & **Yavuz, G. (2016). A self-adaptive artificial bee colony algorithm with incremental population size for large scale optimization. *International Conference on Soft Computing-MENDEL,* Springer, Cham. 111-123.

Uluslararası hakemli dergilerde basılan çalışmalar:

Durmuş, B., **Yavuz, G. ve Aydın, D. (Basım aşamasında). Adaptive IIR filter design using self-adaptive search equation based artificial bee colony algorithm. *Turkish Journal of Electrical Engineering & Computer Sciences*

Aydın, D., **Yavuz, G. ve Stützle, T. (2017). ABC-X: a generalized, automatically configurable artificial bee colony framework. *Swarm Intelligence*, 11(1), 1-38.

Yavuz, G., & Aydin, D. (2016). Angle modulated artificial bee colony algorithms for feature selection. *Applied Computational Intelligence and Soft Computing*, 7.

Ulusal bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler:

Yavuz, G., Aytekin, S. ve Akçay, M. (2012). Apache hadoop ve dağıtık sistemler üzerindeki rolü. *Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, (027), 43-54,

Yavuz, G., Çakır, H. İ., Aytekin, S. ve Özmen, A. (2010). Gömülü sistem ile akıllı gaz algılayıcısı geliştirme. *Gömülü Sistemler ve Uygulamalar Sempozyumu* (GÖMSİS 2010)

Aytekin S., **Yavuz G**. ve Akçay M. (2010). Dağıtık sistemler-Hadoop, *Başarım'10 II. Ulusal Yüksek Başarımlı ve Grid Hesaplama Konferansı*, İstanbul,

**: Doktora tezinden üretilmiş yayınlar.