T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SÜREKLİ FONKSİYONLARIN OPTİMİZASYONU İÇİN DOĞA ESİNLİ ALGORİTMALARIN GELİŞTİRİLMESİ

Hüseyin HAKLI

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Eylül-2013
KONYA

TEZ KABUL VE ONAYI

Hüseyin HAKLI tarafından hazırlanan "SÜREKLİ FONKSİYONLARIN OPTİMİZASYONU İÇİN DOĞA ESİNLİ ALGORİTMALARIN GELİŞTİRİLMESİ" adlı tez çalışması 02/09/2013 tarihinde aşağıdaki jüri tarafından oy birliği / ~~oy çokluğu~~ ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Mehmet ÇUNKAŞ	.....
Danışman Doç. Dr. Harun UĞUZ	.....
Üye Yrd. Doç. Dr. Ömer K. BAYKAN	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Aşır GENÇ
FBE Müdürü

Bu tez çalışması ÖYP Kurum Koordinatörlüğü tarafından 2013-ÖYP-048 nolu proje ile desteklenmiştir.

TEZ BİLDİRİMİ

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

DECLARATION PAGE

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

Hüseyin HAKLI

02.09.2013

ÖZET

YÜKSEK LİSANS TEZİ

SÜREKLİ FONKSİYONLARIN OPTİMİZASYONU İÇİN DOĞA ESİNLİ ALGORİTMALARIN GELİŞTİRİLMESİ

Hüseyin HAKLI

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Harun UĞUZ

2013, 82 Sayfa

Jüri
Doç. Dr. Harun UĞUZ
Doç. Dr. Mehmet ÇUNKAŞ
Yrd. Doç. Dr. Ömer K. BAYKAN

Son yıllarda hızla gelişen doğa esinli algoritmalar birçok alanda ve mühendislik problemlerinde kullanılmaktadır. Bu algoritmaların başlıca bilinenleri Parçacık Sürü Optimizasyonu (PSO) ve Yapay Arı Kolonisi (ABC) algoritmalarının basitlik ve verimliliklerine rağmen bazı yetersizlikleri ve eksikleri bulunmaktadır. Doğa-esinli algoritmalar için bilinen iki önemli nokta vardır: keşfetme(exploration) ve sömürme(exploitation). Bu algoritmaların genel problemi, keşfetme ve sömürme arasındaki dengeyi kurmaktır. Keşfetme kısmı yeni çözümler bulma ve global arama yeteneği ile ilgiliyken, sömürme kısmı daha iyi çözümler aramak için mevcut bilgi kullanma becerisi ve yerel arama yeteneği olarak bilinir.

PSO algoritması yerel araması başarılıyken global arama yeteneğini zayıf kaldığı için yerel minimumlara takılarak başarısız sonuçlar elde etmektedir. ABC algoritması ise global aramayı etkili bir şekilde yaparken yerel arama konusunda ayrı başarıyı gösterememektedir. Algoritmaların bilinen bu sorunlarını çözebilmek adına Levy uçuşu yöntemi kullanılarak PSO algoritmasının global araması, ABC algoritmasının ise yerel araması iyileştiren LFPSO ve LFABC yöntemleri önerildi. Ayrıca farklı karakteristikteki problemlerdeki başarıyı arttırmak için ABC algoritmasında farklı çözüm arama denklemleri kullanılarak yeni bir yöntem ABCVSS önerildi. Önerilen ABCVSS yöntemi gerçek dünya problemi olan enerji talep tahminine uygulanarak Türkiye'nin 2006-2015 yılları arası enerji talep tahmini yapıldı.

Önerilen algoritmaların başarısını gösterebilmek için, algoritmalar belirlenen test fonksiyonlarında orijinal yöntemler ile birlikte karşılaştırıldı. Deneysel sonuçlar önerilen yöntemlerin orijinal yöntemlere göre çözüm kalitesi ve gürbüzlük açısından çok daha başarılı olduğunu açıkça gösterdi.

Anahtar Kelimeler: Doğa-Esinli Algoritmalar, Enerji Talep Tahmini, Optimizasyon, Parçacık Sürü Optimizasyonu, Yapay Arı Kolonisi,

ABSTRACT

MS THESIS

IMPROVE NATURE INSPIRED ALGORITHM FOR CONTINUOUS FUNCTIONS OPTIMIZATION

Hüseyin HAKLI

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Assoc. Prof. Dr. Harun UĞUZ

2013, 82 Pages

Jury
Assoc. Prof. Dr. Harun UĞUZ
Assoc. Prof. Dr. Mehmet ÇUNKAŞ
Asst. Prof. Dr. Ömer K. BAYKAN

In recent years, the rapidly evolving nature-inspired algorithms are used in many areas and engineering problems. Despite the simplicity and efficiency of mainly known of these algorithms which are Particle Swarm Optimization and Artificial Bee colony, they have some deficiencies and weakness. There are two important points that are known for natüre inspired algorithms: exploration and exploitation. The exploration part is concerned the ability of autonomously seeking for the global optimum, whereas the exploitation part is related to the ability of applying the existing knowledge to look for better solutions.

While PSO algorithm is accomplished at local search, because of the ability of global search of PSO is weak, so PSO has problem of being trapped the local minima and it obtains the unsuccesful results. While the ABC algorithm can perform global search better, it is not good enough in local search. To solve these problems of algorithms, LFPSO and LFABC methods are proposed using the Levy Flight method that improves PSO 's global search and ABC' s local search. Furthermore, to increase the success of different characteristic problems, ABCVSS algorithm is proposed using the variable solution search equations in ABC algorithm. The proposed method ABCVSS is applied the problem of energy demand forecast to estimate Turkey's energy demand between 2006-2015 years.

In order to show success of the proposed algorithms, algorithms are compared to original algorithms at determined benchmark functions. Experimental results show that the proposed algorithms are clearly seen to be more successful than the basic algorithms in terms of solution quality and robustness.

Keywords: Nature-Inspired Algorithm, Estimating Energy Demand, Optimization, Particle Swam Optimization, Artificial Bee Colony

ÖNSÖZ

Bu tez çalışmam da bana ilgi ve anlayış gösteren ve sürekli destek olan danışman hocam Doç. Dr. Harun UĞUZ'a, çalışmalarım sırasında desteklerini, eleştirilerini ve yardımlarını esirgemeyen Yrd. Doç. Dr. Ömer Kaan BAYKAN, Arş. Gör. Mustafa Servet KIRAN hocalarıma ve Selçuk Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nün tüm öğretim elemanlarına teşekkür ederim.

Çalışmalarım sırasında sağlamış oldukları destek için ÖYP Kurum Koordinatörlüğü 'ne ve öğrenim bursu için TUBİTAK-BİDEB 'e teşekkürlerimi sunarım.

Çalışmalarım sırasında her konuda gösterdiği ilgi, anlayış ve bana verdiği destek için eşim Özlem HAKLI 'ya, maddi ve manevi yönden beni her zaman destekleyen üzerimde büyük hakları olan aileme, teşekkür ederim.

Hüseyin HAKLI
KONYA-2013

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

c_1	: Kognitif faktör
c_2	: Sosyal faktör
gbest	: Global en iyi değer
pbest	: Kişisel en iyi değer
rand	: [0,1] aralığında rasgele bir sayı
trial	: Çözüm geliştirememe sayacı
v_{ij}	: i. besin kaynağının j. boyutunun yeni değeri
V_i^t	: t anında i. parçacığın hız değerleri
V_{min}, V_{max}	: Parçacıkların hız limitlerinin minimum ve maksimum değerleri
w	: Atalet ağırlığı
X_i^t	: t anında i.parçacığın pozisyon değerleri
X_{min}, X_{max}	: Arama uzayının minimum ve maksimum değerleri
x_{ij}	: i. besin kaynağının j.boyutunun değeri
β	: Levy indeksi
α	: Levy dağılımda ölçek etmeni
ϕ	: [-1,1] aralığında rasgele belirlenen bir sayı
$\oplus$	: Çoklu çarpım

Kısaltmalar

ABC	: Yapay Arı Kolonisi
ABCVSS	: Değişken Arama Teknikleri ile Yapay Arı Kolonisi
ACO	: Karın Kolonisi Optimizasyonu
D	: Boyut
DE	: Diferansiyel Evrim Algoritmaları
EA	: Evrim Algoritmaları
FA	: Ateşböceği Algoritması
GA	: Genetik Algoritmalar
GSYH	: Gayri Safi Yurtiçi Hasıla
LFABC	: Levy Uçuşu ile Yapay Arı Kolonisi
LFPSO	: Levy Uçuşu ile Parçacık Sürü Optimizasyonu
MTEP	: Milyon Ton Eşdeğer Petrol
PS	: Parçacık Sayısı
PSO	: Parçacık Sürü Optimizasyonu
SA	: Tavlama Benzetim
SN	: Besin Kaynağı Sayısı

1. GİRİŞ

1.1. Teze Giriş ve Amacı

Kelime anlamı olarak "mümkün olan en iyiyi bulma" ve "daha iyiyi yapma" anlamına gelen optimizasyon, genel anlamda eldeki kısıtlı imkanları en optimum şekilde kullanabilmek olarak tanımlanır. Ortakçı (2011) ise optimizasyonu, bir uygunluk (amaç, değerlendirme) fonksiyonuna göre belirli aralıktaki sayısal değerlerin en uygununu seçerek kompleks problemleri çözmek olarak tanımlamıştır. Matematikte ise optimizasyon, belirlenen şartlar altında fonksiyonların minimum ve maksimum değerlerinin tespiti ile ilgilenen bir disiplindir. Her hangi bir alana optimizasyon işlemi uygulanarak maksimum performans ve minimum maliyeti sağlayan çözüm elde edilmeye çalışılır. Örneğin; 100kg şeker pancarında maksimum şeker elde etmeye çalışma işlemi maksimizasyon, 100 kg şekeri minimum şeker pancarından elde etme işlemi ise minimizasyon olarak adlandırılabilir. İki işlem için de amaç maliyeti düşürmek ve verimi artırmaktır.

Optimizasyon metotları temel olarak iki kategori altında toplanabilir. Bunlar klasik optimizasyon metotları ve modern sezgisel metotlardır. Klasik optimizasyon metotları en iyi çözümü garanti ederken, modern sezgisel metotlar ise yaklaşık optimal çözüm üretirler (Akay, 2009).

Değişen dünya ve gelişen teknoloji ortamında ortaya çıkan problemler gün geçtikçe zorlaşmaktadır. Klasik optimizasyon teknikleri problemin boyutunun büyük olması, lineer olmaması, çözüm uzayının geniş olması gibi nedenlerden dolayı yetersiz kalmaktadır (Kıran, 2010). Bu nedenle klasik optimizasyon metotları ile gerçek dünya problemlerini çözmek çok zaman almakta ve etkili bir şekilde çözülememektedir. Bu sorunlar insanoğlunu kesin çözüme değil bulanabilecek en iyi çözüme doğru yöneltmektedir. Böylece sezgisel algoritmalar olarak da adlandırılan doğa-esinli birçok algoritma bulunmuştur. Bu algoritmalar karmaşık ve zor olan gerçek dünya problemleri üzerinde başarılı ve zaman bakımından hızlı sonuçlar vermektedir (Sabat ve ark., 2011).

Sezgisel algoritmalar doğada yaşayan canlıların yaşayış ve davranış biçimlerinden etkilenerek önerildiği için doğa-esinli algoritmalar olarak da adlandırılmaktadır. Karıncalar yem arama davranışında yiyecek kaynağı ile yuvaları arasında feronom adında bir kimyasal madde bırakırlar. Böylece yiyecek kaynağı ile yuva arasındaki en yakın mesafeli yolu tespit edebilirler. Karıncaların yiyecek arama

stratejisinden esinlenerek Dorigo ve ark. (1991; Dorigo ve Caro, 1999) Karınca Kolonisi Algoritmasını (Ant Colony Optimization- ACO) önermiştir. Kennedy ve Eberhart ise 1995 yılında kuş ve balık sürülerinin sosyal davranışlarından etkilenerek Parçacık Sürü Optimizasyonunu (Particle Swarm Optimization-PSO) önermiştir. Diğer bir doğa-esinli algoritma olan Yapay Arı Kolonisi (Artificial Bee Colony – ABC) ise arıların kendi aralarında yaptıkları iş bölümü ve iletişimden etkilenerek oluşturulmuştur (Karaboğa, 2005).

Yukarda bahsedilen 3 doğa-esinli algoritma da popülasyon tabanlı ve sürü zekası temelli optimizasyon metotlarıdır. Sürü zekası adından da anlaşılacağı gibi toplu bir şekilde yapılan bir işbirliği sonucu ortaya çıkan üründür. Sürü içindeki bireylerin birbirleri ile iletişimleri ve bilgi alışverişleri sonucu ortaya çıkan bir kolektif çalışmadır. Yukarıda bahsedilen algoritmaların esinlendiği canlıların hepsinde bu özellik vardır. Örneğin; karıncaların geçtikleri yollara bıraktıkları kimyasal madde ile diğer bireylere bilgi sunması, arıların kendi aralarında iş bölümü yaparak bazılarının besin araması, bazılarının besin kaynaklarının yerini yaptıkları dans ile diğer bireylere tarif etmesi gibi.

Son yıllarda birçok alanda sıklıkla kullanılan doğa-esinli algoritmaların da bazı problemleri bulunmaktadır. Genel olarak en bilinen problem, algoritmaların yerel minimum veya maksimuma takılmasıdır.

Doğa esinli algoritmalar için bilinin iki önemli nokta vardır: keşfetme (exploration) ve sömürme (explotation). Bu algoritmaların diğer genel problemi, keşfetme ve sömürme arasındaki dengeyi kurmaktır. Keşfetme kısmı yeni çözümler bulma ve global arama yeteneği ile ilgiliyken, sömürme kısmı tam tersine daha iyi çözümler aramak için mevcut bilgi kullanma becerisi ve yerel arama yeteneği olarak bilinir (Li ve ark., 2012). Literatürde yapılan çalışmalar ise genel olarak keşfetme ve sömürme arasındaki dengenin kurulması ve algoritmaların eksik yönlerinin güçlendirilmesi üzerine kurulmuştur.

Bu tez çalışmasında PSO ve ABC algoritmalarının üzerine çalışmalar yapıldı ve bu algoritmalar iyileştirilmeye çalışıldı. Ayrıca yapılan bir iyileştirme de gerçek bir dünya problemi olan enerji talep tahmini problemine uygulandı.

1.2. Literatüre Katkısı

Son yıllarda sürü zekası algoritmaları üzerinde birçok çalışma yapılmakta ve günümüz gerçek dünya problemlerine uygulanmaktadır. Bu algoritmalar en iyi çözümü

garanti etmese de en iyiye yakın bir çözümü uygun zaman aralıklarında üretebilmektedir. Bu sayede bir ülkenin enerji talep tahmininin yapılması, bir malzemenin en optimum şekilde kullanılması, sağlık alanında belirli teşhisler konulabilmesi gibi birçok alanda kullanılmaktadır.

Gerçek dünya problemlerinin çok farklı özelliklere sahip olması nedeniyle doğa-esinli bu algoritmalar orijinal halleri ile istenilen çözümlere ulaşamamaktadır. Bu problemleri gidermek için bu algoritmaların zayıf yönleri güçlendirilmeye ve geliştirilmeye çalışıldı. Aynı zamanda algoritmalar hibrit olarak kullanılarak uygulanacak probleme göre şekillendirildi.

Bu tez çalışmasında kullanım alanları çok geniş olan bu algoritmalardan iki tanesi PSO ve ABC algoritmaları üzerinde yapılan iyileştirme ve geliştirmeler bulunmaktadır.

İlk olarak yerel minimumlara takılma ve hızlı yakınsama nedeniyle popülasyon çeşitliliği kaybetme gibi sorunlara sahip PSO algoritmasının (Wang ve ark., 2011), Levy dağılımını sağlayan Levy uçuşu yöntemi ile birlikte kullanılması ile LFPSO algoritması önerildi.

İkinci olarak ise global aramayı iyi yaparken yerel aramadaki yeteneği zayıf olan ABC algoritmasının (Gao ve ark., 2012) kaşif arı aşamasında Levy uçuşu kullanılarak yerel arama başarısı artıran LFABC algoritması önerildi.

Son olarak doğa-esinli algoritmaların çeşitli karakteristikteki fonksiyonlara göre başarısının değiştiği gözlemlendi ve bu sorunu çözebilmek adına ABC algoritmasında farklı çözüm arama teknikleri kullanan ve probleme göre hangi çözüm arama tekniği başarılı ise onu seçen ABCVSS yöntemi önerildi.

Yapılan geliştirmeler literatürde sıklıkla kullanılan test fonksiyonlarında test edilerek orijinal yöntemlerle karşılaştırıldı. Önerilen yöntemlerin elde ettiği sonuçların orijinal yöntemlere göre daha iyi oldukları gözlendi. Ayrıca yeni önerilen yöntemlerden bir tanesi enerji talep tahmini problemine uygulanarak Türkiye'nin gelecek yıllardaki tüketeceği enerji talep miktarı tahmin edilmeye çalışıldı. Yapılan bu geliştirmeler farklı alanlarda farklı problemlere uygulanabileceği gibi diğer algoritmalar ile hibrit olarak da kullanabilir.

1.3. Tezin Organizasyonu

Bu tez çalışması 6 bölümden oluşmaktadır.

Birinci bölümde yapılan çalışma ile ilgili genel bilgiler verilmiş, tanıtılmış, amacı ve önemi anlatılmış ve literatüre katkısından bahsedilmiştir.

İkinci bölümde ise tez çalışmasında kullanılan algoritmalar Parçacık Sürü Optimizasyonu (PSO), Yapay Arı Kolonisi (ABC) ve Levy Uçuşu yöntemi anlatılmıştır. Bu algoritmalar ve yöntemin literatür özetleri de bu bölümde verilmiştir.

Üçüncü bölümde ise Levy uçuşu yöntemi kullanılarak geliştirilen PSO algoritması anlatılmış, iki yöntemin karşılaştırılması yapılarak test sonuçları sunulmuştur. Önerilen yöntem hem orijinal PSO hem de 2011' de Omran tarafından önerilen SPSO ile kıyaslanıp, farklı PSO çeşitleri ile karşılaştırılmıştır.

Dördüncü bölümde ise Levy uçuşu yöntemi kullanılarak geliştirilen ABC algoritması anlatılmış, iki yöntemin karşılaştırılması yapılarak test sonuçları gösterilmiştir.

Beşinci bölümde ise ABC algoritması üzerinde farklı çözüm arama tekniklerini kullanılarak yapılan iyileştirme anlatılmış ve sonuçlar paylaşılmıştır. Ayrıca önerilen yöntem enerji talep tahmini problemine uygulanarak Türkiye'nin 2006-2015 yılları arası enerji talep tahmini yapılmıştır.

Son bölümde sonuçlar özet olarak verilmiş ve ileride yapılacak çalışmalar için önerilere yer verilmiştir.

2. MATERYAL ve METOT

2.1. Parçacık Sürü Optimizasyonu

PSO algoritması ilk olarak 1995 yılında balık ve kuş sürülerinin sosyal davranışlarından etkilenilerek Kennedy ve Eberhart tarafından önerilmiştir (Kennedy ve Eberhart, 1995). 2004 yılında bir konferansta verilen bildiri de Eberhart ve ark., PSO 'nun kuş sürülerinin toplanma senaryosundan esinlendiğini belirtmiştir. Bu senaryo kuşların ilk olarak alana rastgele dağıtılması ile başlar. Alanda tek bir yiyecek bulunmaktadır ve kuşlar yiyeceğin yerini bilmemektedir. Fakat bulundukları pozisyon ile kuşlar yiyeceğe ne kadar uzakta bulunduklarını tespit edebilmektedirler. Yiyeceği bulmak için burada ki en kolay yol, yiyeceğe en yakın olan kuşu takip etmektir. PSO algoritması da bu senaryoyu kullanmaktadır.

Gerçekleştirilmesi kolay metotlar ve göz ardı edilebilecek parametre ayarları yakın zamanda PSO algoritmasını oldukça popüler hale getirmiş ve birçok alanda uygulanmaya başlanmıştır. Bu avantajları nedeniyle PSO, fonksiyon optimizasyonu (Wang ve ark., 2012), filtre tasarımı (filter design) (Chang ve ark., 2009; Sarangi ve ark., 2011), bulanık Oransal-İntegral-Türev (Proportional-integral-derivative -PID) kontrolü (Chiou ve ark., 2012), enerji dağıtım tahmini (Yang ve ark., 2010), özellik seçimi (Chuanga ve ark., 2008; Yang ve ark., 2007), yapay sinir ağları (Cavuslu ve ark., 2012), görüntü bölütleme (image segmentation) (Zhang ve ark, 2011), çizelgeleme problemleri (Tasgetiren ve ark., 2007; Pan ve ark., 2008), mantıksal devre tasarımı (Coello ve ark., 2004), insan titreme analizi (Eberhart ve ark., 1999), biyolojik bilgi çıkarımı, ve diğer bilim ve mühendislik problemleri gibi birçok alanda kullanılmıştır ve kullanılmaya da devam etmektedir.

2.1.1. PSO algoritması

PSO algoritması daha önce bahsedildiği gibi kuş ve balık sürülerindeki bireylerin sosyal davranışlarından etkilenerek önerilmiştir (Kennedy ve Eberhart, 1995). Sürülerde bireyler olarak geçen ifade parçacık olarak adlandırılır ve her parçacık *D*-boyutlu değerlerden oluşur. *D* boyutlu bir durum için *i*. parçacığın pozisyon ve hız ifadeleri aşağıdaki gibi temsil edilir.

$X_i=\{X_{i1},\ X_{i2},\ X_{i3,}\ ...,\ X_{iD}\}$ ve $V_i=\{V_{i1}, V_{i2}, V_{i3,}\ ...,\ V_{iD}\}$

Sürüler arasındaki etkileşim, parçacıklarının her birinin en iyi değeri (*pbest*) ve tüm parçacıkların en iyi değeri (*gbest*) ile sağlanır. *D* boyutlu arama uzayı için *i.* parçacığın $pbest_i=\{P_{i1}, P_{i2}, P_{i3,} ..., P_{iD}\}$ şeklinde, *gbest* ise $gbest=\{G_1, G_2, G_{3,} ..., G_D\}$ şeklinde temsil edilir. PSO güncelleme işlemlerini bu değerlere göre yapacağı için her parçacık için pbest değerleri ve tüm sürü için en iyi değer olan gbest değeri tutulmalıdır. PSO başlangıç ve hesaplama şeklinde iki aşamadan oluşur. Başlangıç aşamasında tüm parçacıklar rastgele olarak, belirlenen sınırlar içindeki arama uzayında Denklem (2.1) kullanılarak dağıtılır.

$$X_{i,d} = X_d^{\min} + rand(0,1)(X_d^{\max} - X_d^{\min}) \tag{2.1}$$

Hesaplama aşamasında ise hız ve pozisyon güncellemeleri yapılır. 1995'de sunulan temel PSO 'nun hız güncelleme denklemi (Kennedy ve Eberhart,1995):

$$V_i^{t+1} = V_i^t + c_1 rand_1\left(pbest_i^t - X_i^t\right) + c_2 rand_2\left(gbest^t - X_i^t\right) \tag{2.2}$$

Denklem (2.2)'de c_1 kognitif (cognitive) ve c_2 sosyal (social) faktörler olarak bilinen değerler hızlandırma katsayıları, $rand_1$ ve $rand_2$ değerleri ise [0,1] aralığında rastgele değişkenlerdir. Genellikle eşit ve 2 olarak belirlenen c_1 ve c_2 değerleri istenilen şekilde ayarlanıp parçacıkların daha çok yerel ya da global çözümden etkilenmesi sağlanabilir.

Her parçacık için belirlenen hız değerlerine parçacıklarda çok büyük değişiklikler olmasını ya da sürekli sınır aşmasını engellemek için V_{max} ve V_{min} parametreleri belirlenebilir.

Shi ve Eberhart tarafından 1998 'de PSO' ya keşfetme (exploration) ve sömürme (exploitation) arasındaki dengeyi kurmak için atalet ağırlığı eklendi**:**

$$V_{i,d}^{t+1} = w^t V_{i,d}^t + c_1 rand_1\left(pbest_{i,d}^t - X_{i,d}^t\right) + c_2 rand_2\left(gbest_d^t - X_{i,d}^t\right) \tag{2.3}$$

Atalet ağırlığı parçacıkların bir önceki hız artışlarının yeni hız değerine etkisini kontrol altına alır ve global arama ve yerel arama arasındaki dengeyi kurmada rol oynar. Genelde atalet ağırlığı için maksimum ve minimum değerler de belirlenmektedir.

Denklem (2.3)' de verilen formülle belirlenen hız güncellemeleri parçacıklara eklenerek yeni değerler elde edilir:

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2.4}$$

Hız güncellemeleri yapıldıktan sonra parçacıkların yeni uygunluk değerleri hesaplanır ve gerekli olduğu takdirde *pbest* ve *gbest* güncellemeleri gerçekleştirilerek durdurma kriteri sağlanana kadar aynı işleme devam edilir. PSO algoritmasının sözde (pseudo) kodu aşağıda verilmiştir.

1. Başlangıç parametrelerini (*PS, D,* X_{min}, X_{max}, c_1, c_2, V_{min}, V_{max}) ayarla.
2. *i=1,2,...,PS* , *d=1,2,...,D* olmak üzere parçacıkların başlangıç değerleri Denklem (2.1)' de gibi rasgele şekilde arama uzayında belirle.
3. Her bir parçacık için uygunluk değerlerini belirle ve parçacıkların *pbest* değerlerine ata.
4. *gbest=min(pbest)* (Parçacıkların *pbest*'lerinin en iyisi *gbest* olarak ata)
5. while(sonlandırma kriteri)
6. for *i=1:PS*
   Denklem (2.3) kullanarak parçacıkların hız güncellemesi yap(V_i)
   Denklem (2.4) kullanarak parçacıkların pozisyon güncellemesini yap(X_i)
   Belirlenen yeni parçacık için uygunluk değeri hesapla
   if($fitness_{new}$>$pbestvalue_i$) (Yeni parçacığın çözüm kendi pbestinden iyi ise)
   $pbest_i = X_{new}$
   end //if
   end //for
7. Yeni belirlenen parçacıklar için gbest kontrolü yap
   temp=min(*pbest*)
   if(temp<*gbest*)
   gbest=temp
   end //if
8. end //while

2.1.2. Temel PSO Bileşenleri

PSO' da sürekli bahsedilen ve PSO Algoritması başlığında da değinilen bileşenler biraz daha ayrıntılı olarak anlatılmıştır. Daha detaylı bilgiler için şu tez çalışmalarına bakılabilir (Güner, 2006; Ortakçı, 2011).

2.1.2.1. Başlangıç değerleri

Temel PSO algoritmasında parçacıklar kendileri için belirlenen arama uzayında rasgele bir biçimde dağıtılmaktadır. N sayıda parçacık olduğunu ve boyut sayısının 4 olduğunu kabul edelim. Parçacıkların durumu Çizelge 2.1' deki gibi olacaktır. Sayı değerleri örnek olması açısından rasgele yazılmıştır.

Çizelge 2.1 N parçacık için popülasyon başlangıcı

	1	2	3	4
1.Parçacık	5	1	8	9
2.Parçacık	-1	7	5	-3
3.Parçaçık	4	8	-2	10
….	…	…	…	…
N.Parçacık	2	9	-4	1

2.1.2.2. Uygunluk değerinin hesaplanması

Her bir parçacık, verilen problem için birer aday çözüm olarak adlandırılmaktadır. Parçacıkların hangisinin amaç fonksiyonu için en iyi çözüm olduğunu belirlemek ise verilen amaç fonksiyonun uygunluk değerlerini hesaplamak ile bulunur. Minimizasyon işlemi için en küçük değeri veren parçacık, maksimizasyon işlemi için ise en büyük değeri veren parçacık verilen problem için en uygun aday çözümdür. Uygunluk değeri parçacıkların çözümlerinin kalitesini belirler.

2.1.2.3. Kişisel en iyi değeri

Her bir parçacık yapılan iterasyonlar boyunca o ana kadar bulunan en iyi konum değerini tutar ve buna kişisel en iyi değeri (*pbest*) denir. Her bir parçacık her yeni iterasyon boyunca bulunan yeni konum değeri ile o ana kadar bulunan en iyi konum değeri (*pbest*) kıyaslanır ve yeni konum değeri daha iyi bir sonuç veriyorsa kişisel en iyi

değeri bu yeni konum değeri olarak atanır. Hız güncelleme işleminde kullanılan *pbest* böylece parçacığı kendi en iyi değerine yaklaştırmaya çalışır.

2.1.2.4. Global en iyi değer

Sürüdeki tüm parçacıkların iterasyonlar boyunca o ana kadar bulduğu en iyi konum değerine ise global en iyi değer (*gbest*) denir. Her yeni iterasyonda bulunan uygunluk değeri *gbest* ile karşılaştırılır eğer daha iyi uygunluk değerine sahipse yeni *gbest* değeri olarak belirlenir. Hız güncelleme işleminde kullanılan *gbest* değeri tüm parçacıkları *gbest* değerine yaklaştırmaya çalışır.

2.1.2.5. Hız değeri

Parçacıkların bir sonraki konumlarını belirleyen ve parçacığın çözüm uzayında arama yapmasını sağlayan en önemli etken hız değeridir. Hız değerleri pozitif ve negatif değerler alıp parçacıkların çözüm uzayında çok yönlü hareket ederek arama yapmasını sağlarlar (Ortakçı, 2011). Hız değerinin yüksek ya da düşük belirlenmesi parçacıkların arama uzayında nasıl bir arama yapacağını direk olarak etkilemektedir. Hız değerlerinin yüksek değişimleri algoritma da tutarsızlığa ve bulduğu iyi çözümleri kaybetmesine sebep olabilir. Bu nedenle bu durumu kontrol altına alabilmek için hız değeri için minimum ve maksimum değerler belirlenmekte ve hız değişimi bu sınırlar içinde yapılmaktadır.

Hız değeri bizim belirlediğimiz V_{min} ve V_{max} değerlerini aştığı takdirde bu sınırlara çekilmektedir. V_{min} ve V_{max} değerlerinin belirlenmesi çok önemlidir. Eğer bu değerler yüksek belirlenirse global arama daha iyi yapılırken, tersi küçük değerler seçildiklerinde ise yerel arama daha iyi yapılır. Çok büyük ve çok küçük değerler de ise arama uzayı etkili kullanılamaz ve bölgesel minimumlara takılmalar meydana gelir (Güner, 2006).

2.1.2.6. Atalet ağırlığı

1995' te sunulan ilk PSO 'da atalet ağırlığı bulunmamaktaydı. Daha sonra Shi ve Eberhart (1998) PSO algoritmasına tarafından parçacıkların bir önceki hız artışlarının yeni hız değerine etkisini kontrol altına almak ve global arama ve yerel arama

arasındaki dengeyi kurmak için atalet ağırlığını ekledi. Denklem (2.3)' de gösterildiği gibi bir önceki hız değerinin başına bir çarpan olarak gelen atalet ağırlığı ile PSO algoritmasında büyük bir gelişme meydana gelmiştir. Atalet ağırlığının büyük değerleri, daha genel bir arama yapmasını sağlarken, küçük değerler alması ise daha yerel aramalar yapması sağlanır. Buradan hareketle parçacıkların iterasyon başlarında daha fazla global arama, sonlarına doğru ise daha fazla yerel arama yapılması istenir.

Literatürde farklı şekilde kullanımları olan atalet ağırlığı genellikle, 0.8-1 aralığında bir değerden başlatılıp iterasyon sonlarına doğru 0.2-0.4 aralığındaki değerlere düşürülür.

2.1.2.7. Hızlandırma katsayıları

Hızlandırma katsayılarından c_1 parçacıkların pbest değerine, c_2 ise gbest değerlerine doğru çekilmesini kontrol eden katsayılardır. Hızlandırma katsayılarının büyük değerler alması parçacıkların birbirinden uzaklaşıp ayrılmalarına sebep olurken, küçük değerler alması parçacıkların hareketlerinin kısıtlanmasına ve çözüm uzayının yeterince taranamamasına sebep olur (Ortakçı, 2011). Hızlandırma katsayıları genel olarak eşit ve 2 olarak belirlenir. Katsayılar eşit olarak belirlenmek zorunda değildir, eşit belirlenmesinin sebebi hem gbest hem de pbest den aynı oranda etkilenmesi sağlamak içindir.

2.1.2.8. Sonlandırma kriteri

Sonlandırma kriteri olarak farklı kıstaslar göz önüne alınabilir. Genellikle algoritma belirlenen iterasyon sayısına kadar çalıştırılır ya da amaç fonksiyonu istenen değerin altına düştüğünde veya üstüne çıktığında algoritma sonlandırılabilir.

2.1.3. Genel görüş ve literatür taraması

PSO algoritması çok basit ve kullanışlı olmasının yanı sıra bazı problemlere sahiptir. Genel olarak PSO algoritmasının global arama yeteneğini zayıftır ve yerel minimumlara takılır. Belirtildiği gibi PSO hem yerel hem global durumlardan etkilenerek hız değişimi yapsa da belli bir iterasyon sonrasında parçacıkların birbirine

benzemesi nedeniyle hız değişimleri çok küçük miktarlara düşerek global arama yeteneğini kaybetmesine sebep olur. Bu da PSO ’nun en büyük problemlerinden biri olan yerel minimumlara takılma olasılığını artırır. Bu sorunu engellemek için literatür de birçok çalışma vardır.

J.Liang ve ark. (2006), tüm parçacıkların en iyi bilgilerinin parçacıkların hız güncellemesi için kullanıldığı yeni bir öğrenme stratejisini kullanan kapsamlı öğrenen parçacık sürü optimizasyonunu (CLPSO) önermiştir. Bu strateji PSO algoritmasının erken yakınsama problemini gidermeyi sağlayan popülasyon çeşitliliğine izin verdi. Orijinal PSO’ daki parçacıkların *pbest* ve *gbest* ‘ten öğrenmesi yerine hız güncellemesini tüm parçacıkların *pbesti* veya farklı parçacıkların *pbest* ‘lerini kullanarak ve bunlarından birinin seçimini rastgele yaparak sürüyü çeşitlendirmiştir.

Zhao Xinchao (2010) ise çeşitliliğin kaybolmasını engellemek için hız güncelleme işleminde değişiklik yapmış ve karışık (perturbed) *gbest* güncelleme stratejisini kullanarak karışık parçacık sürü algoritmasını (perturbed particle swarm algorithm) önermiştir.

Diğer bir çalışmada 4 farklı arama stratejisi kullanılarak, hangisinde daha iyi güncelleme yapılıyorsa algoritmanın o stratejiyi kullanmasına devam edilmesi sağlanarak kendi kendine uyarlamalı öğrenme tabanlı bir algoritma önerilmiştir (Wang ve ark., 2011). Böylece farklı türdeki problemler üzerinde tek bir algoritma ile başarılı sonuçlar elde edilmiştir.

Ioannis G. Tsoulos ve ark. (2010) PSO ’nun hız ve etkisini artırmak için durdurma kriteri, benzerlik kontrolü ve bazı bölgesel arama metotlarının şartlı uygulaması modifikasyonlarını eklemiştir**.**

Mendes ve ark. (2004) farklı komşuluk topolojileri ile PSO algoritmasındaki hız güncelleme işlemini sadece en iyi komşu parçacığı kullanmak yerine tüm komşu parçacıkların verilerini kullanarak gerçekleştirmiş ve tamamen bilgili PSO ‘yu (Fully Informed - FIPSO) önermiştir.

Nickabadi ve ark. (2011) PSO’ daki atalet ağırlığını sabit, zamanla değişen ve uyarlamalı olmak üzere incelemiş, parçacıkların arama uzayındaki yerini belirlemek için bir geri bildirim parametresi olarak sürünün başarı oranını kullanmış ve yeni uyarlamalı bir atalet ağırlığı önermiştir.

Zhou ve ark. (2011) PSO algoritmasının kolayca lokal minimumlara takılması ve mutasyon parametresi eksikliği ve sürü çeşitliliğin az olması sebebiyle, tavlama

benzetim (Simulated Annealing - SA) algoritmasından esinlenerek PSO algoritmasına rassal faktörler eklemiştir.

Ratnaweera ve ark. (2004), zamanla değişen hızlandırma katsayılarını ve atalet ağırlığını kullanarak kendi kendini organize eden hiyerarşik parçacık sürü optimizasyonunu (HPSO-TVAC) önermişlerdir. Bu sistemle sadece PSO'nun sosyal ve kognitif kısımları dikkate alınarak (yani önceki hız değerleri kullanılmayarak), her parçacığın hız değişimleri kontrol edilmiş bu hız değişimlerinde duraksama olduğu belirlendiğinde ise parçacığın arama uzayında yeniden dağıtılması sağlanmıştır. Bu da algoritmanın popülasyonundaki çeşitliliği koruyarak yerel minimumlara takılmasını engellemiştir.

Lovbjerg ve ark. (2001), sürü alt popülasyonlara bölünmüş ve bir üreme operatörü sürünün çeşitliliğini artırmak için alt popülasyon içinde veya alt popülasyonların arasında kullanılmıştır.

Van den Bergh ve ark. (2004), her boyutta ayrı ayrı arama yapmak için tek-boyutlu sürüler kullanmış olmalarına rağmen, bu aramaların sonuçları global bir sürü olarak birleştirilerek orijinal PSO'nun çokmodlu (multimodal) fonksiyonlar da performansını önemli ölçüde arttıran kooperatif parçacık sürü optimizasyonunu (CPSO-H) önermiştir.

Sabat ve ark. (2011), PSO' nun yakınsama ve çözüm kalitesini artırmak için birbirinden uzaklaşan parçacıkları bularak onları optimal çözüme doğru hızlandıran ve PSO 'nun öğrenme stratejisi değiştiren tümleşik PSO algoritmasını (ILPSO) sunmuşlardır.

Hız güncelleme değişimlerinden farklı olarak, bölgesel aramayı iyi yapan PSO diğer doğa esinli algoritmalarla hibrit olarak kullanılmış böylece hem global aramayı hem de yerel aramayı iyi yapan hibrit algoritmalar önerilmiştir (Shi ve ark., 2005; Wang ve ark., 2012; Chen ve ark., 2010; Kıran ve ark., 2012a,). Daha birçok PSO çeşitleri PSO 'nun erken yakınsama problemini çözmek için önerilmiştir.

2.2. Yapay Arı Kolonisi

Yapay Arı Kolonisi algoritması 2005 yılında arı kolonilerinin yiyecek arama ve birbiri ile etkileşimlerinden etkilenerek Karaboğa tarafından önerildi. Karaboğa (2005) yayınladığı teknik raporunda arı kolonilerin yiyecek kaynaklarını bulma, değerlendirme, birbirleri ile bilgi alışverişinde bulunma vs. özelliklerini inceleyip bazı kabuller alarak

algoritmayı gerçekleştirdi. Sürü zekası algoritmalarının en bilinenlerinden biri olan ABC algoritması özellikle arıların birbiri arasında ki iş bölümüne odaklanarak yaptıkları görev paylaşımı ve bilgi alışverişini kullanır. Bu işlemi de herhangi bir merkezi otorite olmadan yaptıkları için kendi kendilerine organize olabilmektedirler (Akay, 2009). Algoritma senaryosunda, besin kaynaklarına giden arılar bu besin kaynakları hakkında bilgileri kovanda sürüye yaptıkları dans ile iletir, bu dansı takip eden arılar ise kaynağın yeri ve kalitesi hakkında bilgi aldıktan sonra kendi seçtikleri kaynağa yönelirler, eğer besin kaynağı tükenirse bu arılar rasgele bir şekilde besin kaynağı aramaya devam ederler. ABC algoritması bu senaryoyu kullanmaktadır.

Diğer sürü zekası algoritmalarına göre farklı problemlerde daha etkin ve verimli sonuçlar veren ABC algoritması birçok mühendislik ve gerçek dünya problemine uygulanmaktadır. Ayrıca basitliği ve verimliliği de göz önüne alındığında ABC, fonksiyon optimizasyonu (Xiang ve An, 2013; Kang ve ark., 2013), araç rotalama problemi (Szeto ve ark.,2011), veri kümeleme (Yan ve ark, 2012), görüntü işleme ve bölütleme (Yu ve Duan,2012; Horng, 2011), elektrik yükü problemi (Hong, 2011), çizelgeleme problemleri (Tokmak, 2011; Kıran, 2010), gezgin satıcı problemi (Akça, 2011), ekonomik güç dağıtım problemi (Basu, 2013) ve birçok mühendislik probleminde kullanılmaktadır.

2.2.1. ABC Algoritması

ABC algoritmasını tam olarak açıklayabilmek için ilk olarak gerçek arı kolonilerinin yiyecek arama yöntemi hakkında bilgi vermek daha iyi olacaktır. Akay (2009) tez çalışmasında, Tereshko'nun (2000) arı kolonilerinin yiyecek arama da ki 3 temel bileşenini şu şekilde açıklamıştır;

i) *Yiyecek kaynakları*; Arıların polen, bal ve nektar elde etmek için gittikleri kaynaklardır.
ii) *Görevi belirli işçi arılar*; Bu arılar daha önceden keşfedilen kaynaklara ait nektarın kovana getirilmesinden sorumludurlar. Ayrıca besin kaynağı hakkında bilgiyi sürü içindeki diğer arılarla paylaşırlar.
iii) *Görevi belirsiz işçi arılar*; Bu arılar ise yiyecek kaynağı arayışı içindedirler. Sürü içinde besin kaynağı hakkında verilen bilgiyi alarak besin kaynağına

yönelen gözcü arılar, sadece kendi içsel dürtü ve deneyimleri ile besin kaynaklarına yönelen kaşif arılar olmak üzere iki çeşittir.

Bir arı kolonisinde en başında, potansiyel bir araştırmacı arı ilk olarak görevi belirsiz şekilde nitelendirilebilir. Bu görevi belirsiz arı için iki seçenek vardır (Karaboğa ve Baştürk, 2007):

i) Kendi içgüdüsü veya daha önceki deneyimlere göre kovan etrafında rasgele bir şekilde besin kaynağı arayarak kaşif arı (S) olabilir.

ii) Diğer arıların buldukları besin kaynakları hakkında yapacakları dansı izleyerek gideceği besin kaynağını seçen gözcü arı (R) olabilir.

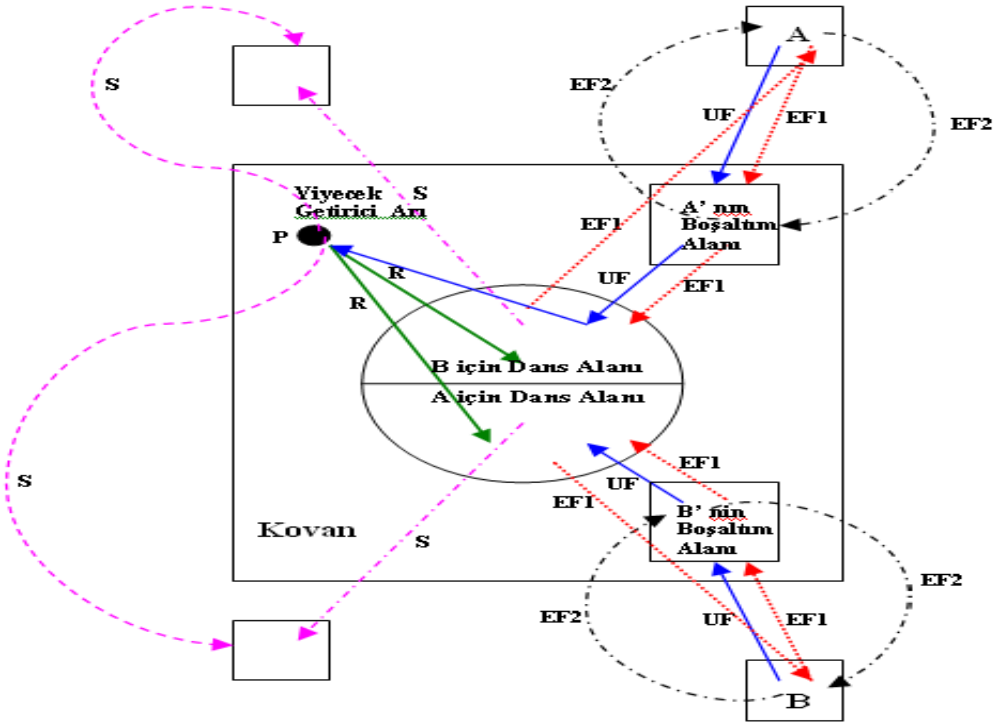
Bu iki seçimden birini yaptıktan sonra belirlenen besin kaynağında giden arı, besin kaynağında aldığı nektar ile tekrar yuvaya döner. Bunun sonucunda artık işçi arı olacaktır. Bu aşamadan sonra işçi arı için 3 seçenek vardır (Karaboğa, 2005):

i) Gittiği kaynağı bırakarak bağımsız izleyici olabilir(UF).

ii) Gittiği kaynağa dönmeden önce dans eder ve arkadaşlarını o kaynağa doğru yönlendirebilir(EF1).

iii) Diğer arıları yönlendirmeden kaynağa gidebilir(EF2).

Yukarda belirtilen açıklamalar Şekil 2.1.' de görülebilir.



Şekil 2.1. Yiyecek Arama Çevrimi (Akay,2009)

Gerçek arı kolonilerinin yiyecek arama davranışlarını yapay arı kolonisi algoritmasına uygularken bazı kabuller yapılmıştır (Karaboğa, 2005). Her bir besin kaynağı için yalnızca bir işçi arı bulunmaktadır. Yani işçi arıların sayısı besin kaynağı sayısına eşittir. Ayrıca işçi arıların sayısı gözcü arılarına sayısına eşittir. Kolonini yarısını işçi arılar oluştururken diğer yarısını da gözcü arılar oluşturur. Besin kaynağı tükenen veya besin kaynağından vazgeçen işçi arılar kaşif arı olur. Besin kaynakları problemin olası çözümünü temsil ederken, besin kaynaklarında nektar miktarı ise çözüm kalitesini gösterir.

ABC algoritması ilk olarak kaşif arılara yiyecek için aranacak besin kaynaklarının rastgele şekilde dağıtılmasıyla başlar. Başlangıçta her bir kaşif arıya (2.5)' de gösterildiği gibi belirlenen arama uzayı içerisinde bir besin kaynağı atanır. Besin kaynağında ki nektarı toplayıp kovana dönen bu arılar görevli arı olurlar. Her bir çözüm *D* boyutlu değerlerden oluşur.

$$x_{ij} = x_j^{\min} + rand(0,1)(x_j^{\max} - x_j^{\min}) \tag{2.5}$$

x_{ij} *i.* işçi arının *j.* boyutunu, $x_j^{\max}$ ve $x_j^{\min}$ ise arama uzayının minimum ve maksimum sınırını belirtir. Böylece Denklem (2.5) ile üretilen besin kaynakları belirlenen arama uzayı sınırları içinde kalır. $i \in 1...SN$, $j \in 1...D$. *SN* yiyecek kaynaklarının sayısını yani koloninin yarısını, *D* ise optimize edilecek olan problemin boyut sayısını ifade eder. Başlangıçta tüm besin kaynakları için *çözüm geliştirememe sayacı (trial)* sıfırlanır.

Başlangıç aşamasından sonra ABC algoritmasını 3 aşamada inceleyebiliriz: işçi arılar, gözcü arılar ve kaşif arılar. Besin kaynakları belirlendikten sonra kovana dönen işçi arılar, bu besin kaynağının kalitesini hafızasında tutar ve besin kaynağının komşuluğunda yeni bir besin kaynağı belirler. Belirlediği yeni besin kaynağının kalitesi, hafızasında tuttuğu besin kaynağından daha iyi ise aç-gözlü seçim algoritmasını kullanarak yeni olanı seçer ve eski kaynağı unutur. İşçi arı yeni besin kaynağı belirleme işlemini Denklem (2.6) 'da belirtildiği gibi yapmaktadır.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{2.6}$$

v_{ij}, x_i besin kaynağının rasgele olarak belirlenen *j*. boyutunun değiştirilmesi ile elde edilen *j*. boyutun yeni değeridir. Böylece x_i komşuluğundan yeni bir besin kaynağı olan v_i belirlenir. x_{kj} ise rasgele olarak belirlenen $k \in \{1,2,3...,SN\}$ komşusunun *j*. boyutunu temsil etmektedir. ϕ_{ij} ise [-1,1] aralığında rasgele belirlenen besin kaynağının her iki yöne de ilerlemesini sağlayan bir sayıdır. Yeniden belirlenen boyut değerinin daha önceden belirlenmiş arama uzayının içinde kalıp kalmadığı kontrol edilerek, eğer sınır değerleri aşmışsa Denklem (2.7) 'de verildiği gibi sınır değerlere çekilir.

$$v_{ij} = \begin{cases} x_j^{\min}, v_{ij} < x_j^{\min} \\ v_{ij}, \quad x_j^{\min} \le v_{ij} \le x_j^{\max} \\ x_j^{\max}, v_{ij} > x_j^{\max} \end{cases} \tag{2.7}$$

Yeni belirlenen besin kaynağının v_i çözüm kalitesi, bir önceki besin kaynağı x_i 'den daha iyi ise işçi arı eski besin kaynağını bırakarak yiyecek arama işlemi için yeni besin kaynağına gider ve çözüm geliştirememe sayıcını sıfırlar. Eğer eski besin kaynağının (x_i) çözüm kalitesi daha iyi ise besin kaynağında herhangi bir değişiklik yapmaz ve bu kaynak için çözüm geliştirememe sayacı 1 artırılır. Besin kaynağının çözüm kalitesini belirlemek için kullanılan uygunluk değeri (2.8)' de verilen denklem ile hesaplanır.

$$fitness_i = \begin{cases} 1/(1+f_i) & f_i \ge 0 \\ 1+abs(f_i) & f_i < 0 \end{cases} \tag{2.8}$$

f_i çözümün amaç fonksiyonu (objective function) sonucunda üretilen değeri belirtirken, $fitness_i$ değeri de o çözümün normalize edilmiş uygunluk değeridir ve çözüm kalitesini gösterir (Karaboğa, 2005; Özdemir, 2012).

İşçi arıların bulundukları besin kaynaklarının çözüm kaliteleri de hesaplandıktan sonra kovana dönerek buldukları besin kaynakları hakkında bilgileri paylaşırlar. Böylece gözcü arıların aşamasına geçilir. Bu aşama da gözcü arılar, işçi arılardan gelen bilgiye bakarak besin kaynaklarından hangisini seçeceğini belirler. ABC algoritmasında

gözcü arıların hangi besin kaynaklarına gideceklerini belirleme de rulet tekerleği seçimi kullanılır. İlk olarak işçi arılardan elde edilen besin kaynaklarının kalitelerinin bilgisi alınarak her bir besin kaynağı için çözüm kalitesine göre bir olasılık oluşturulur. Çözüm kalitesi yüksek olan besin kaynaklarının seçilme olasılığı yüksek olurken, düşük olan kaynakların ise gözcü arılar tarafından seçilme olasılığı azdır. Olasılıkları hesaplamak için (2.9)‘ da ki denklem kullanılmaktadır.

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{2.9}$$

Burada p_i *i.* yiyecek kaynağının seçilme olasılığını, $fitness_i$ *i.* kaynağın uygunluk değerini yani çözüm kalitesini gösterir. Her bir besin kaynağının seçilme olasılığı, o kaynağın uygunluk değerinin tüm kaynakların uygunluk değerlerinin toplamına bölünmesi ile hesaplanır.

Her bir besin kaynağı için olasılıklar belirlendikten sonra, her bir gözcü arı için [0,1] arasında rasgele bir sayı üretilir. Gözcü arı belirlenen bu rasgele sayı ile Denklem (2.9) ’da belirlenen besin kaynaklarının olasılıklarını sırasıyla karşılaştırır. Eğer belirlenen sayı olasılık değerinden büyük ise besin kaynağı seçilmez ve diğer besin kaynağına geçilir. Bu şekilde olasılık değeri rasgele sayıdan büyük olan besin kaynağı bulunana kadar işlem devam eder. Gözcü arı için besin kaynağı seçildikten sonra aynı işçi arılarda olduğu gibi Denklem (2.6) ‘daki gibi bu besin kaynağına komşu bir kaynak seçilir. Seçilen bu besin kaynağının çözüm kalitesi eski çözümden iyi ise eski çözüm terk edilerek gözcü arı yeni besin kaynağını kullanır ve geliştirememe sayacı sıfırlanır. Tam tersi durumda ise geliştirememe sayacı bir artırılır ve gözcü arı eski çözümünü hafızada tutmaya devam eder.

İşçi ve gözcü arıların aşamaları bittikten sonra, kaşif arıların aşamasına geçilir. Bu aşamada ilk olarak çözüm geliştirememe sayaçları (trial) kontrol edilir. Bu kontrol ile besin kaynaklarının terk edilip edilmeyeceği belirlenir. Algoritma başında belirlenecek limit değeri kadar bir besin kaynağında geliştirme yapılmamışsa o besin kaynağı terkedilir ve besin kaynağındaki işçi arı kaşif arı olur. Bu kaşif arı da (2.5) denklemine göre arama uzayında kendine yeni bir besin kaynağı belirler. Temel ABC

algoritmasında her iterasyonda yalnız bir kaşif arının çıkmasına izin verilir (Akay, 2009). Belirtilen durdurma kriteri sağlanana kadar bu 3 aşama tekrarlanır.

ABC algoritmasının sözde(pseudo) kodu aşağıda verilmiştir.

1. Başlangıç parametreleri(*limit* , $x_j^{\max}$ ve $x_j^{\min}$ vb.) ayarla.
2. *i=1,2,...,SN , j =1,2,...,D* olmak üzere besin kaynaklarının başlangıç değerleri Denklem (2.5)' de gibi rasgele şekilde arama uzayında belirle ve çözüm geliştirememe sayaçları sıfırla.
3. Her bir çözüm için amaç fonksiyonundaki değerleri olan f_i hesapla ve Denklem (2.8) ile çözüm kalitesini temsil eden $fitness_i$ değerleri belirle
4. while(sonlandırma kriteri)
5. for *i=1:SN*

   İşçi arılar için Denklem (2.6) ile yeni besin kaynaklarını belirle.

   if(Yeni besin kaynağının çözüm kalitesi > Eski besin kaynağın çözüm kalitesi)

   Eski besin kaynağını hafızadan silerek yerine yeni besin kaynağını seç

   Geliştirememe sayacını sıfırla

   else

   Geliştirememe sayacını bir artır

   end //if

   end //for
6. for i=1:SN

   İşçi arılardan besin kaynakları hakkında gelen bilgi ile Denklem (2.9) ile kaynakların p_i olasılıklarını belirle

   end //for
7. for *i=1:SN*

   if(rand<pi) (Olasılığı rasgele belirlenen sayıdan büyük olan besin kaynağı seç)

   Gözcü arılar için Denklem (2.6) ile yeni besin kaynaklarını belirle.

   if(Yeni besin kaynağının çözüm kalitesi > Eski besin kaynağın çözüm kalitesi)

   Eski besin kaynağını hafızadan silerek yerine yeni besin

kaynağını seç

Geliştirememe sayacını sıfırla

else

Geliştirememe sayacını bir artır

end //if

8. if (max(trial)>limit) (Geliştirememe sayacı limit değerinden büyük olan işçi arıyı seç)
   Denklem (2.5)'e göre kaşif arı için yeni besin kaynağı belirle.
   end//if
9. En iyi çözümü hafızada tut
10. end // while

Genel olarak iki algoritmayı da anlattıktan sonra ABC ve PSO algoritmaları arasında bir kaç farklılıktan söz edebiliriz.

i) İlk olarak PSO algoritması parçacıkların tüm boyutları üzerinde güncelleme işlemi yaparken, ABC algoritması rasgele belirlediği herhangi bir boyut üzerinde değişiklik yaparak yeni çözümler üretir.
ii) Diğer bir farklılık ise PSO algoritması amaç fonksiyonunda elde ettiği değerleri kullanarak çözüm kıyaslaması yaparken, ABC algoritması bu amaç fonksiyonunda gelen değeri doğrudan kullanmayıp (2.8)'de verilen denklem ile elde edilen uygunluk değerini kullanmaktadır.
iii) Son olarak ise PSO algoritması daha iyi bir çözüm elde etse de etmese de parçacıkların güncellemesini yapar, bunun sebebi ise PSO algoritması parçacıkların o zaman kadar ki elde ettiği en iyi çözümleri pbest parametrelerinde tutar. ABC algoritması ise aç gözlü seçimi kullanarak eğer elde edilen yeni çözüm daha iyi ise eskisini unutur, yeni çözümü kullanır. Tersi durumda ise herhangi bir değişiklik yapmadan eski çözümü kullanmaya devam eder.

2.2.2. Genel görüş ve literatür taraması

ABC algoritması birçok sayısal test fonksiyonunda diğer algoritmalara göre daha etkili olmasına rağmen, ABC algoritmasının bazı yetersizlikleri bulunmaktadır. ABC keşfetme için iyi olmasına rağmen, bulduğu sonuçları değerlendirme anlamına gelen sömürme için yeterince iyi değildir. Diğer bir deyişle, ABC algoritması global aramayı daha iyi yapabilirken, yerel arama için zayıf kalmaktadır. PSO algoritmasın da ise tersi bir durumdan bahsetmiştik. Doğa-esinli bu algoritmalar için hangi arama yönüne ağırlık verdiği ya da bu dengeyi nasıl kurduğu önemlidir. Bu denge bize algoritmanın zayıflığı ya da güçlü yönleri hakkında bilgi verir. Literatürde yapılan çalışmalar da genellikle algoritmaların zayıf yönlerini kuvvetlendirmeye yöneliktir.

ABC algoritması diğer algoritmalara göre bu iki ifade arasındaki dengeyi daha iyi kurmaktadır. Fakat buna rağmen yerel arama konusunda ABC algoritmasının güçlendirilmesi daha verimli sonuçlar elde edilmesini sağlayabilir. Bu düşünce ile literatür de ABC algoritmasını iyileştirmek için çeşitli yöntemler kullanılmış, özellikle yerel arama kısmı güçlendirilmeye çalışılmıştır.

Zhu ve Kwong (2010), ABC algoritmasının sömürme kısmını geliştirmek için PSO algoritmasından esinlenerek, çözüm arama denklemine en iyi çözüm bilgisini (gbest) dahil etmişlerdir. Böylelikle PSO algoritmasının tüm parçacıkların için kullandığı gbest değerini kullanarak ABC algoritmasının yerel aramasını kuvvetlendirildi.

Alatas (2010), parametre ayarlaması, sürünün başlangıçta dağıtılması ve işçi arıların keşfe çıkma işlemleri için kaotik haritaları kullanan kaotik ABC algoritmasını (CABC) önerdi. Böylece ABC algoritmasının yakınsama karakteristiğini geliştirmiş ve yerel minimumlardan kaçınılmasını sağlandı.

Xiang ve An (2013), ABC algoritmasının yakınsama performansını artırmak için etkin ve gürbüz yapay arı kolonisi (ERABC) algoritmasını sunmuşlardır. ERABC algoritmasında, arama işlemini hızlandırmak için kombinatoryal çözüm arama denklemi tanıtıldı. Yerel minimumlara takılmayı engellemek için kaşif arılar aşamasında kaotik arama teknikleri kullanıldı ve ayrıca popülasyon çeşitliliğini koruyabilmek amacıyla rulet tekerleği tabanlı ters seçim uygulandı.

Gao ve ark. (2012) , diferansiyel evrim (DE) algoritmasından esinlenerek, ABC algoritmasının sömürme kısmındaki zayıflığını güçlendirmek için her arı için bir önceki iterasyonun en iyi çözümü etrafında arama yapan ABC/Best algoritmasını önerdi.

Ayrıca popülasyon başlangıç dağıtımı ve kaşif arıların üretilmesin kaotik sistemleri ve karşı-tabanlı öğrenme tekniklerini kullandı.

Li ve ark. (2012), PSO algoritmasında ki gibi atalet ağırlığı, hızlandırma katsayıları ve en iyi çözüm değerlerini kullanarak yeni bir çözüm arama denklemi sundu. Daha sonra bu çözüm arama tekniği, orijinal ABC ve GABC 'nin çözüm arama tekniklerini paralel olarak kullanıp, hangi çözüm tekniği daha iyi bir sonuç elde ettiyse o çözüm aday çözüm olarak seçildi. Böylece bu 3 teknik birleşince çeşitli optimizasyon problemi için algoritmaya farklı avantajlar sağladı.

Kang ve ark. (2011), Rosenbrock rotasyonal yön yöntemi ile ABC algoritmasını kombine etti ve RABC 'yi sundu. RABC 'de, keşfetme aşaması aynı ABC algoritmasındaki gibi uygulanırken, sömürme aşamasında Rosenbrok rotasyonal yön yöntemi kullanıldı.

Banharnsakun ve ark. (2011), gözcü arılar için best-so-far (şimdiye kadar en iyi) seçimini kullanarak ABC algoritmasının yakınsama yeteneğini geliştirdi ve sayısal test fonksiyonları ve görüntü kaydetme işlemi üzerinde yöntem test edildi.

ABC algoritması üzerinde bir çok çalışması olan Gao, Liu ile birlikte diferansiyel evrim algoritmasından esinlenerek ABC/Best 'i önerdi. Bu çalışmayı daha önceden yapmış olan Gao ve ark. (2012), bu çalışmasında ABC/Best ' in algoritmanın keşfetme yeteneğini zayıflattığını ve yerel minimumlara takılmasına sebep olduğu belirtti. Orijinal ABC algoritması ise sömürme kısmında zayıf kaldığı için, Gao ve Liu (2012) ABC/Best ve orijinal ABC algoritmaları bir seçim olasılığı aracılığıyla birlikte kullandı. Böylece değiştirilmiş yapay arı kolonisi (MABC) olarak adlandırılan bu algoritma keşfetme ve sömürme kısımları için dengeyi başarılı bir şekilde kurdu.

Liu ve ark. (2012), birlikte öğrenme faktörü tarafından seçilen en yüksek uygunluk değerine sahip iki birey ile aday besin kaynağı üretmeyi ayarlayan birlikte öğrenme stratejisini kullanarak ABC algoritmasını geliştiren bir algoritma önerdi.

Gao ve ark. (2013), ABC algoritmasını besin kaynağı arama denklemi yerine, Genetik algoritmadaki (GA) çaprazlama işlemine benzeyen yeni bir değiştirilmiş çözüm arama denklemine sahip olan CABC 'yi önerdi. Ayrıca bu çalışmada, arama tecrübelerinden daha verimli bilgiler elde edebilmek için ortoganal öğrenme stratejisini çeşitli ABC türlerine uyguladı. Elde ettiği bu algoritmaları (OABC, OGABC, OCABC), farklı ABC türlerinin yanı sıra, evrim algoritmaları (EA), diferansiyel evrim algoritmaları (DE) ve parçacık sürü optimizasyonu (PSO) algoritmalarının farklı türleri ile karşılaştırarak inceledi.

Ayrıca ABC algoritmasının zayıf yönleri geliştirmek için aynı PSO algoritmasında olduğu gibi ABC algoritması da diğer yöntemlerle hibrit olarak kullanıldı (Li ve ark., 2013; Yildiz, 2013; Zang ve ark., 2013).

2.3. Levy Uçuşu

Levy uçuşları, Levy hareketi olarak da bilinir, Gauss olmayan rasgele işlemlerin sabit artışlarla Levy sabit dağılımına göre dağıtıldığı Fransız matematikçi Pierre Lévy tarafından çalışılmış bir sınıfı temsil etmektedir (Chechkin ve ark., 2008).

Akışkanlar dinamiği, deprem analizi, ışınır moleküllerin difüzyonu gibi birçok doğal ve yapay olay Levy uçuşları ile tanımlanabilmektedir (Chen, 2010). Levy uçuşu cilt dokusu ultrasonunda (Pereyra ve Batatia, 2010) ve lazer arama, tespiti, mesafe tayini (Ladar) gibi alanlarda kullanıldı (Al-Temeemy ve ark., 2010). Aynı zamanda Levy uçuşu bu alanların dışında bilgisayar bilimlerinde de birçok konuda önemli bir rol üstlendi. Internet trafik modellerinde (Terdik ve Gyires, 2009), gecikme ve bozulma toleranslı ağlarda (Delay and disruption Tolerant Network-DTN) (Chen, 2010) ve çoklu-robot arama algoritmasında kullanıldı (Sutantyo ve ark., 2010)**.**

Aynı zamanda albatros, bombus arıları, geyik gibi birçok hayvanın yem arama yoluna benzeyen Levy uçuşu (Edwards ve ark., 2007) doğa esinli algoritmalara eklenerek algoritmaların geliştirilmesi sağlandı. Yang ve Deb (2013) Guguk Kuşu Arama' da yeni yavru oluşturmak için Levy uçuşu dağılımını kullandı, ayrıca Xin-She Yang (2010a) Ateş Böceği Algoritmasının (FA-Firefly Algorithm) yeni bir versiyonu olan Levy uçuşu Ateş Böceği Algoritmasını, bu algoritmanın rasgeleleştirmesini düzeltmek için Levy uçuşu arama stratejisi ile kombine etti. Lee and Yao (2001) ise Evrim Algoritmalarında (EA) Levy uçuşun β parametresinin 4 farklı durumu ile 4 aday çözüm oluşturup bu aday çözümler içinden en iyi sonucu hangisi veriyorsa onu alarak mutasyon işlemini gerçekleştirdi.

Genel anlamda, Levy uçuşları adım uzunluğu Levy dağılımına göre belirlenen rasgele bir yürüyüştür, sıklıkla basit bir güç yasası formülü açısından $L(s) \sim |s|^{-1-\beta}$ ' de $0 < \beta < 2$ olduğu bir indekstir. Matematiksel olarak, Levy dağılımının basit bir gösterimi Denklem (2.10) ' da gösterilmiştir (Yang ve Deb, 2013):

$$L(s,\gamma,\mu)=\begin{cases}\sqrt{\frac{\gamma}{2\pi}}exp\left[-\frac{\gamma}{2(s-\mu)}\right]\frac{1}{(s-\mu)^{3/2}} & if\ 0<\mu<s<\infty,\\ 0 & if\ s\le 0\end{cases} \tag{2.10}$$

μ parametresi konum(location) ya da değiştirme (shift) parametresi, $\gamma > 0$ ise ölçek (scale) (dağılım ölçeğini kontrol eden) parametredir.

Genel olarak, Levy dağılımı Fourier transform'una göre şu şekilde tanımlanabilir (Yang, 2010b):

$$F(k)=\exp\left[-\alpha|k|^{\beta}\right],\qquad 0<\beta\le 2, \tag{2.11}$$

α [-1,1] değer aralığında bulunan eğrilik(skewness) ya da ölçek etmeni olarak bilinen parametredir. Kararlılık indeksi β ϵ (0,2] aynı zamanda Levy indeksi olarak da adlandırılır. İntegralin analitik formu birkaç özel durum β' nın genel durumu için bilinmektedir.

Özellikle, $\beta = 1$ için, integral analitik olarak gerçekleştirilir ve Cauchy olasılık dağılımı olarak bilinir.

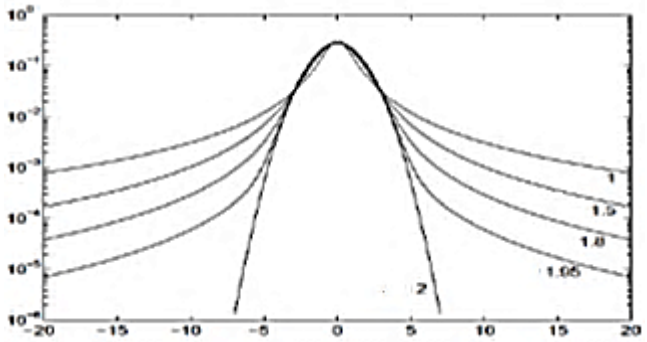
$$F(k)=\exp\left[-\alpha|k|\right], \tag{2.12}$$

Bir diğer özel durum $\beta = 2$ değerini aldığı zaman, dağılım Gauss dağılımına denk gelir.

$$F(k)=\exp\left[-\alpha k^{2}\right], \tag{2.13}$$

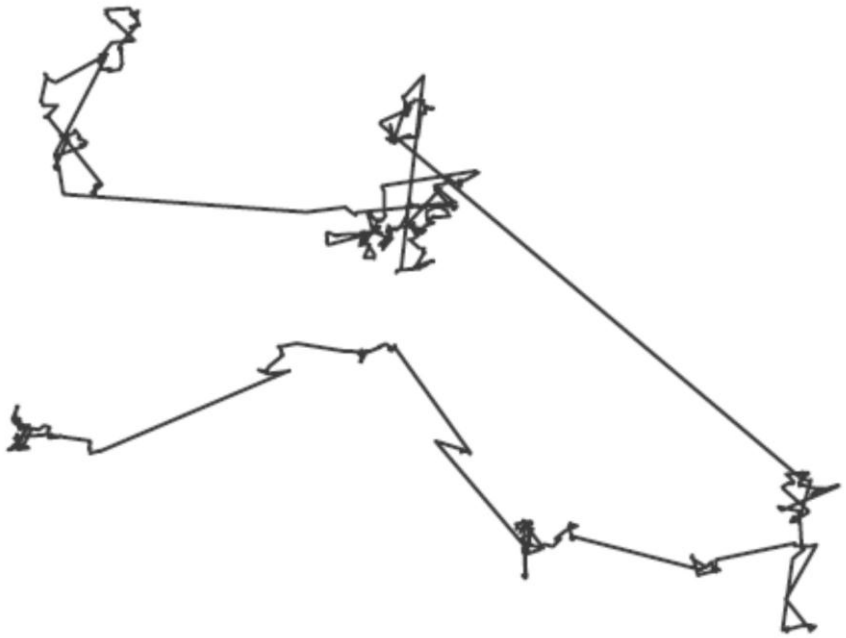
β ve α parametreleri dağılımı belirlemede daha büyük bir rol oynarken, γ ve μ parametreleri daha az rol oynar. β parametresi olasılık dağılımının biçimini kontrol eder, bir şekilde olasılık dağılımın farklı biçimleri elde edilebilir, özellikle kuyruk bölgesi β parametresine bağlıdır. Böylece daha küçük β parametresi, daha uzun kuyruk olacağından dağılımın daha uzun atlamalar yapmasına sebep olur (Lee, 2001). Eğrilik parametresi α 'nın işareti eğri yönünü belirler, pozitif ise sağa, ve negatif ise sola doğrudur. $\alpha = 0$ olduğunda, dağılım simetriktir. Diğer iki parametre genişlik γ ve

kaydırma μ dağıtım tepe noktalarıdır (Al-Temeemy ve ark., 2010). Şekil 2.2' de β parametresinin farklı değerlerinin dağılımı nasıl değiştirdiği görülmektedir. Küçük değerler için daha uzun atlamalar yaparken, büyük değerler için daha kısa atlayışlar yapar.



Şekil 2.2. *β =2,1.95,1.8,1.5 ve 1* değerleri için Levy olasılık yoğunluk fonksiyonları (Al-Temeemy ve ark., 2010).

Ayrıca Şekil 2.3 ' de ise 100 adım için 2 boyutta Levy uçuşunun yolunun nasıl olabileceği ile ilgili örnek verilmektedir.



Şekil 2.3. 100 adımda 2 boyut için Levy Uçuşu dağılımının yol örneği (Yang ve Deb, 2013)

3. LEVY UÇUŞU ile PSO (LFPSO) YÖNTEMİ

3.1. LFPSO Algoritması

Literatürdeki birçok değişik PSO algoritması bulunmasına rağmen, PSO' nun erken yakınsama ve verimsiz sonuçlar üretme sorunları devam etmektedir. Bu sorunları çözerek PSO' nun daha verimli sonuçlar elde etmesini sağlamak amacıyla tez çalışması kapsamında Levy uçuşları yöntemi kullanıldı. Bu yöntemle global arama yeteneği zayıf olan ve belli bir iterasyon sonrasında popülasyon çeşitliliği azalan PSO' nun daha etkili bir şekilde global arama yaparak yerel minimumlardan kaçması sağlandı. Levy uçuşu yöntemi ile parçacıklara rasgele atlayışlar yaptırılarak popülasyonun çeşitliliği artırıldı. LFPSO yönteminde orijinal PSO yöntemine göre 2 değişiklik yapıldı.

- Aynı ABC algoritmasında olduğu gibi tüm parçacıklar için bir çözüm geliştirememe değeri tutuldu ve parçacıkların her yeni iterasyon için *pbest* değerlerinde iyileştirme göstermemeleri durumunda bu değer 1 artırıldı.
- Belirlenen limit değerini aşan parçacıklar ise Levy uçuşu (Levy dağılımı kullanılarak) yöntemi ile yeniden arama uzayında dağıtıldı.

Orijinal PSO' da ki gibi parçacıklar ilk olarak arama uzayında rastgele dağıtılır, hesaplanan uygunluk değerleri her bir parçacık için *pbest* olarak atanır. En iyi uygunluk değerine sahip parçacık ise gbest olarak belirlenir. Orijinal PSO 'dan farklı olarak hız güncellemesi yapılmadan önce her bir parçacık için limit aşımı kontrolü yapılır. Eğer o anki parçacık limit değerini aşmadıysa normal şekilde hız güncellemeleri yapılır ve pozisyonu güncellenir. Parçacık limit değerini aşmışsa limit değeri sıfırlanır, parçacık Levy uçuşu yöntemi ile arama uzayında dağıtılır ve dağıtma işleminden sonra belirlenen arama uzayında sınırı aşan pozisyonlar sınır değerlere çekilir.

Bu işlemler bittikten sonra yeni belirlenen parçacık için uygunluk değeri hesaplanır, hesaplanan yeni uygunluk değerleri parçacığın *pbest* 'in den daha iyi ise *pbest* olarak atanır ve limit değeri sıfırlanır, eğer iyileşme yok ise parçacık için limit değeri 1 artırılır. Aynı şekilde yeni uygunluk değeri *gbest* değerinden küçük ise parçacık *gbest* değeri olarak atanır. Aynı işlemler belirlenen iterasyon şartı sağlanana kadar yapılır.

Aşağıda önerilen algoritmanın adımları verilmektedir.

1. Başlangıç parametrelerini(PS, D, X_{min}, X_{max}, c_1, c_2, V_{min}, V_{max},$trial$) ayarla.
2. $i=1,2,...,PS$, $d=1,2,...,D$ olmak üzere parçacıkların başlangıç değerleri Denklem (2.1)' de gibi rasgele şekilde arama uzayında belirle.
3. Her bir parçacık için uygunluk değerlerini belirle ve parçacıkların *pbest* değerlerine ata.
4. *gbest=min(pbest)* (Parçacıkların pbestlerinin en iyisi *gbest* olarak ata)
5. while(sonlandırma kriteri)
6. for $i=1:PS$
7. if $trial_i < limit$ (Eğer parçacık belirlenen limit değerini aşmadıysa)
   Denklem (2.3) kullanarak parçacıkların hız güncellemesi yap(Vi)
   Denklem (2.4) kullanarak parçacıkların pozisyon güncellemesini yap(Xi)
8. else
   $trial_i$=0
   Parçacık pozisyonları Levy Uçuşu yöntemine göre yeniden dağıt.
   Arama uzayını aşan boyutları belirlenen sınır değerlere(X_{min}, X_{max}) çek.
9. end//if
10. Belirlenen yeni parçacık için uygunluk değeri hesapla
    if($fitness_{new}$>$pbestvalue_i$) (Yeni parçacığın çözüm kendi *pbest*inden iyi ise)
      $pbest_i = X_{new}$
      $trial_i$=0
    else
      $trial_i = trial_i$+1;
    end //if
11. end //for
12. Yeni belirlenen parçacıklar için *gbest* kontrolü yap
    temp=min(*pbest*)
    if(temp<*gbest*)
      gbest=temp
    end //if
13. end //while

X_{max} ve X_{min} arama uzayının maksimum ve minimum değerlerini temsil eder. V_{min} ve V_{max} yapılacak hız artışının maksimum ve minimum limitini temsil eder. *PS* parçacıkların sayısını, *D* test fonksiyonlarının boyutunu, c_1 (kognitif) ve c_2 (sosyal) öğrenme faktörleri, trial ise her bir parçacık için limit değerlerini tutar.

Levy uçuşu kullanılarak dağılım nasıl yapıldığını biraz ayrıntılı şekilde inceleyelim. Levy uçuşu ile parçacığın yeni durumu (Yang ve Deb, 2013);

$$X^{t+1} = X^t + \alpha \oplus Levy(\beta) \tag{3.1}$$

Denklem (3.1) kullanılarak hesaplanır. α, ilgili problemin ölçeklenmesi ile ilişkili adım ölçüsüdür (step size). Önerilen metotta α yerine parçacıkların tüm boyutları için rasgele bir sayı üretilmiş ve denklemin yeni durumu (3.2)' de gösterilmiştir. Bu $\oplus$ sembol çoklu çarpım anlamına gelmektedir.

$$X^{t+1} = X^t + random(size(D)) \oplus Levy(\beta) \tag{3.2}$$

Levy Uçuşu kullanılarak oluşturulan adım ölçüsü *s* 'in örnekleri önemsiz değildir. Yang tarafından detaylandırılan basit bir düzeni şu şekilde özetlenir (Entrywise çarpımından önceki kısım önerilen metottaki gibi güncellenmiştir) (Yang, 2010):

$$s = random(size(D)) \oplus Levy(\beta) \sim 0.01 \frac{u}{|v|^{1/\beta}} (x_j^t - gbest^t) \tag{3.3}$$

u ve *v* normal dağılımlara göre belirlenir. Yani

$$u \sim N\left(0, \sigma_u^2\right)\ v \sim N\left(0, \sigma_v^2\right) \tag{3.4}$$

ile

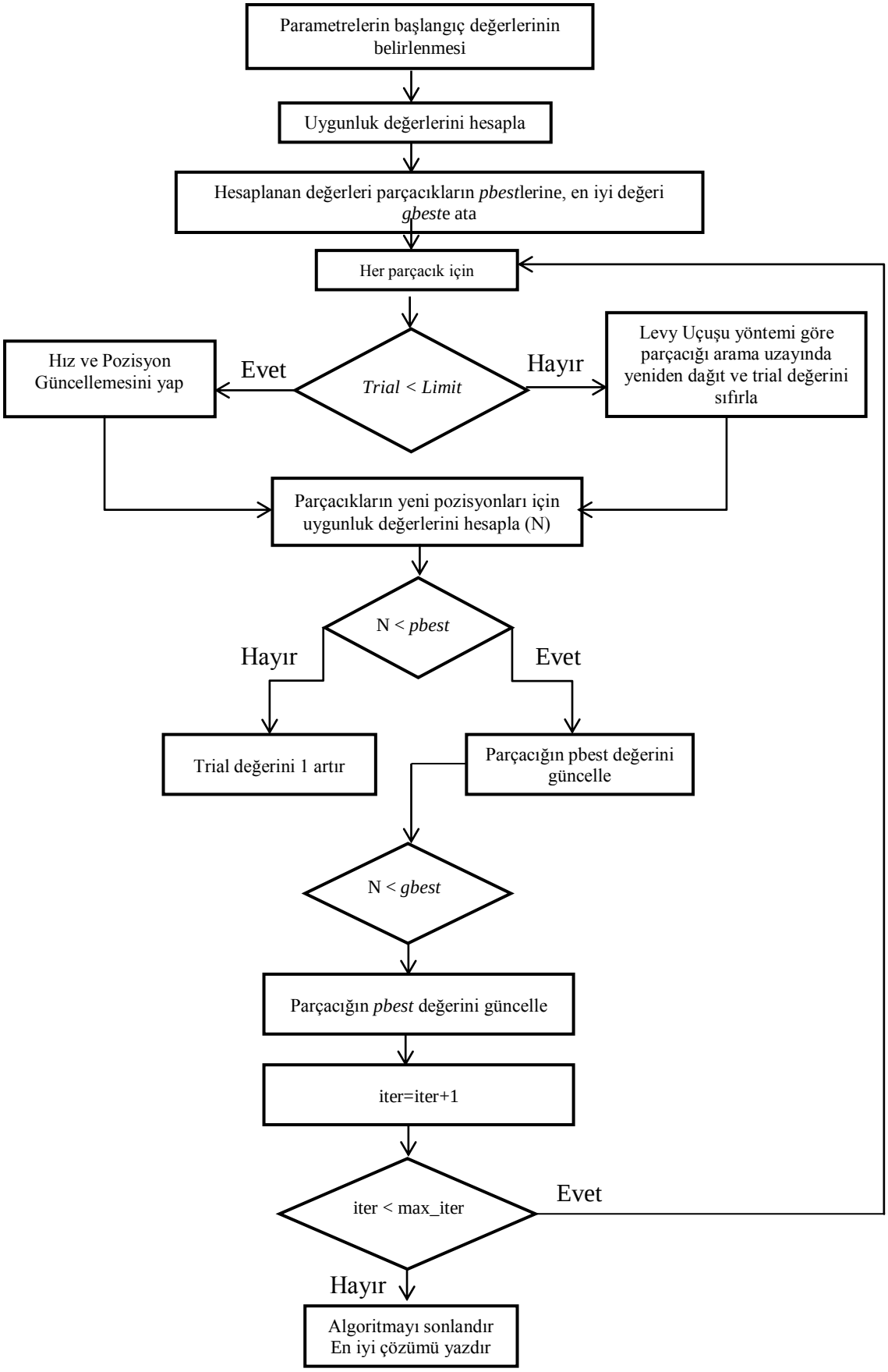
$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma\left[(1+\beta)/2\right]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1 \tag{3.5}$$

Burada Γ standart Gamma fonksiyonudur.

Yukarda elde edilen D boyutlu s değeri, pozisyon güncellemesi yapılacak olan X_i parçacığına eklenerek yeni parçacığın pozisyon değerleri bulunur. Bu yeni parçacık için uygunluk değeri hesaplanır, eğer parçacık *pbest*'inden daha iyi bir çözüm elde ettiyse *pbest* güncellenir aksi halde çözüm geliştirememe sayacı *trial* 1 artırılır. Algoritma kaldığı yerden durdurma kriteri sağlanana kadar devam eder. Şekil 3.1 algoritmanın akış diyagramını göstermektedir.

Levy uçuşu ile dağıtım yapılırken dikkat edilmesi gereken önemli noktalardan bir tanesi β parametresinin aldığı değerdir. Daha önce de belirtildiği gibi Yang ve Deb (2013) çalışmasında yapılan denemelerde β parametresinin farklı değerler de farklı sonuçlar verdiğine değindi. Buradan da her bir farklı karakteristikteki test fonksiyonu için ayrı bir β parametresinin daha etkin bir sonuç verdiği söylenebilir. Ayrıca Lee ve Yao (2001)'nun yaptığı çalışmada ise β parametresi için 4 farklı sabit değer belirlenerek bu değerlerin her biri için dağılım hesaplanıp üretilen yeni yavruların en iyisi seçilerek işlem yapma yoluna gidilmiştir. Bu iki örnekte de görüldüğü gibi β parametresi dağılım durumunu önemli ölçüde etkilemektedir. Bu tez çalışmasında ise β parametresi için sabit bir değer alınmayıp her Levy uçuşu dağılım işlemi için (0,2] aralığında rasgele bir değer alınmaktadır. PSO hem yerel hem global durumlardan etkilenerek hız değişimi yapsa da belli bir iterasyon sonrasında parçacıkların birbirine benzemesi nedeniyle hız değişimleri çok küçük miktarlara düşerek global arama yeteneğini kaybetmesine sebep olur. β parametresinin küçük değerleri için büyük atlamalar yapar bu da her zaman söz ettiğimiz PSO' daki global arama eksikliğini giderip, yerel minimumlara takılmayı engeller. Bir diğer farklılık ise Guguk Kuşu (Cuckoo) Optimizasyonun' daki Levy Uçuşu dağılımında adım ölçüsü hesaplanırken iki rasgele guguk kuşunun farkı alınırken önerilen algoritma LFPSO' da o anki parçacıktan en iyi parçacık (gbest) çıkarılır. Böylece en iyi parçacık değerinin (gbest) değişmeyerek korunması sağlanır. Bu şekilde rasgele alınan β parametresi küçük değerler alırsa parçacıkların arama uzayında çok büyük atlamalar yapabilmesine olanak sağlamakta ve yerel minimumlara sürekli takılmaları engellemektedir, büyük değerler çıkması durumda ise gbest etrafında yakın yeni değerler elde etmeye devam etmektedir. Bu rasgeleleştirme işlemi, adımlar güç yasasına benzeyen Levy dağılımına göre belirlenirse çok daha etkili olur: bundan dolayı, çok sayıda küçük adımlar ve bazen de büyük adım içeren adımlar, büyük farklı atlamalara sebep olur (Yang ve Deb, 2011). Önerilen yöntem de β parametresi her

parçacık için rasgele belirleneceği için büyük atlamaların daha fazla yapılması sağlanmıştır. Yang ve Deb (2011), çalışmasında Guguk Kuşu (Cuckoo) Optimizasyonu anlatırken "PSO ile karşılaştırıldığında, bu uzun atlamalar bazı durumlarda özellikle birçok yerel minimum ancak bir global minimum içeren çok modlu (multimodal) ve lineer olmayan problemler için "guguk" aramanın arama verimliliğini büyük ölçüde artırır" demiş, PSO 'nun zayıflığı olarak bu durumu belirtmiş Guguk Kuşu Optimizasyonu için ise bu durumun bir avantaj olduğunu söylemiştir. Bu tez kapsamında PSO 'nun zayıflığını gidermek için Levy uçuşu kullanılarak rasgele yürüyüşler ile PSO 'nun global araması güçlendirilmiş, yerel minimumlara takılması engellemiş, özellikle birçok yerel minimuma sahip fonksiyonlar için daha başarılı sonuçlar verdiği gözlemlenmiştir.



Şekil 3.1. LFPSO Algoritması Akış Diyagramı

3.2. Testler ve Sonuçlar

Tüm denemeler Windows 7 Profesyonel İşletim Sistemi ortamında Intel i5, 2.4 GHz, 4GB RAM kullanarak Matlab 7.9' da kodlanan program ile gerçekleştirildi.

3.2.1. Orijinal PSO ve LFPSO için test ve sonuçlar

3.2.1.1. Parametre ayarları

Testlerde parçacık sayısı 50 olarak belirlendi. Algoritmalar, Çizelge 3.1' de verilen F1-F9 test fonksiyonları için 10, 30 ve 50 boyutlarında test edildi. İterasyon sayıları ise *(D*10000)/PS* olarak hesaplandı. *PS* parçacık sayısını ifade etmektedir. Çizelge 3.1' de verilen sabit boyutlu fonksiyonlar F10-F23 için ise iterasyon sayısı 10000 olarak belirlendi. Parçacıklar R arama uzayında [X_{min},X_{max}] aralığında başlatılıp, X_{min} ve X_{max} arama uzayının maksimum ve minimum değerleri olarak belirlendi. Hız limitleri V_{min} ve V_{max} değerleri sınır değerlerin (X_{min},X_{max}) %20'si olarak alındı. Çizelge 3.2, 3.3, 3.4 ve 3.5' de gösterilen test sonuçları test fonksiyonlarının 25 çalıştırmada ki minimum değerlerinin ortalama sonuçlarıdır. Durdurma kriteri maksimum iterasyon sayısı olarak belirlendi. Algoritmalar için bu testlerde $c_1 = c_2 = 2$ alındı. Atalet ağırlığı Denklem (3.6) ile hesaplandı. Önerilen LFPSO yöntemi standart PSO ile karşılaştırıldı.

$$w = \left(max\_iter - iter\right) / iter \tag{3.6}$$

LFPSO için ise β değeri belirtildiği gibi her yeni dağıtım işleminde (0,2] aralığında rasgele bir sayı olarak belirlendi. LFPSO için limit değeri yapılan deneysel çalışmalar sonunda 5 ve 10 olarak seçilmesi uygun görüldü. LFPSO-5 limit değerinin 5 olduğunu, LFPSO-10 ise limit değerinin 10 olduğunu gösterir. Çizelge 3.2, 3.3, 3.4 ve 3.5, LFPSO-10'nun LFPSO-5' e göre çok daha başarılı sonuçlar verdiğini göstermektedir. LFPSO' nun limit 5 içinde çalıştırılmasının sebebi limit değerinin sonuçları nasıl etkilediğini göstermektir. Ayrıca elde edilen sonuçlar arasında istatistiksel olarak fark olup olmadığını anlamak için sonuçlar parametrik olmayan Wilcoxon testine tabi tutularak durumları sonuç çizelgelerinde belirtildi.

3.2.1.2. Test fonksiyonları

Tez çalışması kapsamında sürekli değindiğimiz gibi PSO 'nun yerel minimumlara takılması en bilinen sorunu olduğu için test fonksiyonları seçilirken birçok yerel minimum ancak bir global minimum içeren çok modlu (multimodal) fonksiyonlar ağırlıklı olarak belirlenmiştir. Belirlenmiş olan bu fonksiyonlar Çizelge 3.1' de gösterilmektedir. Bu belirleme ile önerilen LFPSO metodunun yerel minimumlara takılmayarak daha iyi sonuçlar bulduğu gösterilmek istenmiştir. F1-F2-F4-F8-F9 fonksiyonları tek optimum nokta içeren tek modlu (unimodal), bu fonksiyonlar dışında kalan diğer test fonksiyonları ise çok modlu fonksiyonlardır.

Çizelge 3.1. Test Fonksiyonları (K:Karakteristik, T: Tek modlu, Ç:Çok modlu, D:Boyut)

No	Fonksiyon	Aralık	K	D	Formülasyon
1	Sphere	[-100,100]	T	D	$f_1 = \sum_{i=1}^{n} x_i^2$
2	Rosenbrock	[-30,30]	T	D	$f_2 = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]$
3	Rastrigin	[-5.12,5.12]	Ç	D	$f_3 = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$
4	Schwefel2.6	[-500,500]	T	D	$f_4 = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{x_i}\right)$
5	Griewank	[-600,600]	Ç	D	$f_5 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
6	Ackley	[-32,32]	Ç	D	$f_6 = -20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right\} - \exp\left\{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right\} + 20 + e$
7	Levy	[-10,10]	Ç	D	$f_7 = \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + \sin^2(3\pi x_1) + \lvert x_n - 1\rvert[1 + \sin^2(3\pi x_n)]$
8	Zakharov	[-5,10]	T	D	$f_8 = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$
9	Schwefel1.2	[-100,100]	T	D	$f_9 = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$
10	Hartmann3	[0,1]	Ç	3	$f_{10} = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$
11	Hartmann6	[0,1]	Ç	6	$f_{11} = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$

12	Bohachevsky1	[-100,100]	Ç	2	$f_{12} = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
13	Bohachevsky2	[-100,100]	Ç	2	$f_{13} = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
14	Bohachevsky3	[-100,100]	Ç	2	$f_{14} = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
15	Michalewicz2	[0,π]	Ç	2	$f_{15} = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10, D = 2$
16	Michalewicz5	[0,π]	Ç	5	$f_{16} = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10, D = 5$
17	Michalewicz10	[0,π]	Ç	10	$f_{15} = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10, D = 10$
18	Perm	[-D,D]	Ç	4	$f_{18} = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k + \beta)((x_i/i)^k - 1)\right]^2$
19	Schaffer	[-100,100]	Ç	2	$f_{19} = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$
20	Shubert	[-10,10]	Ç	2	$f_{20} = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$
21	Shekel5	[0,10]	Ç	4	$f_{21} = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$
22	Shekel7	[0,10]	Ç	4	$f_{22} = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$
23	Shekel10	[0,10]	Ç	4	$f_{23} = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$

3.2.1.3. Sonuçlar ve karşılaştırmalar

Çizelge 3.1' de belirtildiği gibi önerilen LFPSO metodunu test etmek için 23 fonksiyon kullanıldı. Bunların ilk 9 (F1-F9) tanesi 10,30 ve 50 boyutlar için, kalan 14 tanesi ise (F10-F23) Çizelge 3.1' de verilen sabit boyut için çalıştırıldı. Sonuçlar arasında istatistiksel olarak anlamsal fark olup olmadığını anlamak için parametrik olmayan Wilcoxon testi yapıldı. Wilcoxon testi yardımıyla PSO ile LFPSO-5 ve LFPSO-10 yöntemleri karşılaştırıldı. Çizelge 3.2, 3.3, 3.4 ve 3.5' de Wilcoxon testi 0.05 anlamlılık seviyesinde ise + işareti, değil ise – işareti kullanıldı, tüm sonuçlar aynı değeri gösteriyor ya da optimum değeri bulduysa işaret konulmadı.

3.2.1.3.1. 10 boyutlu fonksiyonlar için karşılaştırmalar

Dokuz standart test fonksiyonun (F1-F9) 10 boyut için 2,000 iterasyon ile 25 çalıştırmadaki ortalama sonuçları Çizelge 3.2' de verilmektedir. Çizelge 3.2 ayrıca PSO, LFPSO-5 ve LFPS-10 ilgili test fonksiyonlarının 25 çalıştırmadaki ortalama optimum çözümü, standart sapması ve Wilcoxon testlerinde ki değerleri göstermektedir. Algoritmalar tarafından test fonksiyonları için bulunan en iyi ortalama sonuç ve en iyi standart sapma değeri de kalın olarak gösterilmiştir. Çizelge 3.2 incelendiğinde önerilen LFPSO-10 algoritması ortalama sonuçlarının fonksiyonlar 1, 3, 4, 5, 7 ve 9 için PSO algoritmasına göre daha iyi olduğu görülmektedir. . LFPSO-5 algoritması ise PSO ile kıyaslandığında fonksiyon 3, 4, 5 ve 7' de daha başarılıdır. PSO algoritması diğer algoritmalara göre 2, 6 ve 8 fonksiyonlarında daha iyi sonuç vermektedir. Algoritmaların gürbüzlük açısından kıyaslamalarını standart sapmalara bakarak yapabiliriz. Sonuçları arasındaki tutarsızlığın fazla olması gürbüz olmadığını gösterir. LFPSO-10 algoritması PSO ile fonksiyon 1, 2, 3, 4, 5, 6, 7 ve 9 için karşılaştırıldığında gürbüz bir algoritmadır. LFPSO-10 algoritması fonksiyon 1, 3, 5, 6 ve 9 için sıfır standart sapma göstermektedir. PSO algoritması sadece fonksiyon 8 için daha gürbüz bir sonuç vermiştir. 10 boyut için genel olarak baktığımızda 6 test fonksiyonu için önerilen LFPSO-10 algoritmasının daha başarılı ve gürbüz sonuçlar verdiği gözlemlenmektedir.

Çizelge 3.2. 10 boyut için sonuçlar (Ort:Ortalama, Std. Sap.:Standart Sapma, S: İstatiksel Test İşareti)

F	**Opt.**	**PSO**		**LFPSO-5**			**LFPSO-10**		
		Ort.	Std. Sap.	Ort.	Std. Sap.	S	Ort.	Std. Sap.	S
F1	0,00E+00	2, 34E-70	1,17E-69	3,59E-41	1,70E-40	+	**0,00E+00**	**0,00E+00**	+
F2	0,00E+00	**3,13E+00**	2,03E+00	4,79E+00	**1,81E-01**	+	4,27E+00	2,45E-01	+
F3	0,00E+00	5,37E+00	2,07E+00	**0,00E+00**	**0,00E+00**	+	**0,00E+00**	**0,00E+00**	+
F4	-4,19E+03	-3,29E+03	2,60E+02	**-4,10E+03**	**1,30E+02**	+	-4,02E+03	1,98E+02	+
F5	0,00E+00	6,66E-02	3,31E-02	5,13E-03	1,19E-02	+	**0,00E+00**	**0,00E+00**	+
F6	0,00E+00	**4,01E-15**	1,18E-15	6,15E-15	2,54E-15	+	4,44E-15	**0,00E+00**	-
F7	0,00E+00	**1,50E-32**	**8,38E-48**	**1,50E-32**	**8,38E-48**		**1,50E-32**	**8,38E-48**	
F8	0,00E+00	**6,14E-36**	**2,10E-35**	7,81E-17	1,11E-16	+	4,14E-27	1,59E-26	+
F9	0,00E+00	1,17E-23	2,45E-23	2,42E-11	4,97E-11	+	**0,00E+00**	**0,00E+00**	+

3.2.1.3.2. 30 boyutlu fonksiyonlar için karşılaştırmalar

Dokuz standart test fonksiyonun (F1-F9) 30 boyut için 6,000 iterasyon ile 25 çalıştırmadaki ortalama sonuçları Çizelge 3.3' de verilmektedir. Çizelge 3.3 ayrıca PSO, LFPSO-5 ve LFPS-10 algoritmalarının ilgili test fonksiyonlarının 25 çalıştırmadaki ortalama optimum çözümü, standart sapması ve Wilcoxon testlerinde ki değerlerini göstermektedir. Çizelge 3.3' de görüldüğü gibi LFPSO-10 algoritması PSO algoritmasına göre fonksiyon 1, 2, 3, 4, 5, 7, 8 ve 9 daha iyi ortalama sonuç verirken sadece fonksiyon 6' da daha kötü ortalama sonuç vermiştir. LFPSO-5' in ise fonksiyon 2, 3, 4 ve 8 de PSO' ya göre daha iyi sonuçlar verdiği görülür. Çizelge 5.3' deki standart sapma değerlerinin gösterdiği gibi LFPSO-10 algoritması PSO ile fonksiyon 1, 2, 3, 5, 7, 8 ve 9 için karşılaştırıldığında gürbüz bir algoritmadır. 30 boyut için genel olarak baktığımızda önerilen LFPSO-10 yönteminin PSO yöntemine göre fonksiyon 6 hariç diğer fonksiyonlarda daha başarılı ve gürbüz sonuçlar verdiği görülmektedir.

Çizelge 3.3. 30 boyut için sonuçlar(Ort:Ortalama,Std. Sap.:Standart Sapma, S: İstatiksel Test İşareti)

F	Opt.	PSO		LFPSO-5			LFPSO-10		
		Ort.	Std. Sap.	Ort	Std. Sap.	S	Ort.	Std. Sap.	S
F1	0,00E+00	1,93E-33	9,19E-33	6,16E-24	2,15E-23	+	**1,13E-38**	**4,02E-38**	+
F2	0,00E+00	4,42E+01	2,96E+01	2,47E+01	1,74E-01	-	**2,37E+01**	**2,94E-01**	+
F3	0,00E+00	4,19E+01	1,28E+01	2,12E+00	6,13E+00	+	**1,32E+00**	**5,49E+00**	+
F4	-1,26E+04	-7,76E+03	7,05E+02	-1,12E+04	7,82E+02	+	**-1,13E+04**	**8,94E+02**	+
F5	0,00E+00	1,53E-02	1,93E-02	2,27E-02	2,92E-02	-	**0,00E+00**	**0,00E+00**	+
F6	**0,00E+00**	**9,41E-15**	2,90E-15	1,12E-12	3,75E-12	+	1,41E-14	2,99E-15	+
F7	0,00E+00	7,78E-31	1,44E-30	2,34E-26	8,74E-26	+	**3,26E-31**	**1,27E-30**	+
F8	0,00E+00	2,18E+02	1,67E+02	**9,03E+01**	**8,23E+01**	+	1,81E+02	1,72E+02	-
F9	0,00E+00	1,67E-02	1,58E-02	1,29E-01	1,45E-01	-	**4,81E-03**	**8,15E-03**	+

3.2.1.3.3. 50 boyutlu fonksiyonlar için karşılaştırmalar

Dokuz standart test fonksiyonun (F1-F9) 50 boyut için 10,000 iterasyon ile 25 çalıştırmadaki ortalama sonuçları Çizelge 3.4' de verilmektedir. Çizelge 3.4 ayrıca PSO, LFPSO-5 ve LFPSO-10 algoritmaları için ilgili test fonksiyonlarının 25 çalıştırmadaki ortalama optimum çözümü, standart sapması ve Wilcoxon testlerinde ki değerlerini göstermektedir. Aynı 30 boyuttakine benzer şekilde LFPSO-10 algoritması PSO

algoritmasına göre fonksiyon 1, 2, 3, 4, 5, 7, 8 ve 9 için daha iyi ortalama sonuç verirken sadece fonksiyon 6 için daha kötü sonuç vermiştir. LFPSO-5' in ise fonksiyon 2, 3, 4, 7 ve 8 için PSO 'ya göre daha iyi sonuçlar verdiği görülür.

Çizelge 3.4 50 boyut için sonuçlar(Ort:Ortalama,Std. Sap.:Standart Sapma, S: İstatiksel Test İşareti)

F	Opt.	PSO		LFPSO-5			LFPSO-10		
		Ort.	Std. Sap.	Ort	Std. Sap.	S	Ort.	Std. Sap.	S
F1	0,00E+00	5,88E-24	1,83E-23	7,58E-18	2,33E-17	+	**1,39E-29**	**6,13E-29**	+
F2	0,00E+00	7,66E+01	3,80E+01	4,44E+01	3,42E-01	+	**4,33E+01**	**3,62E-01**	+
F3	0,00E+00	8,81E+01	1,86E+01	5,36E+00	9,54E+00	+	**1,33E+00**	**3,46E+00**	+
F4	2,09E+04	-1,26E+04	9,11E+02	-1,70E+04	1,36E+03	+	**-1,73E+04**	**1,46E+03**	+
F5	0,00E+00	6,50E-03	1,10E-02	1,11E-02	2,29E-02	-	**2,22E-17**	**9,06E-17**	+
F6	0,00E+00	**2,04E-14**	**3,57E-15**	1,81E-10	2,58E-10	+	3,40E-14	5,40E-15	+
F7	0,00E+00	3,74E-01	5,61E-01	4,05E-20	6,38E-20	+	**1,59E-28**	**3,65E-28**	+
F8	0,00E+00	8,24E+02	3,02E+02	**3,93E+02**	**2,03E+02**	+	5,23E+02	2,26E+02	+
F9	0,00E+00	4,24E+02	1,39E+03	8,56E+00	8,02E+00	+	**1,94E+00**	**2,44E+00**	+

Çizelge 3.2, 3.3 ve 3.4 incelendiğinde genel olarak boyut sayısı arttıkça önerilen LFPSO-5 ve LFPSO-10 algoritmalarının PSO'ya göre daha iyi sonuçlar verdiği söylenebilir. Çizelge 3.4' deki standart sapma değerlerinden anlaşılacağı üzere LFPSO-10 yöntemi, PSO ile kıyaslandığında fonksiyon 1, 2, 3, 5, 7, 8 ve 9 için gürbüz bir algoritmadır. Çizelge 3.4 LFPSO-10'nun verilen tüm fonksiyonlar için istatistiksel testi geçtiğini gösterir, bu da iki yöntemin birbirinden farklı olduğunu bize söylemektedir. 50 boyut için genel olarak baktığımızda aynı 30 boyutta olduğu gibi LFPSO-10 yönteminin PSO 'ya göre F6 hariç diğer fonksiyonlarda daha başarılı ve gürbüz sonuçlar verdiği görülmektedir.

3.2.1.3.4. Sabit boyutlu fonksiyonlar için karşılaştırmalar

On dört standart test fonksiyonu (F10-F23) test tablosunda belirtilen sabit boyutları için 10,000 iterasyon ile 25 çalıştırmadaki ortalama sonuçları Çizelge 3.5' de verilmektedir. Çizelge 3.5 ayrıca PSO, LFPSO-5 ve LFPSO-10 ilgili test fonksiyonlarının 25 çalıştırmadaki ortalama optimum çözümü, standart sapması ve Wilcoxon testlerinde ki değerleri göstermektedir. Çizelge 3.5, LFPSO-10 algoritmasının PSO algoritmasına göre belirtilen 14 fonksiyonda da daha iyi ortalama sonuç verdiğini

gösterir. LFPSO-5 algoritması ise PSO algoritmasına göre fonksiyon 18 hariç diğer fonksiyonlarda daha iyi ortalama sonuç vermektedir. Standart sapma değerlerine baktığımız da ise LFPSO-10 fonksiyon 10, 11, 12, 13, 14, 15, 16, 18 ve 19 için PSO 'ya göre daha gürbüz bir algoritma iken, PSO ise LFPSO-10'a göre fonksiyon 17, 21, 22 ve 23 için kötü ortalama sonuç bulmasına rağmen daha tutarlı sonuçlar vermektedir. Genel olarak sabit boyutlu fonksiyonları yorumlarsak LFPSO-10 tüm sabit boyutlu fonksiyonlar için PSO' ya göre daha başarılı sonuçlar verdiği ve bazı fonksiyonlar hariç daha gürbüz bir algoritma olduğu Çizelge 3.5' de görülmektedir.

Çizelge 3.5 Sabit boyutlu fonksiyonlar için sonuçlar
(Ort:Ortalama, Std. Sap.:Standart Sapma, S: İstatiksel Test İşareti)

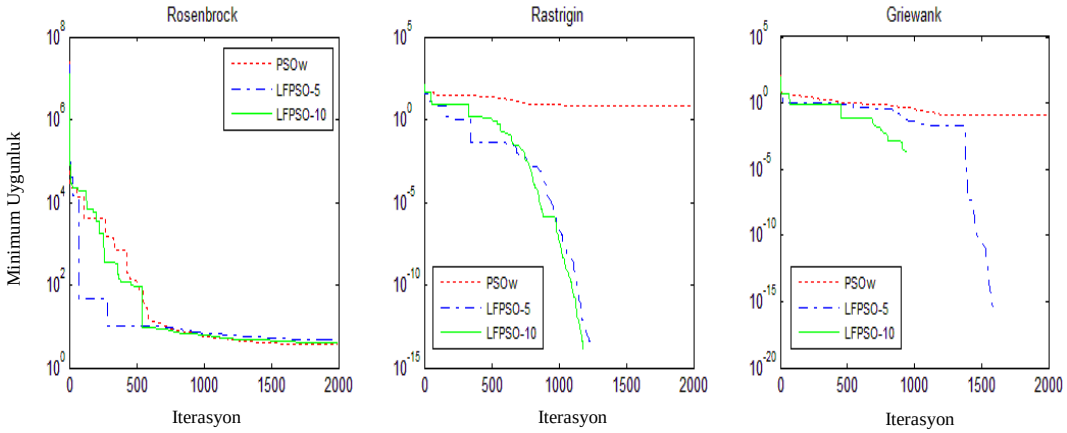
F	Opt.	PSO		LFPSO-5			LFPSO-10		
		Ort.	Std. Sap.	Ort	Std. Sap.	S	Ort.	Std. Sap.	S
F10	-3,86E+00	-3,72E+00	7,07E-02	-3,76E+00	8,64E-02	+	**-3,78E+00**	**4,41E-02**	+
F11	-3,32E+00	-2,06E+00	5,26E-01	-2,79E+00	1,76E-01	+	**-2,84E+00**	**1,59E-01**	+
F12-F13-F14	0,00E+00	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**		**0,00E+00**	**0,00E+00**	
F15	-1,80E+00	-1,57E+00	1,78E-01	**-1,73E+00**	**6,83E-02**	+	-1,71E+00	1,09E-01	+
F16	-4,69E+00	-2,53E+00	3,33E-01	**-2,97E+00**	**3,21E-01**	+	-2,85E+00	3,21E-01	+
F17	-9,66E+00	-3,67E+00	4,29E-01	**-4,14E+00**	**4,05E-01**	+	-4,06E+00	5,01E-01	+
F18	0,00E+00	8,14E-02	1,54E-01	9,66E-02	1,72E-01	-	**5,28E-02**	**1,03E-01**	-
F19	0,00E+00	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**		**0,00E+00**	**0,00E+00**	
F20	-1,87E+02	**-1,87E+02**	**2,32E-14**	**-1,87E+02**	**4,46E-14**		**-1,87E+02**	**4,14E-14**	
F21	-1,02E+01	-1,86E+00	7,82E-01	**-9,92E+00**	**1,19E+00**	+	-9,06E+00	2,59E+00	+
F22	-1,04E+01	-1,85E+00	7,49E-01	**-1,01E+01**	**1,43E+00**	+	-9,02E+00	2,90E+00	+
F23	-1,05E+01	-1,80E+00	5,39E-01	**-1,05E+01**	**4,00E-15**	+	-8,78E+00	3,23E+00	+

3.2.1.4. Yakınsama Grafikleri

10, 30 ve 50 boyutlar için Rosenbrock, Rastrigin ve Griewank fonksiyonlarının, sabit boyutlar için ise Hartmann6 ve Michalewicz5 fonksiyonlarının PSO, LFPSO-5 ve LFPSO-10 yöntemlerindeki yakınsama grafikleri Şekil 3.2, 3.3, 3.4 ve 3.5' de verildi. Genel olarak grafikleri incelediğimiz de önerilen LFPSO yönteminin PSO yöntemine göre daha erken yakınsadığı görülmektedir.

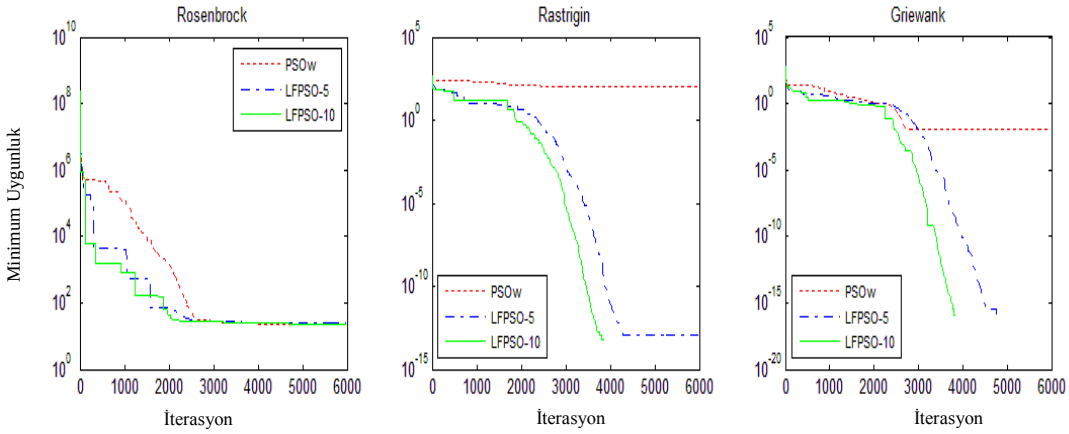
10 boyut için Rosenbrock fonksiyonunda önerilen LFPSO yöntemi daha erken yakınsamasına rağmen PSO algoritmasının daha iyi ortalama sonuç verdiği gözlemlenir. 10 boyuttaki diğer Rastrigin ve Griewank yakınsama grafiklerinde ise önerilen

yöntemin hızlı bir şekilde yakınsadığı ve optimum sonucu bulduğu Şekil 3.2' de görülmektedir.



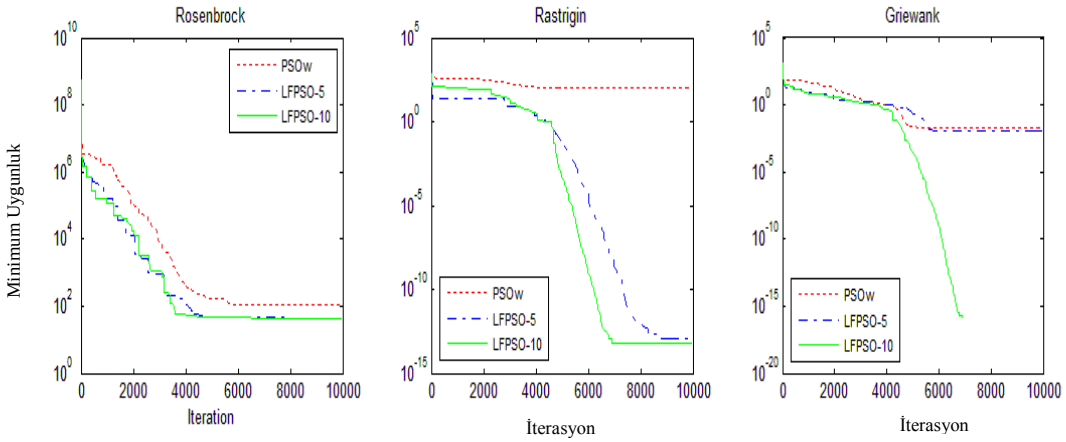
Şekil 3.2. 10 boyut için Rosenbrock, Rastrigin ve Griewank fonksiyonlarının yakınsama grafikleri

30 boyutta ise Rastrigin ve Griewank fonksiyonlarının yakınsama grafiklerinde PSO algoritmasının en iyi değere ulaşamadığı Şekil 3.3' de açıkça görülürken önerilen yöntem LFPSO optimum değerlere ulaştığı görülmektedir. Rosenbrock fonksiyonun da ise önerilen yöntemin daha erken yakınsadığı görülür.



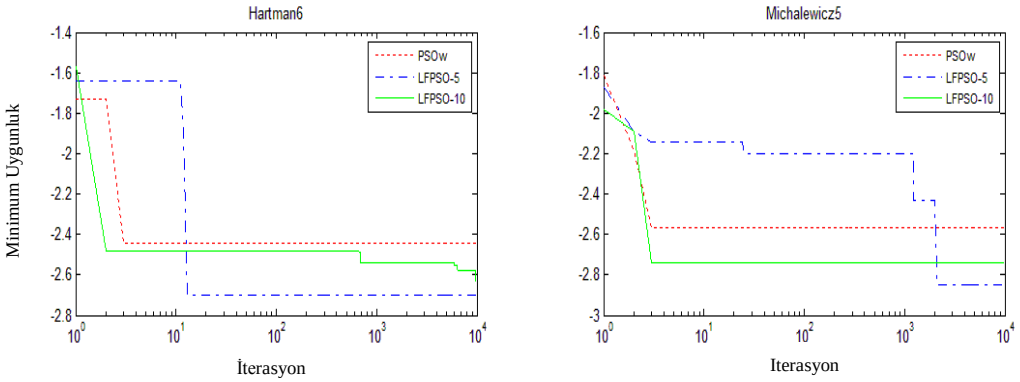
Şekil 3.3. 30 boyut için Rosenbrock, Rastrigin ve Griewank fonksiyonlarının yakınsama grafikleri

50 boyut için ise aynı 30 boyuttaki yorumlar geçerlidir ayrıca Rosenbrock fonksiyonu için daha iyi sonuç elde edildiği Şekil 3.4' de görülmektedir.



Şekil 3.4. 50 boyut için Rosenbrock, Rastrigin ve Griewank fonksiyonlarının yakınsama grafikleri

Sabit boyut için seçilen Hartmann6 ve Michalewicz5 fonksiyonlarının Şekil 3.5' de ki yakınsama grafiklerinde de önerilen yöntem daha hızlı yakınsamış fakat belli bir iterasyondan sonra yöntemler bir sonuçta takılmışlardır.



Şekil 3.5. Sabit boyut için Hartman6 ve Michalewicz5 fonksiyonlarının yakınsama grafikleri

3.2.1.5. Parametrik Olmayan Test Sonuçları

LFPSO-5 ve LFPSO-10 performanslarının PSO ile istatistiksel olarak bir karşılaştırmasını yapmak için parametrik olmayan istatistiksel test gerçekleştirildi. Wilcoxon testi 0.05 anlamlılık seviyesinde kullanıldı ve sonuçlar Çizelge 3.2, 3.3, 3.4

ve 3.5’ de gösterildi. Çizelgelerde Wilcoxon testi 0.05 anlamlılık seviyesinde ise + işareti, değil ise – işareti kullanıldı. İstatistiksel teste uygun olmayan, iki algoritmanın optimum değeri bulması veya aynı sonuçları elde etmesi durumu ise işaret konulmayarak belirtildi.

Sonuçlar incelendiğinde, önerilen LFPSO-10 algoritmasının performansı PSO ile karşılaştırıldığında 10 (F6 ve F7 hariç), 30 (F8 hariç), 50 ve sabit (F18 hariç) boyutlar için anlamlı bir biçimde iyi olduğu görüldü. Bu da iki yöntem arasındaki farkın istatistiksel olarak anlamlı olduğunu gösterir.

Sonuçlar incelendiğinde, önerilen LFPSO-5 algoritmasının performansı PSO ile karşılaştırıldığında 10 (F7 hariç), 30 (F2,F5 ve F9 hariç), 50 (F5 hariç) ve sabit (F18 hariç) boyutlar için anlamlı bir biçimde iyi olduğu görüldü. Bu da iki yöntem arasındaki farkın istatistiksel olarak anlamlı olduğunu gösterir.

3.2.1.5. Tüm Sonuçlar İçin Değerlendirme

Çizelge 3.2, 3.3, 3.4 ve 3.5‘ deki sonuçlar incelendiğinde önerilen LFPSO yönteminin PSO algoritmasına göre açıkça daha başarılı olduğu görülmektedir. Özellikle (F1-F9) arası fonksiyonlar için boyut sayısı arttıkça önerilen yöntem LFPSO’ nun daha etkin olduğu PSO algoritmasının ise boyut sayısı arttıkça etkinliği kaybettiği ve kötü sonuçlar ürettiği gözlemlenmektedir. Ayrıca standart sapmalara bakarak önerilen LFPSO yönteminin PSO’ ya göre daha gürbüz bir yöntem olduğu da görülür. LFPSO-5 ve LFPSO-10 arasındaki farklılığa dikkat çekmek gerekirse limit değerinin 10 seçilmesinin, 5 seçilmesine göre daha başarılı olduğu görülmektedir. Limit değerinin 5 seçilmesi ile amaçlanan limit değerinin algoritma sonuçlarını nasıl etkilediğini göstermektir.

Daha öncede söylendiği gibi yöntemin başarısını daha iyi kavrayabilmek adına test fonksiyonları çoğunlukla birçok yerel minimum ancak bir genel minimum içeren çokmodlu fonksiyonlardan seçildi. Çünkü PSO algoritması yerel minimumlara takılarak çokmodlu fonksiyonlar için başarısız sonuçlar vermekteydi. Önerilen LFPSO yöntemi ile dağılımlar sağlanarak bu takılmalar önlenmeye çalışıldı ve sonuçlardan da görüleceği gibi yöntemin çok modlu fonksiyonlarda başarılı olduğu görüldü.

Önerilen LFPSO yöntemi özellikle Rastrigin, Griewank, Shekel5, Shekel7 ve Shekel10 çok modlu fonksiyonları ve Schwefel1.2, Schwefel2.6 tek modlu fonksiyonlar için iyi bir seçim olacaktır.

3.2.2. SPSO ve LFPSO için testler ve sonuçlar

Orijinal PSO algoritması üzerinde ki çalışmalar sonucu Omran (2011) tarafından önerilen SPSO, rasgele bir topoloji kullanılarak geliştirilmiş geçerli bir standarttır. Daha detaylı bilgi ve kod için http://www.particleswarm.info/Programs.html adresinden ulaşılabilir. Yapılan çalışmalar orijinal PSO' nun birçok eksik noktasını tespit etmiş ve yeni önerilen PSO türlerinin daha gelişmiş bir PSO türü ile karşılaştırmasını neredeyse zorunlu hale getirmiştir. Bu nedenle önerilen yöntem LFPSO, hem SPSO hem de diğer PSO türleri ile kıyaslanarak sonuçlar verilmiştir.

3.2.2.1. Parametre ayarları

SPSO algoritması, parçacık sayısını fonksiyonun boyut değerini kullanarak belirlerken (3.7)'de ki denklemi kullanmaktadır.

$$NP = floor(10 + 2 * sqrt(D))) \tag{3.7}$$

Algoritmalar 30 ve 50 boyut için çalıştırıldı. SPSO algoritmasında 30 boyut için parçacık sayısı 20 iken, 50 boyut için 24 olarak alınmıştır. LFPSO algoritması için parçacık sayısı 40 olarak belirlendi. Durdurma kriteri olarak maksimum çevrim sayısı (*FEs*) kullanıldı ve tüm denemeler için 200,000 olarak ayarlandı. Maksimum çevrim sayısı ve iterasyon sayısı arasındaki ilişki *iterasyon = FEs / PS* ile ifade edilebilir. Amaç fonksiyonun her bir işlemi bir *FEs* olarak adlandırılırken, amaç fonksiyonun tüm parçacıklar için değerlendirilmesi ise iterasyonu ifade etmektedir. Ayrıca algoritmalar matematiksel olarak artık bir anlam ifade etmeyecek kadar küçük bir değer olan 10^{-18} hata oranına ulaştıkları takdir de algoritma durdurularak optimum değerin elde edildiği kabul edildi.

Maksimum çevrim sayısının, parçacık sayısına bölünmesi ile iterasyon sayısı elde edilir. Atalet ağırlığı için LFPSO (3.6)' daki denklemi kullanırken, SPSO ise (3.8)' deki denklemi kullanmaktadır.

$$w = 1 / (2 * log(2)) \tag{3.8}$$

3.2.1.3 numaralı başlık altında verilen sonuçlarda LFPSO' da limit değerinin 10 seçilmesinin, 5 seçilmesine göre daha etkin ve başarılı sonuçlar verdiği belirtilmişti. Bu nedenle SPSO ile karşılaştırma yapılırken, sadece limit değeri 10 olarak seçilen LFPSO-10 yöntemi kullanıldı. SPSO ' da V_{min} ve V_{max} için herhangi bir kriter bulunmamakta iken, LFPSO' da arama uzayının %20'si hız sınırı olarak kullanıldı. SPSO' da $c_1 = c_2 = 0.5 + log(2)$ şeklinde hesaplanmaktadır. Parametre değerleri ile ilgili detaylı bilgi Çizelge 3.6' da verilmektedir.

Çizelge 3.6. SPSO ve LFPSO için parametre değerleri

	SPSO	**LFPSO**
Parçacık Sayısı	30 Boyut için → 20 50 Boyut için → 24	30 Boyut için → 40 50 Boyut için → 40
Maksimum Çevrim Sayısı (FES)	200,000	200,000
Atalet Ağırlığı	w = 0,7213	$w = (max\_iter - iter) / max\_iter$
c_1 ve c_2	$c_1 = c_2$ = 1,1931	$c_1 = c_2$ = 2
Limit	-	10

3.2.2.2. Test fonksiyonları

Test için kullanılan fonksiyonlar Çizelge 3.7' da gösterilmektedir. 10 çok modlu (multimodal), 7 tek modlu (unimodal) ve 4 adet döndürülmüş (rotated) fonksiyon bulunmaktadır. Ayrıca popülasyonların ilk dağıtım aşamasında kullanılan arama uzayı başlangıç aralık değeri de verilmektedir. Parçacıklar ilk olarak bu uzayda dağıtılıp, daha sonra ilk verilen aralık değerine göre kontrol edilmektedir.

Çizelge 3.7. Test fonksiyonları (K:Karakteristik, T: Tek modlu, Ç:Çok modlu)

No	**Fonksiyon**	**Aralık**	**Başlangıç Aralık**	**K**	**Formülasyon**
1	Sphere	[-100, 100]	[-100,50]	T	$f_1 = \sum_{i=1}^{n} x_i^2$
2	Schwefel 2.22	[-10, 10]	[-10,5]	T	$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
3	Rosenbrock	[-10, 10]	[-10,10]	T	$f_3 = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]$
4	Noise	[-1.28, 1.28]	[-1.28, 0.64]	T	$f_4 = \sum_{i=1}^{n} i x_i^4 + random[0,1)$

5	Schwefel 2.26	[-500, 500]	[-500,500]	Ç	$f_5 = 418.98288727243369*\mathrm{n} - \sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
6	Rastrigin	[-5.12, 5.12]	[-5.12,2]	Ç	$f_6 = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
7	Ackley	[-32, 32]	[-32,16]	Ç	$f_7 = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right\} + 20 + e$
8	Griewank	[-600, 600]	[-600, 200]	Ç	$f_8 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
9	Penalized1	[-50, 50]	[-50,25]	Ç	$f_9 = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$ $+ (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)\ \ u_{x_i,a,k,m} = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(x_i - a)^m & x_i < -a \end{cases}$
10	Penalized2	[-50, 50]	[-50,25]	Ç	$f_{10} = \frac{1}{10}\{\sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]$ $+ (x_n - 1)^2[1 + \sin^2(2\pi x_{i+1})]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
11	Rotated Schwefel	[-500, 500]	[-500, 500]	D	$f_{11} = 418.9828*\mathrm{n} - \sum_{i=1}^{n} z_i,$ *where* $z_i = \begin{cases} y_i \sin(\sqrt{\lvert y_i \rvert}), \text{ if } \lvert y_i \rvert \le 500 \\ 0, \quad \text{otherwise} \end{cases}$, $y_i = y_i' + 420.96,$ *where* $y' = M*(x - 420.96)$, M *is an orthogonal matrix*
12	Rotated Rastrigin	[-5.12, 5.12]	[-5.12,2]	D	$f_{12} = \sum_{i=1}^{n}\left[y_i^2 - 10\cos(2\pi y_i) + 10\right]$ *where* $y = M*x$, M *is an orthogonal matrix*
13	Rotated Ackley	[-32, 32]	[-32,16]	D	$f_{13} = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi y_i)\right\} + 20 + e$ *where* $y = M*x$, M *is an orthogonal matrix*
14	Rotated Griewank	[-600, 600]	[-600,200]	D	$f_{14} = \frac{1}{4000}\sum_{i=1}^{n} y_i^2 - \prod_{i=1}^{n}\cos\left(\frac{y_i}{\sqrt{i}}\right) + 1$ *where* $y = M*x$, M *is an orthogonal matrix*
15	SumSquare	[-10, 10]	[-10,10]	T	$f_{15} = \sum_{i=1}^{n} i x_i^2$
16	Step	[-100, 100]	[-100,100]	T	$f_{16} = \sum_{i=1}^{n}\left(\lfloor x_i + 0.5 \rfloor\right)^2$
17	Quartic	[-1.28, 1.28]	[-1.28, 1.28]	T	$f_{17} = \sum_{i=1}^{n} i x_i^4$

18	Levy	[-10, 10]	[-10,10]	Ç	$f_{18}=\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\sin^2(3\pi x_1)+\lvert x_n-1\rvert[1+\sin^2(3\pi x_n)]$
19	Schaffer	[-100, 100]	[-100,100]	Ç	$f_{19}=0.5+\frac{\sin^2(\sqrt{\sum_{i=1}^{n}x_i^2})-0.5}{(1+0.001(\sum_{i=1}^{n}x_i^2))^2}$
20	Alpine	[-10, 10]	[-10,10]	Ç	$f_{20}=\sum_{i=1}^{n}\lvert x_i\cdot\sin(x_i)+0.1\cdot x_i\rvert$
21	Non-Continuous Rastrigin	[-5.12, 5.12]	[-5.12, 5.12]	Ç	$f_{21}=\sum_{i=1}^{n}\left[y_i^2-10\cos(2\pi y_i)+10\right]\ y_i=\begin{cases}x_i & \lvert x_i\rvert<\frac{1}{2}\\ \frac{round(2x_i)}{2} & \lvert x_i\rvert\geq\frac{1}{2}\end{cases}$

3.2.2.3. SPSO ve LFPSO yöntemlerinin sonuç ve karşılaştırmaları

Çizelge 3.7 ’de belirtilen 21 fonksiyon SPSO ve LFPSO yöntemleri karşılaştırmak için kullanıldı. Tüm fonksiyonlar 30 ve 50 boyutta 30 kere çalıştırılarak bulunan sonuçların ortalama değerleri alındı. Sonuçlar arasında istatistiksel olarak anlamsal fark olup olmadığını anlamak için parametrik olmayan Wilcoxon testi yapıldı.. Çizelge 3.8’ de Wilcoxon testi 0.05 anlamlılık seviyesinde ise + işareti, değil ise – işareti kullanıldı, tüm sonuçlar aynı değeri gösteriyor ya da optimum değeri bulduysa işaret konulmadı.

21 standart test fonksiyonun (F1-F21) 30 ve 50 boyut için 200,000 maksimum çevrim sayısı ile 30 denemedeki ortalama sonuçları Çizelge 3.8’ de verilmektedir. Hata oranı olarak 10^{-18} değeri kullanıldı ve algoritmalar bu değeri geçtikten sonra durdurularak optimum değere ulaşıldığı kabul edildi. Çizelge 3.8 ayrıca SPSO ve LFPS-10 ilgili test fonksiyonlarının 30 çalıştırmadaki ortalama optimum çözümü, standart sapmasını ve Wilcoxon testlerinde ki değerlerini göstermektedir. Algoritmalar tarafından test fonksiyonları için bulunan en iyi ortalama sonuç ve standart sapma değeri de kalın olarak gösterilmiştir.

Çizelge 3.8. SPSO ve LFPSO-10 algoritmaların 21 test fonksiyonu için sonuçları (Ort:Ortalama, Std. Sap.:Standart Sapma, Sign: İstatiksel Test İşareti)

F	D	Opt.	SPSO (PSO2007)		LFPSO-10		
			Ort.	Std. Sap.	Ort.	Std. Sap.	Sign
F1	30	0,00E+00	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
	50	0,00E+00	**0,00E+00**	**0,00E+00**	1,45E-14	4,88E-14	+
F2	30	0,00E+00	**0,00E+00**	**0,00E+00**	3,33E-01	1,83E+00	+
	50	0,00E+00	**0,00E+00**	**0,00E+00**	3,33E-01	1,83E+00	+
F3	30	0,00E+00	**8,91E+00**	**1,29E+01**	2,39E+01	3,26E-01	+
	50	0,00E+00	5,89E+01	3,21E+01	**4,40E+01**	**2,20E-01**	+
F4	30	0,00E+00	3,38E-03	1,55E-03	**2,42E-03**	**1,22E-03**	+
	50	0,00E+00	1,57E-02	7,76E-03	**5,75E-03**	**3,04E-03**	+
F5	30	0,00E+00	3,84E+03	6,75E+02	**1,74E+03**	**1,02E+03**	+
	50	0,00E+00	7,20E+03	1,06E+03	**4,49E+03**	**1,37E+03**	+
F6	30	0,00E+00	4,59E+01	1,40E+01	**3,47E+00**	**8,78E+00**	+
	50	0,00E+00	9,67E+01	2,50E+01	**2,17E+01**	**2,51E+01**	+
F7	30	0,00E+00	1,21E+00	9,03E-01	**1,58E-14**	**4,03E-15**	+
	50	0,00E+00	2,27E+00	5,35E-01	**2,35E-10**	**6,25E-10**	+
F8	30	0,00E+00	1,64E-02	2,54E-02	**0,00E+00**	**0,00E+00**	+
	50	0,00E+00	3,10E-02	4,87E-02	**1,26E-02**	**2,36E-02**	+
F9	30	0,00E+00	2,01E-01	4,38E-01	**0,00E+00**	**0,00E+00**	+
	50	0,00E+00	3,82E-01	4,85E-01	**1,80E-17**	**8,32E-17**	+
F10	30	0,00E+00	5,80E-02	2,93E-01	**0,00E+00**	**0,00E+00**	+
	50	0,00E+00	2,38E-01	7,32E-01	**8,78E-11**	**4,71E-10**	+
F11	30	0,00E+00	**5,29E+03**	**9,02E+02**	5,83E+03	5,17E+02	+
	50	0,00E+00	1,00E+04	9,91E+02	**9,86E+03**	**8,14E+02**	-
F12	30	0,00E+00	4,85E+01	1,36E+01	**4,38E+00**	**1,77E+01**	+
	50	0,00E+00	1,04E+02	3,91E+01	**8,05E-15**	**7,72E-15**	+
F13	30	0,00E+00	1,48E+00	8,58E-01	**1,63E-14**	**4,99E-15**	+
	50	0,00E+00	2,26E+00	5,87E-01	**3,53E-10**	**8,88E-10**	+
F14	30	0,00E+00	2,09E-02	2,39E-02	**3,02E-03**	**7,00E-03**	+
	50	0,00E+00	2,42E-02	4,56E-02	**3,69E-03**	**6,89E-03**	-
F15	30	0,00E+00	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
	50	0,00E+00	**0,00E+00**	**0,00E+00**	1,67E+01	9,13E+01	+
F16	30	0,00E+00	3,83E+00	7,24E+00	**0,00E+00**	**0,00E+00**	+
	50	0,00E+00	1,76E+01	3,04E+01	**0,00E+00**	**0,00E+00**	+
F17	30	0,00E+00	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
	50	0,00E+00	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
F18	30	0,00E+00	6,58E-02	1,84E-01	**0,00E+00**	**0,00E+00**	+
	50	0,00E+00	4,54E-01	1,12E+00	**4,31E-15**	**1,96E-14**	+

F19	30	0,00E+00	4,54E-02	1,67E-02	**9,07E-03**	**2,45E-03**	+
	50	0,00E+00	1,03E-01	4,28E-02	**1,82E-02**	**1,27E-02**	+
F20	30	0,00E+00	**2,69E-15**	**1,99E-15**	7,97E-08	4,36E-07	+
	50	0,00E+00	**7,59E-15**	**3,26E-15**	1,74E-05	9,05E-05	+
F21	30	0,00E+00	4,13E+01	1,18E+01	**2,07E+00**	**5,64E+00**	+
	50	0,00E+00	7,39E+01	2,28E+01	**1,41E+01**	**1,82E+01**	+

İlk olarak 30 boyut için iki yöntemi karşılaştırdığımız da, önerilen yöntem LFPSO-10 ‘nun fonksiyon 4,5,6,7,8,9,10,12,13,14,16,18,19 ve 21 olmak üzere 14 fonksiyon da daha başarılı olduğu görülmektedir. SPSO algoritması ise 2,3,11 ve 20 numaralı fonksiyonlarda daha başarılı sonuçlar verirken, geriye kalan 1,15 ve 17 numaralı fonksiyonlar için iki algoritma da hedeflenen değeri bulmaktadır. LFPSO-10 yönteminin 30 boyut için SPSO yöntemine oranla çok daha başarılı ve etkin sonuçlar ürettiği görülmektedir.

50 boyut için sonuçlara baktığımızda, SPSO yöntemi önerilen yönteme göre 1,2,15 ve 20 numaralı fonksiyonlarda daha başarılı olurken, F17 için iki algoritma da optimum değere ulaşmaktadır. Geriye kalan 16 adet test fonksiyonunda önerilen yöntem SPSO yöntemine göre daha başarılı sonuçlar vermektedir. Ayrıca standart sapma değerlerine bakılarak LFPSO-10 algoritmasının SPSO algoritmasına göre daha gürbüz olduğu söylenebilir.

Çizelge 3.8 ’deki tüm sonuçları incelediğimizde LFPSO-10 yönteminin SPSO yöntemine göre test fonksiyonlarının büyük bir çoğunluğun da daha başarılı ve gürbüz sonuçlar verdiği görülmektedir.

Önerilen LFPSO algoritmasının başarısını değerlendirmek için diğer PSO çeşitleri ile karşılaştırması yapılarak sonuçlar Çizelge 3.9 ’da verilmiştir. Verilen PSO algoritmalarının sonuçları Zhan ve ark. (2011) yaptığı çalışmadan alınmıştır. Algoritmaların karşılaştırmasında kullanılan fonksiyonlar, Çizelge 3.7 ’de verilen ilk 14 fonksiyondur. Tüm fonksiyonlar için boyut değeri 30 olarak belirlenmiştir. PSO’nun literatür özetinde bahsedilen CLPSO, HPSO-TVAC, FIPSO ve karşılaştırma yaptığımız SPSO ‘nun yanı sıra LPSO (Kennedy ve Mendes, 2002) ve DMS-PSO (Liang ve Suganthan, 2005) algoritmaları ile kıyaslanmıştır. SPSO algoritmasının yanında 40 yazmasının sebebi ise normal parçacık sayısını denklem (3.7)’ de verildiği gibi hesaplanmayıp, parçacık sayısının direk 40 olarak alındığını göstermektedir.

Tüm algoritmalar için parçacık sayısı 40 olarak belirlendi. Ayrıca algoritmaların hepsi her bir çalıştırma ve tüm fonksiyonlar için maksimum çevrim sayısı olarak

200,000 kullanıldı. İstatistiksel hataları azaltmak amacıyla, karşılaştırmada kullanılan ortalama sonuç ve tüm fonksiyonlar için, algoritmalar birbirinden bağımsız olarak 25 kere çalıştırıldı. Çizelge 3.9' da verilen sonuçlar için herhangi bir hata oranı kullanılmadı. Algoritmalar maksimum çevrim sayısına ulaşıncaya kadar çalıştırıldı.

Çizelge 3.9. LFPSO algoritmasının diğer PSO çeşitleri ile karşılaştırılması (Ort:Ortalama,Std. Sap.:Standart Sapma)

Fonksiyon		CLPSO	HPSO-TVAC	FIPSO	SPSO-40	LPSO	DMS-PSO	LFPSO
1	Ort.	1,58E-12	2,83E-33	2,42E-13	**2,29E-96**	3,34E-14	2,65E-31	4,69E-31
	Std. Sap.	7,70E-13	3,19E-33	1,73E-13	**9,48E-96**	5,39E-14	6,25E-31	2,50E-30
	Sıra	7	2	6	1	5	3	4
2	Ort.	2,51E-08	9,03E-20	2,76E-08	**1,74E-53**	1,70E-10	1,57E-18	2,64E-17
	Std. Sap.	5,84E-09	9,58E-20	9,04E-09	**1,58E-53**	1,39E-10	3,79E-18	6,92E-17
	Sıra	6	2	7	1	5	3	4
3	Ort.	**1,14E+01**	2,39E+01	2,51E+01	1,35E+01	2,81E+01	4,16E+01	2,38E+01
	Std. Sap.	**9,85E+00**	2,65E+01	5,10E-01	1,46E+01	2,18E+01	3,03E+01	3,17E-01
	Sıra	1	4	5	2	6	7	3
4	Ort.	5,85E-03	9,82E-02	4,24E-03	4,02E-03	2,28E-02	1,45E-02	**2,41E-03**
	Std. Sap.	1,11E-03	3,26E-02	1,28E-03	1,66E-03	5,60E-03	5,05E-03	**8,07E-04**
	Sıra	4	7	3	2	6	5	1
5	Ort.	**3,82E-04**	1,59E+03	9,93E+02	3,14E+03	3,16E+03	3,21E+03	1,37E+03
	Std. Sap.	**1,28E-07**	3,26E+02	5,09E+02	7,81E+02	4,06E+02	6,51E+02	6,36E+02
	Sıra	1	4	2	5	6	7	3
6	Ort.	**9,09E-05**	9,43E+00	6,51E+01	4,10E+01	3,51E+01	2,72E+01	4,54E+00
	Std. Sap.	**1,25E-04**	3,48E+00	1,34E+01	1,11E+01	6,89E+00	6,02E+00	1,03E+01
	Sıra	1	3	7	6	5	4	2
7	Ort.	3,66E-07	7,29E-14	2,33E-07	3,73E-02	8,20E-08	1,84E-14	**1,68E-14**
	Std. Sap.	7,57E-08	3,00E-14	7,19E-08	1,90E-01	6,73E-08	4,35E-15	**4,84E-15**
	Sıra	6	3	5	7	4	2	1
8	Ort.	9,02E-09	9,75E-03	9,01E-12	7,48E-03	1,53E-03	6,21E-03	**8,14E-17**
	Std. Sap.	8,57E-09	8,33E-03	1,84E-11	1,25E-02	4,32E-03	8,14E-03	**4,46E-16**
	Sıra	3	7	2	6	4	5	1
9	Ort.	6,45E-14	2,71E-29	1,96E-15	7,47E-02	8,10E-16	2,51E-30	**4,67E-31**
	Std. Sap.	3,70E-14	1,88E-29	1,11E-15	3,11E+00	1,07E-15	1,02E-29	**9,01E-31**
	Sıra	6	3	5	7	4	2	1
10	Ort.	1,25E-12	2,79E-28	2,70E-14	1,76E-03	3,26E-13	2,64E-03	**1,51E-28**
	Std. Sap.	9,45E-13	2,18E-28	1,57E-14	4,11E-03	3,70E-13	4,79E-03	**8,00E-28**
	Sıra	6	2	4	7	3	5	1
11	Ort.	4,39E+03	5,32E+03	4,41E+03	4,57E+03	4,50E+03	**4,04E+03**	5,51E+03
	Std. Sap.	3,51E+02	7,00E+02	9,94E+02	6,28E+02	3,97E+02	**5,68E+02**	5,64E+02
	Sıra	2	6	3	5	4	1	7

	Ort.	8,71E+01	5,29E+01	1,50E+02	4,34E+01	5,34E+01	4,20E+01	**1,79E+00**
12	Std. Sap.	1,08E+01	1,25E+01	1,45E+01	1,74E+01	1,40E+01	9,74E+00	**9,81E+00**
	Sıra	6	5	7	3	4	2	1
	Ort.	5,91E-05	9,29E+00	3,16E-07	9,24E-02	1,55E+00	2,42E-14	**1,65E-14**
13	Std. Sap.	6,46E-05	2,07E+00	1,00E-07	3,20E-01	4,50E-01	1,52E-14	**5,40E-15**
	Sıra	4	7	3	5	6	2	1
	Ort.	7,96E-05	9,26E-03	**1,28E-08**	3,05E-03	1,68E-03	1,02E-02	1,48E-03
14	Std. Sap.	7,66E-05	8,80E-03	**4,29E-08**	5,70E-03	3,47E-03	1,24E-02	6,17E-03
	Sıra	2	6	1	5	4	7	3
	Ort. Sıra	3,93	4,36	4,29	4,43	4,71	3,93	2,36
	Son Sıra	2	4	3	5	6	2	1
Algoritmalar		**CLPSO**	**HPSO-TVAC**	**FIPSO**	**SPSO-40**	**LPSO**	**DMS-PSO**	**LFPSO**

Çizelge 3.8, algoritmaların 14 fonksiyon için 25 bağımsız çalıştırmanın ortalama değerleri, standart sapma ve her algoritma için bulduğu değerlerin derecesine göre verilen sıra numarasını göstermektedir. Karşılaştırılan 7 algoritma buldukları ortalama değerlerine göre en iyiden en kötüye doğru sıralanmaktadır. Belirtilen fonksiyon için en iyi değeri bulan algoritma 1 ile en kötü algoritma ise 7 ile gösterilirken, diğer algoritmalar ise buldukları sonuçlara göre 1-7 arasında sıralanmaktadır. Ayrıca belirtilen fonksiyon için en iyi ortalama değeri bulan algoritma sonuçları ve standart sapma değerleri kalın yazı şekli ile gösterilmiştir.

SPSO-40 algoritması tek modlu (unimodal) fonksiyonlar olan Sphere ve Schwefel2.22 fonksiyonları için en iyi çözümü bulurken, DMS-PSO ise Rotated Schwefel fonksiyonu için ilk sırayı almaktadır. CLPSO algoritması ise Rosenbrock, Schwefel2.26 ve Rastrigin fonksiyonlarında en iyi çözüme sahipken, Ackley,Penalized1 ve Penalized2 fonksiyonlarında aynı başarı gösterememiştir. FIPSO, Rotated Griewank için en iyi sonuca ulaşırken, HPSO-TVAC algoritması hiçbir fonksiyon için en iyi çözümü elde edememiştir.

Önerilen LFPSO yöntemi ise ağırlıklı olarak çok modlu (multimodal) fonksiyonlarda başarılı olduğu gözlenirken Noise, Ackley, Griewank, Penalized1, Penalized2, Rotated Rastrigin ve Rotated Ackley fonksiyonlarında en iyi çözümü bulmuştur. Buna rağmen Rotated Schwefel fonksiyonu için ise en kötü çözüme sahiptir.

Çizelge 3.8' de verilen sonuçları incelediğimizde, önerilen yöntem LFPSO 'nun 14 fonksiyonun 7'sinde en iyi ortalama değeri bulduğu görülürken, 1,2 ve 11 numaralı fonksiyonların dışında ise kalan diğer fonksiyonlarda ilk 3 sırada içinde bulunmaktadır. LFPSO algoritmasının çokmodlu (multimodal) fonksiyonlarda daha başarılı olduğu

görülmektedir. Bu başarı LFPSO yöntemini anlatırken de bahsettiğimiz şekilde, PSO' nun global arama yeteneğinin Levy uçuşu yöntemi sayesinde artırılarak yerel minimumlardan kaçınarak sağlanmıştır.

Algoritmaların buldukları ortalama sonuçlar için yapılan sıralama sonuçlarının ortalama değerine baktığımızda 7 algoritma arasında LFPSO algoritmasının 1. sırada olduğu görülmektedir. Yani verilen 14 test fonksiyonu için Çizelge 3.8' de belirtilen algoritmalar arasında LFPSO yönteminin daha başarılı olduğu görülmektedir. LFPSO ' dan sonra sırasıyla CLPSO ve DMS-PSO, FIPSO, HPSO-TVAC, SPSO-40, son olarak ise LPSO algoritması gelmektedir.

3.2.2.4. Yakınsama Grafikleri

LFPSO ve SPSO algoritmaları için yakınsama grafikleri Şekil 3.6 ve Şekil 3.7 verilmektedir. Şekil 3.6 ' daki yakınsama grafiklerinde fonksiyonlar 50 boyut için, Şekil 3.7 için ise 30 boyut için çalıştırıldı.

Şekil 3.6' yı incelediğimiz de, tekmodlu olan Schwefel2.22 ve SumSquare fonksiyonların da SPSO algoritmasının hızlı bir şekilde yakınsadığı görülürken, LFPSO algoritması ise başarılı bir sonuç üretememektedir. Çizelge 3.8' de 50 boyut için Rosenbrock fonksiyonun SPSO algoritmasın için standart sapmasının yüksek olduğu bu nedenle gürbüz sonuçlar üretemediği görülmektedir. Yakınsama grafiğinde ise SPSO daha hızlı yakınsamasına rağmen ilerleyen iterasyonlarda iki algoritma da yakın değerler elde etmektedir. Tekmodlu olan Noise, çokmodlu olan Rastrigin, Griewank, Penalized2, Schaffer ve döndürülmüş olan Rotated Rastrigin fonksiyonları için genel olarak PSO algoritmasının hızlı yakınsama sebebiyle belirli bir değere takılıp kaldığını, LFPSO algoritmasının ise çözümü geliştirmesini devam ettirdiği görülmektedir. Şekil 3.6 özellikle çokmodlu fonksiyonları incelediğimiz de, PSO 'nun yerel bir minimuma takılarak çözümü geliştiremediğini, önerilen yöntemin ise yerel minimumlardan kaçınarak optimum çözüme doğru ilerlediğini göstermektedir.

Şekil 3.7' yi incelediğimiz de, çokmodlu olan Ackley, Penalized1, Levy ve Non-Continuous Rastrigin fonksiyonları için SPSO algoritmasının yerel minimumlara takıldığını, önerilen yöntemin ise çözümü geliştirmeye devam ettiği görülmektedir. Döndürülmüş fonksiyonlar Rotated Schwefel'ın her iki yöntem için yüksek standart sapma değerlerine sahip olması nedeniyle gürbüz sonuçlar vermemektedir, iki algoritmaların bulduğu sonuçlar anlamsal olarak bir farklılık ifade etmesi sebebiyle

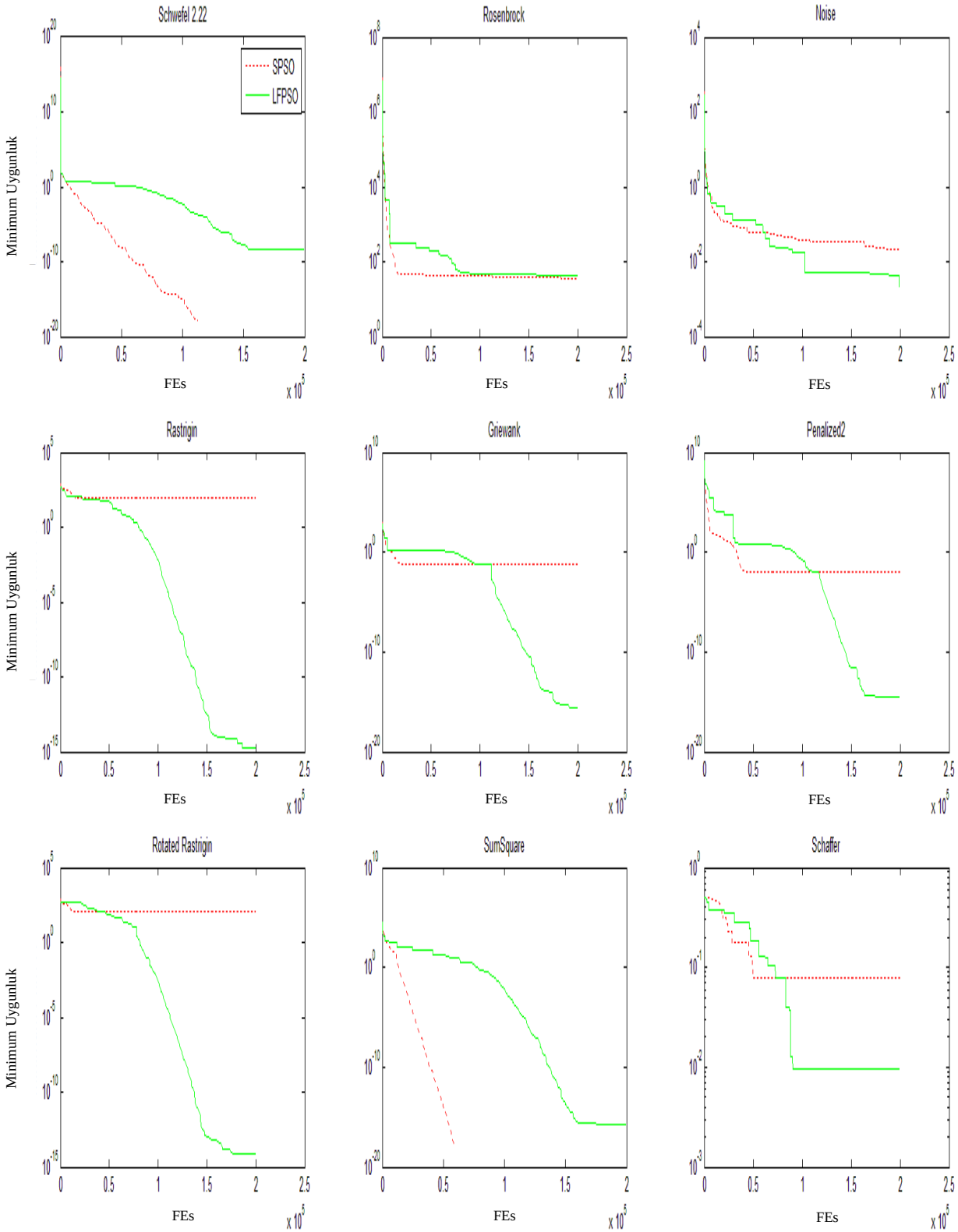
istatistik testi de geçememiştir. Bu nedenle yakınsama grafiğini yorumlamak çok doğru olmaz. Rotated Ackley için ise SPSO' nun yerel minimuma takılıp optimum çözümün uzağında kalırken, LFPSO daha ideal bir çözüm bulmuştur. Tekmodlu olan Schwefel2.26 ve Step fonksiyonların da ise SPSO yönteminin erken yakınsamasına rağmen LFPSO yönteminin daha başarılı sonuçlar elde ettiği görülmektedir.

Genel olarak yakınsama grafikleri yorumladığımız da özellikle çokmodlu ve döndürülmüş fonksiyonlar için SPSO algoritması hızlı bir şekilde yakınsamasına rağmen yerel minimuma takılıp kötü sonuçlar elde etmiştir. LFPSO algoritması ise yerel minimumlara takılmayarak çözümü geliştirmeye devam etmiş SPSO algoritmasına göre daha başarılı sonuçlar elde etmiştir. SPSO ise Schwefel2.22 ve SumSquare tek modlu fonksiyonlarda hızlı bir şekilde yakınsayarak optimum çözüme ulaşırken, LFPSO algoritması ise aynı başarıyı gösterememiştir.

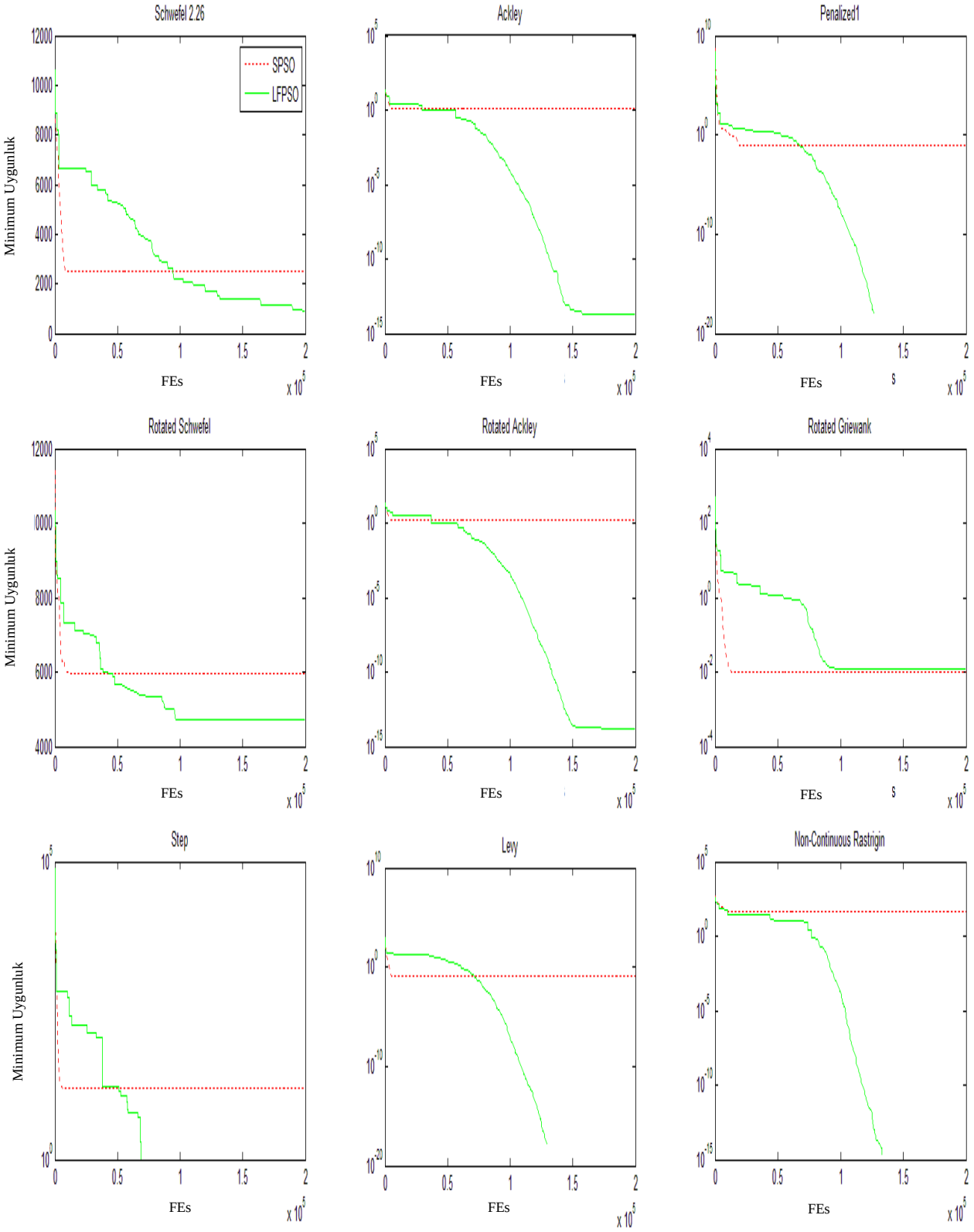
3.2.2.5. Parametrik Olmayan Test Sonuçları

LFPSO performansının SPSO ile istatistiksel olarak bir karşılaştırmasını yapmak için parametrik olmayan istatistiksel test gerçekleştirildi. Wilcoxon testi 0.05 anlamlılık seviyesinde kullanıldı ve sonuçlar Çizelge 3.8' de gösterildi. Çizelgelerde Wilcoxon testi 0.05 anlamlılık seviyesinde ise + işareti, değil ise – işareti kullanıldı. İstatistiksel teste uygun olmayan, iki algoritmanın optimum değeri bulması veya aynı sonuçları elde etmesi durumu ise işaret konulmayarak belirtildi.

Sonuçlar incelendiğinde, önerilen LFPSO algoritmasının performansı SPSO ile karşılaştırıldığında 30 boyut için tüm fonksiyonlarda, 50 boyut için ise F11 ve F14 hariç tüm fonksiyonlarda anlamlı bir biçimde iyi olduğu görüldü. Bu inceleme, LFPSO algoritmasının daha başarılı sonuçlar verdiği fonksiyonlar için söylendi. Bu da iki yöntem arasındaki farkın istatistiksel olarak anlamlı olduğunu gösterir.



Şekil 3.6. 50 boyut için belirtilen fonksiyonlarda SPO ve LFPSO yöntemlerinin yakınsama grafikleri



Şekil 3.7. 30 boyut için belirtilen fonksiyonlarda SPO ve LFPSO yöntemlerinin yakınsama grafikleri

4. YAPAY ARI KOLONİSİNDE KAŞİF ARILAR İÇİN LEVY UÇUŞU (LFABC) YÖNTEMİ

4.1. LFABC Algoritması

ABC algoritması global arama yeteneğini belirten keşfetme kısmında iyiyken, yerel arama yeteneğini belirten sömürme kısmında yeterince iyi değildir (Gao ve ark., 2012). Her bir besin kaynağı etrafında rasgele arayışlar yaparak global aramayı iyi bir şekilde yapabilmesine ve arama uzayını iyi bir biçimde gezebilmesine rağmen, sömürme kısmında bazı sorunlar yaşayarak özellikle çokmodlu (multimodal) fonksiyonlarda yerel minimumlara takılmaktadır.

Orijinal ABC algoritmasında, belli bir besin kaynağında daha önceden belirlenen limit değeri sayısı kadar bir gelişme göstermezse o kaynağa sahip isçi arı kaşif arıya dönüşür. Daha sonra bu kaşif arı sınırları belirlenen arama uzayı içerisinde rasgele olarak dağıtılır. Bu durumda kaşif arılar iterasyonlar sonucunda o ana kadar bulunan sonuçları kullanmadan rasgele olarak arama uzayı içerisinde bir bölge seçmektedir. Bu yeni seçilen besin kaynağının o ana kadar bulunan *GlobalMin*(o anki iterasyondaki en iyi çözüm)' den daha iyi olma olasılığı çok düşüktür. Uğuz ile birlikte yaptığımız çalışma da (2013) kaşif arıların daha etkin kullanılabilmesi için rasgele bir besin kaynağı seçmeleri yerine, *GlobalMin* den etkilenerek Levy uçuşu dağılımını kullanarak yeni bir besin kaynağı bulmaları sağlandı. Bu şekilde iyileştirilemeyen ve vazgeçilen besin kaynağı yerine o ana kadar bulunan en iyi besin kaynağı kullanılarak daha iyi bir çözüm elde edilmek istendi. Yeterince rassal arama yaparak keşfetme kısmı zaten iyi olan ABC algoritmasının sömürme kısmı biraz daha geliştirildi.

LFABC algoritması, orijinal ABC algoritması ile kaşif arı aşaması hariç aynıdır. Bu aşama da orijinal ABC algoritması kaşif arıları (2.5) 'de verildiği gibi arama uzayında rasgele dağıtırken, önerilen yöntem LFPSO algoritmasında da anlatıldığı gibi ve denklem (3.3) hesaplanan adım değerini kullanarak kaşif arıları dağıtır. Bu adım değeri hesaplanırken arıların o zamana kadar buldukları en iyi çözüm olan *GlobalMin* değeri kullanılması ile ABC algoritmasının yerel araması geliştirildi. Diğer bir durum ise LFPSO algoritmasında sıklıkla bahsedilen β parametresinin değeridir. β parametresi (0,2] aralığında alacağı değerler ile Levy dağılımının etkisini (3.3) verilen adım ölçüsünü belirlemektedir. β parametresinin küçük değerleri için adım ölçüsünün değeri artarken, büyük değerleri için ise azalmaktadır. Bu sebeple β parametresi LFABC

yöntemi için LFPSO algoritmasında kullanıldığı gibi (0,2] arasında rasgele bir değer almaması gerekir. LFABC yöntemi için istenen arıların o zamana kadar buldukları en iyi çözüm olan *GlobalMin* etrafında yerel bir aramadır. Eğer β parametresi için küçük değerler seçilirse bu yerel aramayı sağlamak mümkün olmaz. Bu bilgiler doğrultusun da düşünülerek β parametresinin alacağı değeri 1.5 olarak belirlenmiştir.

4.2. Testler ve Sonuçlar

Tüm denemeler Windows 7 Profesyonel İşletim Sistemi ortamında Intel i5, 2.4 GHz, 4GB RAM kullanarak Matlab 7.9' da kodlanan program ile gerçekleştirildi.

4.2.1. Parametre ayarları

Yapılan testlerde algoritmalar için popülasyon 50 olarak belirlendi. Fonksiyonlar için boyut sayısı 50 alındı. Durdurma kriteri olarak maksimum iterasyon sayısı 10,000 seçildi. ABC ve LFABC algoritmaları için limit parametresi 100 olarak ayarlandı. İstatistiksel hataları azaltmak amacıyla, karşılaştırmada kullanılan ortalama sonuç değerlerini belirlerken tüm fonksiyonlar için, algoritmalar birbirinden bağımsız olarak 30 kere çalıştırıldı. LFABC algoritmasındaki β parametresi için 1.5 değeri kullanıldı. Parametre ayarları ile bilgiler Çizelge 4.1' de verilmiştir.

Çizelge 4.1. ABC ve LFABC için parametre verileri

	ABC	**LFABC**
Popülasyon Boyutu	50	50
Limit	100	100
Tüm fonksiyonlar için boyut değeri	50	50
İterasyon Sayısı	10000	10000
Çalışma Sayısı	30	30
Durdurma Kriteri	Maksimum Iterasyon	Maksimum Iterasyon
β parametresi	-	1.5

4.2.2. Test fonksiyonları

Algoritmaları test etmek için kullanılan fonksiyonlar Çizelge 4.2' de verilmektedir. 5 tekmodlu, 4 çokmodlu ve 1 tane de kaydırılmış olmak üzere 10 fonksiyon seçildi.

Çizelge 4.2. Test Fonksiyonları (K: Karakteristik,T:Tekmodlu, Ç:Çokmodlu, KA:Kaydırılmış)

Aralık	K	Fonksiyon	Formülasyon
[-100,100]	T	Sphere	$f_1 = \sum_{i=1}^{n} x_i^2$
[-100,100]	T	Elliptic	$f_2 = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$
[-10,10]	T	Rosenbrock	$f_3 = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]$
[-5.12,5.12]	Ç	Rastrigin	$f_4 = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$
[-600,600]	Ç	Griewank	$f_5 = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$
[-32,32]	Ç	Ackley	$f_6 = -20\exp\left\{ -0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right\} - \exp\left\{ \frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right\} + 20 + e$
[-10,10]	T	SumSquare	$f_7 = \sum_{i=1}^{n} i x_i^2$
[-10,10]	T	Schwefel2.22	$f_8 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
[-10,10]	Ç	Alpine	$f_9 = \sum_{i=1}^{n} \lvert x_i \cdot \sin(x_i) + 0.1 \cdot x_i \rvert$
[-100,100]	KA	Shifted Sphere	$f_{10} = \sum_{i=1}^{n} z_i^2 \quad z = x - o$

4.2.3. Sonuçlar ve karşılaştırmalar

10 test fonksiyonun 50 boyut için 10,000 iterasyon ile 30 çalıştırmadaki ortalama sonuçları Çizelge 4.3' de verilmektedir. Çizelge 4.3 ayrıca ABC ve LFABC ilgili test fonksiyonlarının 30 çalıştırmadaki ortalama optimum çözümü, standart sapması, minimum ve maksimum değeri ve Wilcoxon testlerinde ki değerleri göstermektedir. Algoritmalar tarafından test fonksiyonları için bulunan en iyi ortalama sonuç ve standart sapma değeri de kalın olarak gösterilmiştir.

Çizelge 4.3. ABC ve LFABC algoritmaları için sonuçlar
(Ort:Ortalama, Std. Sap.:Standart Sapma, Min: Minimum, Max: Maksimum, Sign: İstatiksel Test İşareti)

Func.	Opt.		ABC	LFABC	Sign
f_1	0	Ort.	1,44E-15	**6,22E-16**	**+**
		Std. Sap.	1,87E-16	**7,4E-17**	
		Min	8,91E-16	5,16E-16	
		Max	1,85E-15	7,18E-16	
f_2	0	Ort.	1,44E-15	**5,53E-16**	**+**
		Std. Sap.	2,59E-16	**5,83E-17**	
		Min	9,75E-16	4,64E-16	
		Max	2,03E-15	6,93E-16	
f_3	0	Ort.	**0,023337**	0,027578	-
		Std. Sap.	**0,035322**	0,076159	
		Min	0,000754	2,11E-05	
		Max	0,177254	0,420718	
f_4	0	Ort.	5,24E-14	**0**	**+**
		Std. Sap.	1,09E-13	**0**	
		Min	0	0	
		Max	5,19E-13	0	
f_5	0	Ort.	7,8E-15	**0**	**+**
		Std. Sap.	1,03E-14	**0**	
		Min	1,11E-16	0	
		Max	4,27E-14	0	
f_6	0	Ort.	1,17E-13	**3,74E-14**	**+**
		Std. Sap.	1,66E-14	**3,58E-15**	
		Min	9,24E-14	3,2E-14	
		Max	1,63E-13	4,26E-14	
f_7	0	Ort.	1,45E-15	**6,04E-16**	**+**
		Std. Sap.	1,47E-16	**7,55E-17**	
		Min	1,17E-15	4,93E-16	
		Max	1,65E-15	7,18E-16	
f_8	0	Ort.	3,24E-15	**1,58E-15**	**+**
		Std. Sap.	3,5E-16	**8,79E-17**	
		Min	2,54E-15	1,41E-15	
		Max	3,79E-15	1,81E-15	
f_9	0	Ort.	3,37E-08	**7,3E-10**	**+**
		Std. Sap.	1,01E-07	**2,02E-09**	
		Min	8,79E-12	1,65E-15	
		Max	4,7E-07	1,06E-08	
f_{10}	0	Ort.	1,46E-15	**5,97E-16**	**+**
		Std. Sap.	2,21E-16	**7,55E-17**	
		Min	9,98E-16	4,95E-16	
		Max	1,88E-15	7,35E-16	

Çizelge 4.3 incelediğimizde LFABC algoritmasının, ABC algoritmasına göre fonksiyon 3 hariç diğer tüm fonksiyonlar da daha başarılı sonuçlar elde ettiği görülmektedir. Diğer önemli nokta ise çokmodlu olan Rastrigin ve Griewank fonksiyonlarda ABC algoritmasının yerel minimuma takılırken, önerilen yöntem LFABC algoritması optimum değeri elde etmiştir. Bu iki fonksiyon için önerilen yöntem ile istenen geliştirme sağlanmıştır. Diğer fonksiyonlar da ise çok büyük

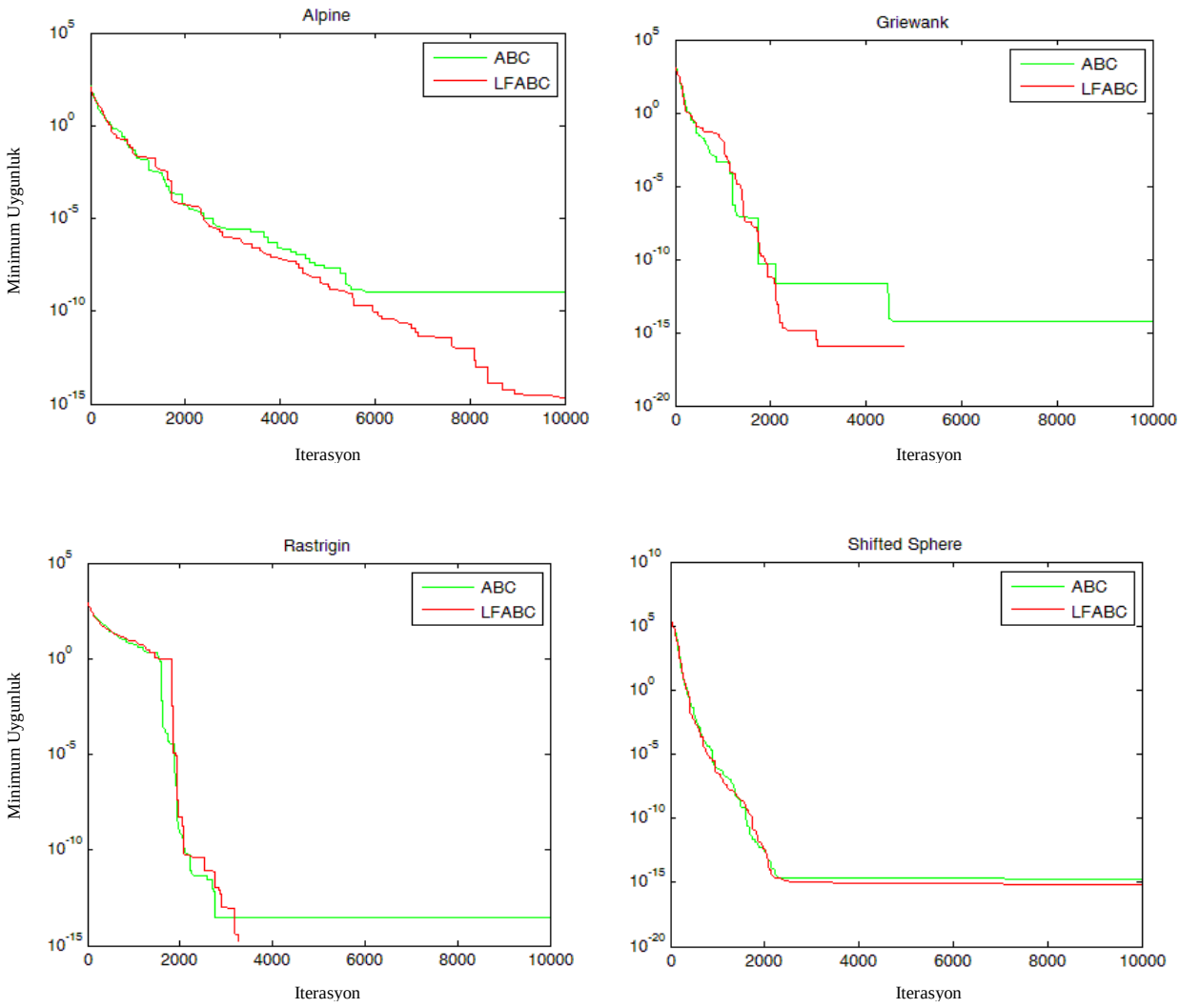
gelişmeler olmasa da LFABC algoritması ABC algoritmasına göre daha başarılı sonuçlar elde etmiştir.

LFABC algoritmasının daha kötü sonuç bulduğu tek fonksiyon Rosenbrock da iki algoritma sonuçlarının istatistiksel testi geçemediği görülmektedir. Bu da iki algoritmanın bu fonksiyon için gürbüz sonuçlar elde edemediğini gösterir.

Sonuçta ise 10 fonksiyonun 9'unda daha etkin sonuç gösteren LFABC algoritmasının ABC algoritması üzerinde başarılı bir geliştirme yaptığı görülmektedir.

4.2.4. Yakınsama grafikleri

Şekil 4.1 ' de ABC ve LFABC algoritmalarının Alpine, Griewank, Rastrigin ve Shifted Sphere fonksiyonları için yakınsama grafikleri verilmiştir. Grafikler incelediğinde Rastrigin ve Griewank için LFABC algoritması optimum değere ulaşırken, ABC algoritmasının yerel bir minimuma takıldığı görülmektedir.



Şekil 4.1. ABC ve LFABC algoritmaları için yakınsama grafikleri

Alpine fonksiyonun yakınsama grafiğinde ise iki algoritma birlikte yakınsamaya devam ederken ABC algoritması belli bir iterasyondan sonra çözüm geliştirmeyi başaramamış, önerilen yöntem ise daha başarılı bir sonuç elde etmiştir.

Son olarak Shifted Sphere fonksiyonun grafiğini incelediğimizde ise iki yöntem de aynı şekilde yakınsarken, LFABC algoritması çok az bir miktar olmasına rağmen daha iyi bir çözüm elde etmiştir.

4.2.5. Parametrik olmayan test sonuçları

LFABC yönteminin performansının ABC ile istatistiksel olarak bir karşılaştırmasını yapmak için parametrik olmayan istatistiksel test gerçekleştirildi. Wilcoxon testi 0.05 anlamlılık seviyesinde kullanıldı ve sonuçlar Çizelge 4.3' de gösterildi. Çizelgelerde Wilcoxon testi 0.05 anlamlılık seviyesinde ise + işareti, değil ise – işareti kullanıldı.

Sonuçlar incelendiğinde, önerilen LFABC algoritmasının performansı ABC ile karşılaştırıldığında Rosenbrock hariç (F3) tüm fonksiyonlarda, anlamlı bir biçimde iyi olduğu görüldü. Bu sonuçta iki yöntem arasındaki farkın istatistiksel olarak anlamlı olduğunu gösterir.

5. DEĞİŞKEN ARAMA TEKNİKLERİ İLE YAPAY ARI KOLONİSİ ALGORİTMASI (ABCVSS)

Algoritmaların test edilmesi için kullanılan test fonksiyonları tekmodlu, çok modlu, döndürülmüş, kaydırılmış vb. farklı karakteristik yapılara sahiptir. Her bir farklı fonksiyonun çözümü için kullanılan yöntemin arama uzayını farklı şekillerde kullanabilmesi, kimi zaman yerel aramaya kimi zaman da global aramaya ağırlık verebilmesi gerekmektedir. Algoritmalar da genellikle tek bir çözüm arama denklemi kullanıldığı için farklı karakteristik özellikli fonksiyonlarda aynı başarıyı gösterememektedirler. Bu sorunu çözebilmek için algoritmanın çözüm arama denklemini problemin yapısına göre değiştirebilmesi ve problemi çözme aşamasında farklı arama tekniklerini kullanabilmesi gerekmektedir. Kıran, M.S. ile yapılan çalışmalarımız da bu sorunu çözebilmek için farklı çözüm arama tekniklerini bir arada kullanarak, ABC algoritmasının farklı karakteristik yapıda ki problemler de başarılı olabilmesini sağlamak amacıyla Değişken Arama Teknikleri ile Yapay Arı Kolonisi (ABCVSS) algoritması önerilmiştir. Böylece algoritmanın yerel arama ve global arama dengesini kurması ve iyileşme hangi çözüm arama tekniğinde devam ediyorsa onun kullanılmasını sağlamıştır.

5.1. ABCVSS Algoritması

ABC algoritması hem işçi hem de gözcü arılar için sadece bir çözüm arama denklemi kullanır. Bu da tekmodlu, çokmodlu vb. gibi farklı yapıda ki fonksiyonlar için yeterli olmamaktadır. Bu sorunu çözebilmek için ABCVSS algoritması hem işçi hem de gözcü arılar için 5 farklı çözüm arama denklemi kullanmaktadır. Bunlar;

$$V_i^j(t+1) = X_i^j(t) + \phi x (X_i^j(t) - X_k^j(t)) \quad i = 1,2,...,SN, i \neq k \; and \; j \in \{1,2,...,D\} \tag{5.1}$$

$$V_i^j(t+1) = X_r^j(t) + \phi x (X_r^j(t) - X_k^j(t)) \quad i = 1,2,...,SN, r \neq k \; and \; j \in \{1,2,...,D\} \tag{5.2}$$

$$V_i^j(t+1) = X_r^j(t) + \phi x (X_i^j(t) - X_k^j(t)) \quad i = 1,2,...,SN, i \neq k \neq r \; and \; j \in \{1,2,...,D\} \tag{5.3}$$

$$V_i^j(t+1) = X_{best}^j(t) + \phi x(X_k^j(t) - X_r^j(t)) \quad i = 1,2,...,SN, i \neq k \neq r \; and \; j \in \{1,2,...,D\} \quad (5.4)$$

$$V_i^j(t+1) = X_i^j(t) + \phi x(X_i^j(t) - X_{mean}^j(t)) \quad i = 1,2,...,SN \; and \; j \in \{1,2,...,D\} \quad (5.5)$$

(5.1)' de çözüm arama denklemi, ABC algoritmasında kullanılan çözüm arama denklemidir. (5.2) (Gao ve ark., 2013) ' de *r* ve *k* indis değerleri ise [1,SN] aralığında rasgele belirlenen komşu arılardır ve o an ki besin kaynağını temsil eden indis *i* ile farklı bir değer almak zorundadır. Aynı şekilde (5.3) (Gao ve ark., 2013) ve (5.4) (Gao ve ark., 2012) çözüm arama denklemlerinde bulunan indis *r* ve *k* [1,SN] aralığında rasgele belirlenen komşu arılarken, $X_{best}^j(t)$ değeri ise o zamana kadar ki bulunan en iyi çözümün *j.* boyutunu temsil etmektedir. Son çözüm arama denklemi (5.5) 'de bulunan $X_{mean}^j(t)$ değeri ise popülasyondaki tüm arıların *j.* boyutlarının ortalama değerini belirtmektedir. ϕ ise [-1,1] aralığında belirlenen rasgele bir değerdir.

Literatürde bulunan farklı çözüm arama denklemleri de test edilerek, en iyi ve verimli çözümleri veren bu 5 çözüm arama denklemi seçilmiştir. İlk olarak orijinal ABC algoritmasında kullanılan çözüm arama denklemi kullanıldı. (5.2) denkleminde ise seçilen boyutun o anki arının *j.* boyut değeri ile değil de rasgele belirlenen komşu arının *j.* boyutuna göre arama yapılarak popülasyon içinde çeşitliliği artırmaktadır. Benzer şekilde (5.2) 'de olduğu gibi (5.3) 'de o anki arının *j.* boyut değerinin farklı bir komşudan çıkarılıp, farklı bir komşu değerine eklemesiyle popülasyon içindeki çeşitliliği sağlamaktadır. Global arama bu eşitlikler ile güçlendirilirken, yerel aramayı geliştirmek için (5.4) kullanıldı. (5.4) o zamana kadar ki bulunan en iyi çözüm değerinin etrafında arama yaparak yerel aramayı geliştirmektedir. (5.5) ise ilk başlarda arıların birbirinden çok uzak bulunması nedeniyle çeşitliliği sağlarken, sonlara doğru arıların iyi çözümlere yaklaşması ile birlikte, tüm arıların belli bir boyutunun ortalama değerlerinin alınmasıyla çözüme katkı sağlamaktadır.

Bu beş çözüm arama denklemi çeşitli ABC türlerinde yapıldığı gibi paralel olarak kullanılmamaktadır. Çünkü 5 çözümün her çevrim de hesaplanması algoritmanın çok fazla zaman harcamasına sebep olur. Hem bu sebeple hem de algoritmanın çözmeye çalıştığı probleme uygun çözüm arama denklemini kendi seçmesi sağlanmak istendi. Çözüm arama denklemlerin her bir çevrim de seçilme işlemi rulet tekerliği seçimi ile yapıldı. Her çözüm arama denklemi için bir sayaç değeri tutuldu. İlk olarak tüm çözüm

arama denklemlerinin sayaç değerleri 1 olarak belirlendi ve olasılıkları (5.6) ’daki gibi hesaplandı.

$$p(e_i) = \frac{C(c_i)}{\sum_{j=1}^{NE} C(c_j)} \quad i = 1,2,...,NE \tag{5.6}$$

$p(e_i)$ *i.* çözüm arama denkleminin seçilme olasılığını temsil ederken, *NE* çözüm arama denklemlerini sayısını ve $C(c_i)$ *i.* çözüm arama denkleminin sayaç değerini göstermektedir. Bu sayaç değeri toplam sayaç değerine bölünürken çözüm arama denklemlerinin seçim olasılıkları belirlenmektedir. Her bir çözüm arama denklemi için olasılıklar belirlendikten sonra, [0,1] arasında rasgele bir sayı üretilir. Belirlenen bu rasgele sayı ile (5.6) ’da belirlenen arama denklemlerinin olasılıklarını sırasıyla karşılaştırır. Eğer belirlenen rassal sayı, olasılık değerinden büyük ise arama denklemi seçilmez ve diğer arama denklemine geçilir. Bu şekilde olasılık değeri rasgele sayıdan büyük olan arama denklemi bulunana kadar işlem devam eder.

Başlangıçta olasılık değerleri eşit olan çözüm arama denklemleri, yukarda anlatılan yöntemle seçilir. ABCVSS algoritmasında ABC algoritmasında farklı olarak çözüm kalitesini hesaplamak için (2.8) ‘deki eşitlik kullanılmaz. ABCVSS aynı PSO algoritmasında olduğu gibi amaç fonksiyonundan gelen uygunluk değerini direkt kullanır. Seçilen çözüm arama denklemine göre besin kaynağının yeni pozisyonu belirlendikten sonra, eski besin kaynağının amaç fonksiyonunda gelen çözümü ile karşılaştırılır. Eğer yeni çözüm daha iyi ise, çözüm arama denkleminin sayaç değeri bir artırılır ve aynı ABC algoritmasındaki açgözlü seçim ile eski çözüm unutularak yeni çözüm hafızaya alınır. Bu işlem her çevrim de devam eder. İşçi arılar aşaması tamamlandıktan sonra gözcü arılara aşamasına geçmeden önce çözüm arama denklemlerinin olasılıkları (5.6) ile tekrar güncellenir. İşçi arılar aşamasının aynısı gözcü arılara da uygulanır. Kaşif arı aşaması ise ABC algoritmasında olduğu gibidir.

ABCVSS algoritmasının adımları aşağıda verilmiştir.

1. Başlangıç parametreleri(*limit* , $C(c_i)$, $x_j^{\max}$ ve $x_j^{\min}$ vb.) ayarla.
2. $i=1,2,...,SN$, $j=1,2,...,D$ olmak üzere besin kaynaklarının başlangıç değerleri Denklem (2.5) ’deki gibi rastgele şekilde arama uzayında belirle ve çözüm geliştirememe sayaçlarını sıfırla.

3. Her bir çözüm için amaç fonksiyonundaki değerleri olan f_i hesapla.
4. while(sonlandırma kriteri)
5. Her bir çözüm arama denklemi için 5.6 ile olasılıkları hesapla.
6. for *i=1:SN*

   Rulet tekerleği yöntemine göre çözüm arama denklemini seç.

   İşçi arılar için seçilen çözüm arama denklemi ile yeni besin kaynaklarını belirle.

   if(Yeni besin kaynağının çözümü> Eski besin kaynağın çözümü)

   Eski besin kaynağını hafızadan silerek yerine yeni besin kaynağını seç

   Geliştirememe sayacını sıfırla

   Seçilen çözüm arama denkleminin sayacını bir artır.

   else

   Geliştirememe sayacını bir artır

   end //if

   end //for
7. for *i=1:SN*

   İşçi arılardan besin kaynakları hakkında gelen bilgi ile Denklem (2.9) ile kaynakların p_i olasılıklarını belirle

   end //for
8. Her bir çözüm arama denklemi için 5.6 ile olasılıkları yeniden hesapla.
9. for *i=1:SN*

   Rulet tekerleği yöntemine göre çözüm arama denklemini seç.

   if(*rand*< p_i) (Olasılığı rasgele belirlenen sayıdan büyük olan besin kaynağı seç)

   Gözcü arılar için seçilen çözüm arama denklemi ile yeni besin kaynaklarını belirle.

   if(Yeni besin kaynağının çözüm kalitesi > Eski besin kaynağın çözüm kalitesi)

   Eski besin kaynağını hafızadan silerek yerine yeni besin kaynağını seç

   Geliştirememe sayacını sıfırla.

   Seçilen çözüm arama denkleminin sayacını bir artır.

   Else

Geliştirememe sayacını bir artır

end //if

10. if (max*(trial)*>*limit*) (Geliştirememe sayacı limit değerinden büyük olan işçi arıyı seç)

    Denklem (2.5)'e göre kaşif arı için yeni besin kaynağı belirle.

    end//if

11. En iyi çözümü hafızada tut
12. end // while

5.2. Testler ve Sonuçlar

Tüm denemeler Windows 8 Profesyonel İşletim Sistemi ortamında Intel i7, 2.4 GHz, 8GB RAM kullanarak Matlab R2012b' de kodlanan program ile gerçekleştirildi.

5.2.1. Parametre ayarları

Yapılan testlerde algoritmalar için koloni sayısı 40 olarak belirlendi. Fonksiyonlar için boyut (*D*) sayısı 60 alındı. Durdurma kriteri olarak maksimum çevrim sayısı (FEs) 300,000 seçildi. ABC ve ABCVSS algoritmaları için limit parametresi *SN*D* ayarlandı. *SN* besin kaynağı sayısıdır, koloninin yarısına ve işçi arıların sayısına eşittir. İstatistiksel hataları azaltmak amacıyla, karşılaştırmada kullanılan ortalama sonuç değerlerini belirlerken tüm fonksiyonlar için, algoritmalar birbirinden bağımsız olarak 30 kere çalıştırıldı. Parametre ayarları ile bilgiler Çizelge 5.1' de verilmiştir.

Çizelge 5.1. ABC ve LFABC için parametre verileri

	ABC	**ABCVSS**
Popülasyon Boyutu	40	40
Limit	SN*D	SN*D
Tüm fonksiyonlar için boyut değeri	60	60
Maksimum Çevrim Sayısı (FES)	300,000	300,000
Çalışma Sayısı	30	30
Durdurma Kriteri	Maksimum Çevrim Sayısı	Maksimum Çevrim Sayısı

5.2.2. Test fonksiyonları

Algoritmaları test etmek için kullanılan fonksiyonlar Çizelge 5.2 'de verilmektedir. Önerilen yöntemin farklı karakteristikteki fonksiyonlar da başarısını ölçebilmek için 2 tekmodlu, 2 çokmodlu, 2 kaydırılmış ve 4 tane de döndürülmüş olmak üzere 10 fonksiyon seçildi.

Çizelge 5.2. Test Fonksiyonları
(K:Karakteristik, T: Tek modlu, Ç:Çok modlu, KA: Kaydırılmış, D: Döndürülmüş)

No	Fonksiyon	Aralık	K	Formülasyon
1	Sphere	[-100, 100]	T	$f_1 = \sum_{i=1}^{n} x_i^2$
2	Elliptic	[-100,100]	T	$f_2 = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$
3	Griewank	[-600, 600]	Ç	$f_3 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
4	Weirstrass	[-0.5,0.5]	Ç	$f_4 = \sum_{i=1}^{D} (\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - D\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k 0.5)],\ a = 0.5,\ b = 3,\ k_{\max} = 20$
5	Shifted Sphere	[-100,100]	KA	$f_5 = \sum_{i=1}^{n} z_i^2 \quad z = x - o$
6	Shifted Alpine	[-10,10]	KA	$f_6 = \sum_{i=1}^{n} \lvert z_i \cdot \sin(z_i) + 0.1 \cdot z_i \rvert \quad z = x - o$
7	Rotated Schwefel	[-500, 500]	D	$f_{11} = 418.9828*\mathrm{n} - \sum_{i=1}^{n} z_i,$ *where* $z_i = \begin{cases} y_i \sin(\sqrt{\lvert y_i \rvert}), & if\ \lvert y_i \rvert \le 500 \\ 0, & otherwise \end{cases}$, $y_i = y_i' + 420.96,$ *where* $y' = M*(x - 420.96),\ M$ *is an orthogonal matrix*
8	Rotated Rastrigin	[-5.12, 5.12]	D	$f_{12} = \sum_{i=1}^{n} \left[y_i^2 - 10\cos(2\pi y_i) + 10 \right]$ *where* $y = M*x,\ M$ *is an orthogonal matrix*
9	Rotated Ackley	[-32, 32]	D	$f_{13} = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi y_i)\right\} + 20 + e$ *where* $y = M*x,\ M$ *is an orthogonal matrix*
10	Rotated Griewank	[-600, 600]	D	$f_{14} = \frac{1}{4000}\sum_{i=1}^{n} y_i^2 - \prod_{i=1}^{n} \cos\left(\frac{y_i}{\sqrt{i}}\right) + 1$ *where* $y = M*x,\ M$ *is an orthogonal matrix*

5.2.3. Sonuçlar ve karşılaştırmalar

10 test fonksiyonun 60 boyut için 300,000 FEs ile 30 çalıştırmadaki ortalama sonuçları Çizelge 5.3 ’de verilmektedir. Çizelge 5.3 ayrıca ABC ve ABCVSS ilgili test fonksiyonlarının 30 çalıştırmadaki ortalama optimum çözümü, standart sapması ve Wilcoxon testlerinde ki değerleri göstermektedir. Algoritmalar tarafından test fonksiyonları için bulunan en iyi ortalama sonuç ve standart sapma değeri de kalın olarak gösterilmiştir.

Çizelge 5.3. ABC ve ABCVSS algoritmaları için sonuçlar
(Ort:Ortalama, Std. Sap.:Standart Sapma, Min: Minimum, Max: Maksimum, Sign: İstatiksel Test İşareti)

F	Opt.	ABC		ABCVSS		
		Ort.	Std. Sap.	Ort	Std. Sap.	Sign
F1	0,00E+00	1,24E-15	1,16E-16	**1,09E-83**	**5,01E-83**	+
F2	0,00E+00	1,15E-15	1,50E-16	**1,01E-82**	**5,54E-82**	+
F3	0,00E+00	3,74E-16	2,92E-16	**0,00E+00**	**0,00E+00**	+
F4	0,00E+00	1,75E-14	1,03E-14	**0,00E+00**	**0,00E+00**	+
F5	0,00E+00	1,16E-15	1,63E-16	**0,00E+00**	**0,00E+00**	+
F6	0,00E+00	1,04E-07	2,24E-07	**2,86E-16**	**3,41E-16**	+
F7	0,00E+00	9,63E+03	5,24E+02	**9,22E+03**	**3,83E+02**	+
F8	0,00E+00	2,87E+02	2,65E+01	**2,50E+02**	**2,80E+01**	+
F9	0,00E+00	1,13E+01	7,84E+00	**2,66E+00**	**8,81E-01**	+
F10	0,00E+00	**8,78E-07**	**1,74E-06**	3,32E-06	7,28E-06	-

Çizelge 5.3 incelediğimizde, ABCVSS algoritmasının Rotated Griewank hariç diğer tüm fonksiyonlarda çok üstün bir başarı sağladığı görülmektedir. Rotated Griewank fonksiyon da ise iki yöntemin elde ettiği sonuçların istatistiksel olarak anlamlı olmadığı görülmektedir. Bu, iki algoritmanın F10 fonksiyonu için birbirine çok benzer sonuçlar ürettiğini gösterir. Özellikle Griewank, Weirstrass ve Shifted Sphere için ABC algoritması uygun değerleri üretemezken, ABCVSS algoritması optimum değerleri 30 çalışma için de elde etmiştir. Diğer fonksiyonlarda ise ABCVSS algoritmasının orijinal ABC algoritmasına göre çok daha iyi sonuçlar elde ederek farklı karakteristikteki fonksiyonlar da başarılı olduğu görülmüştür.

ABCVSS yöntemi enerji tahmin problemine uygulanacağı için test fonksiyonları çok genişletilmemiştir. Sadece yöntemin başarısını gösterebilmek adını farklı tipte ve az sayıda fonksiyon seçilmiştir.

5.2.4. Parametrik olmayan test sonuçları

ABCVSS yönteminin performansının ABC ile istatistiksel olarak bir karşılaştırmasını yapmak için parametrik olmayan istatistiksel test gerçekleştirildi. Wilcoxon testi 0.05 anlamlılık seviyesinde kullanıldı ve sonuçlar Çizelge 5.3' de gösterildi. Çizelgelerde Wilcoxon testi 0.05 anlamlılık seviyesinde ise + işareti, değil ise – işareti kullanıldı.

Sonuçlara bakarak, önerilen ABCVSS algoritmasının performansı ABC ile karşılaştırıldığında Rotated Griewank hariç (F10) tüm fonksiyonlarda, anlamlı bir biçimde iyi olduğu görüldü. Bu da iki yöntem arasındaki farkın istatistiksel olarak anlamlı olduğunu gösterir.

5.3. ABCVSS Algoritmasının Enerji Tahmin Problemine Uygulanması

Bir ülkenin ekonomik, sosyal ve teknolojik gelişimi için birçok şeyin kaynağı olan enerjinin önemi çok büyüktür. Özellikle ekonomisi ve popülasyonu hızla büyüyen Türkiye gibi ülkelerde enerjinin rolü daha fazladır. Avrupa ve Asya kıtalarının arasında bulunan Türkiye genç nüfusu, hızlı kentleşme ve büyüyen ekonomisi ile enerji tüketiminde önemli bir rol oynamaktadır (Ceylan ve Öztürk, 2004).

Yerli enerji kaynakları, Türkiye'nin enerjini tüketimini karşılamamaktadır. Bu nedenle Türkiye enerji ithalatına bağımlı olup enerji kaynaklarının 2/3'ünü dışardan ithal etmektedir (Ceylan ve Öztürk, 2004). 2020 yılında yerel enerji tüketiminin sadece %30 'nun yerel üretim ile karşılanması beklenmektedir (Kıran ve ark., 2012b). Güvenli ve yeterli enerji kaynaklarını başarılı bir şekilde belirlemek Türkiye'nin en önemli enerji politikalarından biridir. Yani Türkiye'nin enerji talep tahminini doğru bir şekilde yapmak çok önemlidir. Eğer ülkelerin gelecekteki enerji talepleri düzgün bir şekilde hesaplanırsa, böylece o ülkeler eldeki enerji kaynakları verimli bir şekilde kullanabilir ve kendi taleplerini karşılayacak enerji kaynaklarını araştırabilirler (Kıran ve Gunduz, 2013).

Enerji talebi tahmini için kullanılan temel göstergeler gayri safi yurtiçi hasıla (GSYH), nüfus, ithalat ve ihracat değerleridir. Ekonomik göstergeler tabanlı enerji talebi tahmininde çeşitli denklemler ile modellenebilmektedir (Toksarı, 2007; Ünler, 2008). Bu denklemler lineer ve lineer olmayan şekilde olabilir. Lineer olmayan

denklemler ekonomik göstergelerin dalgalanmaları sebebiyle enerji talep tahmininde daha iyi sonuçlar üretebilir (Ceylan ve Öztürk, 2004).

Bu tez çalışmasında PSO, ABC ve ABCVSS yöntemleri daha iyi sonuçlar üreten kuadratik denklem kullanılarak Türkiye'nin enerji talep tahmin problemine uygulanacaktır.

Türkiye enerji talebi tahminindeki ilk uygulamaları Devlet Planlama Organizasyonu tarafından basit regresyon teknikleri kullanılarak tamamlandı. 1984' den başlayarak birçok ekonometrik modelli teknikler enerji talep tahmini için kullanıldı. Enerji ve Tabii Kaynaklar Bakanlığı tarafından teknik enerji tahminin analizi en çok kullanılan modeldir. Fakat bu model tarafından belirlenen enerji talep tahminlerinin olduğundan çok fazla talep miktarı hesaplamaktadır (Ünler, 2008). Bu da Türkiye'nin daha fazla ithalat bağımlığı kalmasına ve enerji piyasalarının serbestleştirilmesinin önlenmesine sebep olur.

Bu nedenle enerji talep tahmini, tüketimi vb. alanlarda birçok çalışmalar bulunmaktadır. Genellikle meta-sezgisel olmak üzere yapay zeka yöntemleri ile enerji talep tahmini yapılarak daha başarılı ve tutarlı sonuçlar elde edilmeye çalışılmıştır. Ceylan ve Öztürk (2004) genetik algoritma ile, Toksarı (2007) karınca kolonisi optimizasyonu ile, Ünler (2008) parçacık sürü optimizasyonu ile, Kıran ve ark.(2012, 2013) çeşitli hibrit yöntemler kullanarak Türkiye'nin enerji talebini tahmin etmeye çalışmışlardır. Ayrıca enerji talep tahmini yapıldığı gibi elektrik (Erdoğdu, 2007), gaz (Aras ve Aras, 2004), petrol (Özçelik ve Hepbaşlı, 2006) vb. ihtiyaçlarında tüketim ve talep miktarları tahmin edilmiştir.

Bir ülkenin enerji talebinin en çok etkileyen ekonomi tabanlı 4 gösterge GSYH, nüfus, ihracat ve ithalat verileri olduğu bilinmektedir. 1979 ve 2005 yılları arasında Türkiye için GSYH, nüfus, ihracat, ithalat verileri ve Türkiye'nin enerji talep değeri Çizelge 5.4' verilmektedir. Veriler Türkiye İstatistik Kurumu ve Enerji ve Tabii Kaynaklar Bakanlığından toplanmıştır.

Çizelge 5.4. Türkiye'nin enerji talep, GSYH, nüfus, ithalat ve ihracat verileri

Yıllar	*Enerji Talebi (MTEP)*	*GSYH($10^9)*	*Nüfus (10^6)*	*İthalat($10^9)*	*İhracat($10^9)*
1979	30,71	82,00	43,53	5,07	2,26
1980	31,97	68,00	44,44	7,91	2,91
1981	32,05	72,00	45,54	8,93	4,70

1982	34,39	64,00	46,69	8,84	5,75
1983	35,70	60,00	47,86	9,24	5,73
1984	37,43	59,00	49,07	10,76	7,13
1985	39,40	67,00	50,31	11,34	7,95
1986	42,47	75,00	51,43	11,10	7,46
1987	46,88	86,00	52,56	14,16	10,19
1988	47,91	90,00	53,72	14,34	11,66
1989	50,71	108,00	54,89	15,79	11,62
1990	52,98	151,00	56,10	22,30	12,96
1991	54,27	150,00	57,19	21,05	13,59
1992	56,68	158,00	58,25	22,87	14,72
1993	60,26	179,00	59,32	29,43	15,35
1994	59,12	132,00	60,42	23,27	18,11
1995	63,68	170,00	61,53	35,71	21,64
1996	69,86	184,00	62,67	43,63	23,22
1997	73,78	192,00	63,82	48,56	26,26
1998	74,71	207,00	65,00	45,92	26,97
1999	76,77	187,00	66,43	40,67	26,59
2000	80,50	200,00	67,42	54,50	27,78
2001	75,40	146,00	68,37	41,40	31,33
2002	78,33	181,00	69,30	51,55	36,06
2003	83,84	239,00	70,23	69,34	47,25
2004	87,82	299,00	71,15	97,54	63,17
2005	91,58	361,00	72,97	116,77	73,48

Çizelge 5.4’ de Türkiye’nin sürekli bir gelişim içinde olduğu ekonomik göstergelerin yıllar geçtikçe arttığı gözlemlenebilir. Ekonomik göstergeler tabanlı enerji talep tahmini lineer (5.7) ve kuadratik (5.8) ile farklı biçimlerle modellenmiştir. Lineer biçimi şu şekilde ifade edilebilir;

$$E_{lineer} = w_1 + w_2 X_1 + w_3 X_2 + w_4 X_3 + w_5 X_4 \tag{5.7}$$

Kuadratik denklemli form şu şekilde ifade edilir;

$$\begin{aligned} E_{kuadratik} = {} & w_1 + w_2 X_1 + w_3 X_2 + w_4 X_3 + w_5 X_4 + w_6 X_1 X_2 \\ & + w_7 X_1 X_3 + w_8 X_1 X_4 + w_9 X_2 X_3 + w_{10} X_2 X_4 \\ & + w_{11} X_3 X_4 + w_{12} X_1^2 + w_{13} X_2^2 \\ & + w_{14} X_3^2 + w_{15} X_4^2 \end{aligned} \tag{5.8}$$

Enerji talep tahmininde amaç, veriler için en uygun değerleri bulmaktır. Denklem (5.7) ve (5.8) 'de X_1, X_2, X_3 ve X_4 değerleri GSYH, nüfus, ithalat ve ihracat değerleridir. Bu değerlere göre verilen yıllar için en uygun enerji talebi tahminini yapacak ağırlık değerleri doğa esinli algoritmalar ile hesaplanmaktadır. Kullanılan amaç fonksiyonu (5.9) 'da verilmektedir.

$$\min f(v) = \sum_{r=1}^{R} (E_r^{gözlenen} - E_r^{tah\min})^2 \tag{5.9}$$

$E_r^{gözlenen}$ ve $E_r^{tah\min}$ değerleri gerçek ve tahmin edilen değerleri gösterirken, *r* gözlemlerin sayısını tutmaktadır. 1979-2005 arası toplam da 27 yıl olduğu için her yıl için hata oranı hesaplanıp bunun karesi alınarak toplanır. Bu da bize amaç fonksiyonu değerini vermektedir.

GSYH, nüfus, ithalat ve ihracat göstergelerine dayalı enerji talep tahmini için ABCVSS yöntemi kuadratik model ($ABCVSS_{kuadratik}$) için gerçekleştirildi. Ayrıca yöntemin başarısını karşılaştırmak için orijinal ABC ve PSO algoritmaları için de ($PSO_{kuadratik}$ ve $ABC_{kuadratik}$) problem çalıştırıldı.

Bölüm 5.1 'de ABCVSS algoritması başlığı altında anlatılan ABCVSS algoritması tek bir fark ile aynı şekilde kullanıldı. ABCVSS algoritması tek bir boyut değiştirerek işlem yaptığı için bu tarz problemlere uyum sağlaması için denklem (5.1) 'de verilen ifade aşağıdaki gibi güncellendi.

$$V_i\ (t+1) = X_i\ (t) + \phi x (X_i\ (t) - X_k (t)) \quad i = 1, 2, ..., SN, i \neq k \tag{5.10}$$

(5.1) 'de çözüm arama denklemi tek bir boyut değiştirirken, (5.10) 'da verilen çözüm arama denklemi tüm boyutlar da güncelleme yapmaktadır. ABCVSS yöntemindeki diğer 4 çözüm arama denkleminde herhangi bir değişiklik yapılmamıştır. Aynı şekilde ABC algoritmasında da (5.1) 'deki eşitlik kullanıldığında çok kötü sonuçlar elde edildiği için ABC algoritmasının çözüm arama denklemi de (5.10) 'da ki gibi değiştirilerek tek boyut için değil tüm boyutlar için güncelleme işlemi yapıldı.

Algoritmalar için popülasyon sayısı 100 olarak belirlenirken, iterasyon sayısı olarak 1000 seçildi. PSO algoritması için c_1 ve c_2 2 olarak belirlendi. ABC ve ABCVSS

algoritmaları için limit değeri *SNxD* olarak alındı. Algoritmalar verilen parametreler ile 10 kez test edildikten sonra en iyi sonucu veren değerler kullanılmıştır.

Üç yöntemde 10 kez çalıştırıldı ve elde edilen en iyi, en kötü, ortalama ve standart sapma değerleri Çizelge 5.5 ‘de verildi.

Çizelge 5.5. PSO,ABC,ABCVSS yöntemlerinin kuadratik model için 10 çalıştırma sonuçları

PSO				ABC				ABCVSS			
En iyi	En kötü	Ort.	Std. Sap.	En iyi	En kötü	Ort.	Std. Sap.	En iyi	En kötü	Ort.	Std. Sap.
1664,40	54615	15844	17473	576,52	187940	34278	59567	**57,67**	**1249**	**485,06**	**386,80**

Çizelge 5.5 ’de görüldüğü gibi önerilen yöntem daha başarılı sonuçlar elde ederken PSO algoritmasının başarısının en düşük olduğu görülmektedir. Yöntemler için 10 çalıştırma sonucunda en iyi sonucu veren katsayılar ve amaç fonksiyonu sonuçları 5.11, 5.12 ve 5.13 ‘de verilmektedir.

$$\begin{aligned} PSO_{kuadratik} &= 13,59134 + 0,22627X_1 + 0,40806X_2 + 0,30044X_3 - 0,84979X_4 \\ &\quad - 0,03044X_1X_2 - 0,17613X_1X_3 + 0,13456X_1X_4 + 0,19481X_2X_3 \\ &\quad - 0,12228X_2X_4 - 0,09912X_3X_4 + 0,01502X_1^2 + 0,00529X_2^2 \\ &\quad + 0,26645X_3^2 - 0,20933X_4^2 \end{aligned} \tag{5.11}$$

$$f(v) = 1664,40$$

$$\begin{aligned} ABC_{kuadratik} &= 0,17179 - 0,02700X_1 + 0,42170X_2 - 0,05489X_3 - 0,22583X_4 \\ &\quad + 0,00086X_1X_2 + 0,02313X_1X_3 + 0,01560X_1X_4 - 0,01381X_2X_3 \\ &\quad - 0,00747X_2X_4 - 0,57614X_3X_4 - 0,00298X_1^2 + 0,01159X_2^2 \\ &\quad + 0,12462X_3^2 + 0,44860X_4^2 \end{aligned} \tag{5.12}$$

$$f(v) = 576,52$$

$$\begin{aligned} ABCVSS_{kuadratik} &= -16,47831 + 0,08976X_1 + 0,15425X_2 - 0,06475X_3 + 0,34884X_4 \\ &\quad + 0,00835X_1X_2 + 0,04160X_1X_3 - 0,02684X_1X_4 - 0,03061X_2X_3 \\ &\quad - 0,00317X_2X_4 + 0,01266X_3X_4 - 0,00447X_1^2 + 0,01469X_2^2 \\ &\quad - 0,06613X_3^2 + 0,06695X_4^2 \end{aligned} \tag{5.13}$$

$$f(v) = 57,67$$

Yöntemlerin başarısını ve geçerliliğini karşılaştırmak için 1996-2005 yılları arasındaki veriler ile 5.11, 5.12 ve 5.13 ‘de olan yöntemlerin bulduğu katsayı değerleri ile enerji talep tahminleri yapıldı. Bu tahminler, o yıllar için gözlenen gerçek enerji talep miktarları ile karşılaştırarak hata miktarları ve bağıl hatalar Çizelge 5.6 ’da verildi.

Çizelge 5.6. PSO, ABC, ABCVSS yöntemlerinin 1996-2005 yılları arası tahminlerinin doğrulanması

		PSO			ABC			ABCVSS		
Yıllar	Gözlenen Enerji Talebi	Enerji Talep Tahmini	Bağıl Hata (%)	Hata Miktarı	Enerji Talep Tahmini	Bağıl Hata (%)	Hata Miktarı	Enerji Talep Tahmini	Bağıl Hata (%)	Hata Miktarı
1996	69,86	62,056	-12,58	7,80	67,535	-3,44	2,33	**71,610**	**2,44**	**1,75**
1997	73,78	70,258	-5,01	3,52	68,579	-7,58	5,20	**72,676**	**-1,52**	**1,10**
1998	74,71	65,170	-14,64	9,54	74,269	-0,59	0,44	**75,000**	**0,39**	**0,29**
1999	76,77	68,486	-12,10	8,28	75,616	-1,53	1,15	**75,670**	**-1,45**	**1,10**
2000	80,50	87,912	8,43	7,41	77,001	-4,54	3,50	**78,877**	**-2,06**	**1,62**
2001	75,40	64,406	-17,07	10,99	77,580	2,81	2,18	**76,154**	**0,99**	**0,75**
2002	78,33	80,171	2,30	1,84	75,380	-3,91	2,95	**79,049**	**0,91**	**0,72**
2003	83,84	82,734	-1,34	1,11	90,587	7,45	6,75	**84,731**	**1,05**	**0,89**
2004	87,82	104,114	15,65	16,29	78,397	-12,02	9,42	**84,945**	**-3,38**	**2,87**
2005	91,58	82,984	-10,36	8,60	102,864	10,97	11,28	**92,991**	**1,52**	**1,41**

Çizelge 5.6 ’yı incelediğimiz de, 10 yıl içinde en az hata miktarı ve en az bağıl hataya sahip olan yöntemin ABCVSS olduğu görülmektedir. ABC algoritmasının genellikle yıllar ilerledikçe hata miktarı artmakta olduğu gözlemlenebilir. ABCVSS algoritmasının ise 2004 yılında dışında bağıl hata oranını %2’nin altında tuttuğunu Çizelge 5.6 göstermektedir.

Yöntemlerin doğruluğu hesaplandıktan sonra gelecek yılların enerji talep tahminin yapabilmek için bir senaryo belirlemek gerekmektedir. Belirlenen bu senaryo ile kullanılan GSYH, nüfus, ithalat ve ihracat değerleri hesaplanarak yöntemlerin bulduğu katsayılar ile enerji talep tahmin değerleri belirlenecektir. Kıran ve ark. (2013) çalışmasında 2006-2015 yılları arasında Türkiye için GSYH ‘nin büyüme oranı %6, nüfusun büyüme oranı %0.17, ithalatın büyüme oranı %4.5 ve ihracatın büyüme oranının ise %2 olduğu varsaymışlardır. Bu senaryo sonucu elde edilen verilen Çizelge 5.7 ’de verilmektedir.

Çizelge 5.7. Verilen senaryo için 2006-2015 yılları arasındaki GSYH, nüfus, ithalat ve ihracat değerleri

Yıllar	*GSYH($10^9)*	*Nüfus (10^6)*	*İthalat($10^9)*	*İhracat($10^9)*
2006	382.66	73.09	122.02	74.95
2007	405.62	73.22	127.52	76.45
2008	429.96	73.34	133.25	77.98
2009	455.75	73.47	139.25	79.54
2010	483.10	73.59	145.52	81.13
2011	512.09	73.72	152.06	82.75
2012	542.81	73.84	158.91	84.41
2013	575.38	73.97	166.06	86.09
2014	609.90	74.09	173.53	87.82
2015	646.50	74.22	181.34	89.57

Çizelge 5.8 ’de ise, verilen senaryo ile hesaplanan Çizelge 5.7 ’deki veriler ile yöntemlerin bulduğu katsayılar kullanılarak Türkiye’nin 2006-2015 yılları arasındaki enerji talep miktarının yöntemlere göre tahmin edilen değerleri bulunmaktadır. Çizelge 5.8 incelendiğinde PSO yönteminin belli bir yıldan sonra negatif değerlerde üreterek başarısız olduğu görülmektedir. ABC yönteminde ise çok hızlı artışlar olduğu için yöntemin tutarlılık gösteremediği söylenebilir. Çizelge 5.8 verilen senaryoya göre en gürbüz ve en verimli sonuçların ABCVSS yöntemi ile elde edildiği göstermektedir. Ayrıca 2006-2011 yılları arası gerçek enerji talebi ile yöntemlerin enerji talep tahminleri arasında karşılaştırma yapıldığında en yakın ve başarılı sonuçların ABCVSS yöntemi tarafından elde edildiği görülmektedir.

Çizelge 5.8. PSO, ABC ve ABCVSS yöntemleri ile tahmin edilen Türkiye’nin enerji talebi

Yıllar	*Gözlenen Enerji Talebi*	$PSO_{kuadratic}$	$ABC_{kuadratic}$	$ABCVSS_{kuadratic}$
2006	99,59	66,952	115,988	94,243
2007	107,63	44,170	133,781	96,037
2008	106,34	13,273	157,137	98,476
2009	106,14	-26,842	186,787	101,655
2010	109,27	-77,838	223,785	105,678
2011	114,48	-141,230	269,182	110,702
2012	-	-218,913	324,113	116,811
2013	-	-313,076	390,080	124,216
2014	-	-425,715	468,275	132,987
2015	-	-559,888	560,742	143,397

6.SONUÇLAR ve ÖNERİLER

6.1 Sonuçlar

Bu tez çalışmasında doğa-esinli algoritmaların en bilinenlerinden PSO ve ABC algoritmaları üzerinde çalışılmış ve bu iki algoritmanın geliştirilmesi sağlanmıştır.

İlk olarak yerel minimumlara takılma ve hızlı yakınsama nedeniyle popülasyon çeşitliliği kaybetme gibi sorunlara sahip PSO algoritmasının Levy dağılımını sağlayan Levy uçuşu yöntemi ile birlikte kullanılması ile LFPSO algoritması önerildi. Önerilen yöntem ile PSO algoritmasının global araması iyileştirildi ve özellikle çok modlu fonksiyonlar da başarılı sonuçlar aldığı görüldü. Bazı fonksiyonlar için 10,30 ve 50 boyut için testler yapıldı ve boyut sayısı arttıkça önerilen yöntemin PSO algoritmasına göre daha başarılı olduğu gözlemlendi. İki yöntem için çizilen yakınsama grafiklerinde de PSO algoritması hızlı yakınsamasına rağmen belli bir yerel minimuma takılıp kalırken, LFPSO algoritması daha başarılı sonuçlar üretti. LFPSO algoritması hem orijinal PSO ile hem de 2011 yılında Omran tarafından geliştirilen SPSO ile karşılaştırıldı. Ayrıca literatürde bulunan farklı PSO çeşitleri ile kıyaslanarak önerilen yöntemin başarısı gösterildi. Önerilen yöntemin hem temel PSO ile hem de SPSO ile yapılan istatistiksel testleri ile yöntemler arasında istatistiksel olarak anlamsal bir fark olduğu kanıtlandı.

İkinci olarak ise global aramayı iyi yaparken yerel aramadaki yeteneği zayıf olan ABC algoritmasının kaşif arı aşamasında Levy uçuşu kullanılarak yerel arama başarısı artırıldı. Levy uçuşu β parametresinin etkisi ile PSO' da global aramayı, ABC' de ise yerel aramayı iyileştirmek için kullanıldı. Önerilen LFABC ile ABC yöntemi test fonksiyonları ile kıyaslanarak sonuçlar paylaşıldı. İncelenen sonuçlarda LFABC algoritmasının test fonksiyonlarında az da olsa bir iyileştirme gösterdiği, bazılarında ise optimum değerlere ulaştığı gözlemlendi.

Son olarak doğa-esinli algoritmaların çeşitli karakteristikteki fonksiyonlara göre başarısının değiştiği gözlemlendi ve bu sorunu çözebilmek adına Kıran, M.S. ile birlikte ABC algoritmasında farklı çözüm arama teknikleri kullanan ve probleme göre hangi çözüm arama tekniği başarılı ise onu seçen ABCVSS yöntemi önerildi. Yöntem de kullanılan çözüm arama tekniklerinin bazısı global aramaya ağırlık verirken, bazısı da yerel aramayı güçlendirmeye yönelik seçildi. Bu sayede yöntem probleme yönelik olarak hangi arama tekniğini seçeceğini kendi belirleyerek başarıyı artırdı. Bu yöntem

ile tekmodlu, çokmodlu, kaydırılmış vb. gibi farklı karakteristikteki fonksiyonlarda başarısı gösterildi. Ayrıca bulunan bu yöntem gerçek dünya problemi olan enerji tahmin problemine uygulanarak Türkiye'nin 2006-2015 yılları arasın enerji talep tahmini yapıldı.

Sonuç olarak PSO ve ABC algoritmalarının zayıf ve eksik yönleri incelenerek farklı yöntemler ile bu yönler iyileştirildi. Önerilen yöntemler belirlenen test fonksiyonlarında denenerek sonuçlar gösterildi ve yöntemlerden biri gerçek dünya problemine uygulandı.

6.2 Öneriler

PSO ve ABC algoritmaları üzerinde yapılan çalışmalar sonucu 3 yeni algoritma önerildi.

LFPSO algoritması çizelgeleme, yapay sinir ağları, görüntü segmentasyonu vb. problemlerde kullanılabileceği gibi, ayrıca popülasyon çeşitliliği sağlayan Levy uçuşu yöntemi diğer doğa-esinli algoritmalar ile hibrit olarak kullanılabilir.

LFABC yöntemi ise çok başarılı sonuçlar üretmese de çeşitli ABC türleri ile kombine edilerek bu algoritmalarının başarısının artırılması sağlanabilir. Ayrıca Levy uçuşu yöntemi sadece kaşif arı aşamasında değil de işçi veya gözcü arı aşamalarına da uygulanıp sonuçlar gözlenebilir. Bu konu üzerinde de çalışmalar devam etmektedir.

Önerilen ABCVSS yöntemi ise farklı tipteki fonksiyonlardaki başarısı sebebiyle araç rotalama problemi, veri kümeleme, görüntü işleme ve segmentasyonu, elektrik yükü problemi, ekonomik güç dağıtım problemi vb. gerçek dünya problemlerinde kullanabilir. Çözüm arama denklemleri üzerinde çok daha detaylı bir araştırma yapılarak daha uygun çözüm arama denklemleri kullanılabilir. Ayrıca işçi arı ve gözcü arı aşamaları için ayrı ayrı çözüm arama denklem grupları kullanılarak başarı yükseltilebilir. Bu konu üzerinde de araştırmalar devam etmektedir.

KAYNAKLAR

Akay, B.,2009, Nümerik optimizasyon problemlerinde yapay arı kolonisi (artificial bee colony)algoritmasının performans analizi, Doktora Tezi, *Erciyes Üniversitesi Fen Bilimleri Enstitüsü*, Kayseri.

Al-Temeemy, A. A., Spencer, J. W. and Ralph, J. F., 2010, Levy Flights for Improved Ladar Scanning, *2010 IEEE International Conference on Imaging Systems and Techniques (IST)*, Thessaloniki, Greece, 225-228.

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

Akça, M. R., 2011, Yapay arı kolonisi algoritması kullanılarak gezgin satıcı probleminin Türkiyedeki il ve ilçe merkezlerine uygulanması, Yüksek Lisans Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü,* Konya.

Aras, H. and Aras, N., 2004, Forecasting residential natural gas demand, *Energy Sources*, 26(5), 463-472.

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

Basu M., 2013, Artificial bee colony optimization for multi-area economic dispatch, *Electrical Power and Energy Systems*, 49, 181-187.

Cavuslu, M. A., Karakuzu, C. and Karakayac, F., 2012, Neural identification of dynamic systems on FPGA with improved PSO learning, *Applied Soft Computing*, 12, 2707-2718.

Chang, Y.-P. and Ko, C.-N., 2009, A PSO method with nonlinear time-varying evolution based on neural network for design of optimal harmonic filters, 36, 6809-6816.

Chechkin, A.V., Metzler, R., Klafter, J. and Gonchar, V.Y., 2008, Anomalous Transport: Foundations and Applications, Klages, R. , Radons, G. , and Sokolov, I. M., *John Wiley & Sons*, Weinheim, 129-162.

Chen, Y. , 2010, Research and simulation on Levy Flight model for DTN, *2010 3rd International Congress on Image and Signal Processing*, Yantai, China, 4421-4423.

Chen, M.-R., Li, X., Zhang, X. and Lu Y.-Z., 2010, A novel particle swarm optimizer hybridized with extremal optimization, *Applied Soft Computing*, 10, 367-373.

Ceylan, H. and Ozturk,H. K., 2004, Estimating energy demand of Turkey based on economic indicators using genetic algorithm approach, *Energy Conversion and Management* , 45, 2525-2537.

Chiou, J.-S., Tsai, S.-H. ve Liu, M.-T., 2012, A PSO-based adaptive fuzzy PID-controllers, *Simulation Modelling Practice and Theory*, 26, 49-59.

Chuanga, L.-Y., Chang, H.-W., Tu, C.-J. and Yan, C.-H., 2008, Improved binary PSO for feature selection using gene expression data, *Computational Biology and Chemistry*, 32, 29-38.

Coello, C. A., Luna, E. H. and Aguirre, A. H., 2004, A comparative study of encodings to design combinational logic circuits using particle swarm optimization, *Proceedings of the 2004 NASA/DoD Conference on Evolution Hardware (EH'04)*, 71-78.

Dorigo M., Maniezzo, V. and Colorni, A., 1991, Positive feedback as a search strategy, *Technical Report 91-016, Italy*.

Dorigo, M. and Caro, G.D., 1999, Ant colony optimization: a new meta-heuristic, *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington, DC, 1470–1477.

Eberhart, R. and Hu, X.,1999, Human tremor analysis using particle swarm optimization, *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99*, Washington DC, 1927-1930.

Eberhart, R., Hu, X. and Shi, Y., 2004, Recent advances in particle swarm, *Congress on Evolutionary Computation, CEC2004*, Portland, 90-97.

Edwards, A. M., Phillips, R. A., Watkins, N. W., Freeman, M. P., Murphy, E. J., Afanasyev, V., Buldyrev, S. V., Luz, M. G. E., Raposo, E. P., Stanley, H. E. and Viswanathan, G. M., 2007, Revisiting Lévy flight search patterns of wandering albatrosses, bumblebees and deer, *Nature*, 449, 1044-1048.

Erdoğdu, E., 2007, Electricity demand analysis using co-integration and ARIMA modeling: a case study of Turkey, *Energy Policy*,35,1129-1146.

Gao, W. and Liu, S., 2012, A modified artificial bee colony algorithm, *Computers & Operations Research*, 39, 687-697.

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

Gao, W., Liu, S. and Huang, L., 2013, A novel artificial bee colony algorithm based on modified search equation and orthogonal learning, IEEE Transactions on Cybernetics, 43(3), 1011-1024.

Güner, A. R., 2006, A continuous and a discrete particle swarm optimization algorithm for uncapacitated facility location problem, Yüksek Lisans Tezi, *Fatih Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul.

Haklı, H. and Uğuz, H., 2013, Levy Flight Distribution for Scout Bee in Artificial Bee Colony Algorithm, *Lecture Notes on Software Engineering*, Paris, 254-258.

Hong, W.-C. , 2011, Electric load forecasting by seasonal recurrent SVR (support vector regression) with chaotic artificial bee colony algorithm, *Energy*, 36, 5568-5578.

Horng, M.-H. , 2011, Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation, *Expert Systems with Applications*, 38, 13785-13791.

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

Kang, F., Li, J. and Li, H., 2013, Artificial bee colony algorithm and pattern search hybridized for global optimization, *Applied Soft Computing*, 13, 1781-1791.

Karaboğa, D., 2005, An idea based on honey bee swarm for numerical optimization, *Technical Report-TR06, Kayseri.*

Karaboğa, D. ve Baştürk B., 2008, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*, 8, 687-697.

Kennedy, J. and Eberhart, R., 1995, Particle swarm optimization, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 39-43.

Kennedy, J. and Mendes, R., 2002, Population structure and particle swarm performance, *IEEE Congr. Evol. Comput.*, Honolulu, 1671-1676.

Kıran, M.S.,2010, Arı kolonisi ile şoför-hat-zaman optimizasyonu, Yüksek Lisans Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*, Konya.

Kıran, M. S., Gunduz, M. and Baykan, O. K., 2012a, A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum, *Applied Mathematics and Computation*, 219, 1515-1521.

Kıran, M.S., Ozceylan, E., Gunduz ,M. and Paksoy T.,2012b, A novel hybrid approach based on Particle Swarm Optimization and Ant Colony Algorithm to forecast energy demand of Turkey, *Energy Conversion and Management* , 53, 75-83.

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

Lee, C.-Y. and Yao, X., 2001, Evolutionary Algorithms with Adaptive Levy Mutations,. *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, South Korea, 568-575.

Li, G., Niu, P. and Xiao, X., 2012, Development and investigation of efficient artificial bee colony algorithm for numerical function optimization, *Applied Soft Computing*, 12, 320-332.

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

Liu, Y., Ling, X. and Liu, G., 2012, Improved artificial bee colony algorithm with mutual learning, *Journal of Systems Engineering and Electronics*, 23, 265-275.

Liang, J. J. and Suganthan, P. N., 2005, Dynamic multi-swarm particle swarm optimizer, *Swarm Intelligence Symposium*, California, 124-129.

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

Lovbjerg, M., Rasmussen, T. K. and Krink, T., 2001, Hybrid particle swarm optimizer with breeding and subpopulations, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, 469-476.

Mendes, R., Kennedy, J. and Neves J., 2004, The Fully Informed Particle Swarm: Simpler, Maybe Better, *IEEE Transactions on Evolutionary Computation*, 8(3), 204-210.

Nickabadi, A., Ebadzadeh, M. M. and Safabakhsh R., 2011, A novel particle swarm optimization algorithm with adaptive inertia weight, *Applied Soft Computing*, 11, 3658-3670.

Omran, M., 2011, SPSO 2007 Matlab, http://www.particleswarm.info/Programs.html [Ziyaret Tarihi: 17.07.2013].

Ortakçı,Y., 2011, Parçacık sürü optimizasyonu yöntemlerinin uygulamalarla karşılaştırılması, Yüksek Lisans Tezi, *Karabük Üniversitesi Fen Bilimleri Enstitüsü*, Karabük.

Özçelik, Y. and Hepbaşlı, A., 2006, Estimating petroleum exergy production and consumption using a simulated annealing approach, *Energy Sources B: Econ Plan Policy*, 1(3), 255-265.

Özdemir R., 2012, Yapay arı kolonisi algoritması için yeni seçme ve arama mekanizmalarının geliştirilmesi, Yüksek Lisans Tezi, *Erciyes Üniversitesi Fen Bilimleri Enstitüsü*, Kayseri.

Pan, Q.-K., Tasgetiren, M. F. and Liang, Y.-C., 2008, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Computers & Operations Research*, 35, 2807-2839.

Pereyra, M. A. and Batatia, H., 2010, A Levy Flight Model for Ultrasound in Skin Tissues, *2010 IEEE on Ultrasonics Symposium (IUS)*, San Diego, CA, 2327-2331.

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

Sabat, S. L., Ali, L. and Udgata, S. K., 2011, Integrated Learning Particle Swarm Optimizer for global optimization, *Applied Soft Computing*, 11, 574-584.

Sarangi, A., Mahapatra, R. K. ve Panigrahi, S. P., 2011, DEPSO and PSO-QI in digital filter design, *Expert Systems with Applications*, 38, 10966-10973.

Shi, Y. and Eberhart, R., 1998, A modified particle swarm optimizer, *IEEE World Congress on Computational Intelligence*, Anchorage, 69-73.

Shi, X.H., Liang, Y.C., Lee, H.P., Lu, C. and Wang, L.M., 2005, An improved GA and a novel PSO-GA-based hybrid algorithm, *Information Processing Letters*, 93, 255-261.

Sutantyo, D. K., Kernbach, S., Levi, P. and Nepomnyashchikh, V. A., 2010, Multi-Robot Searching Algorithm Using Levy Flight and Artificial Potential Field, *2010 IEEE International Workshop on Safety Security and Rescue Robotics (SSRR)*, Bremen, Germany, 1-6.

Szeto, W.Y., Wu, Y. and Ho, S.C., 2011, An artificial bee colony algorithm for the capacitated vehicle routing problem, *European Journal of Operational Research*, 215, 126-135.

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M. and Gencyilmaz, G., A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, *European Journal of Operational Research*, 177, 1930-1947.

Terdik, G. and Gyires, T., 2009, Lévy Flights and Fractal Modeling of Internet Traffic, *IEEE/ACM Transactions on Networking*, 120-129.

Tereshko, V., 2000, Reaction-Diffusion model of a honeybee colony's foraging behaviour, 6th International Conference on Parallel Problem Solving from Nature PPSN VI, Paris, 807-816.

Tokmak, M., 2011, Yapay arı kolonisi algoritması ile ders çizelgeleme probleminin çözümü, Yüksek Lisans Tezi, *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü*, Isparta.

Toksarı, D. M., 2007 Ant colony optimization approach to estimate energy demand of Turkey, *Energy Policy*, 35, 3984-3990.

Tsoulos, I. G. and Stavrakoudis, A., 2010, Enhancing PSO methods for global optimization, *Applied Mathematics and Computation*, 216, 2988-3001.

Unler, A., 2008, Improvement of energy demand forecasts using swarm intelligence: the case of Turkey with projections to 2025, *Energy Policy*, 36, 1937-1944.

Van den Bergh, F. and Engelbrecht, A. P., 2004, A cooperative approach to particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, 8(3), 225-239.

Wang, H., Moon, I., Yang, S. and Wang, D., 2012, A memetic particle swarm optimization algorithm for multimodal optimization problems, *Information Sciences*, 197, 38-52.

Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B. and Tian, Q., 2011, Self-adaptive learning based particle swarm optimization, *Information Sciences*, 181, 4515-4538.

Xiang, W. and An, M., 2013, An efficient and robust artificial bee colony algorithm for numerical optimization, *Computers & Operations Research*, 40, 1256-1265.

Xinchao, Z., 2010, A perturbed particle swarm algorithm for numerical optimization, *Applied Soft Computing*,10, 119-124.

Yan, X., Zhu, Y., Zou, W. and Wang, L., 2012, A new approach for data clustering using hybrid artificial bee colony algorithm, *Neurocomputing*, 97, 241–250.

Yang, H.-C., Zhang, S.-B., Deng, K.-Z. and Du, P.-J., 2007, Research into a feature selection method for hyperspectral imagery using PSO and SVM, *J China Univ Mining & Technol*, 17(4), 473-478.

Yang, X., Yuan, J., Yuan, J. ve Mao, H., 2010, An improved WM method based on PSO for electric load forecasting, *Expert Systems with Applications*, 37, 8036-8041.

Yang, X.-S., 2010a, Firefly Algorithm, Levy Flights and Global Optimization, Bramer, M., Ellis, R. and Petridis, M. (Eds.), Research and Development in Intelligent Systems XXVI, *Springer London*, 209-218.

Yang, X.-S., 2010, Engineering Optimization An Introduction with Metaheuristic Applications, *John Wiley and Sons*, New Jersey.

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

Yıldız, A. R., 2013, A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing, *Applied Soft Computing*, 13, 2906-2912.

Yu, J. and Duan, H., 2012, Artificial Bee Colony approach to information granulation-based fuzzy radial basis function neural networks for image fusion, *Optik*, In Press.

Zhan, Z.-H., Zhang, J., Li, Y. and Shi, Y.-H., 2011, Orthogonal learning particle swarm optimization, *IEEE Transactions on Evolutionary Computation*,15(6), 832-847.

Zhang, Y., Huang, D., Ji, M. ve Xie, F., 2011, Image segmentation using PSO and PCM with Mahalanobis distance, *Expert Systems with Applications*, 38, 9036-9040.

Zhang, R., Song, S. and Wu, C., 2013, A hybrid artificial bee colony algorithm for the job shop scheduling problem, *Int. J. Production Economics*, 141, 167-178.

Zhou, D., Gao, X., Liu, G., Mei, C., Jiang, D. and Liu, Y., 2011, Randomization in particle swarm optimization for global search ability, *Expert Systems with Applications*, 38, 15356-15364.

Zhu, G. and Kwong, S., 2010, Gbest-guided artificial bee colony algorithm for numerical function optimization, *Applied Mathematics and Computation*, 217, 3166-3173.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Hüseyin HAKLI
Uyruğu	**:**	T.C
Doğum Yeri ve Tarihi	**:**	Konya – 03.09.1989
Telefon	**:**	0 (555) 403 60 65 – 0 (332) 223 37 28
Faks	**:**	–
e-mail	**:**	hhakli@selcuk.edu.tr, huseyin_hakli22@hotmail.com

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Selçuklu Anadolu Lisesi, Selçuklu, Konya	2007
Üniversite	:	Selçuk Üniversitesi – Bilgisayar Mühendisliği, Selçuklu, Konya	2011
Yüksek Lisans	:	Selçuk Üniversitesi – Bilgisayar Mühendisliği ABD, Selçuklu, Konya	Devam Ediyor
Doktora	:		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2009	IMS Yazılım ve Otomasyon Sistemleri	Stajyer
2010	Türk Kızılayı Konya Şubesi Özel Ticaret Borsası Hastanesi	Stajyer
2011	Necmettin Erbakan Üniversitesi Bilgisayar Mühendisliği Bilgisayar Yazılımı (ÖYP)	Arş. Gör.
2011	Selçuk Üniversitesi Bilgisayar Mühendisliği (ÖYP- Eğitim)	Arş. Gör.

UZMANLIK ALANI
YABANCI DİLLER

İngilizce, KPDS B Sınıf(80), ÜDS(86.250)

BELİRTMEK İSTEĞİNİZ DİĞER ÖZELLİKLER

2011- Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümü Bölüm 1.si

YAYINLAR

Hakli, H. and Uguz, H., 2013, Levy Flight Distribution for Scout Bee in Artificial Bee Colony Algorithm, *Lecture Notes on Software Engineering*, 1(3), 254-258. (Konferans - Yüksek Lisans tezinden yapılmıştır)

Hakli, H., Guraksin, G. E. and Uguz, H., 2013, Training Support Vector Machines by Using Particle Swarm Optimization and a Bone Age Example, *Euro Informs MMXII 26th European Conference on Operational Research*. (Konferans)