T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YARASA ALGORİTMASININ UNIMODAL, MULTIMODAL VE KAYDIRILMIŞ SAYISAL OPTİMİZASYON PROBLEMLERİ (CEC05) ÜZERİNDE GELİŞTİRİLMESİ

Selim YILMAZ

Danışman
Doç. Dr. Ecir Uğur KÜÇÜKSİLLE

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
ISPARTA – 2014

TEZ ONAYI

Selim YILMAZ tarafından hazırlanan **"Yarasa Algoritmasının Unimodal, Multimodal ve Kaydırılmış Sayısal Optimizasyon Problemleri (CEC05) Üzerinde Geliştirilmesi"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak başarı ile savunulmuştur.

Danışman	**Doç. Dr. Ecir Uğur KÜÇÜKSİLLE** Süleyman Demirel Üniversitesi
Jüri Üyesi	**Yrd. Doç. Dr. Habil KALKAN** Süleyman Demirel Üniversitesi
Jüri Üyesi	**Yrd. Doç. Dr. Yavuz CENGİZ** Süleyman Demirel Üniversitesi

Enstitü Müdürü **Doç. Dr. Ahmet ŞAHİNER**

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Selim YILMAZ

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

YARASA ALGORİTMASININ UNIMODAL, MULTIMODAL VE KAYDIRILMIŞ SAYISAL OPTİMİZASYON PROBLEMLERİ (CEC05) ÜZERİNDE GELİŞTİRİLMESİ

Selim YILMAZ

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ecir Uğur KÜÇÜKSİLLE

Optimizasyon bir probleme en uygun çözümler üretebilme çabası olarak tanımlanabilmektedir. Optimizasyonun tarihteki izlerine II. Dünya Savaşı'nda rastlanmaktadır. O dönemde İngiliz Ordusu düşman güçlerini saf dışı bırakmak için ihtiyaç duydukları kaynakların kıtlığı ve bu kaynakların paylaşımı gibi problemler ile karşı karşıya kalmışlardır. İngilizler bu gibi problemlere bilimsel çözümler üretmesi için matematikçilerin oluşturduğu bir ekibe başvurmuşlardır. Bu ekip tarafından geliştirilen yöntemler, İngiltere'nin bu mücadeledeki hava savaşını kazanmasına önemli ölçüde vesile olmuştur. Bu olayda görüldüğü gibi optimizasyon metotları, insanların mevcut kaynakları belirli şartlar altında en iyi şekilde değerlendirme arayışlarından var olmuştur.

Optimizasyon metotlarının önemli bir türü sezgisel algoritmalardır. Sezgisel algoritmalar genellikle doğadan ilham alınarak oluşturulmuş algoritmalardır. Örneğin Parçacık Sürü Optimizasyonu, kuş ya da balık sürülerinin sosyal davranışlarından esinlenmiştir. Yarasa Algoritması, yarasaların avlanma davranışlarına ve karanlık ortamlarda bile hareket edebilmelerine rehberlik eden *ekolokasyon* olarak bilinen bir özellikten ilham alınmış ve Yang tarafından 2010 yılında önerilmiş bir sezgisel algoritmadır.

Bu çalışmada Yarasa Algoritmasının lokal ve global arama karakteristikleri üç farklı yöntem ile güçlendirilmiştir. Geliştirilen algoritmanın performansını ölçmek amacıyla, algoritma sırasıyla standart, karmaşık benchmark fonksiyonları ile üç adet kısıtlı gerçek hayat problemleri üzerinde test edilmiştir. Test kümesi üzerinden elde edilen sonuçlar, geliştirilen algoritmanın, önerilen algoritmaya göre daha iyi olduğunu göstermektedir. Ayrıca bu yöntem kısıtlı gerçek hayat problemleri üzerinde literatürdeki bir takım çalışmalar ile kıyaslanmış ve sonuçlar bu yöntemin literatürdeki çalışmalardan daha etkili olduğunu kanıtlamıştır.

Anahtar Kelimeler: Optimizasyon, sezgisel algoritmalar, Yarasa Algoritması.

2014, 111 sayfa

ABSTRACT

M.Sc. Thesis

MODIFICATION OF BAT ALGORITHM ON UNIMODAL, MULTIMODAL AND SHIFTED NUMERIC OPTIMIZATION PROBLEMS (ALSO KNOWN AS CEC05)

Selim YILMAZ

Süleyman Demirel University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ecir Uğur KÜÇÜKSİLLE

Optimization can be defined as the effort of generating solutions to a problem. The traces of the optimization can be seen in World War II. During the war, British army had faced to some problems of the scarceness and allocation of the sources to defeat numerous targets. The army had requested assistance from a team consisting of mathematicians to produce scientific solutions to these problems. The methods developed by the team were significantly instrumental in the winning of the Air Battle by Britain. As it is in this issue, the optimization methods have arisen from the ideally evaluation efforts of the existing sources of the humans under bounded circumstances.

An important type of the optimization method is heuristic algorithms. Heuristic algorithms are generally proposed by inspiration of the nature. For instance, Particle Swarm Optimization has been inspired by the social behavior of fish schooling or bird flocking. Bat Algorithm is a heuristic algorithm proposed by Yang in 2010 and has been inspired from a property, named *echolocation*, which guides the bats on their hunting behavior and movements even in complete darkness.

In this thesis, local and global search characteristics of Bat Algorithm are improved through three different methods. To validate the performance of the improved algorithm, standard, complicated benchmark test functions and constrained real-world problems are utilized, respectively. The results obtained by these test sets reveal that improved algorithm is better than the standard one. Furthermore, the method proposed in this thesis is compared with the studies in the literature on real-world problems and it is proved that this method is more effective than the studies belonging to the literature on such kind of the problems.

Keywords: Optimization, heuristic algorithms, Bat Algorithm.

2014, 111 pages.

TEŞEKKÜR

"*Yarasa Algoritmasının Unimodal, Multimodal ve Kaydırılmış Sayısal Optimizasyon Problemleri (CEC05) Üzerinde Geliştirilmesi*" isimli yüksek lisans tez çalışmamın yürütülmesinin, sonuçlandırılmasının ve değerlendirilmesinin her aşamasında bilgi, tecrübe ve yardımlarını esirgemeyen değerli danışman hocam Doç. Dr. Ecir Uğur KÜÇÜKSİLLE'ye saygı ve teşekkürlerimi sunarım.

Yüksek Lisans tezimin hazırlanmasında büyük emekleri olan, lisansüstü öğretim yıllarının ders dönemlerinde derslerini alarak bilgilerinden faydalandığım Süleyman Demirel Üniversitesinin değerli öğretim üyeleri hocalarıma emeklerinden ötürü teşekkür ederim.

3712-YL1-13 No'lu Proje ile tezimi maddi olarak destekleyen Süleyman Demirel Üniversitesi Bilimsel Araştırma Projeleri Yönetim Birimi Başkanlığı'na teşekkür ederim.

Hayatım boyunca verdikleri tüm desteklerden ötürü haklarını ödeyemeyeceğim aileme, anlayış ve desteğinden dolayı eşime teşekkür ederim.

Selim YILMAZ
ISPARTA, 2014.

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

ABC	Artificial Bee Colony
BP	Back Propagation
CEC05	Congress of Evolutionary Computation 2005
FDCTM	Fırçasız DC Tekerlekli Motor
GA	Genetik Algoritma
GYA	Geliştirilmiş Yarasa Algoritması
HA	Harmoni Arama Algoritması
KM	K-Medoid Kümeleme Algoritması
LM	Levenberg-Marquardt
NSGA	Non-Dominated Sorting Genetic Algorithm
PSO	Parçacık Sürü Optimizasyonu
YA	Yarasa Algoritması

GİRİŞ

Bir mühendislik sisteminin tasarım, imalat ve onarım sürecinde görev alan uzmanlar belli kısıtlar altında bu sistem üzerinde yönetimsel ve teknolojik kararlar vermek zorunda kalmışlardır. Optimizasyon, mevcut kısıtlar altında en iyi sonucu elde etme girişimidir. Optimizasyon sürecinin en önemli amacı bir sistem üzerinde harcanan enerji ve zamanı en aza indirmek ya da o sistemden maksimum verim elde etmektir. O halde bir sistemin tasarım maliyeti fonksiyon olarak ifade edilirse, optimizasyon bu fonksiyonun belli şartlar altında minimum ya da maksimum değerine ulaşma süreci olarak tanımlanabilir (Rao, 2009).

Optimizasyon, içerisinde karar verme durumu barındıran tüm problemlerin merkezindedir. Bu karar verme aşaması verilen bu problemin içerisinde mevcut birçok alternatif arasından yalnızca birini seçme zorunluluğu getirmektedir. Bu seçim aşaması, en iyi kararı verebilmek için ihtiyaçlar tarafından yönetilmektedir. Seçeneklerin kalitesinin ölçeği bir hedef fonksiyon ya da performans endeksi olarak tanımlanmaktadır. Optimizasyon metotları, verilen hedef fonksiyon içerisinden en iyi kararı verme süreci olarak tanımlanmaktadır. Bilgisayar teknolojisindeki hızlı gelişmeler; kullanıcı dostu yazılımların, yüksek hızlı paralel işlemcilerin, bilhassa yapay sinir ağlarının gelişmesi neticesinde optimizasyon konusu araştırmacılar tarafından oldukça ilgi çekmektedir (Chong ve Zak, 2013).

Görüldüğü üzere yukarıda iki farklı yaklaşım ile tanımlanan optimizasyon kavramının ortak vurgusu “seçme, karar verme, bir sonuca varma” olmuştur. Bu çerçevede, bir sistemin ya da problemin ihtiyacına ulaşabilecek karar değişkenlerin elde edilebilmesi için değişkenlerin belli bir uzay aralığında araştırılması gerekmektedir. Son yıllardaki bilimsel çalışmalarda bu amaca yönelik optimizasyon algoritmaları geliştirilmiştir.

Optimizasyon algoritmaları *deterministik* ve *stokastik* olmak üzere iki alt dala ayrılmaktadır. Deterministik algoritmalar, yapılarında rassallık yaratacak

herhangi bir operatör bulundurmayan algoritmalardır, bu nedenle bu tür algoritmalar aynı şartlar altında her çalıştıklarında aynı sonucu üretmektedirler. Tepe Tırmanma Algoritması ve Levenberg-Marquardt Algoritması (Levenberg, 1944) deterministik algoritmalara örnek verilebilir. Diğer taraftan stokastik algoritmalar rassal yapılarından dolayı her çalışmada aynı şartlar altında çalışsalar bile farklı çözümler elde etmektedirler. Genetik Algoritma (Goldberg, 1989) ve Yarasa Algoritması (Yang, 2010) stokastik arama algoritmalarına örnek olarak gösterilebilmektedir (Yang, 2013).

Çoğu deterministik algoritmalar türev bilgisi kullanmaktadırlar. Fonksiyon değerlerinden ve onun türevlerinden faydalanan bu algoritmalar, tek global optimum bulunduran unimodal fonksiyonlar için idealdirler; ancak birden çok global optimum barındıran multimodal fonksiyonlar ya da türev bilgisinin sıfır olduğu eğimsiz bölgelerden oluşan fonksiyonlarda deterministik algoritmalar sonuç üretememektedirler. Bu türdeki problemler için stokastik algoritmalar tercih edilmektedir. Stokastik algoritmaların rassal yapılarından dolayı yakınsama hızı oldukça yavaştır ancak lokal bölgelerden kurtulma yeteneği bu algoritmaları deterministik algoritmalara nazaran üstün kılmaktadır (Yang, 2010; Noel, 2012).

Stokastik algoritmalar genellikle ikiye ayrılmaktadırlar; *sezgisel* ve *üst sezgisel* algoritmalar. Sezgisel Algoritmalar, oldukça zorlu optimizasyon problemlerinde bile deneme yanılma yoluyla, kabul edilebilir bir zaman dilimi içerisinde, kaliteli sonuçlar elde etmektedirler; fakat hiçbir zaman optimum sonucu bulmayı garanti etmezler. Üst sezgisel algoritmalar, farklı metotları kullanarak arama uzayını etkili bir biçimde arayan yüksek seviyeli algoritmalar olarak tanımlanmaktadır (Blum ve Roli, 2003). Literatürdeki çoğu çalışmada bu iki türdeki algoritma arasında bir ayrım yapılmamaktadır ve içerisinde rassallık bulunduran stokastik algoritmaların hepsi üst sezgisel algoritma olarak tanımlanmıştır (Yang, 2013).

Sezgisel yöntemler kendi içinde altı dala ayrılmaktadırlar (Akyol ve Alataş, 2012), bunlar: Biyoloji tabanlı, fizik tabanlı, sürü tabanlı, sosyal tabanlı, müzik

tabanlı ve kimya tabanlı yöntemlerdir. Genetik Algoritma (Goldberg, 1989) ve Diferansiyel Evrim Algoritması (Storn ve Price, 1997) biyoloji tabanlı; Yerçekimsel Arama Algoritması (Rashedi vd., 2009), Isıl İşlem Algoritması (Kirkpatrick vd., 1983) fizik tabanlı; Parçacık Sürü Optimizasyonu (Kennedy ve Eberhart, 1995), Karınca Koloni Optimizasyonu (Dorigo vd., 2006) sürü tabanlı; Tabu Arama Algoritması (Glover veLaguna, 1997) sosyal tabanlı; Armoni Arama Algoritması (Yang, 2009) müzik tabanlı ve Yapay Kimyasal Reaksiyon Algoritması (Alatas, 2011) kimya tabanlı algoritmalara örnek olarak gösterilebilir.

Görüldüğü üzere sezgisel algoritmaların esin kaynağı genellikle doğa olmuştur, bu nedenle bu tür algoritmalar aynı zamanda doğadan esinlenen algoritmalar olarak da bilinmektedir. Sezgisel algoritmaların arama karakteristiğine yön veren ve davranışlarını belirleyen, hayati öneme sahip, iki bileşen bulunmaktadır: *Exploration* (global arama) ve *exploitation* (lokal arama). Exploration, algoritmanın uzaydaki bilinmeyen bölgeleri keşfedebilme yeteneği olarak tanımlanırken; exploitation, algoritmanın o ana kadar bulduğu sonuçlara odaklanıp bu sonuçları değerlendirerek daha iyi bir sonuca ulaştırma yeteneği olarak tanımlanmaktadır. Bu iki bileşen birbirinin karşıtı gibi görünse de birbirlerinin eksikliklerini gidermektedirler. Algoritmanın exploration kabiliyeti algoritmayı lokal bölgelere takılmasını önleyerek bu bölgelerden kurtulmasını sağlamakta iken, exploitation kabiliyeti algoritmanın optimum bölgelere yakınsama hızını artırmaktadır. Bir algoritmanın zaman ve optimizasyon kalitesi açısından bir problem üzerinde başarı elde edebilmesi bu iki bileşenin dengeli bir biçimde dağılmasına bağlıdır (Gao ve Liu, 2012). Exploration kabiliyeti algoritmada daha ağırlıklı ise algoritmanın optimum bölgelere yakınsama süresi uzayacak, exploitation kabiliyeti daha ağırlıklı ise algoritma optimizasyon sürecinin başlarında lokal bölgelere takılacaktır. Literatürdeki birçok çalışma, sezgisel algoritmaların öncelikle arama uzayını etkin bir biçimde taramaları ve böylelikle algoritmaların optimum bölgelere yakınsayabilmelerinde ihtiyaç duyacakları bilgileri elde etmeleri için optimizasyon sürecinin başlarında exploration kabiliyetlerini; sürecin

sonlarında ise exploitation kabiliyetlerini daha fazla kullanmalarının faydalı olduğunu belirtmiştir (Tan vd., 2009).

Sezgisel algoritmaların kolay uygulanabilir olması, karmaşık matematiksel yapılar içermemesi ve kontrol parametrelerinin az olması bu algoritmaların popülaritesini artırmakta ve araştırmacıların çeşitli optimizasyon problemlerinin çözümünde bu algoritmalardan faydalanmalarına sebep olmaktadır.

Yarasa Algoritması (Yang, 2010), yarasaların avlanma, karanlık ortamlarda hareket etme ya da konumlarını belirleme gibi davranışlarının analiz edilmesi sonucu keşfedilen ekolokasyon kabiliyetlerinden esinlenmiş sürü tabanlı bir sezgisel optimizasyon algoritmasıdır. Yarasa Algoritması az boyutlu optimizasyon problemlerinde oldukça başarılı iken; çok boyutlu problemler üzerinde erken yakınsayarak lokal bölgelere takılma problemi yaşamaktadır dolayısıyla etkili performans sergileyememektedir (Fister vd., 2013). Bu durum, algoritmada exploration ve exploitation bileşenlerinin etkili ve dengeli bir biçimde dağılmadığını işaret etmektedir.

Bu çalışmada, Yarasa Algoritmasının arama kabiliyetine yön veren exploration ve exploitation bileşenlerinden optimizasyon sürecinde dengeli bir biçimde faydalanabilmesi için bir takım modifikasyon ve hibrit yapılar geliştirilmiştir. Geliştirilen algoritma farklı boyutlarda sırasıyla: Unimodal ve multimodal standart benchmark test fonksiyonları; karmaşık benchmark test fonksiyonları; "Kaynaklı kiriş tasarımı", "Gerilim sıkıştırma yay tasarımı" ve "Basınçlı kap tasarımı" isimli gerçek hayat problemleri üzerinde optimizasyon kalitesi baz alınarak test edilmiştir. Ayrıca Geliştirilen algoritmanın literatürdeki değerinin öğrenilmesi amacıyla yukarıda isimleri verilen gerçek hayat problemleri literatürde yer alan çalışmalar ile kıyaslanmıştır.

Test grupları üzerinden alınan optimizasyon sonuçları, Geliştirilen Yarasa Algoritmasının (GYA) üç farklı test grubunda da Yarsa Algoritmasından (YA) daha başarılı olduğunu göstermektedir. GYA'nın gerçek hayat problemleri

üzerinde literatürdeki çalışmalar ile kıyaslanmasının sonucu geliştirilen yöntemin literatürdeki çalışmalardan daha etkin sonuçlar ürettiği görülmektedir.

1. KAYNAK ÖZETLERİ

Yang (2010), Yarasaların sosyal davranışları üzerindeki araştırmalardan esinlenerek Yarasa Algoritmasını (YA) önermiştir. Yang önerdiği bu algoritmayı Genetik Algoritma(GA) ve Parçacık Sürü Optimizasyonu (PSO) algoritması ile farklı benchmark test fonksiyonları üzerinde ve farklı problem boyutlarında mukayese etmiştir. Yang yaptığı test sonuçlarını karşılaştırmış ve sonuç olarak YA'nın GA ve PSO'ya göre daha hızlı yakınsadığı ve daha verimli sonuçlar ürettiğini göstermiştir.

Yang (2011), Yarasa Algoritmasını çok amaçlı problemleri çözmek için geliştirerek çok amaçlı Yarasa Algoritmasını formülize etmiştir. Yang önerdiği bu çalışmanın verimliliğini teyit etmek amacıyla yöntemi önce benchmark test fonksiyonları üzerinde test etmiş ardından "Kaynaklı Kiriş Dizaynı" isimli gerçek hayat optimizasyon problemini çözmek için kullanmıştır. Yang test fonksiyonlarını convex, non-convex ve süreksiz olarak üç farklı yapıdan seçmiştir. 50 popülasyon sayısı ve 5000 çevrim sayısı ile önerdiği yöntemi denemiştir. Sonuçlar; çok amaçlı Yarasa Algoritmasının, fonksiyonlar üzerinde verimli olduğunu göstermiştir.

Khan ve Sahai (2012), Yapay Sinir Ağlarının eğitiminde yeni geliştirilen Yarasa Algoritmasının üstünlüğünü (yakınsama performansı ve çözüm kalitesi bakımından) standart algoritmalar üzerinde gösteren bir çalışma yapmışlardır. Bu çalışmada veri kümesi olarak Proben1 kullanılmıştır (Proben1: Tıp alanında ortaya çıkan sorunlardan oluşan seçilmiş testlerdir). Ayrıca Yarasa Algoritması, Genetik Algoritma ve Parçacık Sürü Optimizasyonu olmak üzere 3 adet popülasyon tabanlı algoritma ve Levenberg-Marquard (LM), Back Propagation (BP) Optimizasyonu olmak üzere 2 adet türev tabanlı algoritma kullanmıştır. Sonuçlar, yeni geliştirilen Yarasa Algoritmasının yapay sinir ağlarının eğitiminde diğer dört yönteme göre daha avantajlı olduğunu göstermiştir.

Bora vd. (2012), Tek ve çok amaçlı *Fırçasız DC Tekerlekli Motor* (FDCTM) isimli gerçek hayat problemini Yarasa Algoritması ile optimize ederek sonuçları

literatürde aynı problem üzerinde farklı metotlar aracılığıyla elde edilmiş sonuçlar ile kıyaslayarak göstermiştir. Ayrıca Yarasa Algoritmasını çok amaçlı optimizasyon problemlerine genişletmek için algoritmayı Non-Dominated Sorting Genetic Algorithm (NSGA) ile birleştirmiştir. Her iki optimizasyon metodunun başarısını öğrenmek için ZDT1 isimli test fonksiyonu kullanmıştır. Ayrıca FDCTM problemi üzerinde her iki algoritmanın başarısını test etmiştir. Bora vd., tek amaçlı Yarasa Algoritmasının FDCTM problemi üzerindeki verimliliğini 95.23% olarak göstermiştir ve bu değerin optimum değere çok yakın olduğunu belirtmiştir.

Yang ve Gandomi (2012), Yarasa Algoritmasının performansını "matematiksel problem, Himmelblau problemi, üç barlı kafes tasarımı problemi, hız azaltıcı tasarım problemi, yapı parametrelerinin tanımlaması" gibi farklı sekiz adet kısıtlı mühendislik optimizasyon problemleri üzerinde araştırmıştır. Yarasa Algoritmasının her bir problem üzerinde elde ettiği performansı literatürde farklı metotlar ile aynı problemler üzerinde optimize edilmiş veriler ile kıyaslayarak Yarasa Algoritmasının bu fonksiyonlar üzerindeki başarısını ölçmüştür.

Lin vd. (2012), Yarasa Algoritmasının arama karakteristiğinin güçlendirilmesi ve arama esnasında lokal optimum noktalara takılmasını engellemek için algoritmaya kaotik dizi ve Lévy Flight yapılarını dahil etmiştir. Lin vd., geliştirdikleri bu yöntemi parametre kestirimi problemi üzerinde kullanmıştır.

Tsai vd. (2012), Yarasaların davranışlarını yeniden analiz etmiş ve Yarasa Algoritmasının genel yapısını bozmadan ilgili operatörleri yeniden tanımlayarak yeni bir sezgisel yöntem algoritması önermiştir. Önerilen algoritmanın hesaplama maliyetini ve yakınsama hızı performansını öğrenmek için algoritmayı bilinen ve yaygın olarak kullanılan üç adet sayısal test fonksiyonu ile test etmiştir. Sonuçlar önerilen algoritmanın Yarasa Algoritmasına göre yakınsama hızını 99.42% geliştirdiğini ayrıca 6.07% hesaplama maliyetini azalttığını göstermiştir.

Musikapun ve Pongcharoen (2012), Yarasa Algoritması tabanlı çok aşamalı, çok makineli, çok ürünlü zaman çizelgeleme aracı geliştirmişlerdir. Bu çalışmada Yarasa Algoritması, zamanında üretim felsefesini ön plana alarak üretimdeki erken ya da gecikme ceza maliyetlerini minimuma indirgemeyi amaçlamaktadır. Çalışmanın sayısal deneyleri, finans sektöründe faaliyet gösteren bir şirketten elde edilmiştir. Sonuçlar uygun parametreler ayarlandıktan sonra Yarasa Algoritması ile performansın 8.37% geliştiğini göstermektedir.

Komarasamy ve Wahi (2012), Yarasa Algoritması ile K-Means Kümeleme Algoritması'nı (KM) birleştirerek yeni bir metot önermiştir. Bu metodu kümeleme işlemleri üzerinde kullanmıştır. Yöntemde KM'nin en büyük sınırlılıklarından biri olan küme merkezi bulma işlemi Yarasa Algoritması ile gerçekleştirilmiş ardından KM ile kümeleme yapısı oluşturulmuştur. Deneyler üç farklı veri seti üzerine kurulmuştur. Sonuçlar yeni metodun Yarasa Algoritmasının yakınsama hızını artırdığını ayrıca KM algoritmasına başlangıç küme merkezi belirlenmesi konusunda katkıda bulunduğunu göstermiştir.

Nakamura vd. (2012), yarasaların davranışlarını temele alan doğadan esinlenmiş yeni bir özellik seçim tekniği tanıtmışlardır. Bu yaklaşımın özü, yarasaların keşfetme kabiliyetleri ile En Uygun Orman Yolu sınıflandırıcısının hızını birleştirmek olmuştur. Beş adet genel veri kümesi üzerine kurulan deney sonuçları önerilen yaklaşımın, iyi bilinen sürü tabanlı tekniklerden (Parçacık Sürü Optimizasyonu, Yerçekimsel Arama Algoritması ve Ateşböceği Algoritması) daha iyi olduğunu göstermiştir.

Akhtar vd. (2012), Yarasa Algoritmasını kullanarak video görüntüsünde hareket halindeki insan vücudunun takibi problemi üzerinde çalışmıştır. Yarasa Algoritmasının bu problem üzerindeki performansı Particle Filter, Annealed Particle Filter ve Parçacık Sürü Optimizasyonu ile karşılaştırmalı olarak sınanmıştır. Sonuçlar Yarasa Algoritmasının bu problem üzerinde diğer yöntemlere oranla daha başarılı olduğunu göstermiştir.

Fister vd. (2013), Yarasa Algoritmasının çok boyutlu optimizasyon problemleri üzerindeki sınırlılıklarını aşmak amacıyla bu algoritmayı Diferansiyel Evrim Algoritması ile hibritlemiştir. Geliştirilen hibrit yöntemin performansını test etmek amacıyla beş adet sayısal test fonksiyonu kullanılmıştır. Test fonksiyonları üzerinde elde edilen sonuçlar, hibrit yapının Yarasa Algoritmasının performansının artışına katkı sağladığını kanıtlamaktadır.

Yilmaz ve Kucuksille (2013), Yarasa Algoritmasının unimodal ve multimodal fonksiyonlar üzerinde global ve lokal arama kabiliyetlerinin yetersizliklerini gidermek amacıyla algoritmaya üç adet modifikasyon uygulamıştır. Geliştirilen yöntemin performansını ölçmek için farklı boyutlarda beş adet unimodal beş adet multimodal olmak üzere on adet benchmark test fonksiyonundan faydalanmıştır. Test sonuçları, geliştirilen yöntemin Yarasa Algoritması üzerinde on adet fonksiyonun dokuzunun tüm problem boyutlarında başarılı olduğunu kalan bir fonksiyonun ise sadece bir problem boyutunda başarı sağlayamadığını dolayısıyla yöntemin oldukça etkili olduğunu işaret etmektedir.

Gandomi vd. (2013), Yarasa Algoritmasını kısıtlı optimizasyon fonksiyonlarını çözmek için geliştirmiştir. Geliştirilen Yarasa Algoritmasının verimliliği, klasik kısıtlı benchmark fonksiyonları ile denenmiştir. Bunun yanı sıra geliştirilen yöntemin performansını test etmek amacıyla literatürde mevcut üç adet kısıtlı mühendislik problemi kullanılmıştır. Yöntemi var olan başka algoritmalar ile kıyaslamıştır. Çalışma, Yarasa Algoritması tarafından elde edilen sonuçların mevcut diğer optimizasyon problemlerinde elde edilen sonuçlardan daha iyi olduğunu göstermektedir. Gandomi vd., ayrıca Yarasa Algoritmasını diğer algoritmalardan farklı kılan arama özelliklerini analiz etmiştir.

Taha ve Tang (2013), Özellik azaltma işlemi üzerinde Yarasa Algoritmasınından faydalanmıştır. On üç adet veri seti ile algoritmanın verimliliğini test etmiştir. Deneysel sonuçlar önerilen metodun performansının diğer özellik seçim metodlarıyla kıyaslandığında eşit ya da daha iyi olduğunu göstermiştir.

Xie vd. (2013), Yarasa Algoritmasının erken yakınsama ve zayıf lokal arama problemlerini ele almıştır. Algoritmanın yakınsama hızını artırmak amacıyla Diferansiyel Evrim Algoritmasının diferansiyel operatöründen faydalanmıştır. Ayrıca algoritmadaki bireylerin farklılaşarak arama uzayının daha etkili taraması ve lokal optimum noktalardan kurtulabilmesi için Lévy Flight dağılımından faydalanılmıştır. Geliştirilen yöntemin performansını ölçmek amacıyla on dört adet benchmark test fonksiyonu kullanmıştır. Test sonuçları, yöntemin sadece etkili ve uygulanabilir olduğunu değil, aynı zamanda çok boyutlu problemlerde optimum noktaya mükemmel bir yakınsama yeteneğine sahip olduğunu göstermiştir.

Sakthivel vd. (2013), Yarasa Algoritmasını gerçek hayat problemlerinden olan "Ekonomik Dağıtım Problemi" üzerinde minimum yakıt maliyetini elde etmek amacıyla kullanmıştır. Elde edilen sonuçlar literatürde var olan diğer çalışmaların sonuçlarıyla kıyaslanmıştır. Sonuçlar, Yarasa Algoritmasının son zamanlarda önerilen diğer algoritmalardan daha iyi performans sergilediğini göstermiştir.

Wang ve Guo (2013), Yarasa Algoritmasını sayısal optimizasyon problemleri için Harmoni Arama (HA) Algoritması ile hibritlemiştir. Yarasa Algoritmasının yakınsama performansını artırmak amacıyla Harmoni Algoritmasındaki akort ayarı operatörü hibrit yapıya eklenmiştir. Harmoni Arama Yarasa Algoritmasının (HA-YA) başarısını kanıtlamak için on dört adet sayısal optimizasyon problemi kullanılmıştır. HA-YA çoğu durumlarda standart YA'ya göre daha iyi sonuçlar ürettiği ya da en azından oldukça rekabetçi olduğunu göstermiştir. Wang ve Guo ayrıca HA-YA'nın başarısını literatürde mevcut diğer sürü tabanlı optimizasyon metotlarıyla (Karınca Kolonisi Optimizasyonu, Diferansiyel Evrim Algoritması, Genetik Algoritma, Harmoni Arama Algoritması, Parçacık Sürü Optimizasyonu, Biyocoğrafya Tabanlı Optimizasyon) kıyaslamıştır. Sonuçlar HA-YA'nın diğer optimizasyon metotlarına kıyasla daha başarılı olduğunu göstermektedir.

Yang (2013), Yarasa Algoritması ile Cuckoo Search Algoritmasının performansını karşılaştırmalı bir şekilde analiz etmiştir. Çalışma sonuçları, her iki algoritmanın çok geniş yelpazedeki uygulamalar üzerinde oldukça başarılı olduğunu göstermiştir.

Sood ve Bansal (2013), Yarasa Algoritması ile K-Medoid Kümeleme Algoritmasını birlikte kullanarak yeni bir hibrit metot önermişlerdir. Bu metodu kümeleme işlemi üzerinde kullanmıştır. Yarasa Algoritmasının bu çalışmadaki rolü K-Medoid Algoritmasının başlangıç değerlerinin oluşturulmasına katkı sağlamak olmuştur. Ayrıca bu çalışmada önerilen metod ile K- Medoid Kümeleme Algoritmasının performans farkları belirtilmiştir.

Baziar vd. (2013), linear olmayan Mikro-Grid enerji yönetim problemine Yarasa Algoritmasını kullanarak optimal çözüm bulmayı amaçlamıştır. Yarasa Algoritmasını bu problem üzerinde daha etkili kılmak ve algoritmanın erken yakınsama problemini ortadan kaldırmak için iki adet modifikasyon önermiştir. Önerilen bu metodun performansını, literatürde aynı problem üzerinde mevcut diğer optimizasyon algoritmaları (Yarasa Algoritması, Genetik Algoritma, Parçacık Sürü Optimizasyonu) ile kıyaslamıştır. Sonuçlar geliştirilen metodun orijinal Yarasa Algoritmasının yanı sıra diğer optimizasyon algoritmalarından daha üstün olduğunu göstermiştir.

Banu ve Chandrasekar (2013), Yarasa Algoritmasını temel alan yeni bir yöntem önermiştir. Bu yöntemi kayıt tekilleştirme problemi üzerinde kullanmıştır. Önerilen bu metod literatürde daha önceden aynı problem üzerine uygulanmış Genetik Algoritma ile kıyaslanmıştır. Sonuçlar, önerilen yöntemin bu problem üzerindeki performansının Genetik Algoritmaya kıyasla daha etkili olduğunu ortaya koymuştur.

Fister vd. (2013), Yarasa Algoritmasınınının sayısal benchmark fonksiyonlar üzerinde daha etkili arama yapabilmesi amacıyla algoritmayı Diferansiyel Evrim Algoritması ve Random Forest makine öğrenme metodu ile hibritlemiştir. Fister vd., geliştirdikleri yöntemi dört tanesi multimodal bir tanesi unimodal olmak

üzere beş adet benchmark test fonksiyonu ile test etmiştir. Sonuçlar geliştirilen yöntemin standart Yarasa Algoritmasına göre daha etkili olduğunu ortaya koymuştur.

Biswal vd. (2013), Yarasa Algoritmasından faydalanarak bir termik santralin üretim maliyetini optimize etmişlerdir. Biswal vd., bu problemin test aşamasını üç ve altı üniteli termal sistem üzerine kurmuşlardır. Bu problem üzerinde elde ettikleri sonuçları Zeki Su Damlacıkları ve Parçacık Sürü Optimizasyonu algoritmaları ile kıyaslamışlardır. Üç ve altı üniteli termal sistem üzerine uygulanan optimizasyon işleminin sonuçları, Yarasa Algoritmasının bu problem üzerinde diğer iki algoritmadan daha etkin olduğunu göstermiştir.

Marichelvam vd. (2013), Yarasa Algoritmasını kullanarak "Çok Aşamalı Hibrit Akış İstasyonu" isimli üretim zamanlama problemini optimize etmiştir. Bu problem; demir, çelik, seramik, kimya, tekstil gibi içerisinde birçok istasyon barındıran ve seri üretim yapan bir fabrikanın zamanlama problemi olarak tanımlanmaktadır. Marichelvam vd., Yarasa Algoritmasının bu problem üzerindeki performansını araştırmış ve literatürdeki diğer çalışmalar ile kıyaslamıştır. Sonuçlar Yarasa Algoritmasının bu problem üzerinde Genetik ve Parçacık Sürü Optimizasyonu algoritmalarına oranla daha iyi olduğunu göstermiştir.

2. YARASA ALGORİTMASININ GELİŞTİRİLMESİ

2.1. Yarasa Algoritması

Sezgisel algoritmalar, çoğunlukla araştırmacıların gözlemleri sonucu doğadan ilham alınarak ya da doğayı taklit ederek önerilen ve birçok probleme esnek çözümler üretebilen algoritmalardır. Örneğin *Yapay Arı Kolonisi Algoritması* (Karaboga ve Basturk, 2007) bal arılarının nektar ararken aralarında yaptıkları iş birliğinden ilham alınarak oluşturulmuş, *Genetik Algoritma* (Tang vd., 1996) ise Darwin yasasına ve genetik bilimine dayanarak oluşturulmuş, *Karınca Kolonisi Algoritması* (Dorigo ve Di Caro, 1999) ise karıncaların avlanma davranışlarından esinlenerek oluşturulmuştur.

Yarasa Algoritması, Yang tarafından 2009 yılında önerilmiş sezgisel bir algoritmadır (Yang, 2010). Yarasa Algoritması yarasaların çoğunlukla avlanma esnasında faydalandıkları ekolokasyon karakteristiklerine dayanmaktadır.

2.1.1. Yarasaların ekolokasyon karakteristikleri

Türlerinin tamamı olmasa bile çoğu türdeki yarasalar, avlarının konumlarını belirlemeleri, aralarında iletişim kurmaları, tamamen karanlık ortamlarda bile etraflarındaki çok ince nesneleri algılayarak onlara çarpmayacak şekilde hareket etmeleri ve karanlık ortamlarda hareket eden farklı türdeki böcek türlerini ayırt etmeleri için *ekolokasyon* olarak adlandırılan bir çeşit radar kullanırlar (Fenton, 2004; Hagen, 2009).

Yarasalar da dahil ekolokasyon kullanan tüm canlılar (yunuslar, balinalar, kır fareleri ve bazı kuş türleri) belli bir frekansta sinyaller yayarlar, bu sinyaller çoğunlukla insan kulağının algılayabileceği (yaklaşık 20 kHz) üst sınırların da ötesindedir ve bu sinyaller "ultrasonik" olarak adlandırılmaktadır. Yarasalar yaydıkları sinyallerde çok yüksek perdeden (>200kHz), düşük perdeye (~10 kHz) –ki bunlar ultrasonik grubunda yer almaz çünkü insan kulağı bu sesleri

kolayca fark edebilir - kadar geniş yelpazedeki frekanslar barındırır (Fenton, 2004).

Yarasalar, yaydıkları sinyallerin çevrelerindeki nesnelere çarpması sonucu oluşan yankıları analiz ederek çevrelerindeki nesneleri belirler, kategorize eder ve yerini tespit ederler (Schnitzler ve Kalko, 2001).

- **Belirleme:** Yarasalar, ekolokasyon sinyali ya da koklama, duyma, görme duyularıyla ilgili hedefin ya da avın varlığına karar verebilirler.

- **Kategorize etme:** Yarasalar ekolokasyon sinyalinin oluşturduğu yankılar aracılığıyla hedef avlarını kategorize edebilirler. Hedeflerin büyüklüğü, şekli, dokusu vb. özellikleri karmaşık bir şekilde kodlanmış halde yankılar içerisinde barınır. Yankının genliği ve frekans modülasyonu avın hareketleri hakkında bilgi verir.

- **Yer Tespiti:** Ekolokasyon, bir hedefin dikey, yatay konumu ve yarasa ile olan uzaklık bilgisini yarasaya bildirir. Sinyalin yarasa tarafından yayılmaya başladığı an ile yankının yarasaya ilk ulaştığı an arasındaki zaman farkı hedef avın yarasa ile olan mesafesini bildirir. Binoral yankılar hedef ile olan yatay açıyı bildirir iken, monaural yankılar hedefe olan dikey açıyı bildirir.

2.1.2. Yarasa Algoritmasının yapısı

Yang, yarasaların ekolokasyon karakteristiklerinden bazılarını ideal hale getirmiştir ve algoritmanın oluşumu için aşağıdaki kuralları temel almıştır.

- Yarasaların avlarına olan uzaklıklarını öğrenmeleri ayrıca av/yiyecek ile nesneleri/engelleri ayırt edebilmeleri için ekolokasyon kullanırlar.

- Yarasalar yiyeceklerini aramak için; v_i hızında, x_i konumunda, sabit f_i frekansında (ya da değişken λ dalga boyunda) , A_0 ses şiddeti ile rastgele

uçarlar. Hedeflerine olan yakınlığına bağlı olarak sinyal yayılım oranlarını $r\epsilon[0,1]$ ve yaydıkları sinyallerin frekans boylarını (ya da dalga boylarını) otomatik olarak ayarlayabilirler.

- Ses şiddeti pek çok yönden farklılık göstermesine rağmen, algoritmada bu faktör pozitif değerden A_0 minimum değere A_{min} farklılık göstermesi beklenmektedir.

2.1.2.1. Yarasa popülasyonunun oluşturulması

Algoritmadaki arama uzayı, av/yiyecek kaynaklarının bulunduğu bir bölge olarak kabul edilmektedir. Algoritma bu yiyecek kaynaklarının en kalitelisinin ya da optimumunun yarasalar tarafından bulunabilmesini amaçlamaktadır. Hedeflenen yiyecek kaynaklarının uzayda hangi bölgede oldukları bilinmediği için N popülasyon sayısında ve d boyutta yarasa popülasyonu arama uzayına rastgele dağılmaktadır. Uzaya dağılan yarasalar bulundukları konuma ait av/yiyecek kaynaklarının uygunluk değerini hesaplamakta ve her bir yarasanın uygunluk değeri hafızada tutulmaktadır.

$$x_{i,j} = x_{minj} + \varphi(x_{maxj} - x_{minj}) \tag{2.1}$$

Yukarıdaki denklemde; $i = 1,2 \dots N \quad j = 1,2 \dots d$ iken $x_{i,j}$ i. yarasanın j. boyutunu ifade etmektedir, x_{minj} ve x_{maxj} sırasıyla j. boyutun alabileceği minimum ve maksimum değerleri ifade etmektedir. φ ise [0,1] aralığında rastgele dağılmış bir değeri temsil etmektedir.

2.1.2.2. Yarasaların arama uzayı içerisindeki hareketleri

Yarasaların uzaya rastgele dağılımından sonra sahip oldukları uygunluk değerleri tüm popülasyonun bir sonraki hareket yönü ve şiddetine etki etmektedir. Yarasalar, belli bir frekansa (f_i) ve popülasyondaki en iyi bireyin sahip olduğu çözüm değerine (x_*) bağlı olarak ürettikleri hız (v_i) değeri ile bir sonraki adımda bulunacakları konumlarını belirlemektedirler.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (2.2)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i \quad (2.3)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (2.4)$$

Eşitlik 2.2-2.4'de f_i, i. yarasaya ait frekans değerini; f_{min} ve f_{max} sırasıyla minimum ve maksimum frekans değerlerini; β, [0,1] aralığında rastgele dağılmış bir değeri; x_*, popülasyon içerisinde t. süreye kadar en iyi bireyin çözüm değerlerini; v_i^t, t. anda i. bireyin hızını temsil etmektedir.

Algoritmanın hız ve konum güncelleme işlemleri frekansın adım aralığını kontrol etmesi açısından Parçacık Sürü Optimizasyonu (Kennedy ve Eberhart, 1995) ile bir dereceye kadar benzerlik taşımaktadır (Yang, 2010).

2.1.2.3. Algoritmanın yerel arama kabiliyeti

Yang, algoritmanın yerel arama kabiliyetini artırmak amacıyla popülasyondaki bireylerin mevcut av/yiyecek kaynaklarının civarlarında daha kaliteli kaynaklara yönelmesine olanak sağlayacak yapıyı algoritmaya dâhil etmiştir. Bu amaçla belli şartları sağlayan her birey/çözüm, popülasyon içerisinden uygunluk değerinin kalitesine bağlı olarak bir başka çözüm seçmekte ve seçtiği çözümün etrafında yeni kaynağa yönelmektedir.

$$x_{new} = x_{old} + \varepsilon \bar{A}^t \quad (2.5)$$

Eşitlik 2.5'te ε [-1,1] aralığında rastgele üretilen bir değeri; x_{old}, belli mekanizmalar (rulet tekerleği, sıralı seçim, elitizm vb.) aracılığıyla popülasyon içerisinden uygunluk değerinin kalitesine bağlı olarak seçilen bireyin çözüm değerlerini; $\bar{A}^t$, t anında tüm yarasaların ortalama ses şiddeti değerini temsil etmektedir.

2.1.2.4. Ses şiddeti ve sinyal yayılım oranı

Yarasaların ekolokasyon ile ürettikleri sesin şiddeti ve sinyal yayılım oranları iterasyon ilerledikçe ve istenilen hedefe yaklaştıkça güncellenmesi gerekmektedir. Yarasa avına yaklaştıkça ses şiddeti (A) azalırken, sinyal yayılım oranı (r) artmaktadır (Şekil 2.1).

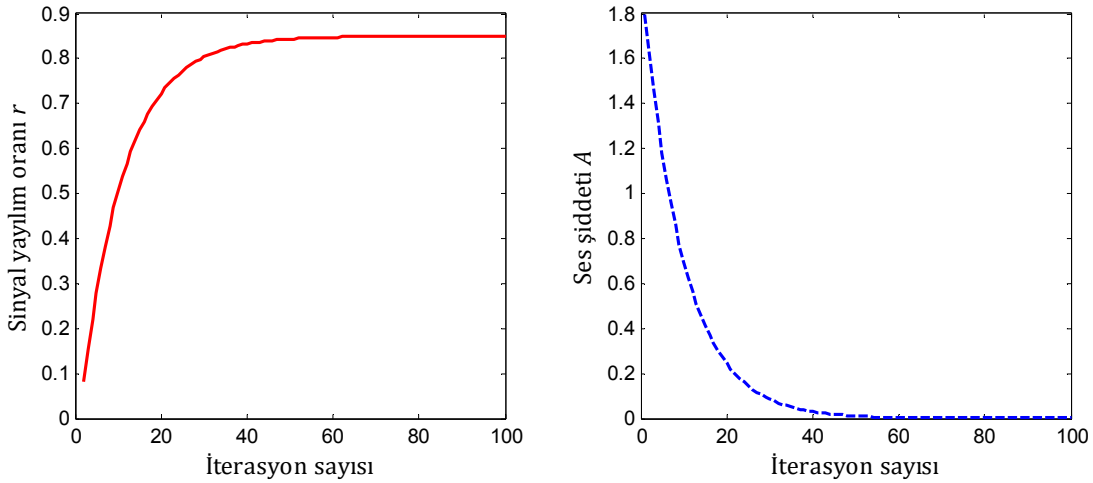
$$A_i^{t+1} = \alpha A_i^t \tag{2.6}$$

$$r_i^{t+1} = r_i^0(1 - e^{-\gamma t}) \tag{2.7}$$

Sırasıyla eşitlik 2.6 ve 2.7'deki α ve γ değerleri sabit değerlerdir. $0 < \alpha < 1$ ve $\gamma > 0$ durumunda aşağıdaki bağıntı elde edilmektedir:

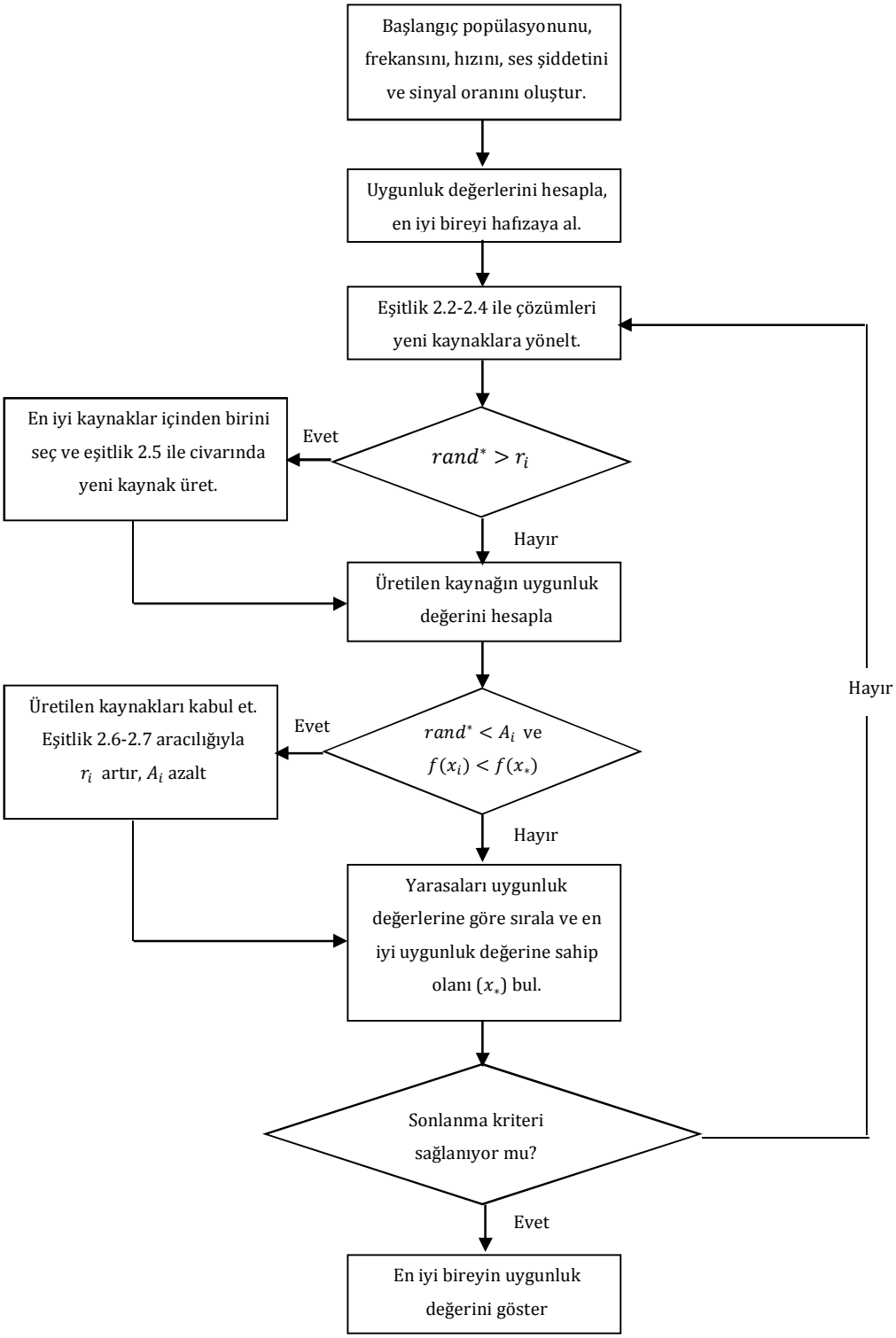
$$A_i^t \to 0, \qquad r_i^t \to r_i^0, \quad t \to \infty \text{ iken} \tag{2.8}$$

Burada başlangıç ses şiddeti (A_i^0) ve sinyal yayılım (r_i^0) oranı değerleri rastgele belirlenmekte ve genelde $A_i^0 \epsilon [1,2], r_i^0 \epsilon [0,1]$ aralığında değer almaktadır.



Şekil 2.1. Sinyal yayılım oranı ve ses şiddeti değerlerinin iterasyon sayısına bağlı değişimi

Yukarıda anlatılan tüm bölümlerin akış diyagramı gösterimi Şekil 2.2'de belirtilmiştir.



* 0 ile 1 aralığında üretilmiş rastgele bir değer

Şekil 2.2. Yarasa Algoritmasının akış şeması gösterimi

2.1.2.5. Yarasa algoritmasının adımları

Yarasa Algoritmasının sayısal fonksiyonlar üzerinde sözde kod gösterimi aşağıda verilmiştir.

1. Eşitlik 2.1 ile tüm yarasa popülasyonunu (x) arama uzayına dağıt, başlangıç hızlarını $v = 0$ olarak belirle.
2. Tüm yarasanın bulduğu av/yiyecek kaynağının kalitesini, uygunluk değerini, ($fitness^0$) hesapla. En iyi uygunluk değerini ($fitness_{best}$) hafızada tut.
3. Her bir yarasaya ait başlangıç frekans (f), ses şiddeti (A) ve sinyal yayılım oranı (r) değerlerini belirle.
4. **repeat**
5. **for** $i = 1$ to N **do**
6. Eşitlik 2.2-2.4 aracılığıyla i. yarasayı arama uzayında başka bir noktaya yerleştir.
7. **if** $rand > r_i$ **then**
8. Popülasyon içerisinden uygunluk değerine bağlı olarak en iyi yarasalar içerisinden birini seç (x_m).
9. i. bireyi eşitlik 2.5 ile arama uzayında seçilen yarasanın (x_m) civarında bir noktaya gönder.
10. **end if**
11. Yarasanın konumlandığı noktadaki av/yiyecek kaynağının kalitesini, uygunluk değerini, ($fitness_i^t$) hesapla.
12. **if** $rand < A_i$ **and** $fitness_i^t < fitness_i^{t-1}$ **then**
13. Yeni noktadaki av/yiyecek kaynaklarını kabul et.
14. Eşitlik 2.6-2.7 ile sinyal yayılım oranını artır, ses şiddetini azalt.
15. **end if**
16. **if** $fitness_i^t < fitness_{best}^t$ **then**
17. Bulunan kaynağı en iyi kaynak olarak güncelle.
18. **end if**
19. **end for**
20. **until** Durma kriteri

2.2. Geliştirilmiş Yarasa Algoritması

Yarasa Algoritması, keşfetme (exploration) ve keşfedilen çözümleri iyileştirme (exploitation) kabiliyetleri zayıf olan bir algoritmadır. Algoritma genelde az boyutlu benchmark fonksiyonlarda etkili bir performans sergilerken, fonksiyondaki boyut sayısı arttıkça erken yakınsama problemleri baş göstermekte ve algoritmanın optimizasyon performansı zayıflamaktadır (Fister vd., 2013).

Bu çalışmada standart Yarasa Algoritmasının yukarıda belirtilen sorunlarının üstesinden gelmek için algoritmaya üç adet geliştirme yapısı önerilmiştir. Bu yapılar sırasıyla aşağıda anlatılmaktadır.

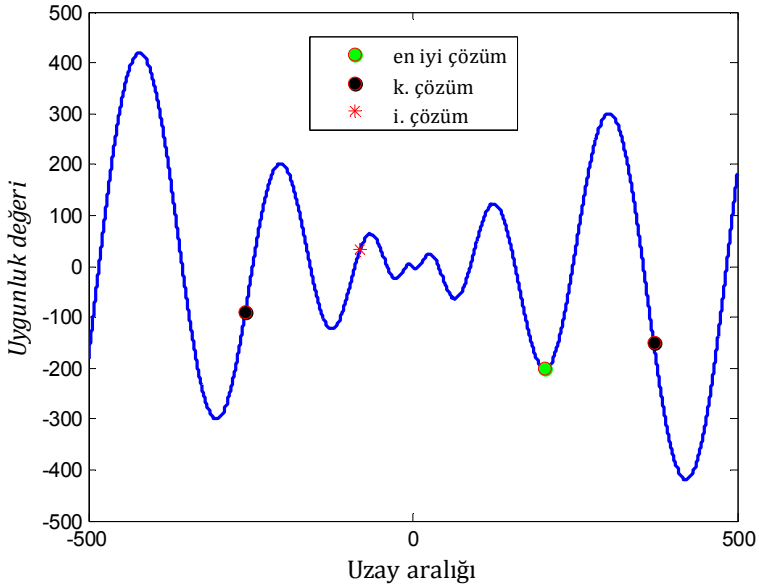
1) Yarasa Algoritmasının hız güncelleme denklemi (Eşitlik 2.3) analiz edildiğinde eşitliğin iki terimden oluştuğu görülmektedir. Birinci terim popülasyonun hızını belirleyen; ikinci terim ise en iyi bireyin rehberliğinde *i*. bireyin hız vektörüne etki eden terimdir. Eşitliğin birinci terimi algoritmanın global arama yapmasına (exploration), ikinci terim ise lokal arama yapmasına (exploitation) katkı sağlamaktadır. Hız güncelleme eşitliğine yalnızca birinci terimin etki etmesi durumunda algoritmadaki bireyler hızlarını ve doğrultularını koruyarak arama uzayının sınırlarına dayanacaklardır dolayısıyla yakınsama hızı oldukça yavaşlayacak ve optimizasyon performansı zayıflayacaktır. Diğer taraftan yalnızca ikinci terimin eşitliğe etki ettiği durumda ise algoritma en iyi bireyin rehberliğinde uzayda bir bölgeye yakınsayacak ve erken yakınsama problemiyle karşı karşıya kalacaktır. Dolayısıyla bu geliştirme yapısında hız güncelleme denkleminin uygun bir yapı ile geliştirilerek algoritmanın global (keşfetme) ve lokal arama (sömürme) kabiliyetleri arasında bir denge oluşturulması amaçlanmıştır.

   Daha önceden belirtildiği gibi, Yarasa Algoritmasının hız ve konum güncellemeleri Parçacık Sürü Optimizasyonu ile bir takım benzerlikler taşımaktadır. Yarasa Algoritmasında popülasyonun hız faktörü etkisini

(Eşitlik 2.3'ün birinci terimi) iterasyon ile beraber azaltarak lokal arama kabiliyetlerini ön plana çıkarmak amacıyla, Parçacık Sürü Optimizasyonu Algoritmasındaki hız vektörüne uygulanan atalet ağırlığı katsayısı Yarasa Algoritmasına uygulanmıştır (Parçacık Sürü Optimizasyonu Algoritması ve geliştirilmiş versiyonu sırasıyla bölüm 2.2.2 ve bölüm 2.2.2.1'de detaylı bir şekilde anlatılmaktadır). Bu modifikasyonda amaç Yarasa Algoritmasının optimizasyon süreci başlarında global arama kabiliyetini ön plana çıkarmak, atalet ağırlık katsayısı değerinin optimizasyon sürecinin sonuna doğru azalması ile lokal arama kabiliyetini ön plana almak olmuştur. Ayrıca bu modifikasyon yapısı ilk olarak Yilmaz ve Kucuksille (2013) tarafından Yarasa Algoritmasının sürekli optimizasyon problemleri üzerinde geliştirilmesi amacıyla kullanılmıştır. Modifikasyon ile eşitlik 2.3 aşağıdaki gibi güncellenmiştir.

$$v_i^t = w(v_i^{t-1}) + (x_i^t - x_*)f_i \tag{2.9}$$

2) Bir önceki geliştirme basamağında açıklandığı gibi hız denkleminin (Eşitlik 2.3) ikinci terimi bireylerin en iyi çözüm rehberliğinde uzayda hareket etmesine olanak tanımaktadır. Yalnızca bu terimin kullanılması algoritmanın erken yakınsamasına ve uzayda istenmeyen lokal noktalara takılmasına sebep olmaktadır. Bu nedenle atalet ağırlığı modifikasyonu algoritmaya dâhil edilmiştir. Atalet ağırlığı, optimizasyon süreci başlarında bireylerin hızlarının etkisini (birinci terim) yüksek tutarken, sürecin sonlarına doğru bu etkiyi oldukça azaltmakta dolayısıyla ikinci terimin etkisini artırmaktadır. Bu durumda optimizasyon sürecinin sonlarına doğru elde edilen en iyi birey istenmeyen bir lokal noktaya takılmış ise uzayda arama yapacak diğer bireylerin o lokal nokta civarında olması kaçınılmaz olacaktır (birinci terimin etkisinin çok az olmasından dolayı). Yarasa Algoritmasının schwefel fonksiyonu (fonksiyon denklemi için bkz. Çizelge 3.9'da 23 numaralı fonksiyon) üzerindeki popülasyon dağılımı Şekil 2.3'te gösterilmektedir.



Şekil 2.3. Yarasa Algoritmasındaki popülasyonun schwefel fonksiyonu üzerinde dağılımı

Şekil 2.3'te yeşil nokta popülasyondaki en iyi bireyi, siyah nokta popülasyondaki *k*. bireyi, kırmızı yıldız işareti ise *i*. bireyi belirtmektedir. Şekilde görüldüğü üzere yeşil nokta ile ifade edilen birey popülasyondaki en iyi bireydir fakat bu birey lokal bir bölgeye takılmıştır. Daha önce de belirtildiği gibi optimizasyon sürecinin sonlarında *i*. bireyin uzayda arama yapmasına en fazla popülasyondaki en iyi birey rehberlik edecektir. Bu durumda, yukarıdaki şekle göre *i*. birey bir sonraki adımda en iyi bireyin bulunduğu lokal alana doğru hareket edecektir. Oysa ki *k* gibi en iyi olmayan ancak umut vadeden bireylerin de rehber olması *i*. bireyin uzayda daha uygun bölgelere yönelimini sağlayacaktır. Bu durumu sağlamak için eşitlik 2.9'a en iyi bireyin dışında etki edecek bir başka rehber birey (k) ilave edilmiş ve eşitlik aşağıdaki gibi güncellenmiştir.

$$v_i^t = w(v_i^{t-1}) + (x_i^t - x_*) f_i \zeta_1 + (x_i^t - x_k^t) f_i \zeta_2 \tag{2.10}$$

$$\zeta_2 = 1 - \zeta_1 \tag{2.11}$$

yukarıdaki eşitlikte k, $(i \neq k)$ popülasyonda uygunluk değeri yüksek olan bireyler içerisinden rastgele seçilmiş bir bireyi; x_k^t, t anında k. bireyin

çözüm değerini; ζ_1 öğrenme faktörü, 0-1 aralığında değişen ağırlık katsayısını temsil etmektedir. Eşitlikten de anlaşılacağı üzere ζ_1 değeri 1'e yaklaştıkça en iyi bireyin, 0'a yaklaştıkça *k*. bireyin *i*. bireye olan etkisi fazla olacaktır. Bu etkinin optimizasyon süresi boyunca sabit kalması algoritmanın problemdeki global noktaya yakınsayamamasına dolayısıyla optimizasyon performansının zayıflamasına sebep olacaktır. Bu nedenle ζ_1 faktörünün başlangıçta belirlenen değerden 1'e doğru güncellenmesi gerekmektedir. Böylece *k*. bireyin etkisi optimizasyon sürecinde giderek azalacak en iyi bireyin etkisi giderek artacaktır. Aşağıda ζ_1 faktörünün optimizasyon süresince güncellenmesini sağlayan eşitlik verilmiştir.

$$\zeta_1 = 1 + (\zeta_{init} - 1)\frac{(iter_{max} - iter)^n}{(iter_{max})^n} \tag{2.12}$$

Eşitlik 2.12'de ζ_{init} başlangıç etki değerini; $iter_{max}$, maksimum iterasyon sayısını; $iter$, iterasyon sayısını ve n nonlineerlik katsayısını temsil etmektedir.

Bölüm 3.1.5'te ζ_1 parametresinin Yarasa Algoritması üzerinde eğitilmesiyle oluşan sonuçlar verilmiştir.

3) Yarasa Algoritmasının kaba (pseudo) kodu analiz edildiğinde (bkz. bölüm 2.1.2.5), algoritmanın lokal arama işlevinin sadece 7. işlem adımını sağlayan bireyler tarafından gerçekleştirildiği görülmektedir. Diğer bir deyişle 0-1 aralığında rastgele üretilen bir değerin *i*. bireyin ses şiddeti değerinden büyük olması durumunda *i*. birey lokal arama yapabilecektir. Aksi takdirde *i*. birey uzayı keşfetmeye, uzayda yeni bölgelere yönelmeye devam edecektir. Eşitlik 2.7 ve Şekil 2.1'den anlaşılacağı üzere *i*. bireye ait sinyal yayılım oranı değeri optimizasyon süresince sürekli olarak artmaktadır. Dolayısıyla optimizasyon süresince algoritmanın lokal arama kabiliyeti azalacaktır. Bu durumun önüne geçmek için Yarasa Algoritması, Yabani Ot Optimizasyonu ile

birleştirilmiştir (Yabani Ot Optimizasyonu bölüm 2.2.1'de detaylı bir şekilde anlatılmaktadır).

Optimizasyon sürecinin her aşamasının sonunda Yarasa Algoritmasında bulunan tüm bireyler Yabani Ot Optimizasyonundaki bireyleri temsil etmektedirler. Bölüm 2.2.1'de de anlatıldığı gibi bireyler önceden belirlenen maksimum üretilecek tohum sayısı kadar bulundukları çevrede çoğalmaktadırlar. Yeni üretilen bireylerin uygunluk değeri hesaplandıktan sonra popülasyondaki tüm bireyler (ebeveyn ve yeni üretilen bireyler) uygunluk kalitesine göre sıralanmakta ve umut vadeden bireyler bir sonraki iterasyonda algoritmanın popülasyonunu oluşturmaktadırlar.

Yarasa Algoritmasının geliştirilmesinde katkıda bulunan yöntemlerin açıklamaları aşağıda detaylı bir şekilde anlatılmıştır.

2.2.1. Yabani ot optimizasyonu

Yabani Ot Optimizasyonu, Mehrabian ve Lucas (2006) tarafından yabani ot kolonilerinden ilham alınarak önerilmiş sezgisel bir optimizasyon algoritmasıdır.

Son derece güçlü ve istilacı büyüme alışkanlığıyla çevresi için ciddi bir tehdit oluşturan yabani ot kolonisi, zirai açıdan kültür bitkilerinin yetişmesine engel teşkil etmektedir. Aşağıda yabani otların tarımsal alanlardaki dayanıklılığını ve bulundukları çevrelere kolayca adapte olabilirliğini kanıtlayan bazı özellikleri verilmiştir (Mehrabian ve Lucas 2006).

- Binlerce yıldır toprağın işlenmesine ve yabani otlarla mücadele edilmesine rağmen günümüzde halen bu tür otlar bulunmaktadır.
- Tarımsal alanlarda elli yılı aşkın süredir herbisit kullanılmasına rağmen bu alanlarda yabani otların yeniden var olduğu ve büyüdüğü gözlenmektedir.

- Yeni yabani ot türlerinin tüm dünya genelinde hızlı bir şekilde yayıldığı görülmektedir.
- İnsanlar artık bu tür yabani otları yeni bir kategoride görmek zorunda kalmıştır: Herbiside dirençli otlar.

Yabani otların güçlülüğünü ve istilacılığını kanıtlayan bu özellikler, yabani otların kolonileşme davranışlarının taklit edilerek güçlü bir sezgisel algoritmaya dönüştürülmesinde en büyük etken olmuşlardır.

Yabani otların kolonileşme davranışının taklit edilerek algoritma yapısına uyarlanabilmesi için kolonileşme sürecinde bazı varsayımlar oluşturulmuştur:

1. Belirli sayıdaki bitki tohumu arama uzayına rastgele dağılacaktır (yabani ot populasyonun oluşturulması).
2. Her tohum çiçekli bitki haline gelecek ve uygunluk değerine bağlı olarak kendi tohumlarını üretecektir (popülasyonun çoğalması).
3. Üretilen tohumlar arama uzayına rastgele dağıtılacak ve gelişerek bitki haline gelecektir (uzaysal dağılım).
4. Yukarıda belirtilen işlem basamakları önceden tanımlanan maksimum bitki sayısına ulaşana kadar devam edecek. Bitki popülasyonu bu sayıya ulaştığında ya da bu sayıyı aştığında en iyi uygunluk değerine sahip bitkiler hayatta kalacak diğerleri popülasyonun dışında kalacaklardır (rekabetçi kıyaslama).
5. Tüm bu süreç belirlenen maksimum iterasyon sayısı kadar devam edecektir.

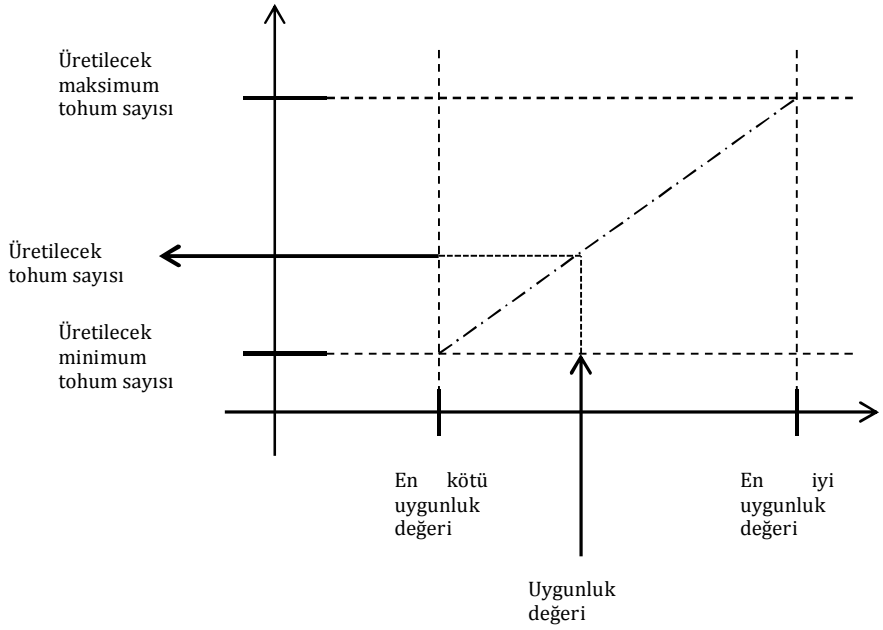
Yukarıda özetlenen bu aşamalar aşağıda detaylı bir şekilde anlatılmaktadır.

2.2.1.1. Yabani ot popülasyonun oluşturulması

Yabani ot popülasyonu (çözüm kümesi), tıpkı yarasa algoritmasında olduğu gibi başlangıç esnasında belirlenen alana (arama uzayına) rastgele dağılırlar.

2.2.1.2. Popülasyonun çoğalması

Popülasyon içerisinde yer alan her bir çözüm, kendi uygunluk değerinin popülasyon içerisinde bulunan diğer çözümlerin uygunluk değerleri oranına göre tohum (yeni çözüm) üretecektir. Diğer bir deyişle, popülasyondaki bir bireyin uygunluk değeri ne kadar iyi ise o kadar fazla tohum üretme hakkı olacaktır (Şekil 2.4).



Şekil 2.4. Yabani ot kolonisinin tohum üretme işlemi

Evrimsel optimizasyon algoritmalarında en iyi uygunluk değerlerine sahip olan çözümler diğer çözümlere nazaran daha uygulanabilir çözümler olarak kabul edilmektedir. Bu nedenle uygunluk değeri düşük olan çözümlerin geliştirilmesine imkân sunulmamaktadır. Fakat bu bakış açısı evrimsel algoritmaların olasılıksal karakteristiğine ters düşmektedir. Çünkü bazen optimizasyon sürecinde uygulanabilirliği düşük olan çözümler uygulanabilir çözümlerden daha fazla yararlı bilgi taşıyabilmektedirler (Yuchi ve Kim, 2005). Yabani Ot Optimizasyonunda tohum üretimi sürecine sadece uygulanabilir bireylerin değil tüm popülasyonun dâhil edilmesi belirtilen bu durumu ortadan kaldırmaktadır. Algoritmada tohum üretim işlemi eşitlik 2.13'e göre yapılmaktadır.

$$s = \left\lfloor s_{min} + (s_{max} - s_{min}) \left(\frac{u_i - u_{worst}}{u_{best} - u_{worst}} \right) \right\rfloor \quad (2.13)$$

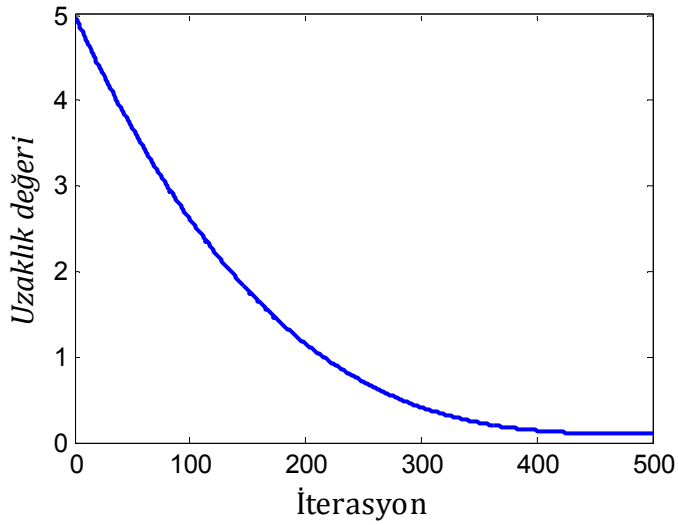
Yukarıdaki eşitlikte s_{max} ve s_{min} sırasıyla, üretilecek maksimum ve minimum tohum sayısını; u_{worst} ve u_{best} sırasıyla en kötü ve en iyi uygunluk değerlerini, u_i ise *i*. bireyin uygunluk değerini belirtmektedir.

2.2.1.3. Uzaysal dağılım

Bir önceki aşamada üretilen yeni tohumlar (çözümler) eşitlik 2.14 ile arama uzayında ebeveyn bitkilerin civarlarına konuşlanmaktadır.

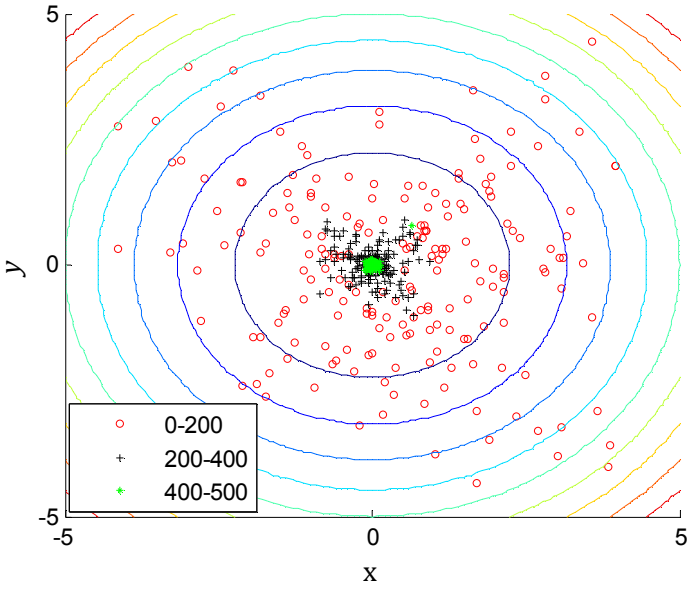
$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n} (\sigma_{initial} - \sigma_{final}) + \sigma_{final} \quad (2.14)$$

Burada $iter_{max}$ maksimum iterasyon sayısını, $iter$ iterasyon sayısını, $\sigma_{initial}$ ve σ_{final} sırasıyla başlangıç ve bitiş uzaklık değerlerini, n ise nonlineerlik katsayısını belirtmektedir. Şekil 2.5'te üretilen tohumların ebeveynlerine olan uzaklığını (σ_{iter}) gösteren grafik verilmiştir ($iter_{max} = 500$, $\sigma_{initial}$=5, σ_{final}=0 ve n=3 iken).



Şekil 2.5. Üretilen tohumların ebeveynlerine olan uzaklıklarının iterasyona göre değişimi

Eşitlik 2.14'te görüldüğü gibi üretilen yeni tohumların ebeveynlerine olan uzaklığı iterasyon ilerledikçe azalacaktır. Şekil 2.6'da beş yüz iterasyon boyunca üretilen tohumların orijin noktasında bulunan ebeveynlerine olan uzaklığını gösterilmektedir.



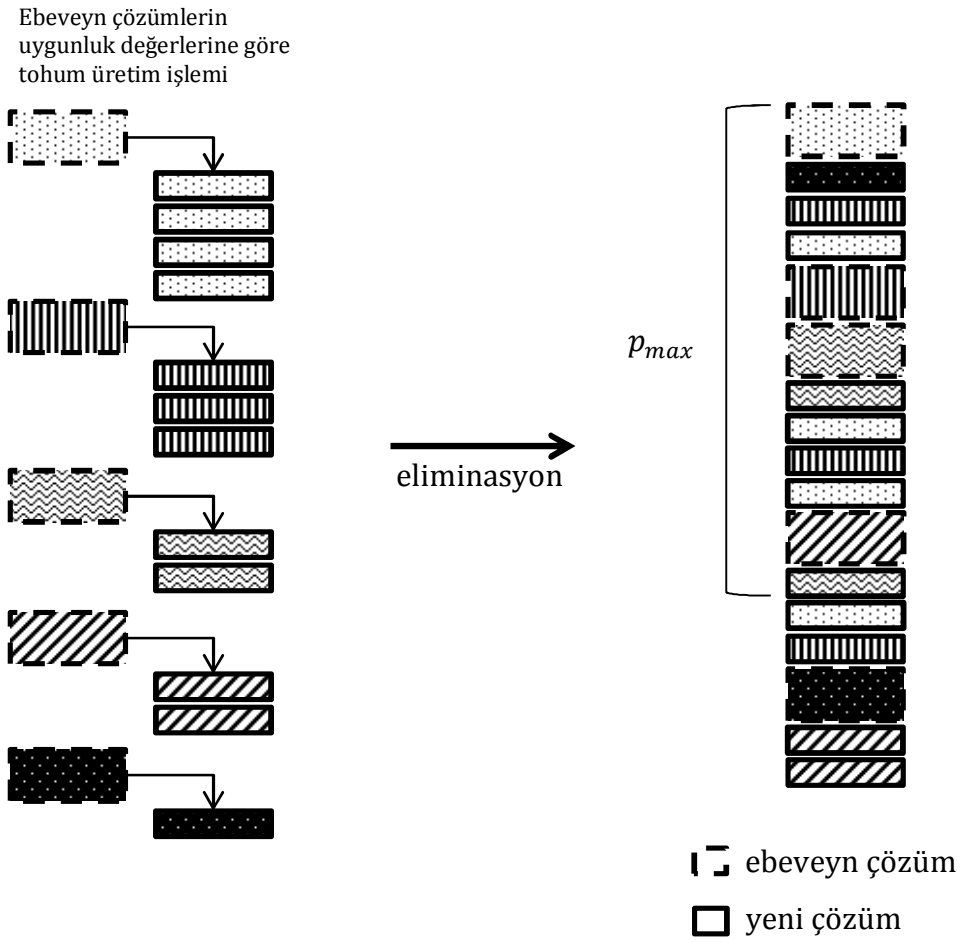
Şekil 2.6. Tohumların iterasyonlar boyunca orijin noktasına olan uzaklıkları

Yukarıdaki şekilde kırmızı, siyah ve yeşil işaretçiler sırasıyla ilk iki yüz, iki yüz ile dört yüz ve son yüz iterasyon boyunca tohumların orijine olan uzaklığı belirtmektedir.

2.2.1.4. Rekabetçi kıyaslama

Ebeveynler (çözümler) kendi tohumlarını ürettikten sonra, önceden belirlenen maksimum popülasyon sayısına (p_{max}) ulaşılıp ulaşılmadığı kontrol edilir. Eğer üretilen tohum sayısı ile ebeveynlerin sayıları toplamı bu sayıya eşit ya da bu sayıdan fazla ise yeni üretilen tohumlarla birlikte tüm popülasyon herhangi bir mekanizma ile eliminasyona tabi tutulur. Eliminasyon işleminde, umut vadeden çözümlerin varlıklarını koruyup onların yeni tohumlar üretebilmeleri; kalitesi düşük olan çözümlerin ise popülasyondan ayrılmaları beklenmektedir. Bu aşamada uygunluk değeri yüksek olan ebeveyn çözümleri ile yeni üretilen çözümler bir sonraki iterasyonda arama yapacak popülasyonu oluşturmaktadır.

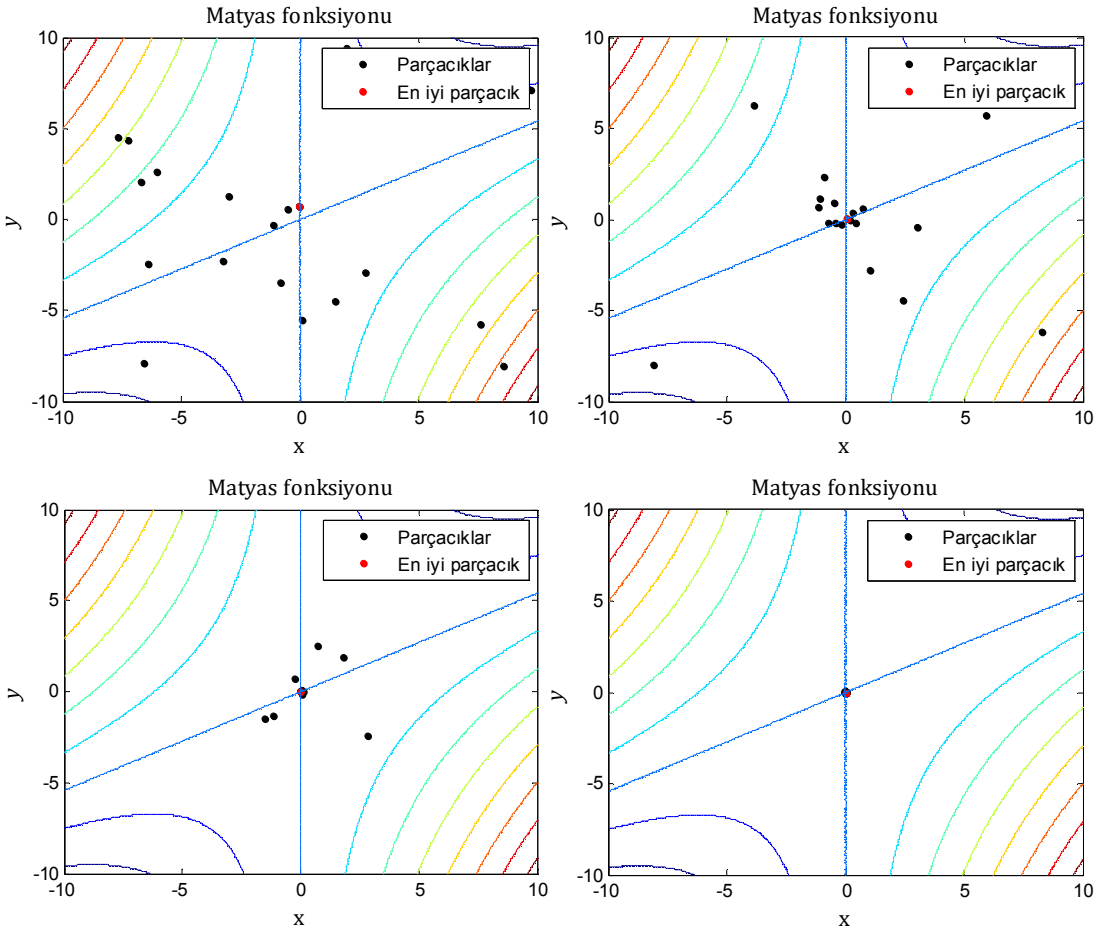
Yabani Ot Optimizasyonunun bir problemi optimize etme prensibi Şekil 2.7'de anlatılmaktadır. Şekilde yer alan açık nokta desenli ebeveyn çözüm popülasyonda uygunluk değeri en iyi olan çözümdür ve en fazla tohum üretmiştir. Koyu nokta desenli çözüm ise uygunluk değeri en kötü çözümdür ve sadece bir adet tohum üretebilmiştir. Tohum üretme işleminin hemen ardından toplam popülasyon sayısı (ebeveyn ve üretilen tohumlar) önceden belirlenen maksimum popülasyon sayısını (p_{max}) aştığı için eliminasyon işlemine tabi tutulmuştur. Uygunluk değerleri en iyi çözümler bir sonraki iterasyonda popülasyonu oluşturmuştur. Şekil dikkatlice incelendiğinde uygunluk değeri en düşük olan koyu nokta desenli ebeveyn çözümün ürettiği tohumun uygunluk değeri oldukça iyi çıktığı görülecektir. Bu da önceki bölümde anlatılan "*optimizasyon sürecinde uygulanabilirliği düşük olan çözümler uygulanabilir çözümlerden daha fazla bilgi taşıyabilmektedirler*" görüşünü desteklemektedir.



Şekil 2.7. Yabani ot optimizasyonunun tohum üretim ve eliminasyon işlemleri

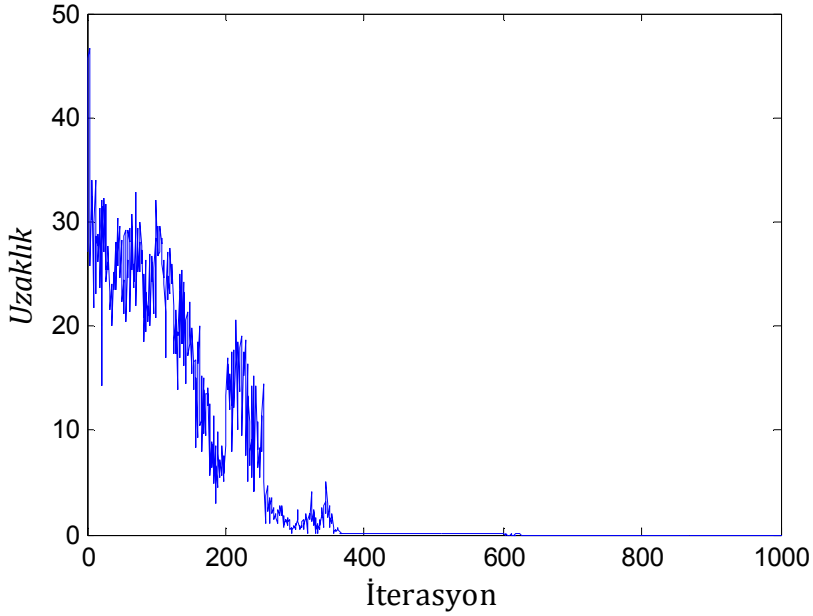
2.2.2. Parçacık sürü optimizasyonu

Parçacık sürü optimizasyonu (PSO), Kennedy ve Eberhart (1995) tarafından kuş ya da balık sürülerinin havada meydana getirdikleri sosyal davranışlardan esinlenerek önerilmiş popülasyon tabanlı sezgisel bir algoritmadır. Şekil 2.8, sürüdeki bireylerin matyas fonksiyonunun (fonksiyon denklemi için bkz. 11'da 8 numaralı fonksiyon) optimum noktasına ulaşırken oluşturdukları kümelenme davranışını göstermektedir. Şekil, sırasıyla bireylerin uzaydaki başlangıç esnasındaki (sol üst), yüzüncü (sağ üst), ikiyüzüncü (sol alt) ve son iterasyondaki (sağ alt) koordinatlarından oluşmaktadır. Şekil analiz edildiğinde, bireylerin/parçacıkların uzaya rastgele dağılımlarının ardından iterasyonlar ilerledikçe birbirlerine ve optimum noktaya doğru hareket ettikleri görülmektedir.



Şekil 2.8. Parçacıkların arama uzayında aralarında etkileşim kurarak optimum noktaya yönelimi

Parçacıkların birbirleri arasındaki uzaklık değerlerinin iterasyona bağlı değişimi Şekil 2.9'da verilmiştir. Daha önceden de belirtildiği gibi iterasyon ilerledikçe parçacıkların birbirlerine olan uzaklığı giderek azalmaktadır.



Şekil 2.9. Parçacıkların birbirine olan uzaklıklarının iterasyona göre değişimi

Algoritma, kuş ya da balık sürüsünün yiyecek aramak için uzaya rastgele dağılmasıyla başlamaktadır. *N* adet parçacık ve *d* boyuttan oluşan popülasyonun matrisi eşitlik 2.15 ve matristeki *i*. parçacık eşitlik 2.16'daki gibi ifade edilmektedir.

$$x = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} \end{bmatrix} \tag{2.15}$$

$$x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}] \tag{2.16}$$

Parçacıklar, uzayın hangi noktalarında ve aradıkları yiyeceklere ne kadar uzaklıkta olduklarını öğrenmek için bulundukları koordinatların uygunluk değerlerini hesaplamaktadırlar. Parçacıklar, aradıkları yiyeceğin (optimum noktanın) uzaydaki koordinatlarını bilmediklerinden dolayı her iterasyonda iki

"en iyi" uygunluk değerine sahip parçacık ile etkileşim kurarak uzayda hareket etmektedirler. Bunlardan bir tanesi, parçacığın kendisinin o ana kadar ulaştığı en iyi koordinat bilgisini (*pbest*) belirtirken; diğeri, popülasyondaki herhangi bir parçacık tarafından o ana kadar ulaşılmış en iyi koordinat bilgisini (*gbest*) belirtmektedir. *i*. parçacığın *pbest* ve popülasyonun *gbest* değerleri sırasıyla eşitlik 2.17 ve 2.18'de gösterilmektedir. Algoritmada bir parçacığın her iterasyondaki hız değişimini veren bir vektör bulunmaktadır. *i*. parçacığın hız değişim vektörü eşitlik 2.19'da verilmiştir.

$$pbest_i = \left[P_{i,1}, P_{i,2}, \dots, P_{i,d}\right] \tag{2.17}$$

$$gbest = [P_1, P_2, \dots, P_d] \tag{2.18}$$

$$v_i = \left[v_{i,1}, v_{i,2}, \dots, v_{i,d}\right] \tag{2.19}$$

Algoritmadaki parçacıklar uzayda eşitlik 2.20 ve 2.21'e göre hareket etmektedirler. Eşitlik 2.20'de *i*. parçacığın *t*. süredeki yönelim hızı hesaplanmaktadır. Eşitlik 2.21'de ise *i*. parçacığın, eşitlik 2.20'de hesaplanan yönelim hızına göre, uzayda yeni konumuna hareket etmesi sağlanmaktadır.

$$v_i^t = v_i^{t-1} + c_1\rho_1(pbest_i^{t-1} - x_i^{t-1}) + c_2\rho_2(gbest^{t-1} - x_i^{t-1}) \tag{2.20}$$

Yukarıdaki eşitliğin ikinci terimi *bilişsel*, üçüncü terimi ise *sosyal* terim olarak ifade edilmektedir. Eşitlikte c_1 ve c_2 *ivmelenme katsayısı* olarak adlandırılan birer öğrenme faktörleridir ve sırasıyla parçacığın *t*. iterasyondaki kendi tecrübesine (bilişsel) ya da popülasyondaki en iyi parçacığın tecrübesine (sosyal) göre hareket etmesini sağlamaktadırlar. p_1 ile p_2 0 ve 1 aralığında rastgele üretilen değeri temsil etmektedir.

$$x_i^t = x_i^{t-1} + v_i^t \tag{2.21}$$

2.2.2.1. Geliştirilmiş parçacık sürü optimizasyonu

Parçacık sürü optimizasyonu algoritmasının performansını geliştirmek amacıyla Shi ve Eberhart (1998) algoritmanın hız vektörünün hesaplandığı denklem (Eşitlik 2.20) üzerinde modifikasyon geliştirmiştir.

Eşitlik 2.20 incelendiğinde bu eşitliğin üç kısımdan oluştuğu görülmektedir. Birinci kısım parçacığın önceki hızını, ikinci ve üçüncü kısım parçacığın *t*. andaki hızın değişimine etki eden faktörleri temsil etmektedir. Buradan hareketle, ikinci ve üçüncü kısımların hız vektörüne etki etmediği varsayıldığında, parçacığın hızının ve yönünün korunarak o doğrultuda uzayın sınırına kadar ilerlediği görülecektir. Aynı şekilde eşitliğin birinci kısmının etkisiz olduğu varsayıldığında, parçacıklar uzayda yalnızca bulundukları alanları arayabileceklerdir. Böyle bir durumda uzayın optimum çözüm noktaları yalnızca başlangıç noktalarından biri olduğunda algoritma bu optimum noktayı bulmuş olacaktır; aksi takdirde algoritmanın optimum noktayı bulması imkansız olacaktır. Sonuç olarak eşitliğin birinci kısmının algoritmaya global arama, ikinci ve üçüncü kısmının ise lokal arama yapma kabiliyeti kazandırdığı görülmektedir (Shi ve Eberhart, 1998).

Yukarıdaki paragrafta verilen bu bilgiler geliştirilen yapının en önemli nedenini oluşturmaktadır. Shi ve Eberhart, algoritmada global ve lokal arama kabiliyetlerini etkin kılmak için *atalet ağırlığı* olarak adlandırılan bir katsayı vektörü (w) geliştirmişlerdir. Geliştirdikleri yapı ile eşitlik 2.20 aşağıdaki gibi güncellenmiştir.

$$v_i^t = w(v_i^{t-1}) + c_1\rho_1(pbest_i^{t-1} - x_i^{t-1}) + c_2\rho_2(gbest^{t-1} - x_i^{t-1}) \qquad (2.22)$$

Katsayı vektörü (w), aldığı değere göre algoritmadaki parçacıkların arama yapma karakteristiklerine doğrudan etki etmektedir. $w < 0.8$ olduğu durumda algoritma hızın etkisini azaltarak lokal arama kabiliyetini; $w > 1.2$ olduğu durumda ise algoritma hızın etkisini artırarak global arama kabiliyetini ön plana çıkarmaktadır. Ayrıca ilk durumda parçacıkların global optimum noktasına

yakınsaması ikinci duruma nazaran daha hızlı gerçekleşmekte; ancak daha önceden de belirtildiği gibi ulaşılmak istenen optimum çözüm kümesi başlangıç noktalarının civarlarında yer alması gerekmektedir. Shi ve Eberhart, bu durumu kanıtlamak amacıyla farklı atalet ağırlığı değerleri ile Schaffer (Çizelge 3.9'daki 25 no.'lu fonksiyon) fonksiyonunu optimize etmiştir. Optimizasyon aşamasında algoritmanın çalışma sayısı 30, parçacık sayısı 20 ve maksimum iterasyon sayısı 4000 olarak belirlenmiştir. Çizelge 2.1'de PSO'nun farklı atalet ağırlığı değerlerinde Schaffer fonksiyonunu optimize etme süresini göstermektedir. Çizelge 2.1'in en sağındaki sütunu 1.4 değerinden 0 değerine lineer azalan atalet ağırlığı değeri ile PSO'nun optimizasyon süresi açısından performansını göstermektedir. Çizelge 2.1'de yer alan boş hücreler PSO'nun Schaffer fonksiyonunu çalışma esnasında 4000 iterasyon içerisinde optimize edemediğini temsil etmektedir. PSO'nun farklı atalet ağırlığı değerlerinde toplam başarısız optimizasyon süreci sayısı Çizelge 2.2'de gösterilmektedir.

Aşağıdaki çizelgeler incelendiğinde, atalet ağırlığı değeri 1.05 olduğunda algoritmanın tüm çalışma süreleri içerisinde global optimum noktayı bulduğu görülmektedir. Bununla beraber atalet ağırlığı küçük bir değer aldığında ($w < 0.9$) ortalama iterasyon sayısında ciddi bir azalış olduğu görülmektedir. Ancak Çizelge 2.2'den de anlaşılacağı üzere başarısız optimizasyon işlemi sayısı artmaktadır. Bu iki durum da yukarıda anlatılan bilgiyi desteklemektedir. Ayrıca bu çizelgelerden, sabit atalet ağırlığı yerine lineer azalan atalet ağırlığı değerinin belirlenmesinin daha uygun olduğu sonucu çıkarılabilir. Çünkü, bu değişken ağırlık değerlerinde otuz çalışma süresinin hepsinde global optimum nokta bulunmuştur ve ortalama iterasyon sayısının diğerlerine oranla daha az olduğu görülmüştür.

Çizelge 2.1. Farklı atalet ağırlığı değerleri ile Schaffer fonksiyonunun global optimum noktasının bulunması için geçen iterasyon süreleri (Shi ve Eberhart, 1998)

Çalışma No	Atalet ağırlıkları										
	1.4	1.2	1.1	1.05	1	0.95	0.9	0.85	0.8	0	1.4-0
1	684	819	989	994	864	680	633	1639	138	115	433
2		2521	2571	1117	885	2278	985	281			1382
3	1534	1632	2645	1636	2086	1264	3131	2480	158		982
4		480		2415	2250	908	252	1131	122		1482
5	577	3222	174	1564	1638	540	342	1562	120		898
6	854	984	1808	2040	984	799	1253	414			1384
7	2733	3371	1578	1982	1143	659	1076	116	262	96	1354
8	1498	3886	1351	826	796		878	353			1558
9	965	2085	272	1843	654	1158	1806	260			1268
10	3423	847	149	2097	748	2079	344	1587	147		1341
11		1660	2024	3153	341	1597	480	2416	147	158	1435
12		1361	960	481		1056	1298	206	174		534
13		610	1565	3724	1678	3283	1122	1733	163		621
14	3599	372	1260	833	1168	917	756	2644			524
15			1158	2767	1449	423	777	279	251		1462
16	3221	813	587	2954	1002	1746	474	1474		84	1508
17		1740	1477	2323	3789	651	1262	44		402	1362
18	718		1544	3065	2278	1458	1615	162		151	1510
19	1174	735	377	2179	342	712	933	337	3274		956
20		925	1744	673	1234	3458		700	1965	95	1571
21	2766	319	463	1672	274	2425	787	203			680

Çizelge 2.1. Farklı atalet ağırlığı değerleri ile Schaffer fonksiyonunun global optimum noktasının bulunması için geçen iterasyon süreleri (Shi ve Eberhart, 1998) (Devam)

22		621	2071	2934	2153	2165	3383	555	192	283	1431
23		1071	1074	2243	316	3639	948	1030		150	1361
24	3313	2038	1535	1994	1264	619	1660		319	102	1125
25	266	2137	1437	2336	2055	468	3159		167		1516
26	837	952	563	1455	628	2038	431	312		139	1450
27	745	962	1596	262	1878	508		697			834
28		3245	657	786	1159	919	1969				479
29	3354	2702	1673	3800	2772	1955	470	640			1567
30	1753	1476	3768	1206	691	328	2681		170	340	1591
Ortalama	1790.21	1556.64	1347.24	1911.8	1328.24	1404.48	1246.60	894.423	485.56	176.25	1186.6

Çizelge 2.2. Farklı atalet ağırlığı değerlerinin otuz çalışma sayısı içinde başarısız olduğu çalışma sayıları

Ağırlıklar	1.4	1.2	1.1	1.05	1	0.95	0.9	0.85	0.8	0
No	11	2	1	0	1	1	2	4	14	18

2.2.3. Geliştirilmiş yarasa algoritmasının adımları

Aşağıda Geliştirilmiş Yarasa Algoritmasının sayısal fonksiyonlar üzerinde sözde kod gösterimi aşağıda verilmiştir. Ayrıca geliştirilen algoritmanın akış diyagramı Şekil 2.10'da verilmiştir.

1. Başlangıç popülasyonunu, problem boyutunu belirle (N, d).
2. Yarasa Algoritmasına ait maksimum-minimum frekans, sinyal yayılım oranı ile ses şiddeti parametrelerini belirle (f_{max}, f_{min}, r, A).
3. Yabani Ot Optimizasyonuna ait maksimum-minimum üretilecek tohum sayısı, başlangıç-bitiş standart sapmaları, nonlineerlik katsayısı parametrelerini belirle $(s_{max}, s_{min}, \sigma_{initial}, \sigma_{final}, n)$.
4. Popülasyonun uzaya dağılımının iyileştirmesine katkı sağlayan ζ_{init} başlangıç parametresini belirle.
5. Eşitlik 2.1 ile tüm yarasa popülasyonunu (x) arama uzayına dağıt, başlangıç hızlarını $v = 0$ olarak belirle.
6. Tüm yarasanın bulduğu av/yiyecek kaynağının kalitesini, uygunluk değerini, $(fitness^0)$ hesapla. En iyi uygunluk değerini $(fitness_{best})$ hafızada tut.
7. **repeat**
8. Belirlenen atalet ağırlığı mekanizmasına göre bölüm 3.1.4'teki eşitliklerden birini kullanarak atalet ağırlığı değerini (w) güncelle.
9. ζ_{init} değerine göre eşitlik 2.12 ile ζ_1 parametresini güncelle.
10. **for** $i = 1$ to N **do**
11. Popülasyondaki en iyi yarasalar içerisinden birini (x_k) rastgele seç $(x_k \neq x_*)$.
12. Eşitlik 2.2, 2.10 ve 2.4 ile i. yarasayı arama uzayında başka bir noktaya yerleştir.
13. **if** $rand > r_i$ **then**
14. Popülasyon içerisinden uygunluk değerine bağlı olarak en iyi yarasalar içerisinden birini seç (x_m).
15. i. bireyi eşitlik 2.5 ile arama uzayında seçilen yarasanın (x_m) civarında bir noktaya gönder.

16. **end if**

17. Yarasanın konumlandığı noktadaki av/yiyecek kaynağının kalitesini, uygunluk değerini, ($fitness_i^t$) hesapla.

18. **if** $rand < A_i$ **and** $fitness_i^t < fitness_i^{t-1}$ **then**

19. Yeni noktadaki av/yiyecek kaynaklarını kabul et.

20. Eşitlik 2.6-2.7 ile sinyal yayılım oranını artır, ses şiddetini azalt.

21. **end if**

22. **if** $fitness_i^t < fitness_{best}^t$ **then**

23. Bulunan kaynağı en iyi kaynak olarak güncelle.

24. **end if**

25. **end for**

26. Eşitlik 2.14'ile σ_{iter} değeri üret.

27. Uygunluk değerlerine göre yarasa popülasyonunun yarısını seç ve seçilen popülasyonu Yabani Ot Optimizasyonunun başlangıç popülasyonu olarak kullan.

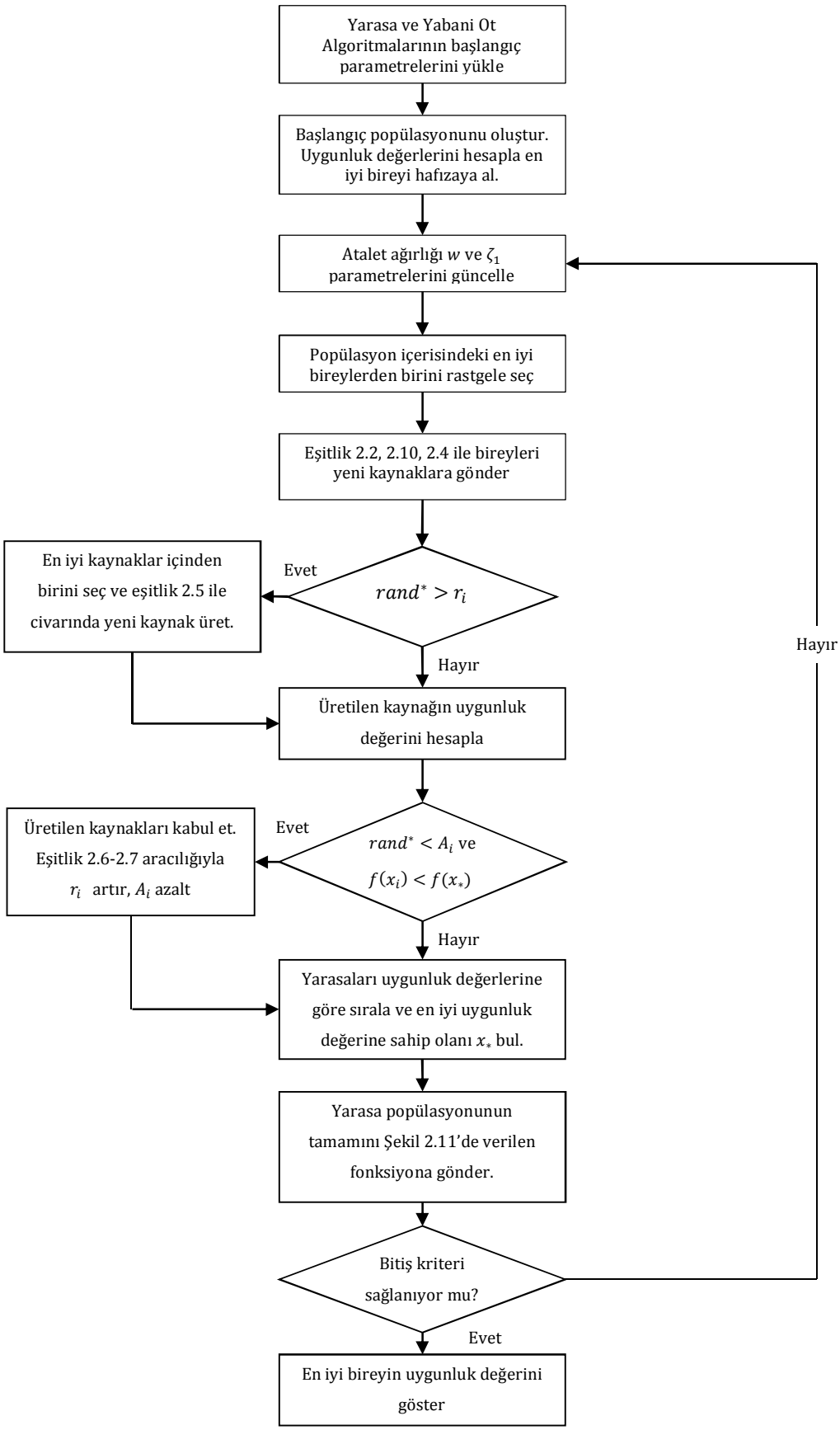
28. Eşitlik 2.13 ile popülasyondaki her bireyin üreteceği yeni tohumlarının sayısını (s) belirle.

29. Belirlenen s ve σ_{iter} değerlerine göre popülasyonu çoğalt ve uzaya dağıt. Üretilen yeni bireylerin ses şiddeti, sinyal yayılım oranları parametrelerine başlangıç değerlerini ata ve uygunluk değerlerini hesapla.

30. Yabani Ot Optimizasyonu ile üretilen yeni bireyleri Yarasa Algoritmasının popülasyonuna ekle.

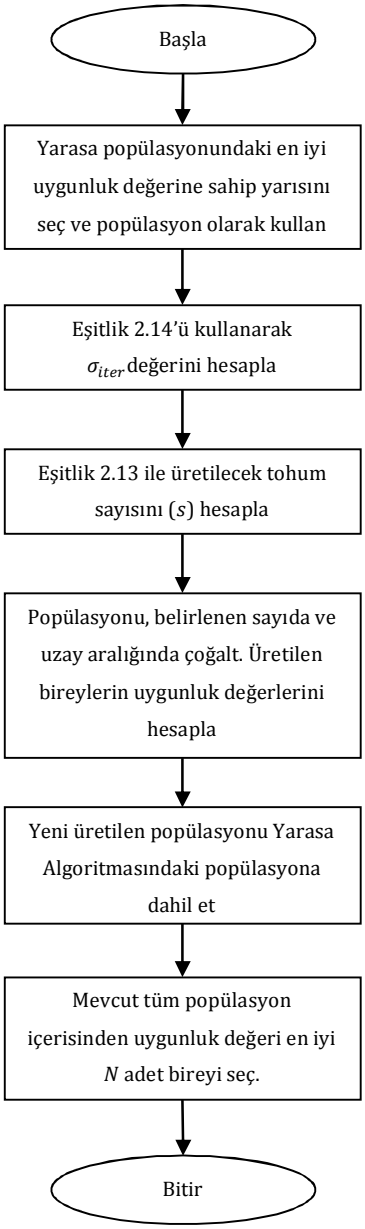
31. Birleştirilerek oluşan yeni popülasyonu sırala ve içerisinden en iyi N adet bireyi seç. Seçilen N adet bireyi bir sonraki iterasyonda kullan.

32. **until** Durma kriteri



* 0 ile 1 aralığında üretilmiş rastgele bir değer

Şekil 2.10. Geliştirilmiş Yarasa Algoritmasının akış şeması gösterimi



Şekil 2.11. Yabani Ot optimizasyonu fonksiyonunun akış şeması gösterimi

3. ARAŞTIRMA BULGULARI

GYA'da yapılan geliştirmelerin algoritmaya katkısı ve etkinliklerinin ayrı ayrı belirlenmesi amacıyla her geliştirme aşamasının ardından GYA, bazı standart test fonksiyonları üzerinde test edilmiştir. Ayrıca geliştirmeler ikişerli şekilde farklı kombinasyonlarda da test edilerek sonuçları gösterilmiştir.

Geliştirilen Yarasa Algoritmasının performansını ölçmek amacıyla her iki algoritma (YA, GYA) farklı problem boyutlarında 50 adet unimodal ve multimodal sayısal benchmark fonksiyonları (Çizelge 3.9) üzerinde test edilmiştir. Ayrıca her iki algoritma, *Congress of Evolutionary Computation 2005* konferansında tanıtılan ve (CEC05) olarak bilinen, karmaşık sayısal benchmark fonksiyonları üzerinde de sınanmıştır (Suganthan vd., 2005). GYA'nın standart ve karmaşık fonksiyon kümeleri üzerinde standart versiyona olan üstünlüğünün daha net anlaşılabilmesi için her iki fonksiyon kümesine uygulanan algoritmaların sonuçlarına *t-test* değerlendirme formu uygulanmıştır. *T-test*, iki örneklem üzerinden alınan sonuçlar arasında bir fark olup olmadığının ve sonuçların deneysel hataya ya da tesadüfe bağlı oluşup oluşmadığının anlaşılmasına yardımcı olmaktadır. *T-test* işleminin sonucu (t) pozitif ya da negatif olabilir. t değerinin pozitif olması GYA'nın YA'ya oranla daha etkili olduğunu; negatif olması YA'nın GYA'ya göre daha etkili olduğunu belirtmektedir. Bu çalışmada güven düzeyi %95 tutulmuştur bu durumda da $t_{0.05}=1.96$ olmaktadır. Diğer bir deyişle t değeri 1.96'dan büyükse GYA'nın YA'ya; -1.96 dan daha küçükse YA'nın GYA'ya göre daha etkili olduğu anlaşılmaktadır. t değerinin bu iki değer arasında kalması halinde ($-1.96<t<1.96$) her iki yöntemin birbirine üstünlük kuramadığı anlaşılmaktadır. t değeri eşitlik 3.1'e göre hesaplanmaktadır.

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{SD_1^2}{n_1 - 1} + \frac{SD_2^2}{n_2 - 1}}} \tag{3.1}$$

Burada $\overline{X_1}$, SD_1 ve n_1 sırasıyla YA'nın fonksiyonlar üzerinde aldığı ortalama değer, standart sapma ve örneklem sayısını belirtirken; $\overline{X_2}$, SD_2 ve n_2 sırasıyla GYA'nın fonksiyonlar üzerinde aldığı ortalama değer, standart sapma ve örneklem sayısını (çalışma sayısı) belirtmektedir.

Geliştirilen yöntemin kısıtlandırılmış gerçek hayat problemleri üzerindeki verimliliğini teyit etmek amacıyla yöntem, Yarasa Algoritması ile karşılaştırmalı olarak 3 adet kısıtlı gerçek hayat problemleri (Kaynaklı Kiriş Tasarımı, Gerilim Sıkıştırma Yay Tasarımı, Basınçlı Kap Tasarımı) üzerinde test edilmiştir. Ayrıca geliştirilen yöntemin bu problemler üzerindeki optimizasyon performansı literatürdeki (Gandomi vd., 2013) çalışmalar ile kıyaslanmıştır.

3.1. Algoritmadaki parametrelerin eğitimi

Yarasa Algoritmasının ve Geliştirilmiş Yarasa Algoritmasının test fonksiyonları üzerinde en iyi şekilde performans sergilemeleri amacıyla her iki algoritma içerisinde bulunan parametreler Çizelge 3.1'de verilen benchmark fonksiyonları üzerinde optimize edilerek eğitilmiştir. Eğitim aşamasında algoritmanın popülasyon sayısı, problem boyutu, fonksiyon hesaplama sayısı ve çalışma zamanı sayısı sırasıyla, 50, 10, $2x10^3$ ve 20 olarak belirlenmiştir. Her bir çalışma zamanının ardından elde edilen sonuçların ortalama değeri göz önünde bulundurulmuştur.

Algoritmada kullanılan her parametre aldığı değere göre diğer parametreleri belli bir miktarda etkilemektedir. Diğer bir deyişle, bir parametre eğitilirken aldığı optimum değer diğer parametrelerin aldığı değere bağlı olmaktadır. Bu tez çalışmasında öncelikle ses şiddeti ve sinyal yayılım oranı faktörleri daha sonra sırasıyla; maksimum frekans, atalet ağırlığı faktörü, ζ_1 ve ζ_2 katsayı faktörü eğitilmiş ve bu faktörlerin optimum değerleri elde edilmiştir.

Çizelge 3.1. Parametre eğitiminde kullanılan unimodal ve multimodal fonksiyonlar

No	**Fonksiyon**	**C**	**Denklem**
1	Sphere	U	$f(x) = \sum_{i=1}^{n} x_i^2$
2	Rosenbrock	U	$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$
3	Ackley	M	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$
4	Griewangk	M	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
5	Rastrigin	M	$f(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$
6	Zakharov	U	$f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$
7	Step	U	$f(x) = \sum_{i=1}^{n}([x_i + 0.5])^2$
8	Dixon-price	U	$f(x) = (x_i - 1)^2 + \sum_{i=1}^{n} i(2x_i^2 - x_{i-1})^2$
9	Easom	M	$f(x) = -(-1)^n(\prod_{i=1}^{n}\cos^2(x_i))\exp[-\sum_{i=1}^{n}(x_i - \pi)^2]$
10	Michalewicz	M	$f(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}$

3.1.1. Ses şiddeti parametresinin eğitimi

Algoritmadaki ses şiddeti (A), algoritmanın lokal arama performansını doğrudan etkilerken çözümlerin kabul edilmesinde de aktif rol oynar. Bu parametre, 0 ve 1 aralığında unimodal ve multimodal fonksiyonlar ile test edilmiş, sonuçlar normalize edilmiş şekilde Çizelge 3.2'de verilmiştir. Görüldüğü üzere, bazı istisnalar dışında, algoritma, ses şiddeti faktörü değeri arttıkça daha iyi sonuçlar üretmektedir. A faktörü 0.9 ve 1 olduğunda algoritma bu on fonksiyon üzerinde birbirine yakın en iyi ortalama değerleri üretmiştir. Dolayısıyla bu parametre, her iki algoritmanın (YA, GYA) test edilmesi aşamasında, bu iki değerin (0.9 – 1) ortalaması 0.95 olarak belirlenmiştir.

Çizelge 3.2. Ses şiddeti, ***A*** parametresinin sayısal fonksiyonlar üzerinde eğitilmesi

	A										
No	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	1.28	1.18	1.16	1.03	1.13	1.45	1.01	**1**	1.14	1.04	1.13

Çizelge 3.2. Ses şiddeti, $\boldsymbol{A}$ parametresinin sayısal fonksiyonlar üzerinde eğitilmesi (Devam)

2	5.32	3.98	4.55	2.74	4.43	3.83	3.62	3.35	2.23	1.26	**1**
3	1.22	1.21	1.22	1.23	1.19	1.17	1.07	1.10	**1**	1.06	**1**
4	1.24	1.18	**1**	1.12	1.18	1.18	1.42	1.28	1.17	1.09	1.10
5	1.83	1.65	1.35	1.37	1.41	1.19	1.18	1.21	1.02	**1**	1.05
6	3.47	4.41	4.97	4.87	4.13	2.96	3.53	3.47	2.18	1.40	**1**
7	1.32	1.35	1.42	1.17	1.15	**1**	1.14	1.36	1.37	1.07	1.17
8	18.3	16.4	17.2	19.0	6.00	6.92	11.4	4.18	4.33	1.34	**1**
9	1.99	1.99	1.98	1.95	1.97	1.98	1.70	1.59	1.47	1.64	**1**
10	1.05	1.12	1.07	1.08	1.12	1.06	1.07	1.05	1.02	1.01	**1**
Ort*	3.71	3.44	3.60	3.55	2.37	2.27	2.72	1.95	1.69	**1.19**	**1.04**

* ortalama

3.1.2. Sinyal yayılım oranı parametresinin eğitimi

Sinyal yayılım oranı (r), algoritmanın exploration ya da exploitation kabiliyetini aktif kılmada önemli rol oynamaktadır. Bu değer optimizasyon süreci boyunca azaldıkça algoritma, lokal arama yoğunluğunu azaltmakta global arama yoğunluğunu artırmaktadır. Sinyal yayılım oranı parametresi 0-1 aralığında test edilerek eğitilmiştir. Sonuçlar normalize edilmiş şekilde Çizelge 3.3'te gösterilmiştir. Sonuçlara göre algoritmanın minimum değerlere ulaşma olasılığı sinyal yayılım oranı parametresi ile beraber, bazı istisnalar dışında, artmaktadır. Bu faktör 0.8 ile 0.9 değeri aldığında algoritmanın bu fonksiyonlar üzerindeki ortalama optimizasyon performansı birbirine çok yakın ve diğerlerine göre daha iyi çıkmaktadır. Bu nedenle bu değer her iki algoritmanın (YA, GYA) test edilmesi aşamasında bu iki değerin (0.8 – 0.9) ortalaması 0.85 olarak belirlenmiştir.

Çizelge 3.3. Sinyal yayılım oranı, $\boldsymbol{r}$ parametresinin sayısal fonksiyonlar üzerinde eğitilmesi

	r										
No	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	17.79	7.73	4.27	3.44	2.30	2.3	1.6	1.6	1.3	**1**	1.2
2	360.7	56.79	29.6	11.3	10.1	7.5	4.1	1.7	**1**	1.3	2.7
3	2.13	1.79	1.54	1.39	1.35	1.2	1.1	1.1	**1**	**1**	1.0

Çizelge 3.3. Sinyal yayılım oranı, $\boldsymbol{r}$ parametresinin sayısal fonksiyonlar üzerinde eğitilmesi (Devam)

4	11.43	5.57	4.51	3.08	2.06	1.8	1.7	1.2	1.4	**1**	1.1
5	2.95	1.90	1.74	1.44	1.46	1.1	1.2	1.1	**1**	1.09	1.5
6	1.19	1.62	1.29	1.06	1.08	1.2	**1**	1.0	1.7	2.04	2.2
7	14.29	6.72	3.89	3.26	2.69	1.5	1.6	1.2	1.0	1.17	**1**
8	332.3	147.7	16.5	6.10	2.06	7.2	1.2	1.0	1.5	**1**	4.4
9	1.97	1.79	1.32	**1**	1.66	1.1	1.0	1.4	1.1	1.51	1.9
10	**1**	1.01	**1**	1.02	1.01	1.0	1.0	1.0	1.0	1.06	1.0
Ort	74.58	23.26	6.57	3.30	2.58	2.6	1.5	1.2	**1.2**	**1.2**	1.8

* ortalama

3.1.3. Maksimum frekans parametresinin eğitimi

Frekans, algoritmadaki adım aralığı faktörünü temsil eden bir parametredir. Bu faktör ne kadar büyük olursa algoritmanın istenen optimum bölgeleri kaçırma ihtimali o kadar yüksek; diğer taraftan ne kadar küçük olursa algoritmanın istenen bölgelere belirlenen süre içerisinde ulaşma ihtimali oldukça düşük olacaktır. Optimum frekans değeri probleme göre değişmektedir. Örneğin uzay aralığı geniş olan bir problemin frekans değerinin, uzay aralığı dar olana göre nispeten büyük olması beklenmektedir. Bu tezde minimum frekans değeri olarak 0 seçilmiştir. Maksimum frekans değeri ise 1 ile 10 aralığında seçilen benchmark fonksiyonları üzerinde eğitilerek belirlenmiştir. Çizelge 3.4'ten de anlaşılacağı üzere maksimum frekans değeri arttıkça algoritmanın fonksiyonlar üzerinde optimizasyon performansı düşmektedir ve en iyi performansı $f_{max} = 1$ iken vermektedir. Dolayısıyla frekans faktörünün maksimum değeri 1 olarak belirlenmiştir.

Çizelge 3.4. Maksimum frekans ($\boldsymbol{f_{max}}$) parametresinin sayısal fonksiyonlar üzerinde eğitilmesi

	f_{max}									
No	1	2	3	4	5	6	7	8	9	10
1	**1**	3.24	5.33	6.16	7.02	9.54	11.64	10.80	10.73	13.56
2	**1**	1.76	2.03	5.56	17.92	89.75	63.09	77.42	107.1	207.9
3	**1**	2.98	5.34	6.05	7.54	8.14	8.19	8.40	8.11	7.60
4	**1**	1.64	2.44	2.31	3.91	3.95	4.42	4.92	5.74	4.81

Çizelge 3.4. Maksimum frekans (f_{max}) parametresinin sayısal fonksiyonlar üzerinde eğitilmesi (Devam)

5	**1**	1.05	1.05	1.40	1.47	1.29	1.74	1.43	1.71	1.98
6	1.81	4.74	3.32	17.55	31.22	7.17	24.03	33.63	97.79	**1**
7	**1**	2.84	3.45	5.28	7.09	5.98	6.97	9.35	7.16	8.86
8	**1**	1.15	**1**	1.70	**1**	5.53	2.56	9.84	5.97	3.77
9	4.12	**1**	2.47	5.67	5.68	10.33	4.00	3.90	5.50	5.51
10	1.01	1.03	1.03	**1**	1.04	1.03	**1**	**1**	1.04	1.03
Ort*	**1.39**	2.14	2.74	5.26	8.38	14.27	12.76	16.06	25.09	25.60

* ortalama

3.1.4. Atalet ağırlığı parametresinin eğitimi

Atalet ağırlığı faktörü önceden de belirtildiği üzere iterasyon ilerledikçe algoritmanın adım aralığını kısarak global aramadan lokal aramaya geçiş yapmasını sağlayan bir faktördür. Bu atalet ağırlığı faktörünün güncellenmesini sağlayacak literatürde önerilmiş farklı birçok mekanizma bulunmaktadır. Bu tezde, literatürde önerilmiş (Nickabadi vd., 2011) bazı atalet ağırlığı güncelleme mekanizmaları seçilerek karşılaştırma yoluyla en uygun mekanizma belirlenmiş ve geliştirilen algoritmaya uygulanmıştır. Karşılaştırılan atalet ağırlığı güncelleme mekanizmaları aşağıda açıklanmıştır.

a. **Rastgele atalet ağırlığı:** Çoğu zaman algoritmanın lokal ya da global arama yapması gerektiği önceden kestirilemez. Bu nedenle rastgele seçilen bir atalet ağırlığı faktörü (w) bu problemi ortadan kaldırabilir (Nickabadi vd., 2011).

$$w = 0.5 + \frac{\eta}{2} \tag{3.2}$$

Yukarıdaki eşitlikte η, 0 ve 1 arasında rastgele değer alan bir vektörü temsil etmektedir. Atalet ağırlığı sadece η değerine bağlı olarak tüm iterasyon boyunca 0 ve 1 aralığında değer üretmektedir (Şekil 3.1(a)).

b. Lineer azalan atalet ağırlığı: Bu mekanizma ile atalet ağırlığı değeri belirlenen bir üst sınırdan (w_{max}) alt sınıra (w_{min}) iterasyon ilerledikçe azalmaktadır (Şekil 3.1(b)).

$$w^t = \frac{t_{max} - t}{t_{max}}(w_{max} - w_{min}) + w_{min} \tag{3.3}$$

Burada t o andaki iterasyon sayısı bilgisini verirken; t_{max}, maksimum iterasyon sayısını vermektedir.

c. Non-lineer azalan atalet ağırlığı: Lineer azalan atalet ağırlığına benzer bir şekilde atalet değerini üst sınırdan alt sınıra doğru nonlineer olarak azaltmaktadır (Şekil 3.1(c)). Ancak bir modülasyon indeksi (n) ile bu faktörün azalış hızını yavaşlatıp hızlandırabilmektedir. Eğitim aşamasında $n = 3$ olarak belirlenmiştir.

$$w^t = \left\{\frac{(t_{max} - t)^n}{(t_{max})^n}\right\}(w_{max} - w_{min}) + w_{min} \tag{3.4}$$

d. Kaotik lineer azalan atalet ağırlığı: Feng vd. (2007), çalışmasında lineer azalan atalet ağırlığı mekanizmasına kaotik terim (z) ekleyerek kaotik azalan lineer atalet ağırlığı mekanizmasını kullanmıştır. Bu yöntem lineer azalan ve rastgele atalet ağırlığı mekanizmalarının hibritlenmesi gibi çalışmaktadır (Şekil 3.1(d)).

$$w^t = \frac{t_{max} - t}{t_{max}}(w_{max} - w_{min}) + (w_{min})z \tag{3.5}$$

$$z = 4z(1 - z) \tag{3.6}$$

Burada z değeri başlangıçta [0,1] aralığında rastgele üretilir, ilerleyen iterasyonlarda eşitlik 3.6'ya göre hesaplanır.

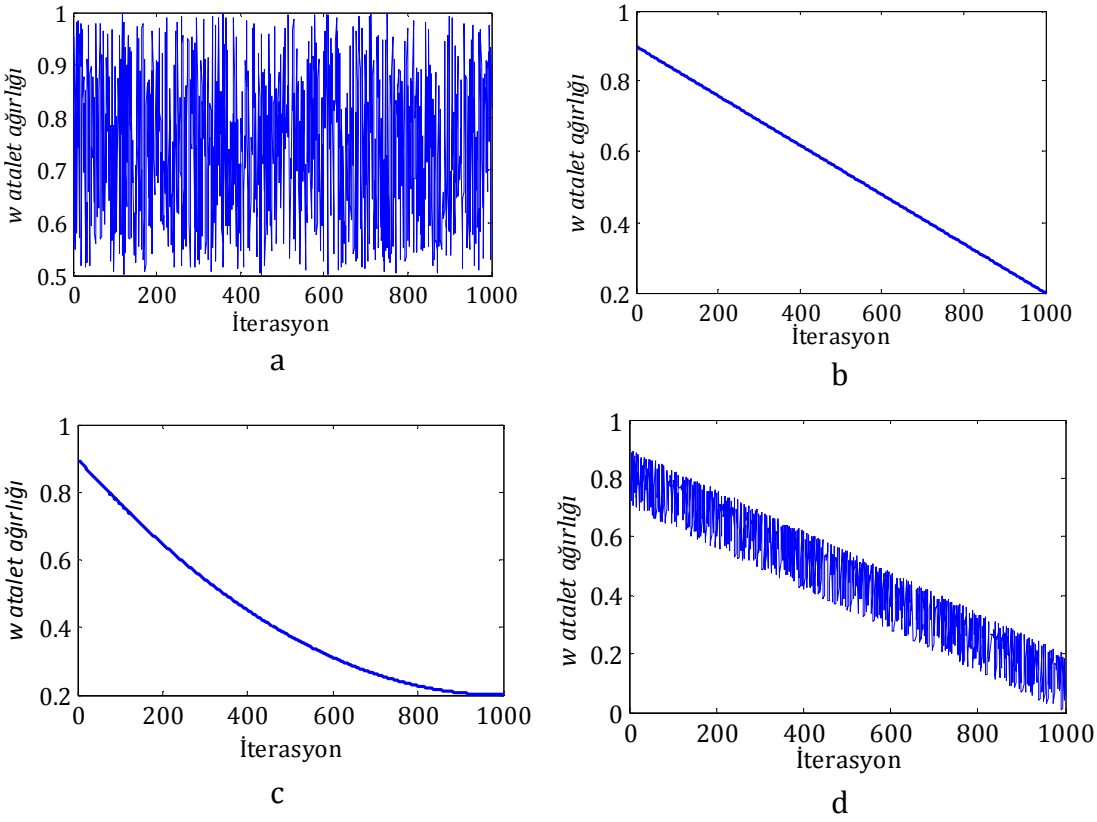
e. Sugeno azalan atalet ağırlığı: Sugeno mekanizması ise non-lineer azalan atalet ağırlığı mekanizmasına benzer şekilde atalet ağırlığı

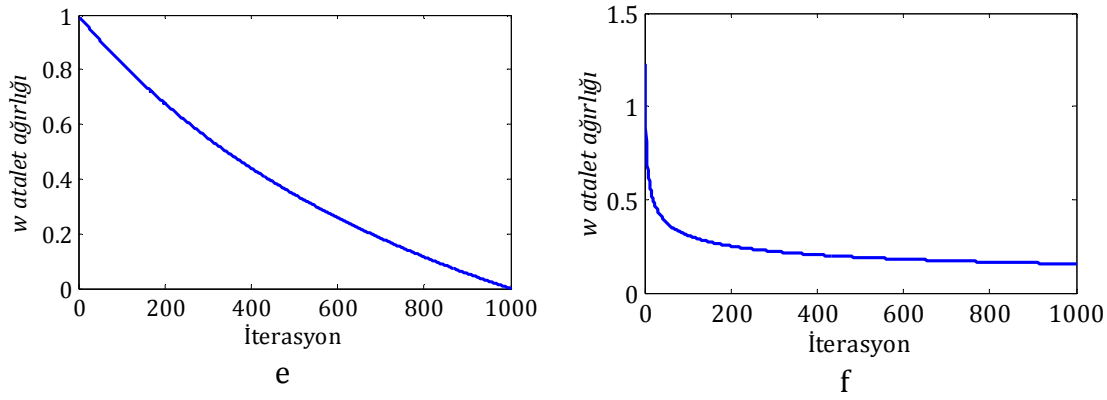
faktörünü azaltmaktadır (Şekil 3.1(e)). Bu mekanizma maksimum iterasyon sayısı ve o anki iterasyon değerine göre atalet ağırlığını hesaplar.

$$w = \frac{1 - \frac{t}{t_{max}}}{1 - s\frac{t}{t_{max}}} \qquad s > -1 \tag{3.7}$$

f. İterasyona bağlı azalan atalet ağırlığı: Fan ve Chiu (2007), PSO algoritmasında bu mekanizmayı önermiştir. Bu mekanizmada atalet ağırlığı faktörü yaklaşık olarak 1.23 değerinden başlayarak çok kısa süre içerisinde azalmaktadır ve iterasyon ilerledikçe atalet ağırlığı faktörleri arasındaki fark azalmaktadır (Şekil 3.1(f)).

$$w = \left(\frac{2}{t}\right)^{0.3} \tag{3.8}$$





Şekil 3.1. Atalet ağırlığı faktörünün farklı mekanizmalarda iterasyona bağlı değişimi

Çizelge 3.5'ten de anlaşılacağı üzere lineer (b) ve kaotik lineer (d) azalan atalet ağırlığı güncelleme mekanizmaları ile Yarasa Algoritması en çok (dört) minimum uygunluk değerine ulaşmıştır. Ancak Yarasa Algoritmasının bu fonksiyonlar üzerindeki ortalamaları göz önüne alındığında, bu algoritmanın en iyi performansı non-lineer (c) azalan atalet ağırlığı mekanizması ile elde ettiği görülmektedir. Dolayısıyla bu çalışmada nonlineer azalan atalet ağırlığı mekanizması, *w* faktörünün güncellenmesi için seçilmiştir.

Çizelge 3.5. Atalet ağırlığı faktörü, ***w*** parametresinin sayısal fonksiyonlar üzerinde eğitilmesi

No	a	b	c	d	e	f
1	1.34e-02	6.66e-23	1.14e-05	**1.14e-24**	3.63e-19	9.86e+07
2	5.87	1.06	0.82	**0.80**	5.91	5.12e+09
3	2.98	**0.36**	0.60	1.92	0.58	16.35
4	**2.58**	4.25	3.31	3.96	6.67	120.43
5	37.16	20.80	**20.05**	33.83	22.44	77.27
6	3.69e-03	**8.75e-18**	4.08e-02	3.57e-02	1.34e-01	0.03
7	40.30	41.30	**28.45**	42.40	75.50	9.60e+07
8	0.63	0.67	0.63	**0.60**	0.67	20.77
9	-0.80	**-1.00**	-0.95	**-1.00**	**-1.00**	-0.59
10	-7.68	**-7.73**	-7.50	-7.62	-7.71	-5.58
Ort*	8.11	5.97	4.55	7.49	10.30	5.32e+08

* ortalama

3.1.5. ζ_1 ve ζ_2 katsayı parametrelerinin eğitilmesi

Algoritmanın arama karakteristiğine etki eden bir diğer önemli faktör ise ζ_1 ve ζ_2 parametreleridir. Eşitlik 2.10'un ikinci terimi algoritmanın yakınsama hızına üçüncü terimi ise uzayı daha etkin bir şekilde keşfetmesine yardımcı olmaktadır. Dolayısıyla bu parametrenin uygun bir şekilde belirlenmesi algoritmanın arama uzayını keşfetme kabiliyetini ve yakınsama performansını artıracaktır.

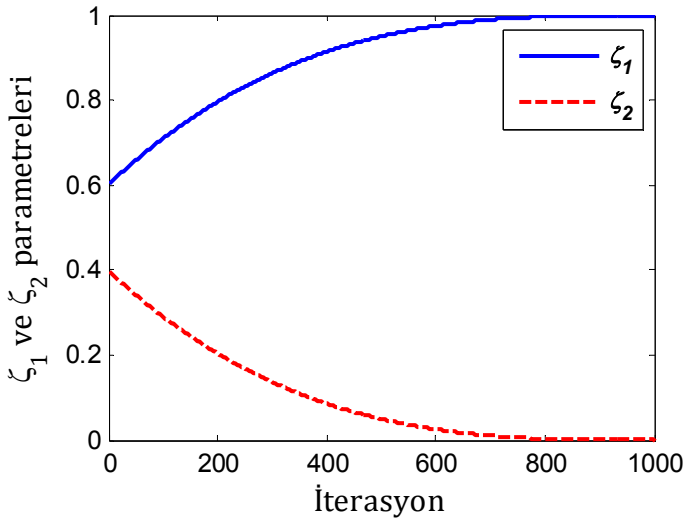
Çizelge 3.6'daki sonuçlara bakıldığında ζ_1 faktörü 0.5 ile 0.8 aralığında ve ζ_2 faktörü de 0.5 ile 0.2 aralığında iken algoritmanın ortalama optimizasyon performansının diğerlerine göre daha iyi sonuç verdiği görülebilmektedir. ζ_1 faktörü 0.6 ve ζ_2 faktörü 0.4 iken algoritma fonksiyonlar üzerinde en iyi ortalama optimizasyon performansı elde etmiştir. Bu nedenle ζ_1 faktörü 0.6 olarak ζ_2 faktörü ise 0.4 olarak belirlenmiştir.

Çizelge 3.6. **ζ_1** ve **ζ_2** parametrelerinin sayısal fonksiyonlar üzerinde eğitilmesi

ζ_1	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ζ_2	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
1	5.32	4.08	3.80	2.91	1.8	1.5	1.45	**1**	1.05	1.1	1.5
2	86.9	54.8	37.7	56.4	7.8	4.0	**1**	5.8	2.11	3.8	2.5
3	7.73	6.66	4.49	3.20	3.2	2.5	**1**	1.3	1.76	1.0	2.3
4	2.85	2.36	2.00	1.54	1.4	1.3	1.21	1.0	1.12	**1**	1.1
5	1.19	1.16	1.10	1.01	**1**	1.0	1.06	1.0	1.01	1.0	1.0
6	4.04	4.92	1.16	1.15	1.1	**1**	1.19	1.3	1.21	1.1	1.0
7	5.47	3.34	2.49	2.32	1.5	1.8	1.43	1.6	**1**	1.4	1.6
8	1.14	2.56	1.05	1.04	1.0	1.0	1.04	1.0	1.04	**1**	1.0
9	4.22	4.27	4.26	4.91	2.3	2.2	1.69	**1**	1.01	3.6	2.9
10	1.02	1.04	1.01	**1**	1.0	1.0	1.09	1.0	1.03	1.0	1.0
Ort	11.9	8.53	5.91	7.55	2.2	1.7	**1.2**	1.6	**1.2**	1.6	1.6

* ortalama

Optimum başlangıç değerleri verilen ζ_1 ve ζ_2 faktörlerinin iterasyona bağlı değişimi Şekil 3.2'de gösterilmiştir.



Şekil 3.2. $\boldsymbol{\zeta_1}$ ve $\boldsymbol{\zeta_2}$ parametresinin iterasyon ile değişimi

3.2. Geliştirmelerin algoritmaya olan katkısının analiz edilmesi

Yarasa Algoritmasının standart ve karmaşık sayısal benchmark test fonksiyonları ile gerçek hayat problemleri üzerinde standart versiyonuna oranla daha etkili sonuçlar üretebilmesi için algoritma üç farklı yapı ile geliştirilmiştir. Bu bölümde algoritmaya dâhil olan her bir geliştirmenin algoritmaya olan katkısı ayrı ayrı araştırılmıştır. Bu amaçla her bir geliştirme yapısı Çizelge 3.1'de belirtilen on adet benchmark test fonksiyonu üzerinde tek tek ve ikişerli şekilde otuz defa çalıştırılarak test edilmiştir. Çizelge 3.7'de algoritmaya eklenen geliştirme yapıları başlıklar halinde belirtilmiştir. Testte Çizelge 3.10'da belirtilen parametre değerleri kullanılmıştır. Çizelge 3.8'de farklı geliştirme yapıları ile uygulanan Yarasa Algoritmasının bazı benchmark fonksiyonlar üzerindeki ortalama maliyet (uygunluk) değeri sonuçları gösterilmektedir. YA ve GYA ile birlikte tüm geliştirme yapılarının fonksiyonlar üzerinde ayrı ayrı yakınsama grafikleri Şekil 3.3'te verilmiştir.

Çizelge 3.7. Yarasa Algoritmasına uygulanan geliştirmeler

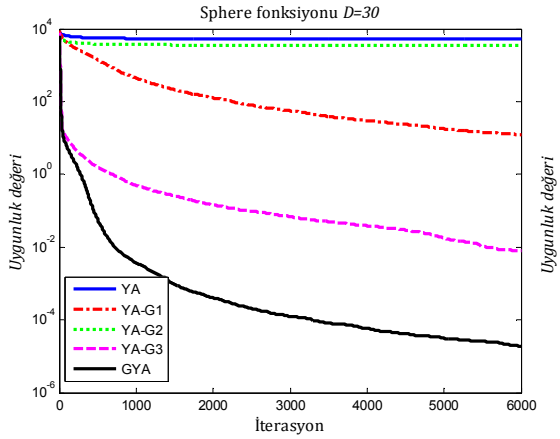
No	Açıklama
G1	Atalet ağırlığı modifikasyon
G2	Global arama modifikasyonu
G3	Yabani Ot Optimizasyonu hibritlemesi

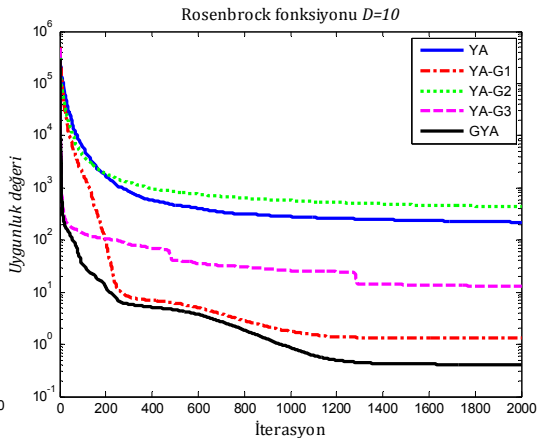
Çizelge 3.8. Yarasa Algoritmasına etki eden geliştirmelerin algoritmaya olan katkısının benchmark fonksiyonları üzerinde performansı

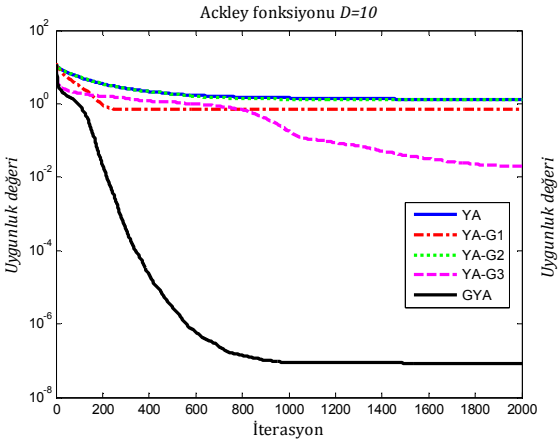
No	D	YA	G1	G2	G3	G1-G2	G1-G3	G2-G3	G1-G2-G3
1	10	4.68e+02	4.69e-27	3.12e+02	2.02e-04	7.19e-28	3.87e-06	2.08e-04	9.95e-24
	30	5.22e+03	1.23e+01	3.54e+03	8.07e-03	2.54e-00	2.17e-05	8.98e-03	1.85e-05
	50	9.55e+03	4.59e+02	7.79e+03	1.17e-02	1.65e+02	2.07e-05	1.01e-02	2.18e-05
2	10	2.20e+02	1.34e-00	4.38e+02	1.30e+01	4.87e-00	9.95e-01	1.03e+01	4.13e-01
	30	1.84e+05	3.01e+01	5.69e+04	5.16e+01	2.99e+01	2.99e+01	6.02e+01	2.80e+01
	50	8.96e+05	2.51e+02	4.57e+05	6.69e+01	8.65e+01	5.16e+01	8.03e+01	5.54e+01
3	10	1.31e-00	6.92e-01	1.26e-00	2.01e-02	3.33e-01	9.44e-02	1.99e-02	8.56e-08
	30	9.44e-00	5.68e-00	8.22e-00	1.80e-01	4.10e-00	8.09e-01	2.04e-01	2.25e-01
	50	1.08e+01	8.62e-00	1.00e+01	1.96e-00	7.16e-00	2.01e-00	2.01e-00	1.76e-00
4	10	1.10e+01	4.15e-00	1.03e+01	9.87e-01	2.73e-00	1.70e-00	6.40e-01	1.14e-00
	30	6.08e+01	6.86e-00	5.33e+01	1.41e-02	4.00e-00	8.62e-03	5.41e-03	6.90e-03
	50	1.11e+02	2.58e-01	1.01e+02	6.59e-03	1.32e+01	3.61e-03	3.87e-03	5.26e-03
5	10	3.08e+01	2.29e+01	2.96e+01	1.73e+01	1.69e+01	2.21e+01	9.83e-00	1.19e+01
	30	1.18e+02	1.21e+02	1.08e+02	9.71e+01	9.16e+01	1.19e+02	5.16e+01	5.73e+01
	50	2.47e+02	2.43e+02	2.05e+02	1.91e+02	1.92e+02	2.35e+02	1.16e+02	1.20e+02
6	10	3.00e-02	4.14e-06	8.33e-02	4.99e-04	5.88e-25	1.07e-05	2.79e-04	1.41e-11
	30	2.23e+02	5.69e-05	2.07e+02	6.91e-03	3.86e-05	4.85e-05	7.41e-03	4.60e-05
	50	8.01e+02	3.11e-00	7.12e+02	2.35e-02	1.35e-00	3.66e-04	2.41e-02	1.46e-04

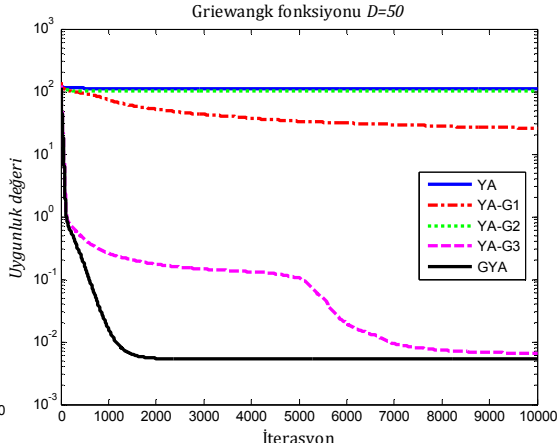
Çizelge 3.8. Yarasa Algoritmasına etki eden geliştirmelerin algoritmaya olan katkısının benchmark fonksiyonları üzerinde performansı (Devam)

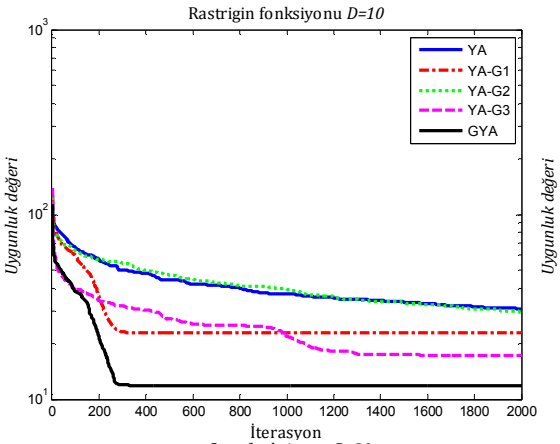
7	10	6.05e+02	3.40e+01	4.19e+02	0.00e-00	9.33e-00	0.00e-00	0.00e-00	0.00e-00
	30	5.09e+03	1.91e+03	4.38e+03	6.33e-01	9.03e+02	1.63e-00	4.00e-01	8.33e-01
	50	1.18e+04	5.30e+03	9.02e+03	7.70e-00	3.32e+03	9.30e-00	7.00e-00	7.67e-00
8	10	7.34e-01	6.44e-01	7.58e-01	7.04e-01	6.44e-01	5.78e-01	6.95e-01	6.67e-01
	30	1.02e+01	6.71e-01	9.37e-00	6.41e-00	6.68e-01	6.68e-01	6.70e-00	6.68e-01
	50	4.73e+02	6.78e-01	2.45e+02	3.17e+02	6.93e-01	8.69e-01	1.98e+02	8.28e-01
9	10	-8.14e-01	-1.00e-00	-9.08e-01	-1.00e-00	-1.00e-00	-1.00e-00	-9.99e-01	-1.00e-00
	30	-3.24e-07	-6.74e-02	-7.29e-12	-6.66e-02	-1.68e-01	-3.34e-01	-2.95e-01	-3.01e-01
	50	-2.70e-40	-6.78e-07	-1.59e-39	-3.47e-12	-3.40e-07	-1.07e-04	-1.05e-04	-3.35e-02
10	2	-1.80e-00	-1.80e-00	-1.80e-00	-1.80e-00	-1.80e-00	-1.80e-00	-1.80e-00	-1.80e-00
	5	-3.71e-00	-4.30e-00	-3.82e-00	-4.34e-00	-4.47e-00	-4.16e-00	-4.51e-00	-4.38e-00
	10	-5.61e-00	-7.56e-00	-5.75e-00	-7.79e-00	-8.09e-00	-7.37e-00	-8.48e+00	-8.17e-00

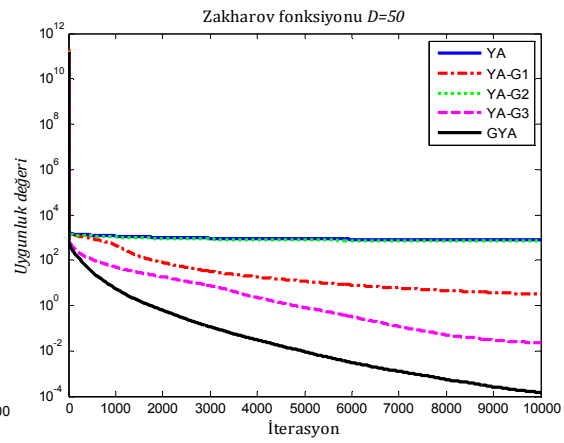

Sphere fonksiyonu D=30
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA

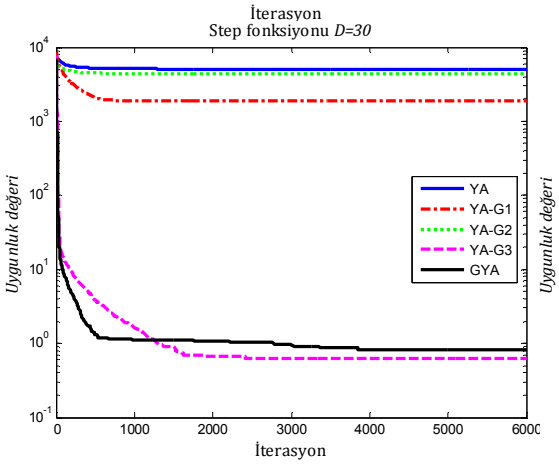

Rosenbrock fonksiyonu D=10
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA

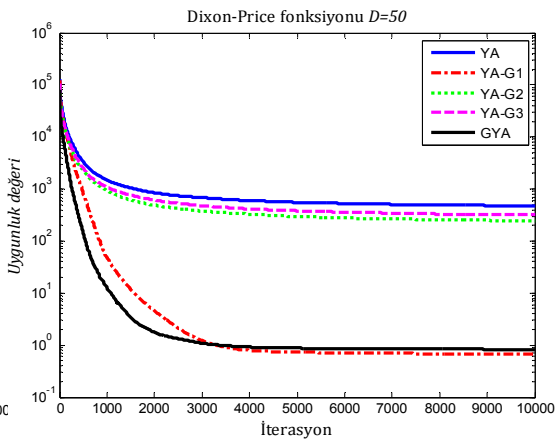

Ackley fonksiyonu D=10
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA

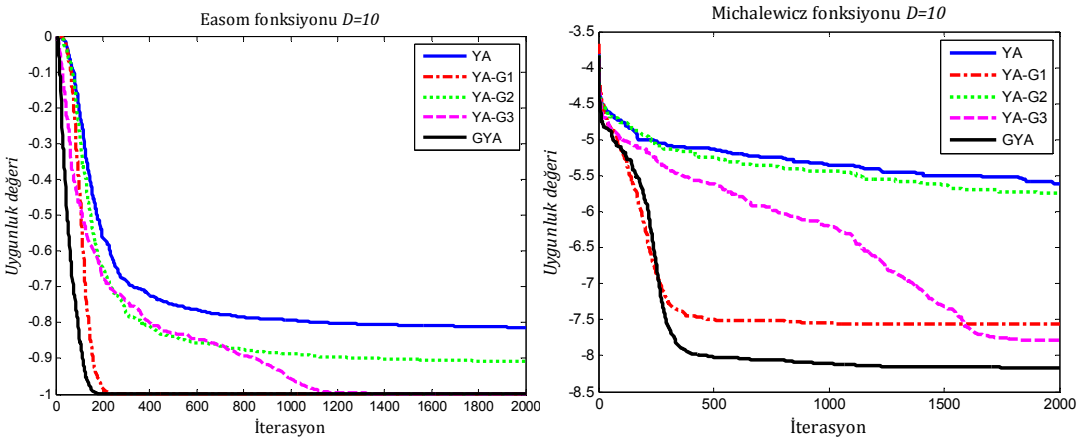

Griewangk fonksiyonu D=50
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA


Rastrigin fonksiyonu D=10
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA


Zakharov fonksiyonu D=50
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA


Step fonksiyonu D=30
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA


Dixon-Price fonksiyonu D=50
Uygunluk değeri
İterasyon
YA
YA-G1
YA-G2
YA-G3
GYA



Şekil 3.3. Standart Yarasa Algoritması ve geliştirme yapılarının fonksiyonlar üzerindeki yakınsama grafikleri

3.3. Yöntemin Standart Fonksiyon Kümesi Üzerinde Test Edilmesi

3.3.1. Standart benchmark test fonskiyonları

Standart ve Geliştirilmiş Yarasa Algoritması, sayısal benchmark test fonksiyonları üzerinde test edilmiştir. Bu test aşamasında kullanılan 50 adet fonksiyon için literatürden (Karaboga ve Akay, 2009) faydalanılmıştır. Bu test seti içerisinde bulunan benchmark fonksiyonları farklı boyutlarda unimodal ve multimodal fonksiyonlardan oluşmaktadır. Unimodal fonksiyonlar, algoritmanın yakınsama performansının ölçülmesi için kullanılırken; multimodal fonksiyonlar, algoritmanın erken yakınsama probleminin ya da lokale takılma sorununun olup olmadığının öğrenilmesi için kullanılmaktadır.

Çizelge 3.9. Test için kullanılan unimodal ve multimodal benchmark fonksiyonları (Karaboga ve Akay, 2009)

No	Fonksiyon	D	C	Min.	Denklem
1	Stepint	5	U	0	$f(x) = 25 + \sum_{i=1}^{n} \lfloor x_i \rfloor$
2	Step	*	U	0	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
3	Sphere	*	U	0	$f(x) = \sum_{i=1}^{n} x_i^2$
4	SumSquares	*	U	0	$f(x) = \sum_{i=1}^{n} i x_i^2$
5	Quartic	*	U	0	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$
6	Beale	2	U	0	$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$

Çizelge 3.9. Test için kullanılan unimodal ve multimodal benchmark fonksiyonları (Karaboga ve Akay, 2009) (Devam)

7	Easom	2	U	-1	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$
8	Matyas	2	U	0	$f(x) = 0.26(x_1^2+x_2^2) - 0.48x_1x_2$
9	Colville	4	U	0	$f(x) = 100(x_1^2-x_2)^2+(x_1-1)^2+(x_3-1)^2+90(x_3^2-x_4)^2+10.1((x_2-1)^2+(x_4-1)^2)+19.8(x_2-1)(x_4-1)$
10	Trid6	6	U	-50	$f(x) = \sum_{i=1}^{n}(x_i-1)^2 - \sum_{i=1}^{n} x_i x_{i-1}$
11	Trid10	10	U	-210	$f(x) = \sum_{i=1}^{n}(x_i-1)^2 - \sum_{i=1}^{n} x_i x_{i-1}$
12	Zakharov	*	U	0	$f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$
13	Powell	24	U	0	$f(x) = \sum_{i=1}^{n/k}(x_{4i-3}+10x_{4i-2})^2 + 5(x_{4i-1}+x_{4i})^2 + (x_{4i-2}+x_{4i-1})^4 + 10(x_{4i-3}+10x_{4i})^4$
14	Schwefel2.22	*	U	0	$f(x) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$
15	Schwefel1.2	*	U	0	$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$
16	Rosenbrock	*	U	0	$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1}-x_i)^2 + (x_i-1)^2]$
17	Dixon-Price	*	U	0	$f(x) = (x_i-1)^2 + \sum_{i=1}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$
18	Fox-holes	2	M	0.998	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$
19	Branin	2	M	0.398	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos x_1 + 10$
20	Bohachevsky1	2	M	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
21	Booth	2	M	0	$f(x) = (x_1+2x_2-7)^2 + (2x_1+x_2-5)^2$
22	Rastrigin	*	M	0	$f(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$
23	Schwefel	*	M	-12569.5	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{x_i}\right)$
24	Michalewicz2	2	M	-1.8013	$f(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}$
25	Michalewicz5	5	M	-4.6877	
26	Michalewicz10	10	M	-9.6602	
27	Schaffer	2	M	0	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{\left(1+0.001(x_1^2+x_2^2)\right)^2}$
28	SHCB	2	M	-1.03163	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
29	Bohachevsky2	2	M	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
30	Bohachevsky3	2	M	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$

Çizelge 3.9. Test için kullanılan unimodal ve multimodal benchmark fonksiyonları (Karaboga ve Akay, 2009) (Devam)

31	Shubert	2	M	-186.73	$f(x) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right)\left(\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right)$
32	Goldstein-price	2	M	3	$f(x) = [1 + (x_1 + x_2 + 1)^2(9 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
33	Kowalik	4	M	0.00031	$f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)^2}{b_i^2 + b_i x_3 + x_4}\right)$
34	Shekel5	4	M	-10.15	$f(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$
35	Shekel7	4	M	-10.4	$f(x) = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$
36	Shekel10	4	M	-10.53	$f(x) = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$
37	Perm	4	M	0	$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k + \beta\right)\left(\left(\frac{x_i}{i}\right)^k - 1\right)\right]^2$
38	PowerSum	4	M	0	$f(x) = \sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n} x_i^k\right) - b_k\right]^2$
39	Hartman3	3	M	-3.86	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
40	Hartman6	6	M	-3.32	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
41	Griewank	*	M	0	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
42	Ackley	*	M	0	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$
43	Penalized1	30	M	0	$f(x) = \frac{\pi}{n}\{10\sin^2\pi y_1 + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$
44	Penalized2	30	M	0	$f(x) = 0.1\{\sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(5, 100, 4)$

Çizelge 3.9. Test için kullanılan unimodal ve multimodal benchmark fonksiyonları (Karaboga ve Akay, 2009) (Devam)

45	Langerman2	2	M	-1.08	$f(x)=\sum_{i=1}^{m} c_i \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j - a_{ij})^2\right)\right)\cos\left(\pi\sum_{j=1}^{n}(x_j - a_{ij})^2\right)$
46	Langerman5	5	M	-1.5	
47	Langerman10	10	M	-	
48	FletcherPowell2	2	M	0	$f(x)=\sum_{i=1}^{n}(A_i - B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$
49	FletcherPowell5	5	M	0	
50	FletcherPowell10	10	M	0	

U: Unimodal, M: Multimodal,
* 10, 30, 50

3.3.2. Deneysel Test Sonuçları

Bu test grubunda kullanılan parametre değerleri bölüm 3.1'de eğitilmiş ve eğitilmiş değerlerin sonuçları Çizelge 3.10'da verilmiştir. Her iki yöntem eğitilmiş başlangıç parametreleri ile Çizelge 3.9'da belirtilen kısıtsız, standart sayısal benchmark test fonksiyonları üzerinde uygulanmıştır. Sonuçlar, Çizelge 3.11'de gösterilmiştir. Çizelge 3.11'de sırasıyla problemin boyutu ile algoritmaların o fonksiyon üzerinde bulduğu minimum, maksimum, ortalama ve standart sapma değerleri sütunlar halinde belirtilmiştir. Ayrıca çizelgenin son iki sütununda sırasıyla *t-test* sonuçları ile bu sonuçlara bağlı olarak fonksiyonlar üzerinde üstünlük sağlayan algoritmanın ismi yer almaktadır.

Algoritmaların bazı unimodal ve multimodal fonksiyonlar üzerindeki yakınsama performansları sırasıyla Şekil 3.4 ve Şekil 3.5'te verilmiştir.

Çizelge 3.10. YA ve GYA parametrelerinin tüm fonksiyonlar üzerindeki başlangıç değerleri

Parametreler	YA	GYA
Çalışma süresi, RT	30	30
Popülasyon, N	50	50
Ses şiddeti, A	0.95	0.95
Sinyal yayılım oranı, r	0.85	0.85
Ses şiddeti ve sinyal yayılım oranı güncelleme faktörleri, $a-\gamma$	0.9	0.9
Minimum-maksimum frekans, $f_{min}-f_{max}$	0-1	0-1
Atalet ağırlığı başlangıç ve bitiş değerleri, $w_{min}-w_{max}$	-	0.9-0.2
Atalet ağırlığı modülasyon indeksi, n	-	2
ζ_1 ve ζ_2	-	0.6-0.4

Çizelge 3.10. YA ve GYA parametrelerinin tüm fonksiyonlar üzerindeki başlangıç değerleri (Devam)

ζ_1 ve ζ_2 modülasyon indeksi	-	3
Minimum üretilecek popülasyon, τ_{min}	-	0
Maksimum üretilecek popülasyon, τ_{max}	-	4

Çizelge 3.11. Standart ve Geliştirilmiş Yarasa Algoritmalarının temel benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi

		YA				GYA					
No	D	Minimum	Maksimum	Ortalama	Std. S.	Minimum	Maksimum	Ortalama	Std. S.	*t*	
1	5	0	0	0	0	0	0	**0**	0	~	-
2	10	5.40e+01	1.35e+03	5.22e+02	3.48e+02	0	0	**0**	0	8.0755	GYA
	30	2.99e+03	8.82e+03	5.44e+03	1.34e+03	0	3.00e-00	**9.66e-01**	1.06e-00	21.8336	GYA
	50	6.39e+03	1.54e+04	1.08e+04	2.61e+03	1.00e-00	1.80e+01	**7.13e-00**	3.60e-00	22.3300	GYA
3	10	1.55e+01	1.12e+03	4.37e+02	2.83e+02	1.64e-35	1.70e-30	**1.13e-31**	3.91e-31	8.2932	GYA
	30	2.13e+03	1.04e+04	4.65e+03	2.04e+03	6.97e-06	3.70e-05	**2.06e-05**	6.25e-06	12.2558	GYA
	50	6.14e+03	1.89e+04	1.06e+04	3.24e+03	1.08e-05	2.84e-05	**2.08e-05**	4.38e-06	17.7032	GYA
4	10	3.30e-02	1.69e-01	9.78e-02	3.61e-02	2.17e-34	1.88e-27	**6.41e-29**	3.43e-28	14.5803	GYA
	30	8.92e-01	6.55e+01	1.16e+01	1.38e+01	2.49e-04	1.11e-03	**5.43e-04**	1.77e-04	4.5556	GYA
	50	3.05e+01	3.49e+02	1.53e+02	9.00e+01	6.30e-04	3.14e-03	**1.13e-03**	6.40e-04	9.2078	GYA
5	10	2.80e-02	1.67e-01	9.12e-02	3.51e-02	2.96e-04	4.42e-03	**1.69e-03**	1.06e-03	13.7219	GYA
	30	3.92e-01	3.04e-00	1.48e-00	5.79e-01	1.18e-02	8.42e-02	**3.77e-02**	1.69e-02	13.4846	GYA
	50	4.48e-01	1.04e+01	4.85e-00	2.91e-00	1.16e-02	1.21e-01	**4.88e-02**	2.68e-02	8.8870	GYA
6	2	9.92e-06	8.10e-04	2.56e-04	2.45e-04	0	0	**0**	0	5.6373	GYA
7	2	-1.00e+00	0	-9.66e-01	1.83e-01	-1.00e-00	-1.00e-00	**-1.00e-00**	0	0.9895	-
8	2	6.08e-07	4.77e-05	1.22e-05	1.31e-05	1.98e-40	4.87e-37	**5.72e-38**	9.91e-38	5.0176	GYA

Çizelge 3.11. Standart ve Geliştirilmiş Yarasa Algoritmalarının temel benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

9	4	8.27e-02	5.50e-00	1.06e-00	1.53e-00	0	0	**0**	0	3.7345	GYA
10	6	-4.99e+01	-3.13e+01	-4.80e+01	4.30e-00	-5.00e+01	-5.00e+01	**-5.00e+01**	0	2.4549	GYA
11	10	-8.50e-00	1.47e+03	3.85e+02	3.59e+02	-2.10e+02	-2.10e+02	**-2.10e+02**	0	8.9346	GYA
12	10	1.11e-02	7.48e-02	3.46e-02	1.59e-02	2.05e-35	1.04e-09	**3.49e-11**	1.91e-10	11.6985	GYA
	30	4.64e+01	4.60e+02	1.71e+02	1.00e+02	2.78e-05	7.02e-05	**4.40e-05**	1.00e-05	9.1331	GYA
	50	4.78e+02	1.24e+03	8.66e+02	2.15e+02	4.88e-05	1.45e-03	**2.06e-04**	3.01e-04	21.6782	GYA
13	24	2.18e-01	5.13e-00	1.70e-00	1.17e-00	4.91e-04	1.07e-02	**3.02e-03**	2.21e-03	7.7893	GYA
14	10	1.24e-01	4.33e-01	3.01e-01	7.48e-02	2.15e-05	1.79e-01	**1.04e-02**	3.23e-02	19.2010	GYA
	30	4.92e-01	1.44e+01	4.09e-00	3.42e-00	3.15e-02	1.70e-00	**3.11e-01**	3.99e-01	5.8988	GYA
	50	1.10e+01	4.46e+01	2.07e+01	7.14e-00	4.50e-01	2.78e-00	**1.34e-00**	7.17e-01	14.5371	GYA
15	10	1.50e+02	2.18e+03	1.05e+03	5.01e+02	1.91e-35	5.54e-12	**1.84e-13**	1.01e-12	11.3544	GYA
	30	4.86e+03	2.82e+04	1.26e+04	5.66e+03	6.41e-05	2.60e-04	**1.18e-04**	3.96e-05	12.0651	GYA
	50	1.66e+04	1.43e+05	3.95e+04	2.37e+04	1.69e-03	3.19e-02	**1.07e-02**	6.06e-03	8.9742	GYA
16	10	6.72e-00	9.50e+03	7.46e+02	1.93e+03	4.19e-12	3.98e-00	**1.32e-01**	7.27e-01	2.0763	GYA
	30	2.54e+03	2.30e+06	1.98e+05	4.29e+05	8.34e-01	2.94e+01	**2.11e+01**	5.96e-00	2.4871	GYA
	50	1.26e+05	2.53e+06	1.04e+06	6.12e+05	3.94e+01	1.01e+02	**5.45e+01**	2.16e+01	9.1957	GYA
17	10	6.93e-01	8.20e-01	7.56e-01	3.50e-02	6.66e-01	6.66e-01	**6.66e-01**	1.79e-09	13.8873	GYA
	30	1.28e-00	7.45e+01	1.12e+01	1.44e+01	6.66e-01	6.69e-01	**6.67e-01**	6.48e-04	3.9463	GYA

Çizelge 3.11. Standart ve Geliştirilmiş Yarasa Algoritmalarının temel benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

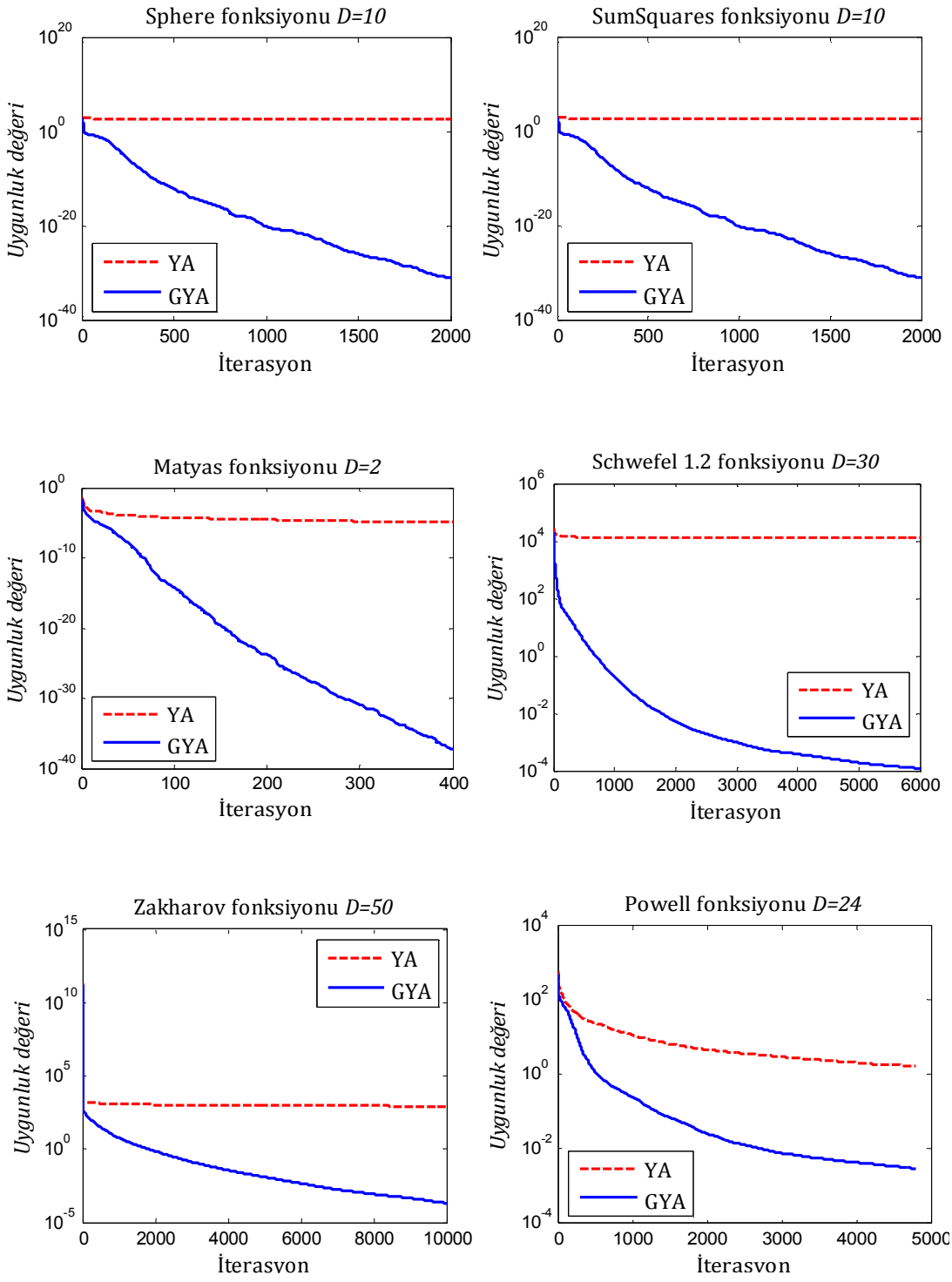
	50	1.15e+02	3.86e+03	4.92e+02	6.97e+02	6.67e-01	1.53e-00	**6.99e-01**	1.56e-01	3.8015	GYA
18	2	9.98e-01	9.98e-01	9.98e-01	2.20e-13	9.98e-01	9.98e-01	**9.98e-01**	2.22e-16	3.8986	GYA
19	2	3.97e-01	3.99e-01	3.98e-01	3.11e-04	3.97e-01	3.97e-01	**3.97e-01**	0	4.7628	GYA
20	2	4.29e-05	5.85e-03	1.76e-03	1.54e-03	0	0	**0**	0	6.1474	GYA
21	2	2.04e-05	1.99e-03	4.52e-04	5.41e-04	0	0	**0**	0	4.5002	GYA
22	10	1.31e+01	4.37e+01	3.06e+01	7.58e-00	3.97e-00	1.98e+01	**1.01e+01**	4.14e-00	12.7979	GYA
	30	8.61e+01	1.96e+02	1.42e+02	2.67e+01	1.99e+01	6.76e+01	**3.44e+01**	1.28e+01	19.5803	GYA
	50	8.42e+01	3.42e+02	2.25e+02	8.43e+01	3.18e+01	8.35e+01	**5.15e+01**	1.40e+01	10.9442	GYA
23	10	-2.52e+03	-1.41e+03	-2.03e+03	2.67e+02	-3.39e+03	-1.58e+03	**-2.55e+03**	4.14e+02	5.6800	GYA
	30	-4.58e+03	-2.57e+03	-3.59e+03	4.85e+02	-8.97e+03	-4.51e+03	**-6.98e+03**	9.61e+02	16.9875	GYA
	50	-7.27e+03	-3.34e+03	-4.84e+03	8.78e+02	-1.43e+03	-6.35e+03	**-1.12e+04**	1.56e+03	19.2871	GYA
24	2	-1.80e-00	-1.79e-00	-1.80e-00	3.28e-03	-1.80e-00	-1.80e-00	**-1.80e-00**	9.03e-16	5.2415	GYA
25	5	-4.10e-00	-3.11e-00	-3.67e-00	2.46e-01	-4.69e-00	-3.41e-00	**-4.45e-00**	3.33e-01	10.2037	GYA
26	10	-6.21e-00	-4.97e-00	-5.68e-00	3.32e-01	-9.41e-00	-6.08e-00	**-8.14e-00**	9.26e-01	13.4588	GYA
27	2	2.60e-03	7.81e-02	1.91e-02	1.92e-02	0	9.71e-03	**5.50e-03**	4.89e-03	3.7028	GYA

Çizelge 3.11. Standart ve Geliştirilmiş Yarasa Algoritmalarının temel benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

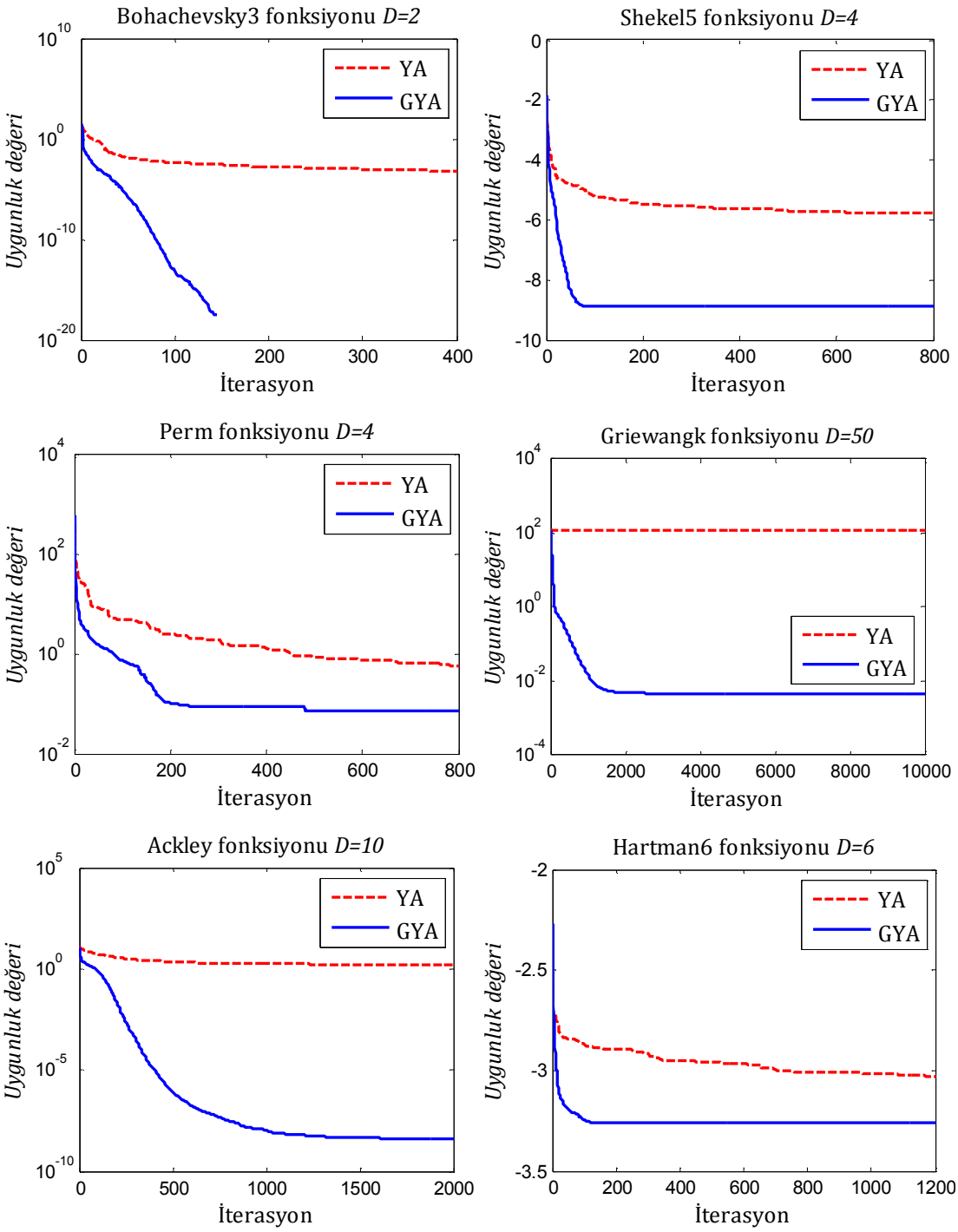
28	2	-1.03e-00	-1.02e-00	-1.03e-00	7.30e-04	-1.03e-00	-1.03e-00	**-1.03e-00**	6.18e-16	5.8509	GYA
29	2	5.27e-05	9.24e-03	2.55e-03	2.25e-03	0	0	**0**	0	6.0833	GYA
30	2	5.51e-05	2.32e-03	6.84e-04	6.44e-04	0	0	**0**	0	5.7218	GYA
31	2	-1.86e+02	-1.85e+02	-1.86e+02	3.53e-01	-1.86e+02	-1.86e+02	**-1.86e+02**	3.73e-14	5.2554	GYA
32	2	3.00e-00	3.13e-00	3.04e-00	3.19e-02	3.00e-00	3.00e-00	**3.00e-00**	0	1.3869	-
33	4	6.66e-04	2.03e-02	4.17e-03	6.87e-03	3.07e-04	2.03e-02	**3.07e-03**	6.90e-03	0.6117	-
34	4	-9.96e-00	-2.61e-00	-5.77e-00	3.14e-00	-1.01e+01	-2.63e-00	**-8.89e-00**	2.36e-00	4.2673	GYA
35	4	-1.02e+01	-2.70e-00	-6.90e-00	3.14e-00	-1.04e+01	-2.76e-00	**-9.46e-00**	2.46e-00	3.4403	GYA
36	4	-1.03e+01	-2.36e-00	-7.85e-00	3.22e-00	-1.05e+01	-5.17e-00	**-1.01e+01**	1.36e-00	3.5741	GYA
37	4	2.52e-02	1.27e-00	5.51e-01	3.52e-01	1.00e-20	4.72e-01	**7.20e-02**	1.44e-01	6.7898	GYA
38	4	9.33e-03	2.52e-01	9.41e-02	7.32e-02	7.87e-07	4.07e-04	**1.51e-04**	1.45e-04	6.9132	GYA
39	3	-3.86e-00	-3.81e-00	-3.85e-00	9.27e-03	-3.86e-00	-3.86e-00	**-3.86e-00**	2.62e-15	7.0995	GYA
40	6	-3.12e-00	-2.90e-00	-3.02e-00	5.29e-02	-3.32e-00	-3.20e-00	**-3.25e-00**	6.04e-02	15.5573	GYA

Çizelge 3.11. Standart ve Geliştirilmiş Yarasa Algoritmalarının temel benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

41	10	2.82e-00	2.20e+01	1.18e+01	5.44e-00	5.66e-02	2.67e-00	**9.04e-01**	6.57e-01	10.7187	GYA
	30	3.31e+01	9.63e+01	6.45e+01	1.82e+01	5.10e-07	4.43e-02	**7.30e-03**	9.46e-03	19.0546	GYA
	50	7.05e+01	1.86e+02	1.16e+02	2.74e+01	5.77e-07	2.21e-02	**4.68e-03**	6.21e-03	22.9103	GYA
42	10	9.09e-02	7.98e-00	1.63e-00	2.32e-00	4.44e-15	1.26e-07	**4.21e-09**	2.30e-08	3.8019	GYA
	30	4.49e-00	1.25+01	9.17e-00	2.07e-00	2.41e-03	1.50e-00	**4.58e-01**	5.78e-01	21.8369	GYA
	50	8.54e-00	1.34e+01	1.08e+01	1.28e-00	1.02e-00	2.57e-00	**1.80e-00**	4.11e-01	35.9427	GYA
43	30	7.68e-00	6.21e+04	2.14e+03	1.13e+04	2.64e-07	9.00e-00	**9.99e-01**	1.84e-00	1.0192	-
44	30	6.66e+01	3.51e+02	1.71e+02	6.41e+01	1.87e-06	1.93e-02	**7.74e-03**	9.64e-03	14.4012	GYA
45	2	-1.08e-00	-1.08e-00	-1.08e-00	2.29e-04	-1.08e-00	-1.08e-00	**-1.08e-00**	6.51e-16	7.5889	GYA
46	5	-1.49e-00	-4.82e-01	-9.53e-01	3.88e-01	-1.49e-00	-9.07e-01	**-1.18e-00**	2.80e-00	2.6005	GYA
47	10	-7.97e-01	-2.74e-01	-4.80e-01	1.81e-01	-1.49e-00	-2.74e-01	**-6.38e-01**	3.04e-00	2.3956	GYA
48	2	4.36e-02	1.94e-00	5.64e-01	4.27e-01	0	0	**0**	0	7.1099	GYA
49	5	3.29e+01	1.61e+03	2.93e+02	4.51e+02	0	2.52e+02	**7.70e+01**	1.00e+02	2.5197	GYA
50	10	3.03e+02	9.43e+03	2.74e+03	2.67e+03	1.71e-71	6.60e+02	**1.92e+02**	2.34e+02	5.1303	GYA



Şekil 3.4. Standart ve Geliştirilmiş Yarasa Algoritmalarının bazı unimodal fonksiyonlar üzerindeki yakınsama performansı



Şekil 3.5. Standart ve Geliştirilmiş Yarasa Algoritmalarının bazı multimodal fonksiyonlar üzerindeki yakınsama performansı

3.4. Yöntemin Karmaşık Fonksiyon Kümesi Üzerinde Test Edilmesi

3.4.1. Karmaşık benchmark test fonksiyonları

Bu test aşamasında Yarasa Algoritması ve geliştirilmiş versiyonu, önceki test aşamasında kullanılan standart sayısal benchmark fonksiyonlarına oranla optimize edilmesi daha zor olan, CEC05 (Congress of Evolutionary Computation 2005) olarak bilinen karmaşık benchmark fonksiyonları üzerinde test edilmiştir. Testte kullanılan 25 adet benchmark test fonksiyon kümesi literatürden (Suganthan vd., 2005) alınmıştır. Bu test kümesinde bulunan test fonksiyonları unimodal, multimodal, kaydırılmış, döndürülmüş ve hibritlenmiş fonksiyonlardan oluşmaktadır (bkz. Çizelge 3.12). Bu test grubundaki fonksiyonların en önemli özelliği her fonksiyonun global optimum noktasına ait çözüm değerleri uzayın farklı noktalarında bulunmasıdır.

Çizelge 3.12. Test için kullanılan karmaşık benchmark fonksiyon karakteristikleri (Suganthan vd., 2005)

No(f)	Karakteristik	Açıklama
1	U (Temel)	Kaydırılmış Sphere fonksiyonu
2	U (Temel)	Kaydırılmış Schwefel 1.2 fonksiyonu
3	U (Temel)	Kaydırılmış, döndürülmüş Elliptic fonksiyonu
4	U (Temel)	Kaydırılmış Schwefel 1.2 fonksiyonu (gürültülenmiş uygunluk değeri)
5	U (Temel)	Schwefel 1.2 fonksiyonu (uzay sınırına kaydırılmış global optimum çözümü)
6	M (Temel)	Kaydırılmış Rosenbrock fonksiyonu
7	M (Temel)	Kaydırılmış, döndürülmüş Grewangk fonksiyonu (sınırlandırılmamış arama uzayı)
8	M (Temel)	Kaydırılmış, döndürülmüş Ackley fonksiyonu (uzay sınırına kaydırılmış global optimum noktası)
9	M (Temel)	Kaydırılmış Rastrigin fonksiyonu
10	M (Temel)	Kaydırılmış, döndürülmüş Rastrigin fonksiyonu
11	M (Temel)	Kaydırılmış, döndürülmüş Weierstrass fonksiyonu
12	M (Temel)	Schwefel 2.13 fonksiyonu
13	M (Genişletilmiş)	Genişletilmiş, geliştirilmiş Griewangk ve Rosenbrock fonksiyonu
14	M (Genişletilmiş)	Kaydırılmış, döndürülmüş, genişletilmiş Scaffer fonksiyonu

Çizelge 3.12. Test için kullanılan karmaşık benchmark fonksiyon karakteristikleri (Suganthan vd., 2005) (Devam)

15	M (Hibrit)	Hibrit bileşim fonksiyonu (Rastrigin, Weierstrass, Grieangk, Ackley, Sphere fonksiyonları)
16	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (döndürülmüş f15 fonksiyonu)
17	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (uygunluk değerine gürültü eklenmiş f16 fonksiyonu)
18	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (Ackley, Rastrigin, Sphere, Weierstrass, Grieangk fonksiyonları)
19	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (global optimum havzası daraltılmış f18 fonksiyonu)
20	M(Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (global optimum noktası uzay sınırına kaydırılmış f18 fonksiyonu)
21	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (f14, Rastrigin, f13, Weierstrass, Griewangk fonksiyonları)
22	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (döndürme matrisi koşulları artırılmış f21 fonksiyonu)
23	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (sürekli olmayan f21 fonksiyonu)
24	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (Weierstrass, f14, f13, Ackley, Rastrigin, Griewangk fonksiyonları)
25	M (Hibrit)	Döndürülmüş hibrit bileşim fonksiyonu (sınırlandırılmamış arama uzaylı f24 fonksiyonu)

U: Unimodal, M: Multimodal,

Test fonksiyonlarının (7 ve 25 numaralı fonksiyonlar dışında) global optimum noktaları arama uzayı sınırları içerisindedir ve başlangıç çözüm kümeleri arama uzayında rastgele oluşturulmuşlardır. 7 ile 25 numaralı fonksiyonların başlangıç çözüm kümeleri sırasıyla [0, 600] ve [2, 5] aralığında rastgele oluşturulmuşlardır. Ayrıca bu iki fonksiyonun global optium noktaları başlangıç noktalarının dışında yer almaktadır. YA ve GYA tüm fonksiyonlar üzerinde 10, 30 ve 50 boyutlarda test edilmiştir. Maksimum fonksiyon hesaplama sayısı (Λ_{max}) eşitlik 3.9'a göre belirlenmiştir ve belirtilen boyutlar için sırasıyla 100.000, 300.000 ve 500.000 değerlerini almıştır.

$$\Lambda_{max} = (10.000)d \tag{3.9}$$

Yukarıdaki eşitlikte d, optimize edilecek benchmark fonksiyonun boyutunu belirtmektedir.

3.4.2. Deneysel test sonuçları

Önceki test aşamasında olduğu gibi bu test aşamasında da Standart ve Geliştirilmiş Yarasa Algoritması Çizelge 3.10'da belirtilen parametre başlangıç değerleri ile CEC05 fonksiyonları üzerinde test edilmiştir. Sonuçlar Çizelge 3.13'te gösterilmiştir.

Algoritmaların bazı temel unimodal ve multimodal karmaşık fonksiyon kümesi üzerindeki yakınsama performansları sırasıyla Şekil 3.6 ve Şekil 3.7'de verilmiştir.

Çizelge 3.13. Standart ve Geliştirilmiş Yarasa Algoritmalarının karmaşık benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi

		YA				GYA					
No	D	Minimum	Maksimum	Ortalama	Std. S.	Minimum	Maksimum	Ortalama	Std. S.	*t*	
1	10	1.17e+03	7.00e+03	3.54e+03	1.81e+03	0	1.34e-11	**4.46e-13**	2.44e-12	10.5233	GYA
	30	1.84e+04	6.85e+04	4.53e+04	9.97e+03	8.28e-06	3.26e-05	**1.99e-05**	5.65e-06	24.4722	GYA
	50	7.38e+04	1.37e+05	1.03e+05	1.41e+04	1.14e-05	2.85e-05	**2.08e-05**	4.37e-06	39.2942	GYA
2	10	9.45e+02	1.19e+04	6.10e+03	3.07e+03	0	2.82e-11	**9.41e-13**	5.15e-12	10.7030	GYA
	30	3.55e+04	1.20e+05	6.65e+04	1.92e+04	5.58e-05	2.14e-04	**1.19e-04**	4.07e-05	18.6345	GYA
	50	1.02e+05	4.16e+05	2.02e+05	6.46e+04	2.88e-03	1.87e-02	**7.48e-03**	3.92e-03	16.8350	GYA
3	10	3.13e+06	1.45e+08	2.82e+07	3.05e+07	1.33e-04	6.93e+03	**5.15e+02**	1.26e+03	4.9886	GYA
	30	2.86e+08	1.17e+09	6.27e+08	2.16e+08	1.19e+04	2.71e+05	**6.75e+04**	4.48e+04	15.6134	GYA
	50	1,66e+09	4,49e+09	2,56e+09	6,89e+08	1.18e+05	4.24e+05	**2.46e+05**	7.86e+04	20.0089	GYA
4	10	2.39e+03	2.38e+04	9.56e+03	4.45e+03	4.32e-10	9.68e-04	**8.56e-05**	2.29e-04	11.5579	GYA
	30	4.18e+04	1.33e+05	7.66e+04	2.38e+04	5.29e-02	3.01e-01	**1.91e-01**	6.68e-02	17.3207	GYA
	50	1.33e+05	3.97e+05	2.30e+05	7.55e+04	2.50e+01	6.80e+02	**1.62e+02**	1.28e+02	16.4022	GYA
5	10	1.49e+02	6.80e+03	3.11e+03	2.24e+03	1.09e-11	1.74e-02	**8.43e-04**	3.19e-03	7.4778	GYA
	30	1.81e+04	3.89e+04	2.81e+04	5.45e+03	1.15e+03	3.95e+03	**2.57e+03**	7.37e+02	24.9534	GYA
	50	3.09e+04	4.57e+04	3.81e+04	4.07e+03	3.00e+03	9.17e+03	**6.05e+03**	1.23e+03	40.6518	GYA
6	10	5.31e+06	8.79e+08	1.74e+08	2.13e+08	9.73e-11	6.04e-00	**1.14e-00**	1.97e-00	4.3824	GYA
	30	3.16e+09	3.33e+10	1.60e+10	8.14e+09	2.50e-00	1.34e+03	**2.75e+02**	4.10e+02	10.5715	GYA
	50	2.16e+10	5.41e+10	3.90e+10	9.00e+09	3.38e+01	1.31e+03	**3.20e+02**	4.02e+02	23.3347	GYA

Çizelge 3.13. Standart ve Geliştirilmiş Yarasa Algoritmalarının karmaşık benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

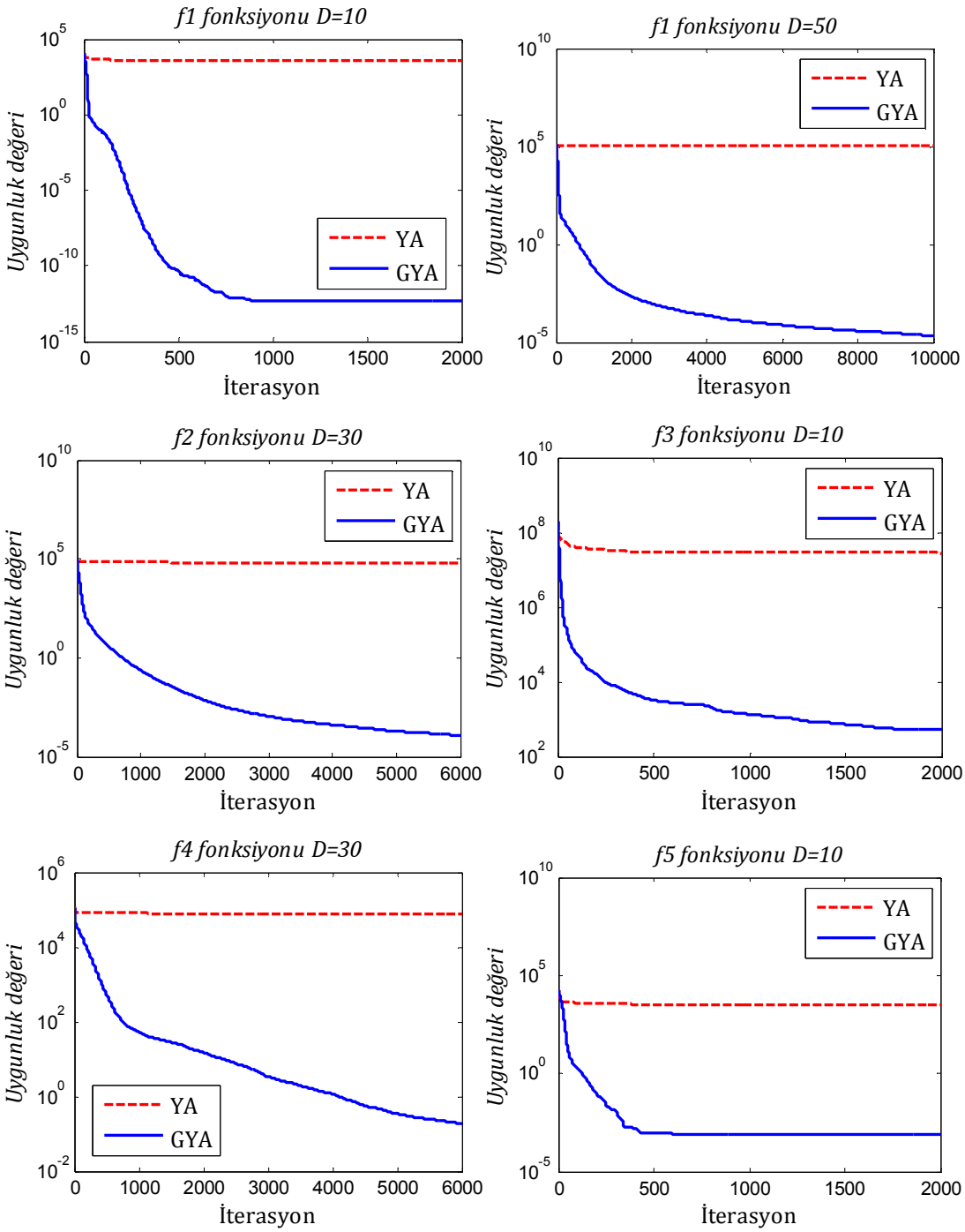
7	10	2.45e+02	2.35e+03	1.05e+03	5.28e+02	5.66e-02	5.00e-00	**8.34e-01**	1.21e-00	10.7156	GYA
	30	5.60e+03	1.07e+04	7.46e+03	1.26E+03	5.54e-06	4.67e-02	**1.40e-02**	1.16e-02	31.9114	GYA
	50	1.00e+04	1.53e+04	1.28e+04	1.68e+03	3.01e-06	3.19e-02	**5.75e-03**	8.13e-03	41.2080	GYA
8	10	2.02e+01	2.06e+01	2.04e+01	9.34e-02	2.00e+01	2.04e+01	**2.02e+01**	9.34e-02	10.7058	GYA
	30	2.09e+01	2.11e+01	2.10e+01	5.65e-02	2.04e+01	2.08e+01	**2.06e+01**	1.07e-01	17.0280	GYA
	50	2.11e+01	2.13e+01	2.12e+01	4.40e-02	2.06e+01	2.10e+01	**2.08e+01**	1.32e-01	14.8429	GYA
9	10	1.15e+01	4.82e+01	2.81e+01	8.36e-00	1.99e-00	2.39e+01	**1.12e+01**	5.64e-00	8.9782	GYA
	30	7.48e+01	1.70e+02	1.12e+02	1.96e+01	2.29e+01	1.17e+02	**5.22e+01**	2.18e+01	11.0316	GYA
	50	1.84e+02	3.32e+02	2.54e+02	3.40e+01	6.47e+01	1.82e+02	**1.24e+02**	3.04e+01	15.4173	GYA
10	10	1.79e+01	4.69e+01	3.31e+01	6.79e-00	4.97e-00	2.79e+01	**1.33e+01**	5.14e-00	12.4867	GYA
	30	1.03e+02	2.13e+02	1.57e+02	2.56e+01	2.89e+01	9.25e+01	**5.75e+01**	1.60e+01	17.7469	GYA
	50	2.62e+02	4.57e+02	3.42e+02	5.00e+01	9.25e+01	2.20e+02	**1.39e+02**	3.19e+01	18.4542	GYA
11	10	8.04e-00	1.09e+01	9.57e-00	7.13e-01	3.54e-00	8.62e-00	**5.77e-00**	1.31e-00	13.6992	GYA
	30	3.90e+01	4.27e+01	4.10e+01	9.39e-01	2.24e+01	3.40e+01	**2.96e+01**	2.98e-00	19.6902	GYA
	50	6.92e+01	7.72e+01	7.45e+01	1.79e-00	4.65e+01	6.79e+01	**5.85e+01**	5.48e-00	14.9662	GYA
12	10	1.68e+02	2.08e+04	3.42e+03	4.78e+03	1.19e-02	2.01e+04	**2.05e+03**	4.51e+03	1.1227	-
	30	8.04e+03	9.35e+04	3.67e+04	1.93e+04	9.99e+01	8.11e+04	**1.37e+04**	1.85e+04	4.6321	GYA
	50	5.98e+04	6.52e+05	4.12e+05	1.26e+05	1.02e+04	3.02e+05	**7.12e+04**	5.89e+04	13.1521	GYA
13	10	1.70e-00	3.95e-00	3.01e-00	5.28e-01	4.47e-01	1.89e-00	**9.71e-01**	4.10e-01	16.4360	GYA
	30	1.31e+01	1.99e+01	1.67e+01	1.64e-00	2.49e-00	7.80e-00	**4.21e-00**	1.22e-00	32.8173	GYA

Çizelge 3.13. Standart ve Geliştirilmiş Yarasa Algoritmalarının karmaşık benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

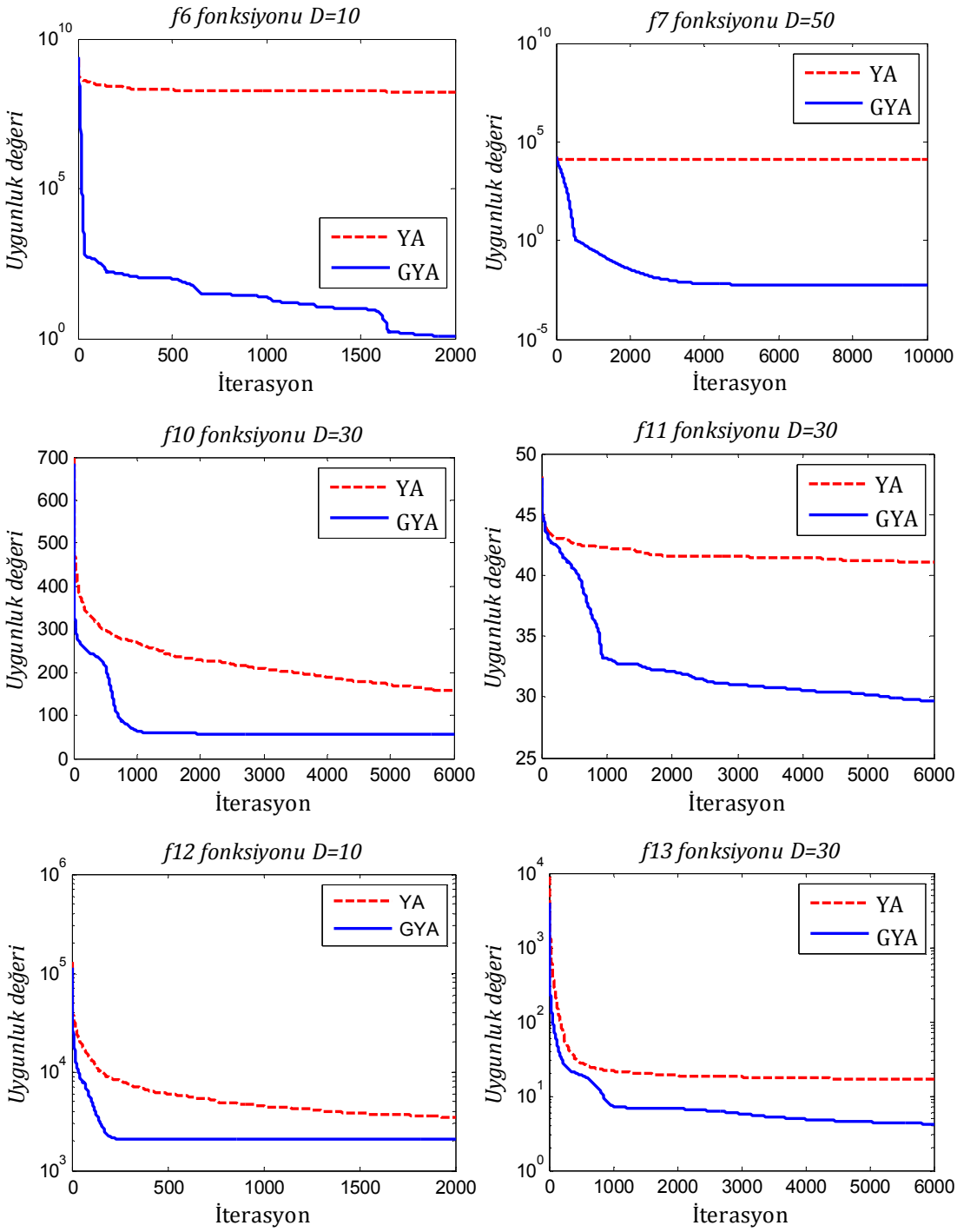
	50	2.98e+01	9.02e+01	4.24e+01	1.21e+01	7.07e-00	2.57e+01	**1.23e+01**	3.99e-00	12.7418	GYA
14	10	3.34e-00	4.42e-00	3.98e-00	2.59e-01	2.55e-00	4.01e-00	**3.34e-00**	4.43e-01	6.75092	GYA
	30	1.29e+01	1.41e+01	1.36e+01	3.32e-01	1.22e+01	1.40e+01	**1.29e+01**	3.88e-01	7.67885	GYA
	50	2.19e+01	2.39e+01	2.33e+01	4.74e-01	2.14e+01	2.30e+01	**2.22e+01**	4.37e-01	9.28414	GYA
15	10	2.31e+02	8.01e+02	4.17e+02	1.23e+02	9.12e+01	8.00e+02	**3.39e+02**	1.64e+02	2.0523	GYA
	30	2.33e+02	7.61e+02	5.59e+02	1.16e+02	2.66e+02	6.49e+02	**5.08e+02**	9.54e+01	1.8390	-
	50	3.75e+02	8.96e+02	6.36e+02	1.35e+02	1.71e+02	6.34e+02	**4.63e+02**	1.00e+02	5.5508	GYA
16	10	1.37e+02	2.31e+02	1.68e+02	1.89e+01	9.99e+01	4.07e+02	**1.26e+02**	5.42e+01	3.8968	GYA
	30	1.41e+02	5.62e+02	3.32e+02	1.39e+02	7.94e+01	5.02e+02	**3.11e+02**	1.67e+02	0.5421	-
	50	2.11e+02	4.99e+02	3.34e+02	8.56e+01	1.39e+02	5.01e+02	**2.70e+02**	1.06e+02	2.5236	GYA
17	10	1.37e+02	2.45e+02	1.84e+02	2.17e+01	1.03e+02	1.56e+02	**1.24e+02**	1.30e+01	12.8538	GYA
	30	1.79e+02	8.66e+02	3.88e+02	1.70e+02	9.49e+01	5.37e+02	**3.24e+02**	1.64e+02	1.4618	-
	50	3.55e+02	4.97e+02	6.91e+02	1.05e+02	1.73e+02	5.25e+02	**3.17e+02**	9.41e+01	6.8707	GYA
18	10	3.12e+02	1.07e+03	7.94e+02	2.76e+02	3.00e+02	1.04e+03	**7.45e+02**	3.05e+02	0.6375	-
	30	9.12e+02	1.06e+03	9.71e+02	3.38e+01	8.17e+02	9.00e+02	**8.23e+02**	1.51e+01	21.5186	GYA
	50	9.23e+02	1.12e+03	1.06e+03	4.60e+01	8.58e+02	9.88e+02	**9.11e+02**	4.08e+01	13.3641	GYA
19	10	3.14e+02	1.04e+03	8.04e+02	2.83e+02	3.00e+02	1.02e+03	**7.24e+02**	2.53E+02	1.1347	-
	30	9.14e+02	1.03e+03	9.53e+02	3.05e+01	9.00e+02	9.17e+02	**9.09e+02**	3.15e+00	7.7151	GYA
	50	9.11e+02	1.13e+03	1.06e+03	5.45e+01	9.00e+02	9.77e+02	**9.28e+02**	2.57e+01	11.4547	GYA

Çizelge 3.13. Standart ve Geliştirilmiş Yarasa Algoritmalarının karmaşık benchmark fonksiyonları üzerinde karşılaştırmalı gösterimi (Devam)

20	10	3.15e+02	1.04e+03	8.08e+02	2.82e+02	3.00e+02	1.00e+03	**7.62e+02**	2.37e+02	0.6682	-
	30	9.10e+02	1.05e+03	9.66e+02	3.68e+01	9.07e+02	9.48e+02	**9.12e+02**	7.10e-00	7.7357	GYA
	50	1.00e+03	1.13e+03	1.07e+03	3.41e+01	9.00e+02	9.81e+02	**9.32e+02**	2.80e+01	16.5832	GYA
21	10	3.11e+02	1.22e+03	8.70e+02	3.31e+02	2.00e+02	1.19e+03	**7.00e+02**	3.05e+02	2.0306	GYA
	30	5.05e+02	1.20e+03	1.04e+03	2.35e+02	5.00e+02	1.17e+03	**6.40e+02**	2.49e+02	6.2544	GYA
	50	1.07e+03	1.26e+03	1.21e+03	4.20e+01	5.00e+02	1.20e+03	**8.05e+02**	2.82e+02	7.7132	GYA
22	10	7.69e+02	8.48e+02	9.78e+02	7.26e+01	3.00e+02	8.35e+02	**7.63e+02**	9.18e+01	3.9033	GYA
	30	9.17e+02	1.06e+03	1.18e+03	5.24e+01	8.54e+02	9.92e+02	**9.17e+02**	2.89e+01	12.8320	GYA
	50	1.08e+03	1.31e+03	1.17e+03	6.63e+01	8.99e+02	1.00e+03	**9.73e+02**	2.72e+01	14.8622	GYA
23	10	5.54e+02	9.61e+02	1.26e+03	2.52e+02	5.59e+02	1.24e+03	**8.98e+02**	2.40e+02	0.9689	-
	30	6.53e+02	1.11e+03	1.22e+03	1.48e+02	5.34e+02	1.17e+03	**6.22e+02**	1.88e+02	11.0006	GYA
	50	1.13e+03	1.27e+03	1.24e+03	2.54e+01	5.67e+02	1.21e+03	**9.01e+02**	2.19e+02	8.1642	GYA
24	10	5.09e+02	8.46e+02	1.03e+03	2.26e+02	2.00e+02	5.00e+02	**3.33e+02**	1.12e+02	10.9884	GYA
	30	6.06e+02	1.12e+03	1.27e+03	1.45e+02	2.00e+02	9.96e+02	**5.38e+02**	3.78e+02	7.7588	GYA
	50	1.24e+03	1.34e+03	1.28e+03	1.84e+01	2.00e+02	1.24e+03	**5.01e+02**	4.28e+02	9.8195	GYA
25	10	2.01e+02	1.28e+03	6.90e+02	2.99e+02	3.66e+02	9.00e+02	**3.98e+02**	9.75e+01	4.9987	GYA
	30	6.08e+02	1.39e+03	1.21e+03	1.97e+02	2.12e+02	1.18e+03	**2.48e+02**	1.77e+02	19.4752	GYA
	50	1.37e+03	1.51e+03	1.45e+03	4.16e+01	2.26e+02	1.19e+03	**3.09e+02**	2.36e+02	25.5351	GYA



Şekil 3.6. Standart ve Geliştirilmiş Yarasa Algoritmalarının bazı unimodal karmaşık fonksiyonlar üzerindeki yakınsama performansı



Şekil 3.7. Standart ve Geliştirilmiş Yarasa Algoritmalarının bazı multimodal karmaşık fonksiyonlar üzerindeki yakınsama performansı

3.5. Yöntemin Gerçek Hayat Problemleri Üzerinde Test Edilmesi

Bu test aşamasında literatürden (Gandomi vd., 2013) gerçek hayattaki bazı mühendislik problemleri seçilerek geliştirilen yöntemin performansının kısıtlı problemler üzerinde ölçülmesi amaçlanmıştır. Geliştirilen yöntemin etkinliğinin

bu tür problemler üzerinde daha net anlaşılabilmesi için yöntemin bu problem üzerindeki performansı Gandomi vd. (2013)'nin kısıtlı gerçek hayat problemleri üzerine yaptığı çalışmada yer alan sonuçlar ile mukayese edilmiştir. Adil bir kıyaslama yapılabilmesi için popülasyon sayısı, fonksiyon hesaplama sayısı değerleri bu çalışmayla eşit tutulmuştur ve bu değerler Çizelge 3.14'te gösterilmiştir. Ayrıca sonuçların kararlı ve istikrarlı olabilmesi için her iki yöntem 30 defa çalıştırılmıştır.

Algoritmaların maliyet fonksiyonuna penaltı metodunda yer alan ceza katsayı puanı dâhil edilerek Geliştirilmiş ve Standart Yarasa Algoritmaları kısıtlı problemlerin çözümüne uygun hale getirilmiştir. Bu amaçla eşitlik 3.10'da verilen denklem algoritmaların problemin maliyet değerini hesaplamalarında esas alınmıştır.

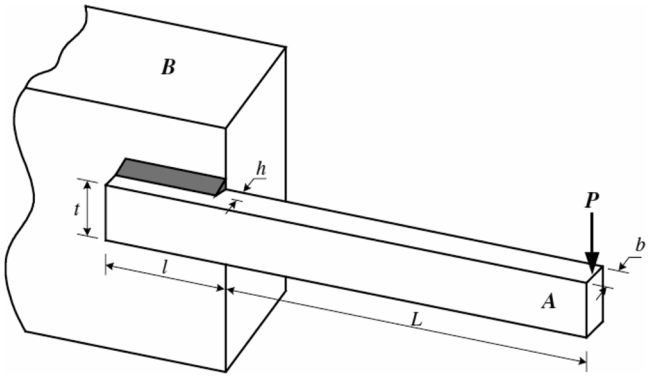
$$\Pi(x) = f(x) + \sum_{i=1}^{M} \mu_i \phi_i^2(x) + \sum_{j=1}^{M} v_j \psi_j^2 \tag{3.10}$$

Yukarıdaki denklemde ϕ_i eşitlik bildiren kısıtların maliyet değerini; ψ_j, eşit olmayan kısıtların maliyet değerini temsil etmektedir. μ_i ve v_j ise eşitlik sağlanıp sağlanamaması durumuna bağlı olarak problemin maliyet değerine etki edecek sabit penaltı parametresini temsil etmektedir.

Çizelge 3.14. Gerçek hayat problemlerinde kullanılan genel parametre değerleri

Problem	Popülasyon sayısı	Fonksiyon hesaplama sayısı
Kaynaklı kiriş tasarımı	25	50000
Gerilim-sıkıştırma yay tasarımı	10	5000
Basınçlı kap tasarımı	25	15000

3.5.1. Kaynaklı kiriş tasarımı problemi



Şekil 3.8. Kaynaklı kiriş örneği

Bu problemin amacı belli kısıtlar altında minimum maliyetle kanyaklı kiriş tasarımı elde etmektir. Şekil 3.8'de A kirişinden ve bu kirişin B nesnesine tutunması için ihtiyaç duyulan kaynaktan oluşan bir kaynaklı kiriş yapısını temsil etmektedir. Problem dört adet tasarım değişkeninden oluşmaktadır (x_1, x_2, x_3, x_4). Şekil 3.8'de kaynak inceliği *h*, (x_1); kaynak bağlantı uzunluğu *l*, (x_2); kiriş genişliği *t*, (x_3); kiriş inceliği *b*, (x_4) ile temsil edilmektedir. Bu değerlerin alabileceği değer aralıkları sırasıyla $0.125 \leq x_1 \leq 5$ ve $0.1 \leq x_2, x_3, x_4 \leq 10$ olarak tanımlanmıştır. Problemin amaç fonksiyonu eşitlik 3.11'de, kısıtları ise eşitlik 3.12-3.18'de belirtilmiştir.

$$f(x) = (1 + C_1)x_1^2 x_2 + C_2 x_3 x_4 (L + x_2) \tag{3.11}$$

$$g_1(x) = \tau(x) - \tau_d \leq 0 \tag{3.12}$$

$$\tau(x) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2} \tag{3.12a}$$

$$\tau' = \frac{6000}{\sqrt{2}x_1 x_2} \tag{3.12b}$$

$$\tau'' = \frac{MR}{J} \tag{3.12c}$$

$$M = 6000(14 + \frac{x_2}{2}) \tag{3.12d}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{3.12e}$$

$$J = 2\left\{x_1 x_2 \sqrt{2}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{3.12f}$$

$$g_2(x) = \sigma(x) - \sigma_d \leq 0 \tag{3.13}$$

$$\sigma(x) = \frac{504000}{x_4 x_3^2} \tag{3.13a}$$

$$g_3(x) = x_1 - x_4 \leq 0 \tag{3.14}$$

$$g_4(x) = C_1 x_1^2 + C_2 x_3 x_4 (L + x_2) - 5 \leq 0 \tag{3.15}$$

$$g_5(x) = 0.125 - x_1 \leq 0 \tag{3.16}$$

$$g_6(x) = \delta(x) - \delta_d \leq 0 \tag{3.17}$$

$$\delta(x) = \frac{2.1952}{x_4 x_3^3} \tag{3.17a}$$

$$g_7(x) = P - P_c(x) \leq 0 \tag{3.18}$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{E}{4G}}}{28}\right) \tag{3.18a}$$

Yukarıdaki fonksiyon ve kısıtlarda yer alan parametrelerin açıklamaları ve değerleri Çizelge 3.15'te verilmiştir.

Çizelge 3.15. Kaynaklı kiriş tasarımı probleminin parametre değerleri

Parametre	Tanım	Değer
C_1	Kaynaklı malzemenin hacim başı maliyeti	0.10471 \$/in^3
C_2	Metal çubuğun hacim başı maliyeti	0.04811 \$/in^3
τ_d	Kaynaklı malzemenin kesme gerilimi	13600 psi
σ_d	Çubuk malzemenin normal gerilimi	30000 psi

Çizelge 3.15. Kaynaklı kiriş tasarımı probleminin parametre değerleri (Devam)

δ_d	Çubuk sonundaki sapma	0.25 in
E	Çubuk stokunun Young katsayısı	$30x10^6$ psi
G	Çubuk stokunun kesim katsayısı	$12x10^6$ psi
P	Koşulların oluşumu	6000 pound
L	Kirişin çıkıntı uzunluğu	14 in

Geliştirilen yöntem, Yarasa Algoritması (her iki yöntemde Çizelge 3.10'da verilen parametreler kullanılmıştır) ve Gandomi vd. (2013)'nin çalışması ile bu problem üzerinde karşılaştırmalı olarak test edilmiştir. Sonuçlar Çizelge 3.16'da verilmiştir. Ayrıca yöntemin (GYA) bu problem üzerindeki başarısı literatürdeki çalışmalar ile tek tek kıyaslanarak sonuçları Çizelge 3.17'de verilmiştir.

Çizelge 3.16. GYA'nın kaynaklı kiriş tasarımı problemi üzerindeki performansının karşılaştırmalı değerleri

Metot	Minimum	Maksimum	Ortalama	Ortanca	Std. S.
YA	1.8550794	2.6437666	2.2349807	2.2146606	0.2133583
YA*	1.7312065	2.3455793	1.8786560	-	0.2677989
GYA	1.7248525	1.7851591	1.7283549	1.7249882	0.0112228

* Gandomi vd. (2013)

Çizelge 3.17. GYA'nın kaynaklı kiriş tasarımı problemi üzerindeki performansının literatür ile karşılaştırmalı değerleri

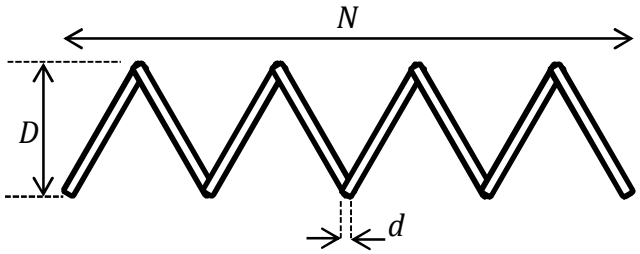
Araştırma	x_1	x_2	x_3	x_4	Maliyet
Coello (1999)	0.2088	3.4205	8.9975	0.2100	1.7483
Leite ve Topping (1998)	0.2489	6.1097	8.2484	0.2485	2.4000
Deb (2000)	-	-	-	-	2.3800
Deb (1991)	0.2489	6.1730	8.1789	0.2533	2.4331
Lemonge ve Barbosa (2004)	0.2443	6.2117	8.3015	0.2443	2.3816
Barbosa ve Lemonge (2002)	0.2442	6.2231	8.2915	0.2444	2.3814
Bernardino vd. (2007)	0.2443	6.2202	8.2915	0.2444	2.3812
Bernardino vd. (2008)	0.2444	6.2183	8.2912	0.2444	2.3812
Atiqullah ve Rao (2000)	0.2471	6.1451	8.2721	0.2495	2.4148
Hedar ve Fukushima (2006)	0.2444	6.2158	8.2939	0.2444	2.3811

Çizelge 3.17. GYA'nın kaynaklı kiriş tasarımı problemi üzerindeki performansının literatür ile karşılaştırmalı değerleri (Devam)

Liu (2005)	0.2444	6.2175	8.2915	0.2444	2.3810
Hwang ve He (2006)*	0.2231	1.5815	12.846	0.2245	2.2500
Parsopoulos ve Vrahatis (2005)	-	-	-	-	1.9220
He vd. (2004)	0.2444	6.2175	8.2915	0.2444	2.3810
Zhang vd. (2009)	0.2443	6.2201	8.2940	0.2444	2.3816
Coello (2000)	-	-	-	-	1.8245
Runarsson ve Yao (2000)	0.2758	5.0053	8.6261	0.2758	2.5961
Gandomi vd. (2011)	0.2015	3.562	9.0414	0.2057	1.7312
Montes ve Ocana (2008)	0.2057	3.4711	9.0367	0.2057	2.3868
Lee ve Geem (2005)	0.2442	6.2231	8.2915	0.2443	2.3807
Siddall (1972)	0.2444	6.2819	8.2915	0.2444	2.3815
Kaveh ve Talatahari (2010)	0.2058	3.4681	9.0380	0.2057	1.7249
Akhtar vd. (2002)	0.2407	6.4851	8.2399	0.2497	2.4426
Ray ve Liew (2003)	0.2444	6.2380	8.2886	0.2446	2.3854
Ragsdell ve Philips (1976)	0.2536	7.1410	7.1044	0.2536	2.3398
Zhang vd. (2008)	0.2444	6.2175	8.2915	0.2444	2.3810
Aragon vd. (2010)	0.2444	6.2186	8.2915	0.2444	2.3811
Datta ve Figueira (2011)	0.1875	1.7821	8.2500	0.2500	1.9553
Sadollah vd. (2013)	0.2057	3.4704	9.0366	0.2057	1.7249
Gandomi vd. (2013)	0.2015	3.5620	9.0414	0.2057	1.7312
GYA	0.2057	3.4704	9.0366	0.2057	1.7249

* kısıtların ihlal edildiği çalışmalar

3.5.2. Gerilim-sıkıştırma yay tasarımı problemi



Şekil 3.9. Yay tasarımı örneği

Çok bilinen, bir diğer mühendislik problemi olan yay tasarımı probleminin amacı minimum ağırlıkta bir yay tasarımı oluşturmaktır (Arora, 1989). Problem üç adet tasarım değişkeninden (x_1, x_2, x_3) ve dört adet kısıttan (g_1, g_2, g_3, g_4) oluşmaktadır. Şekil 3.9'da tel çapı *d*, (x_1); ortalama bobin çapı *D*, (x_2); aktif rulo sayısı *N*, (x_3) ile temsil edilmektedir. Bu değişkenlerin alabileceği sınır değerleri sırasıyla $0.05 \leq x_1 \leq 1$, $0.25 \leq x_2 \leq 1.3$ ve $2 \leq x_3 \leq 15$'dir. Problemin amaç fonksiyonu eşitlik 3.19'da ve kısıtları eşitlik 20-23'de verilmiştir.

$$f(x) = (x_3 + 2)x_2x_1^2 \tag{3.19}$$

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \tag{3.20}$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leq 0 \tag{3.21}$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{3.22}$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{3.23}$$

Geliştirilmiş Yarasa Algoritması, Yarasa Algoritması (her iki yöntemde Çizelge 3.10'da verilen parametreler kullanılmıştır) ve Gandomi vd. (2013)'nin çalışması ile yay tasarımı problemi üzerinde karşılaştırmalı olarak test edilmiştir. Sonuçlar Çizelge 3.18'de verilmiştir. Ayrıca yöntemin (GYA) bu

problem üzerindeki başarısı literatürdeki çalışmalar ile tek tek kıyaslanarak sonuçları Çizelge 3.19'da verilmiştir.

Çizelge 3.18. GYA'nın yay tasarımı problemi üzerindeki performansının karşılaştırmalı değerleri

Metot	Minimum	Maksimum	Ortalama	Ortanca	Std. S.
YA	0.01300848	0.03123316	0.02183271	0.02386520	0.00732093
YA*	0.01266522	0.0168954	0.01350052	-	0.00142027
GYA	0.01266613	0.01777192	0.01298877	0.01273126	0.00092165

* Gandomi vd. (2013)

Çizelge 3.19. GYA'nın yay tasarımı problemi üzerindeki performansının literatür ile karşılaştırmalı değerleri

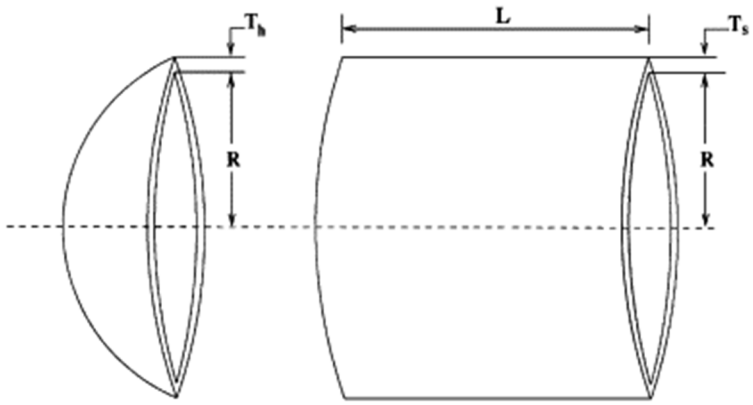
Araştırma	x_1	x_2	x_3	Maliyet
Arora (1989)*	0.0534	0.3991	9.1854	0.0127
Aragon vd. (2010)	0.0516	0.3551	11.384	0.0126
Barbosa ve Lemonge (2002)	0.0511	0.3443	12.070	0.0126
Belegundu (1982)	0.0500	0.3159	14.250	0.0128
Bernardino vd. (2007) *	0.0516	0.3560	11.329	0.0126
Bernardino vd. (2008)	0.0514	0.3505	11.661	0.0126
Coello (1999)	0.0514	0.3516	11.632	0.0127
Coello (2000)	0.0514	0.3516	11.632	0.0127
Coello ve Becerra (2004)	0.0500	0.3174	14.031	0.0127
Coello ve Montes (2001)	0.0519	0.3639	10.890	0.0126
Dos Santos Coelho (2010)	0.0515	0.3525	11.538	0.0126
He ve Wang (2007)	0.0517	0.3576	11.244	0.0126
He vd. (2004) *	0.0514	0.3513	11.608	0.0126
Hedar ve Fukushima (2006)	0.0517	0.3580	11.213	0.0126
Hsu ve Liu (2007) *	0.0523	0.3731	10.364	0.0126
Hu vd. (2003) *	0.0514	0.3513	11.608	0.0126
Huang vd. (2007)	0.0516	0.3547	11.410	0.0126
Montes ve Coello (2008)	0.0516	0.3553	11.397	0.0127
Montes ve Ocana (2008)	0.0518	0.3599	11.107	0.0126

Çizelge 3.19. GYA'nın yay tasarımı problemi üzerindeki performansının literatür ile karşılaştırmalı değerleri (Devam)

Parsopoulos ve Vrahatis (2005)	-	-	-	0.0131
Ray ve Liew (2003)	0.0521	0.3681	10.648	0.0126
Ray ve Saini (2001)	0.0504	0.3215	13.979	0.0130
Runarsson ve Yao (2000)	0.0516	0.3554	11.375	0.0126
Zhang vd. (2008)	0.0516	0.3567	11.289	0.0126
Sadollah vd. (2013)*	0.0516	0.3559	11.344	0.0126
Mahdavi vd. (2007)	0.0519	0.3639	10.890	0.0126
Gandomi vd. (2013)	0.0516	0.3567	11.288	0.0126
GYA	0.0519	0.3620	10.980	0.0126

* kısıtların ihlal edildiği çalışmalar

3.5.3. Basınçlı kap tasarımı problemi



Şekil 3.10. Basınçlı kap örneği

Her iki ucu yarı küresel kafa tarafından kapatılmış basınçlı kabın tasarımı Şekil 3.10'da verilmiştir. Bu tasarım probleminin amacı; şekillendirme, kaynaklama ve malzeme maliyeti dâhil toplam maliyet değerini minimize etmektir. Problem dört adet tasarım değişkeninden (x_1, x_2, x_3, x_4) oluşmaktadır. Şekil 3.10'da gövde kalınlığı T_s, (x_1); kafa kalınlığı T_h, (x_2); iç yarıçapı R, (x_3) ve kabın gövde bölümünün uzunluğu L, (x_4) ile gösterilmiştir. Tasarım değişkenlerinden gövde kalınlığı (x_1) ve kafa kalınlığı (x_2) değişkenleri ayrık ve 0.0625'in tam katları iken kafa kalınlığı (x_3) ve iç yarıçapı (x_4) değişkenleri süreklidir ve alabileceği değer aralıkları şu şekildedir: $1x0.0625 \leq x_1, x_2 \leq 99x0.0625$, $10 \leq x_3, x_4 \leq 200$.

Yarasa Algoritması sürekli değişkenli optimizasyon problemleri için uygundur ve optimizasyon sürecinde kullandığı tasarım değişkenleri reel değerler almaktadır. Basınçlı kap tasarımı problemindeki iki tasarım değişkeninin (x_1 ve x_2) sürekli olmayıp ayrık değişken olması nedeniyle algoritma, bu değişkenlerin optimizasyon süresi boyunca elde ettiği reel değerleri en yakın tamsayı değerlerine yuvarlayarak, kesikli değişkenli problemlere uyarlanmıştır.

Problemin amaç fonksiyonu ve kısıtları sırasıyla eşitlik 3.24 ve eşitlik 3.25-3.28'de verilmiştir.

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \quad (3.24)$$

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0 \quad (3.25)$$

$$g_2(x) = -x_2 + 0.00954x_3 \leq 0 \quad (3.26)$$

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \quad (3.27)$$

$$g_4(x) = -x_4 - 240 \leq 0 \quad (3.28)$$

Geliştirilen yöntem Yarasa Algoritması (her iki yöntemde Çizelge 3.10'da verilen parametreler kullanılmıştır) ve Gandomi vd. (2013)'nin çalışması ile basınçlı kap tasarımı problemi üzerinde karşılaştırmalı olarak test edilmiştir. Sonuçlar Çizelge 3.20'de verilmiştir. Ayrıca yöntemin (GYA) bu problem üzerindeki başarısı literatürdeki çalışmalar ile tek tek kıyaslanarak sonuçları Çizelge 3.21'de verilmiştir.

Çizelge 3.20. GYA'nın basınçlı kap tasarımı problemi üzerindeki performansının karşılaştırmalı değerleri

Metot	Minimum	Maksimum	Ortalama	Ortanca	Std. S.
YA	6412.35	7929.97	7168.06	7205.50	462.240
YA*	6059.71	6318.95	6179.13	-	137.223
GYA	6059.71	6370.77	6173.67	6090.52	142.334

* Gandomi vd. (2013)

Çizelge 3.21. GYA'nın basınçlı kap tasarımı problemi üzerindeki performansının literatür ile karşılaştırmalı değerleri

Araştırma	Minimum	Ortalama	Maksimum	Std. S.
Akhtar vd. (2002)	6171.00	6335.05	6453.65	-
Aragon vd. (2010)	6390.55	7694.06	6737.06	357.00
Barbosa ve Lemonge (2002)	6065.82	8248.00	6632.37	515.00
Bernardino vd. (2007)	6060.13	6845.49	6385.94	-
Bernardino vd. (2008)	6059.85	7388.16	6545.12	124.00
Cagnina vd. (2008)	6059.71	-	-	-
Cai ve Thierauf (1997)	7006.93	-	-	-
Cao ve Wu (1997)	7108.62	-	-	-
Coello (1999)	6228.74	-	-	-
Coello (2000)	6288.75	6293.84	6308.15	7.4133
Coello (2000)	6263.79	-	-	97.944
Coello ve Montes (2002)	6177.25	-	-	130.93
Coello ve Cortes (2004)	6061.12	6734.09	7368.06	457.99
Coello ve Montes (2001)	6059.95	6177.25	6469.32	130.93
Deb (1997)	6410.38	-	-	-
Dos Santos Coelho (2010)	6059.71	-	-	-
Fu vd. (1991)	8048.60	-	-	-
Ben ve Bean (1997)	6303.50	8065.66	10569.7	821.30
He ve Wang (2007)	6061.08	6147.13	6363.80	86.454
He vd. (2004)	6059.71	6289.93	-	305.78
Homaifar vd. (1994)	6295.11	8098.03	9528.07	831.69
Hu vd. (2003)*	6059.13	-	-	-
Huang vd. (2007)	6059.73	6085.23	6371.05	43.013
Joines ve Houck (1994)	6273.28	8092.87	10382.1	1017.9
Kannan ve Kramer (1994)*	7198.04	-	-	-
Lemonge ve Barbosa (2004)	6060.18	-	-	-
Li ve Chang (1998)	7127.30	-	-	-
Li ve Chou (1993)	7127.30	-	-	-
Litinetski ve Abramzon (1998)	7197.70	-	-	-
Michalewicz ve Attia (1994)	6572.62	8164.56	9580.51	789.65

Çizelge 3.21. GYA'nın basınçlı kap tasarımı problemi üzerindeki performansının literatür ile karşılaştırmalı değerleri (Devam)

Montes ve Coello (2008)	6059.75	6850.00	7332.88	426.00
Montes ve Ocana (2008)*	6059.73	6081.78	6150.13	67.241
Montes vd. (2007)*	6059.70	6059.70	-	0
Parsopoulos ve Vrahatis (2005)	6544.27	9032.55	11638.2	995.57
Ray ve Liew (2003)	6171.00	6335.05	-	-
Runarsson ve Yao (2000)	6832.58	8012.61	7187.31	267.00
Sandgren (1988)	7980.89	-	-	-
Sandgren (1990)	8129.10	-	-	-
Shih ve Lai (1995)	7462.10	-	-	-
Tsai vd. (2002)	7079.04	-	-	-
Wu ve Chow (1995)	7207.49	-	-	-
Yun (2005)	7198.42	-	-	-
Zahara ve Kao (2009)*	5930.31	5946.79	5960.06	9.1614
Zhang ve Wang (1993)	7197.70	-	-	-
Datta ve Figueira (2011)*	5850.38	-	-	-
Sadollah vd. (2013)*	5889.32	6200.64	6392.50	160.34
Mahdavi vd.(2007)*	5849.76	-	-	-
Mashinchi vd. (2011)	6059.71	-	-	-
Gandomi vd. (2013)	6059.71	6179.13	6318.95	137.22
GYA	6059.71	6173.67	6370.77	142.33

* kısıtların ihlal edildiği çalışmalar

4. TARTIŞMA ve SONUÇLAR

Bu çalışmada Yarasa Algoritmasının global ve lokal arama karakteristikleri 3 farklı yöntemle güçlendirilmiştir. Geliştirilen yöntemin performansının teyit edilmesi amacıyla, sürekli değişkenli kısıtsız standart (Karaboga ve Akay, 2009) ve karmaşık (Suganthan vd., 2005) test fonksiyonları ile sürekli ve süreksiz değişkenli kısıtlı 3 adet gerçek hayat problemleri (kaynaklı kiriş tasarımı, gerilim-sıkıştırma yay tasarımı ve basınçlı kap tasarımı) kullanılmıştır. Standart ve karmaşık benchmark test fonksiyonları üzerinde Geliştirilmiş Yarasa Algoritması, standart Yarasa Algoritması ile sırasıyla bölüm 3.3.2 ile bölüm 3.4.2'de, gerçek hayat problemleri üzerinde Geliştirilmiş Yarasa Algoritması literatürdeki çalışmalar ile bölüm 3.5'de kıyaslanmıştır.

Bu bölümde sırasıyla, algoritmaya uygulanan her bir geliştirme yapısının algoritmaya olan katkısı; GYA ve YA'nın standart ve karmaşık fonksiyonlar üzerindeki karşılaştırmaları; kısıtlı, gerçek hayat problemleri üzerinde GYA'nın YA'ya ve özellikle literatürde belirtilen çalışmalara oranla ne kadar başarılı olduğu detaylı bir şekilde analiz edilerek yorumlanmıştır.

4.1. Geliştirme Yapılarının Analiz Edilmesi

Yarasa Algoritması üzerinde uygulanan her bir geliştirme yapısının verimliliğinin ayrı ayrı test edilmesi amacıyla Çizelge 3.1'de verilen fonksiyonlar kullanılmıştır. Bu çizelgede yer alan 1, 2, 6, 7, 8 numaralı fonksiyonlar unimodal iken, 3, 4, 5, 9, 10 numaralı fonksiyonlar ise multimodaldir. Çizelge 3.8'de bu fonksiyonlar üzerinde elde edilen sonuçlar verilmiştir.

Sonuçların daha anlamlı analiz edilebilmesi için 10, 30, 50 ve 2, 5, 10 boyutlarında test edilen fonksiyon sonuçlarının ortalaması esas alınmıştır. Ortalama değerler Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Geliştirme yapılarının 10, 30, 50 ve 2, 5, 10 boyutlarındaki ortalama değerleri

No	**YA**	**G1**	**G2**	**G3**
1*	5.08e+03	1.57e+02	3.88e+03	6.66e-03
2*	3.60e+05	9.41e+01	1.71e+05	4.38e+01
3+	7.18e-00	5.00e-00	6.49e-00	7.20e-01
4+	6.09e+01	3.76e-00	5.49e+01	3.36e-01
5+	1.32e+02	1.29e+02	1.14e+02	1.02e+02
6*	3.41e+02	1.04e-00	3.06e+02	1.03e-02
7*	5.83e+03	2.41e+03	4.61e+03	2.78e-00
8*	1.61e+02	6.64e-01	8.50e+01	1.08e+02
9+	-2.71e-01	-3.56e-01	-3.03e-01	-3.56e-01
10+	-3.71e-00	-4.55e-00	-3.79e-00	-4.64e-00
No	**G1-G2**	**G1-G3**	**G2-G3**	**G1-G2-G3(GYA)**
1*	5.58e+01	1.54e-05	6.43e-03	1.34e-05
2*	4.04e+01	2.75e+01	5.03e+01	2.79e+01
3+	3.86e-00	9.71e-01	7.45e-01	6.62e-01
4+	6.64e-00	5.71e-01	2.16e-01	3.84e-01
5+	1.00e+02	1.25e+02	5.91e+01	6.31e+01
6*	4.50e-01	1.42e-04	1.06e-02	6.40e-05
7*	1.41e+03	3.64e-00	2.47e-00	2.83e-00
8*	6.68e-01	7.05e-01	6.85e+01	7.21e-01
9+	-3.89e-01	-4.45e-01	-4.31e-01	-4.45e-01
10+	-4.79e-00	-4.44e-00	-4.93e-00	-4.78e-00

* unimodal, + multimodal

- Her bir geliştirme yapısı (G1, G2 ve G3) tüm fonksiyonlar (unimodal ve multimodal) üzerinde standart Yarasa Algoritmasından daha iyi sonuçlar üretmiştir.

- Şekil 3.3 ve Çizelge 4.1'Çizelge den anlaşılacağı üzere G3 geliştirme yapısı (Yabani Ot Optimizasyonu hibrit yapısı), 8 numaralı fonksiyon (Dixon-Price) dışındaki tüm unimodal fonksiyonlar üzerinde G1 (atalet ağırlığı modifikasyonu) ve G2 (global arama modifikasyonu) geliştirme

yapılarına göre daha iyi sonuç elde etmiştir. Bu fonksiyonlar üzerinde G3'den sonra sırasıyla G1 ve G2 yapıları etkili olmuştur. 8 numaralı fonksiyon üzerinde ise sırasıyla G1, G2 ve G3 geliştirme yapıları daha etkin performans göstermektedir.

- Multimodal fonksiyonlar üzerinde elde edilen sonuçlar analiz edildiğinde G3 geliştirme yapısı; 3, 4, 5 ve 10 numaralı fonksiyonlarda en üstün performans göstermekte, 9 numaralı fonksiyon üzerinde G1 geliştirme yapısı ile aynı sonucu üretmektedir. Beklendiği üzere G2 geliştirme yapısının multimodal fonksiyon gruplarında elde ettiği optimizasyon performansı unimodal fonksiyonlar üzerinde elde ettiği başarıdan daha fazladır.

- G1, G2 ve G3 yapılarının farklı kombinasyonlar halinde uygulandığı durumlarda unimodal fonksiyonlar üzerinde en iyi başarıyı G1-G3; multimodal fonksiyonlar üzerinde ise G2-G3 göstermektedir.

4.2. Algoritmaların Standart Fonksiyon Kümesi Üzerindeki Performanslarının Analizi

YA ve GYA'nın unimodal ve multimodal standart benchmark test fonksiyonları (bkz. Çizelge 3.9) üzerindeki performansları bölüm 3.3'te test edilmiştir.

Her iki algoritmanın bu test fonksiyon kümesi üzerindeki performanslarının daha anlamlı yorumlanabilmesi için her bir fonksiyon üzerinde elde edilen ortalama ve standart sapma değerleri kullanılarak *t-test* uygulanmış ve her bir optimizasyon işlemi için t değeri hesaplanmıştır (Eşitlik 3.1). Hesaplanan t değerine göre ilgili fonksiyon üzerinde hangi yöntemin daha etkili olduğu Çizelge 3.11'de belirtilmiştir.

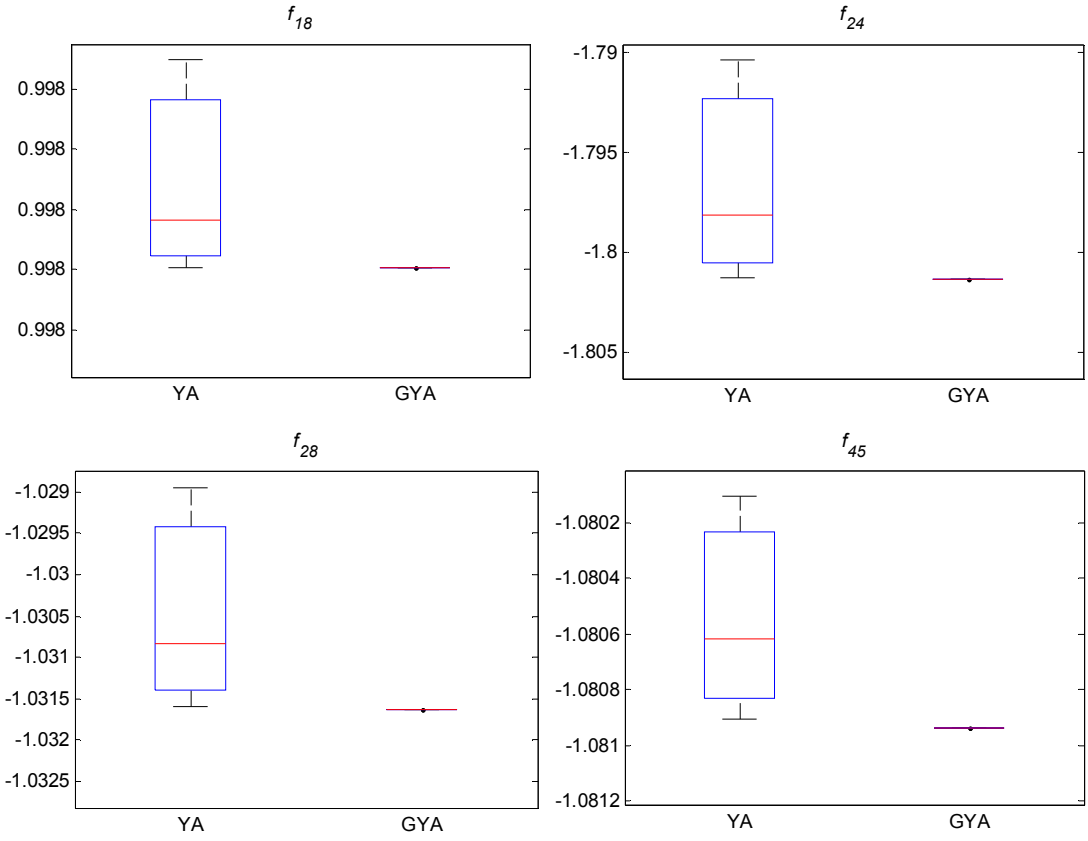
YA ve GYA, Çizelge 3.9'da verilen 50 adet benchmark test fonksiyonlarını farklı boyutlar ile (d =2, 3, 4, 5, 6, 10, 24, 30, 50) toplamda 76 defa optimize etmiştir. t değerlerinden anlaşılacağı üzere 76 defa uygulanan optimizasyon işlemlerinin

71'inde GYA, YA'ya oranla daha başarı iken, 5'inde ise her iki algoritma birbirine üstünlük sağlayamamıştır. Yüzdelik olarak gösterildiğinde GYA optimizasyon işlemlerinin %93.4'ünde YA'ya oranla daha başarılı olmuştur.

Bir problemin boyutu arttıkça onu optimize eden metodun performansı giderek zayıflamaktadır, bu durumun 2 nedeni vardır. Birinci neden, problemin çözüm uzayı problemin boyutu arttıkça genişlemektedir ve kısıtlı süre içinde umut vadeden tüm bölgelerin aranması için daha etkili stratejiler gerekmektedir. İkinci neden, boyut artışıyla birlikte problemin karakteristiği de değişmektedir. Örneğin Rosenbrock fonksiyonu 2 boyutta unimodal iken boyut sayısı artırıldıkça multimodal yapıya dönüşmektedir (Táng vd. 2010). Bu bilgiler çerçevesinde, Çizelge 3.11'de yer alan sonuçlara bakıldığında, genel olarak boyut sayısı arttığında YA ve GYA'nın performanslarının zayıfladığı görülebilmektedir. Örneğin YA ve GYA; 2, 3, 4, 5, 12, 14, 15, 16, 17, 22, 42 numaralı fonksiyonların d =10, 30, 50 boyutları üzerinde test edilmiş ve bu fonksiyonlar üzerindeki en iyi performansı 10 boyutlu; en kötü performansı ise 50 boyutlu optimizasyon işlemlerinde sergilemiştir.

Çizelge 3.11'deki *minimum* değerlerine bakıldığında, 76 optimizasyon işleminin 75'i dikkate alındığında (Benchmark test kümesi içinde yer alan 47 numaralı 10 boyutlu Langerman fonksiyonunun optimum uygunluk değerinin bilinmediğinden bu fonksiyon değerlendirme dışında tutulmuştur) YA ve GYA sırasıyla 9 ve 21 defa global optimum değerlerini yakaladığı görülmektedir. Bu değerler yüzdelik cinsten gösterilecek olursa, YA ve GYA, fonksiyonların global optimum değerlerini bulmada sırasıyla %12 ve %36 başarılı olmuştur.

YA ve GYA; 18, 24, 28 ve 45 numaralı fonksiyonlarda aynı ortalama değerlerini yakalamayı başarmışlardır; ancak bu fonksiyonlar üzerinde elde edilen standart sapma değerleri göz önüne alındığında GYA'nın ürettiği standart sapma değerleri daha küçük olduğundan GYA'nın, YA'ya göre bu fonksiyonlarda daha başarılı olduğu kabul edilebilmektedir. Şekil 4.1'de verilen bu fonksiyonlara ait istatistik değerleri de bu durumu kanıtlamaktadır.



Şekil 4.1. 18, 24, 28 ve 45 numaralı fonksiyonların en iyi, en kötü ve ortalama değerlerinin grafiksel karşılaştırmalı

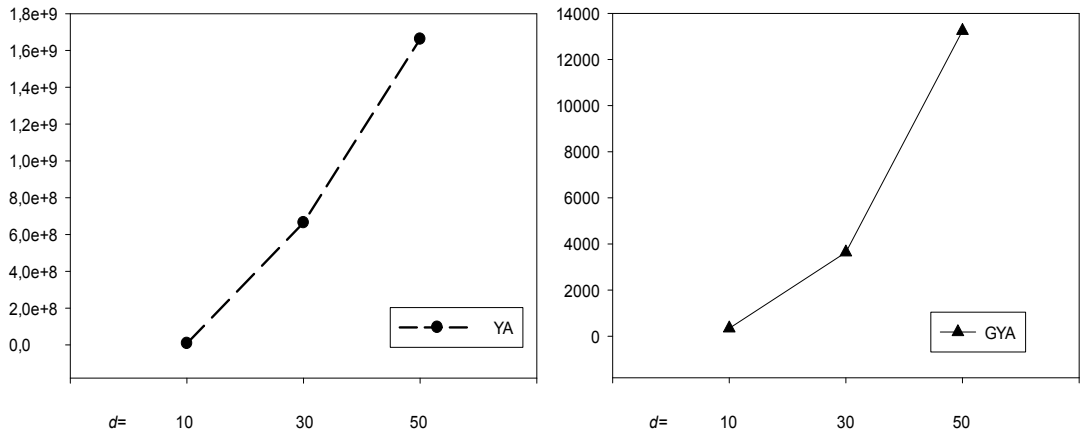
4.3. Algoritmaların Karmaşık Fonksiyon Kümesi Üzerindeki Performanslarının Analizi

YA ve GYA'nın performansları, unimodal ve multimodal standart benchmark test fonksiyonları dışında, optimize edilmesi daha zor olan, karmaşık test fonksiyonları üzerinde farklı boyutlar ile (d =10, 30, 50) bölüm 3.4'te test edilmiş sonuçlar Çizelge 3.13'te verilmiştir. Test aşamasında kullanılan 25 adet karmaşık fonksiyonlar karakteristik yapılarına göre 4 farklı gruba ayrılmıştır: unimodal temel fonksiyonlar (1-5), multimodal temel fonksiyonlar (6-12), genişletilmiş fonksiyonlar (13,14) ve hibrit fonksiyonlar (15-25) (bkz. Çizelge 3.12).

YA ve GYA, 25 adet karmaşık fonksiyon kümesine, farklı boyutlarda toplamda 75 defa optimizasyon işlemi uygulamıştır. Çizelge 3.13'teki t değerinden de anlaşılacağı üzere 75 optimizasyon işleminin 67'sinde GYA, YA'ya oranla daha

başarılı olmuştur. 8 optimizasyon işleminde ise YA ve GYA birbirine üstünlük sağlayamamıştır. Yüzdelik olarak ifade etmek gerekirse, GYA optimizasyon işlemlerinin %89.3'ünde YA'ya göre daha iyi performans sergilemiştir. %10.7'sinde ise her iki yöntem birbirine üstünlük kuramamıştır. Fonksiyonların karakteristik yapıları üzerinde GYA, YA ile kıyaslandığında optimizasyon işlemlerinin; unimodal temel fonksiyonlarda %100'ünde, multimodal temel fonksiyonlarda %95.2'sinde, genişletilmiş fonksiyonlarda %100'ünde ve hibrit fonksiyonlarda %78.7'sinde daha başarılı olduğu görülmektedir.

YA ve GYA'nın performansları, standart benchmark test fonksiyonlarında olduğu gibi karmaşık benchmark test fonksiyonlarında da, fonksiyon boyutu arttıkça azalış göstermektedir. 25 fonksiyonun 10, 30 ve 50 boyut üzerinden elde edilen değerlerinin ortalaması alınarak oluşturulan Şekil 4.2, YA ve GYA'nın bu boyutlardaki performanslarını gözler önüne sermektedir.



Şekil 4.2. YA ve GYA'nın karmaşık fonksiyonlar üzerinde 10, 30 ve 50 boyutta elde ettiği ortalama performans

Çizelge 4.2'de GYA'nın YA'ya göre test fonksiyonları üzerinde elde ettiği başarı yüzdeleri (BY) verilmiştir. Başarı yüzdesi eşitlik 4.1'de verilen denkleme göre hesaplanmıştır.

$$BY = 100 - \frac{100\left(\overline{f_{GYA}^{p}} - f_{best}^{p}\right)}{\overline{f_{YA}^{p}} - f_{best}^{p}} \qquad (4.1)$$

Yukarıdaki eşitlikte $\overline{f_{GYA}^{p}}$ ve $\overline{f_{YA}^{p}}$ sırasıyla GYA ve YA'nın p fonksiyonu üzerinde ortalama uygunluk değerlerini belirtmektedir. f_{best}^{p}, p fonksiyonunun optimum uygunluk değerini göstermektedir.

Çizelge 4.2. GYA'nın YA'ya kıyasla karmaşık fonksiyonları optimize etme performansının yüzdelik gösterimi

p	BY	p	BY
1	100.00	14	6.07
2	100.00	15	18.75
3	99.99	16	15.26
4	99.95	17	28.51
5	87.56	18	12.36
6	100.00	19	8.96
7	100.00	20	8.28
8	1.64	21	31.26
9	52.58	22	13.82
10	60.62	23	26.77
11	24.96	24	57.78
12	80.77	25	71.44
13	71.82		

Yukarıdaki çizelgeden anlaşılacağı üzere GYA, YA ile kıyaslandığında karmaşık fonksiyonların tamamını daha başarılı bir şekilde optimize etmeyi başarmıştır. GYA'nın başarı performansı %1.64 ile %100 aralığında değişkenlik göstermektedir. GYA'nın tüm fonksiyonlar üzerindeki yüzdelik başarı performanslarının ortalaması %51.17 olarak hesaplanmaktadır. Başarı yüzdeleri karmaşık fonksiyonların karakteristiklerine (bkz. Çizelge 3.12) göre gruplandırılarak yorumlandığında GYA, YA'ya oranla; unimodal temel fonksiyonlar üzerinde ortalama %97.50, multimodal temel fonksiyonlarda %60.08, genişletilmiş fonksiyonlarda %38.95 ve hibrit fonksiyonlarda %26.65 daha başarılı optimizasyon performansı sergilemiştir. Görüldüğü üzere karmaşık fonksiyonların karakteristik yapıları zorlaştıkça, optimize edilmesi de giderek zor hale gelmektedir.

4.4. Algoritmaların Gerçek Hayat Problemleri Üzerindeki Performanslarının Analizi

Geliştirilmiş Yarasa Algoritmasının eşitlik ve eşitsizlik kısıtları içeren 3 adet gerçek hayat problemi üzerinde performansı ölçülmüştür. Her bir problem üzerinde GYA'nın performansı, standart YA ve literatürdeki çalışmalar ile kıyaslanmıştır.

4.4.1. Kaynaklı kiriş tasarımı problemi üzerinde performans analizi

GYA'nın kısıtlı kaynaklı kiriş tasarımı probleminin çözümündeki optimizasyon performansı standart YA ile kıyaslanmış sonuçlar Çizelge 3.16'da gösterilmiştir. Çizelge 3.16'daki *minimum*, *maksimum* ve *ortalama* değerleri üzerinden kıyaslama yapıldığında GYA'nın, YA ve Gandomi vd. (2013)'e göre daha başarılı olduğu görülmektedir.

GYA'nın bu problem üzerindeki performansı ayrıca literatürdeki 30 adet çalışma ile kıyaslanmış, sonuçlar Çizelge 3.17'de verilmiştir. Sonuçlardan da anlaşılacağı üzere GYA, kaynaklı kiriş tasarımı maliyeti açısından literatürdeki çalışmalara göre daha uygun değeri üretmiştir. Hwang ve He (2006) çalışmasında kısıtlara dikkat edilmemesine rağmen GYA'dan daha kötü performans sergilediği görülmektedir. Şekil 4.3'te GYA ile literatürdeki çalışmaların (kısıtların ihlal edildiği çalışmalar göz ardı edilmiştir) kaynaklı kiriş problemi üzerinde elde ettikleri maliyet değerlerinin karşılaştırmalı gösterimi verilmiştir.

4.4.2. Gerilim-sıkıştırma yay tasarımı problemi üzerinde performans analizi

GYA'nın gerilim-sıkıştırma yay tasarımı problemi üzerindeki performansı YA ile kıyaslanmış ve sonuçlar Çizelge 3.20'de gösterilmiştir. Çizelge 3.20'deki *minimum*, *maksimum* ve *ortalama* değerlerine bakıldığında GYA, YA'ya göre daha iyi optimizasyon performansı sergilemiştir. Ancak Gandomi vd. (2013) ile

kıyaslandığında GYA *minimum* ve *ortalama* değer açısından daha iyi sonuç üretirken *maksimum* değer açısından daha kötü performans sergilemektedir.

GYA dışında bu problem ayrıca literatürdeki 27 adet çalışma ile karşılaştırılmış, optimizasyon sonuçları Çizelge 3.19'da verilmiştir. Çizelge 3.19'da görüldüğü üzere literatürdeki 27 çalışmanın 6 tanesinde kısıtlar göz önünde bulundurulmamıştır. Çizelge 3.19'da verilen sonuçlara bakıldığında GYA literatürdeki 14 çalışma ile aynı değeri üretmiştir (kısıtları aşan çalışmalar göz önünde bulundurulmamıştır). Şekil 4.4'te GYA ile literatürdeki çalışmaların (kısıtların ihlal edildiği çalışmalar göz ardı edilmiştir) gerilim-sıkıştırma yay tasarımı problemi üzerinde elde ettikleri maliyet değerlerinin karşılaştırmalı gösterimi verilmiştir.

4.4.3. Basınçlı kap tasarımı problemi üzerinde performans analizi

Diğer sürekli değişkenli, kısıtlı problemlerin tersine basınçlı kap tasarımı problemi sürekli ve süreksiz değişkenli, kısıtlı bir gerçek hayat problemidir. Bölüm 3.5.3'de de anlatıldığı üzere bu çalışmada YA ve GYA bu tür problemlerin optimize edilmesi için uyarlanmıştır. GYA'nın performansı basınçlı kap tasarımı problemi üzerinde öncelikle Yarasa Algoritması ve Gandomi vd. (2013) ile karşılaştırılmış ve sonuçlar Çizelge 3.20'de verilmiştir.

Çizelge 3.20'deki *minimum*, *maksimum* ve *ortalama* değerlere bakıldığında GYA, YA'ya göre daha iyi performans sergilediği görülmektedir. Ancak Gandomi vd. (2013)'nin çalışması ile kıyaslandığında GYA; *minimum* değer bazında eşit, *maksimum* değer bazında daha kötü sonuç üretmektedir. Ancak *ortalama* değere bakıldığında GYA daha iyi sonuç vermektedir. Sonuç olarak GYA'nın bu problem üzerinde Gandomi vd. (2013)'nin çalışmasından daha iyi olduğu söylenebilmektedir.

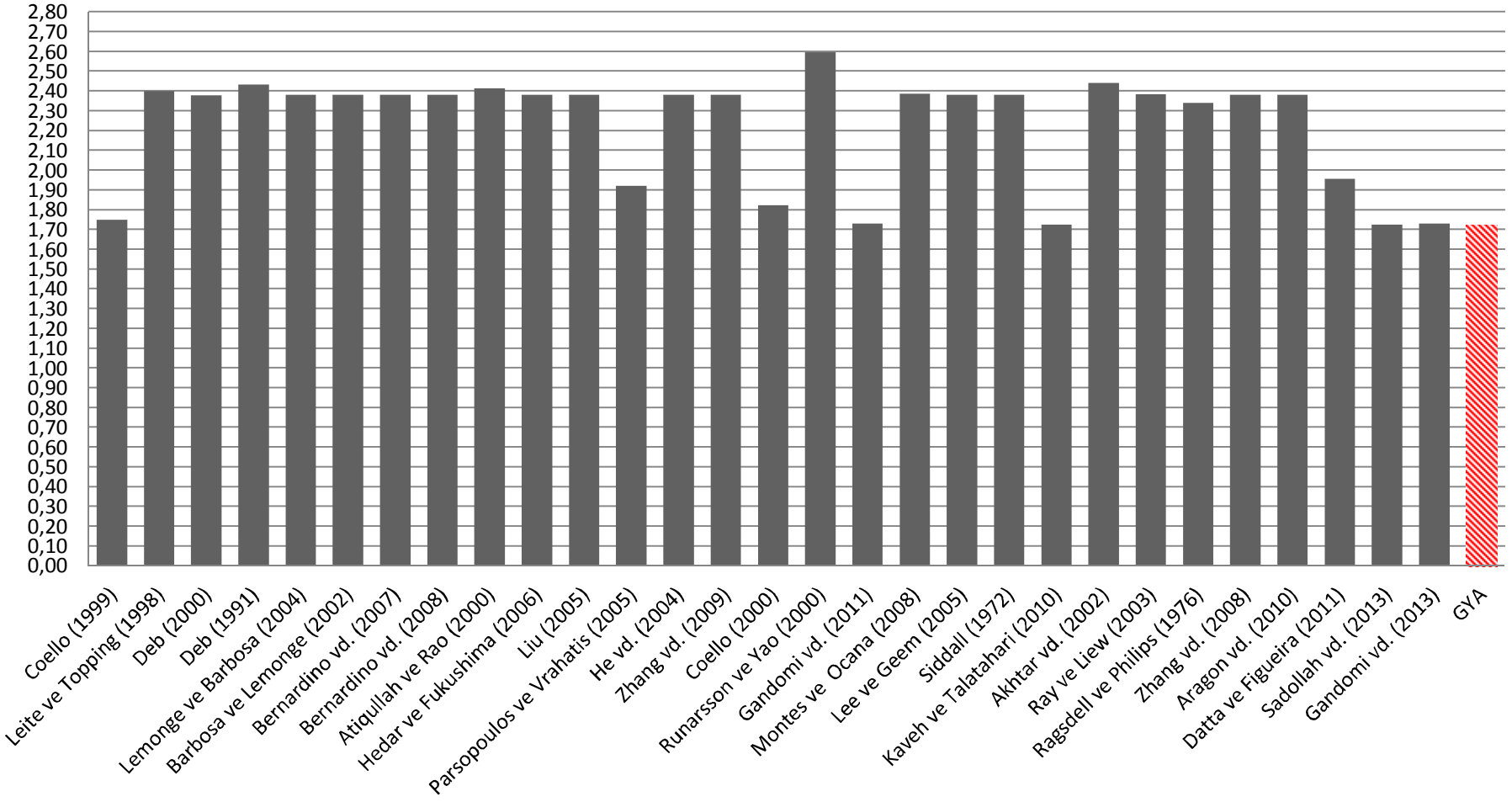
GYA'nın performansı, YA ve Gandomi vd. (2013)'nin çalışmasının dışında literatürdeki 50 adet çalışma ile de kıyaslanmış ve sonuçlar Çizelge 3.21'de

verilmiştir. Çizelge 3.21'den de anlaşılacağı üzere literatürdeki 50 adet çalışmanın 8 tanesinin kısıtları ihlal ettiği görülmektedir.

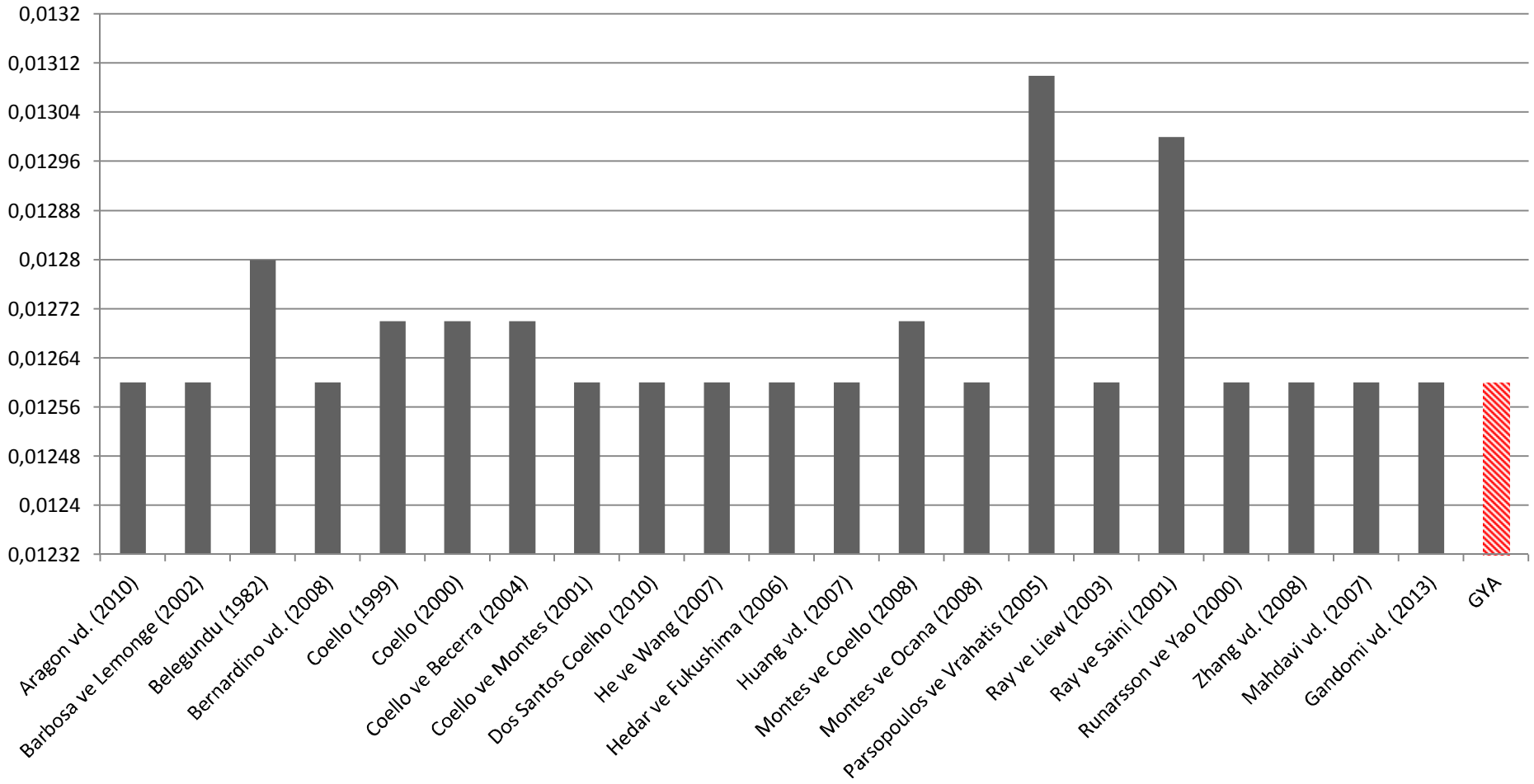
Mahdavi vd. (2007), Datta ve Figuaira. (2011), Sadollah vd. (2013), Zahara ve Kao. (2009), Hu vd. (2003) ve Montes vd. (2007)'nin çalışamaları basınçlı kap problemi üzerinde GYA'dan daha iyi sonuç üretmiştir. Ancak tüm bu çalışmalar problemde verilen kısıtları ihlal etmiştir. Ayrıca bu çalışmaların bulduğu minimum maliyet değerlerinin, problemin minimum maliyet değerinden daha küçük olması bu çalışmaların kısıtları ihlal ettiğini açıkça kanıtlamaktadır. Kısıtların ihlal edildiği çalışmalar göz ardı edildiğinde, GYA'nın literatürde kıyaslanan 5 adet çalışma ile birlikte en uygun değeri ürettiği görülmektedir. Şekil 4.5'te GYA ile literatürdeki çalışmaların (kısıtların ihlal edildiği çalışmalar göz ardı edilmiştir) basınçlı kap tasarımı problemi üzerinde elde ettikleri maliyet değerlerinin karşılaştırmalı gösterimi verilmiştir.

Bu tez çalışmasında, Yarasa Algoritmasının eksikliklerinden olan lokal ve global arama kabiliyetleri bazı metotlar ile geliştirilmiş ve geliştirilmiş olan algoritmanın verimliliği sürekli değişkenli, kısıtsız standart ve karmaşık benchmark test fonksiyonları üzerinde sınanmış, bu fonksiyon kümeleri üzerinde algoritmanın geliştirilmiş versiyonunun, standart versiyonuna göre daha başarılı olduğu kanıtlanmıştır. Ayrıca Geliştirilmiş Yarasa Algoritmasının kesikli değişkenli ve kısıtlı gerçek hayat problemleri üzerindeki performansı, literatürde çok iyi bilinen, araştırmacılar tarafından sıklıkla kullanılan üç adet mühendislik dizayn problemleri üzerinde, literatürdeki çalışmalar ile karşılaştırılmış ve Geliştirilmiş Yarasa Algoritmasının, kısıtlar barındıran gerçek hayat problemleri üzerinde literatürdeki birçok yöntemden daha etkili sonuçlar ürettiği görülmüştür.

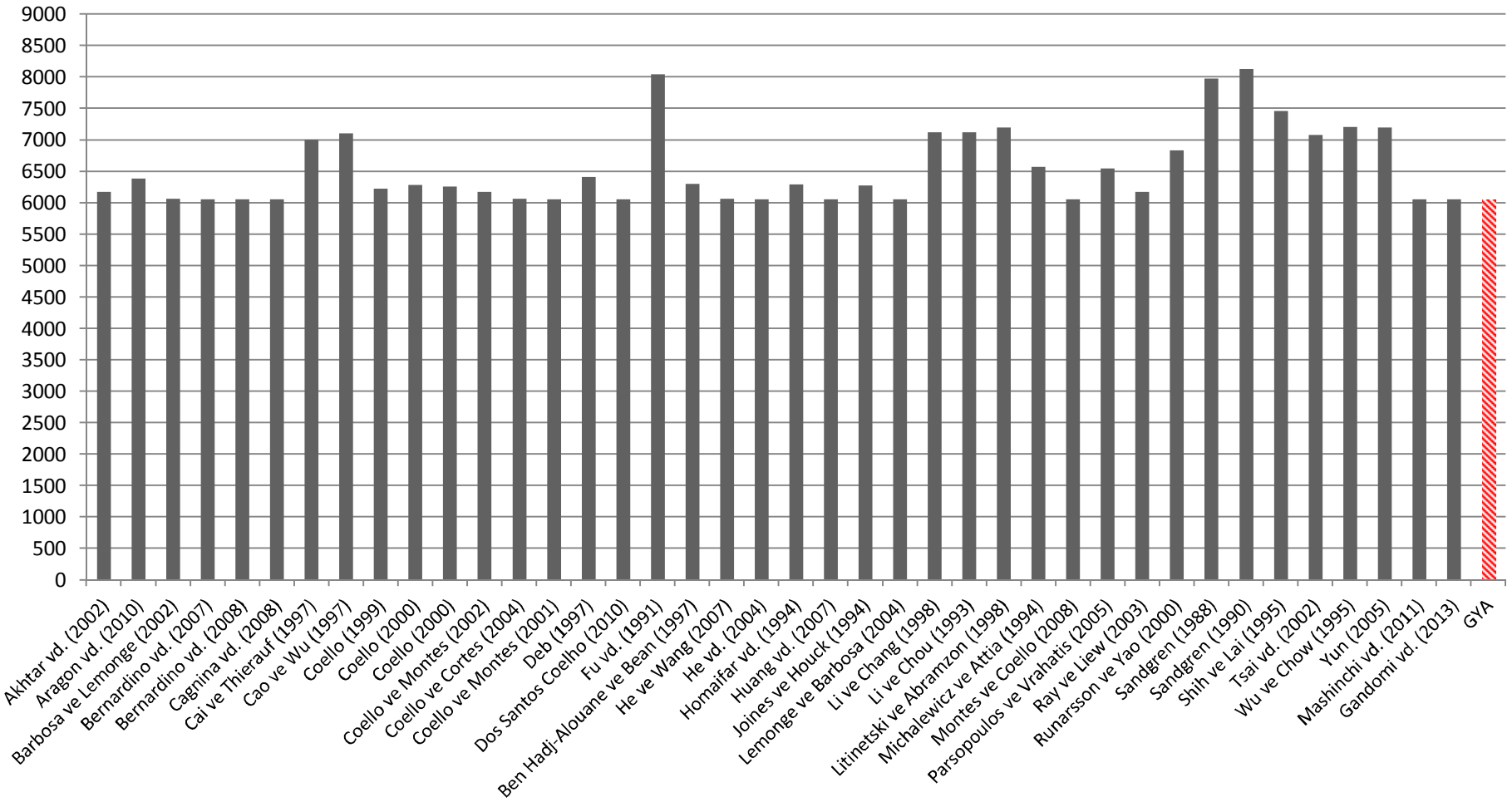
Gelecekteki çalışmalarda; Geliştirilmiş Yarasa Algoritmasının, *Gerçek Zamanlı Ters Sarkaç* ve *XOR (Exclusive Or)* gibi sistem tanılama problemleri üzerindeki performansının araştırılması planlanmaktadır.



Şekil 4.3. Kaynaklı kiriş tasarımı problemi üzerinde GYA'nın literatür ile kıyaslanması sonucu elde edilen maliyet değerlerinin gösterimi



Şekil 4.4. Gerilim-sıkıştırma yay tasarımı problemi üzerinde GYA'nın literatür ile kıyaslanması sonucu elde edilen maliyet değerlerinin gösterimi



Şekil 4.5. Basınçlı kap tasarımı problemi üzerinde GYA'nın literatür ile kıyaslanması sonucu elde edilen maliyet değerlerinin gösterimi

KAYNAKLAR

Akhtar, S., Tai, K., Ray T., 2002. A Socio-Behavioural Simulation Model for Engineering Design Optimization. Engineering Optimization, 34(4), 341–354.

Akhtar, S., Ahmad, A.R., Abdel-Rahman, E. M., 2012. A Metaheuristic Bat-Inspired Algorithm for Full Body Human Pose Estimatio. Computer and Robot Vision (CRV), 2012 Ninth Conference, 27-30 May., Toronto, 369-375.

Akyol, S., Alataş, B., 2012. Güncel Sürü Zekâsı Optimizasyon Algoritmaları. Nevşehir Üniversitesi Fen Bilimleri Enstitü Dergisi, 1(1), 36-50.

Alatas, B., 2011. Artificial Chemical Reaction Optimization Algorithm for global optimization. Expert Systems with Applications, 38(10), 13170-13180.

Aragon, V.S., Esquivel, S.C., Coello, C.A.C., 2010. A modified version of a T-Cell Algorithm for constrained optimization problems. International Journal for Numerical Methods in Engineering, 84(3), 351-378.

Arora, J. (Ed.), 1989. Introduction to Optimum Design. Elsevier Science,728s.

Atiqullah, M.M., Rao, S.S., 2000. Simulated Annealing and Parallel Processing: An Implementation for Constrained Global Design Optımızation. Engineering Optimization, 32(5), 659-685.

Banu, F., Chandrasekar, C., 2013. An Optimized Approach of Modified Bat Algorithm to Record Deduplication. International Journal of Computer Applications, 62(1), 10-15.

Barbosa, H.J.C., Lemonge, A.C.C., 2002. An Adaptive Penalty Scheme in Genetic Algorithms For Constrained Optimiazation Problems. Hitoshi, I. (Ed.), Frontiers in Evolutionary Robotics (9-38), I-Tech Education and Publishing, 596s, China.

Baziar, A., Abdollah, K.F., Jafar, Z., 2013. A Novel Self Adaptive Modification Approach Based on Bat Algorithm for Optimal Management of Renewable MG. Journal of Intelligent Learning Systems and Applications, 5(1), 11-18.

Belegundu, A.D. (Ed.), 1982. A Study of Mathematical Programming Methods for Structural Optimization. University of Iowa, 724s.

Ben, A., Bean, J.C., 1997. A Genetic Algorithm for the Multiple-Choice Integer Program. Operations Research, 45(1), 92-101.

Bernardino, H.S., Barbosa, I.J.C., Lemonge, A., 2007. A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. Evolutionary Computation: Congress on Computational Intelligence, 25-28 Sept., Singapore, 646-653.

Bernardino, H.S., Barbosa, I.J.C., Lemonge, A., Fonseca, L.G., 2008. A new hybrid AIS-GA for constrained optimization problems in mechanical engineering. Evolutionary Computation: IEEE World Congress on Computational Intelligence, 1-6 June, Hong Kong, 1455-1462.

Biswal, S., Barisal, A.K., Behera, A., Prakash, T., 2013. Optimal power dispatch using BAT algorithm. Energy Efficient Technologies for Sustainability (ICEETS): 2013 International Conference on, 10-12 April, Nagercoil, 1018-1023.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv., 35(3), 268-308.

Bora, T., Coelho, L., Luiz, L., 2012. Bat-Inspired Optimization Approach for the Brushless DC Wheel Motor Problem. Magnetics, IEEE Transactions on, 48(2), 947-950.

Cagnina, L.C., Esquivel, S.C., Coello, C.A.C., 2008. Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica. 32(3), 319–326.

Cai, J., Thierauf, G., 1997. Evolution strategies in engineering. Engineering Optimization, 29(1), 177-199.

Cao, Y.J., Wu, Q.H., 1997. Evolutionary programming. Evolutionary Computation: IEEE International Conference on, 13-16 April, Indianapolis, IN., 443-446.

Chong, E.K.P., Zak, S.H. (Ed.), 2013. An Introduction to Optimization (4th Edition). Wiley, 640s.

Coello, C.A.C., 1999. Self-adaptive penalties for GA-based optimization. Evolutionary Computation: Proceedings of the 1999 Congress on, 6-9 July, Washington DC., 573-580.

Coello, C.A.C., 2000. Constraint-Handling using an Evolutionary Multiobjective Optimization Technique. Civil Engineering and Environmental Systems, 17(4), 319-346.

Coello, C.A.C., 2000. Use of a self-adaptive penalty approach for engineering optimization. Computers in Industry, 41(2), 113-127.

Coello, C.A.C., Montes, E.M., 2001. Use of dominance-based tournament selection to handle constraints in genetic algorithms. In Intelligent Engineering Systems through Artificial Neural Networks (ANNIE2001), 177-182.

Coello, C.A.C., Montes, E.M., 2002. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics, 16(3), 193–203.

Coello, C.A.C., Becerra, R.L., 2004. Efficient evolutionary optimization through the use of a cultural algorithm. Engineering Optimization, 36(2), 219-236.

Coello, C.A.C., Cortes, N.C., 2004. Hybridizing a genetic algorithm with an artificial immune system for global optimization. Engineering Optimization, 36(5), 607-634.

Datta, D., Figueira, J.R., 2011. A real-integer-discrete-coded particle swarm optimization for design problems. Applied Soft Computing, 11(4), 3625–3633.

Deb, K., 1991. Optimal design of a welded beam via genetic algorithms. AIAA Journal, 29(11), 2013-2015.

Deb, K., 1997. GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. Dasgupta, D., Michalewicz, Z. (Ed.), Evolutionary Algorithms in Engineering Applications (497-514), Springer Berlin Heidelberg, 555s.

Deb, K., 2000. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186(2-4), 311-338.

Dorigo, M., Caro D.G., 1999. The ant colony optimization meta-heuristic. Corne, D., Dorigo, M. Glover, F., Dasgupta, D. Moscato, P., Poli, R., Price, K.V. (Ed.), New Ideas in Optimization (11-32), McGraw-Hill Ltd., 493s.

Dorigo, M., Birattari, M., Stützle, T., 2006. Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique. IEEE Computational Intelligence Magazine, 1(4), 28-39.

Dos-Santos, C.L., 2010. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Systems with Applications, 37(2), 1676-1683.

Fan, S.K.S., Chiu, Y.Y., 2007. A decreasing inertia weight particle swarm optimizer. Engineering Optimization, 39(2), 203-228.

Feng, Y. Gui-fa T., Ai-Xin W., Yong-Mei Y., 2007. Chaotic Inertia Weight in Particle Swarm Optimization. Innovative Computing, Information and Control: ICICIC '07. Second International Conference on, 5-7 Sept., Kumamoto, 475-475.

Fenton, M.B., 2004. Bat Natural History and Echolocation. Brigham, R.M., Elisabeth, K.V.K., Gareth, J., Stuart, P., Herman, A.L. (Ed.), Bat Echolocation Research tools, techniques and analysis (2-6), Bat Conservation International, Austin, TX, 167s.

Fister, I., Fister, D., Fister, I., 2013. Differential Evolution Strategies with Random Forest Regression in the Bat Algorithm. Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion, 6-10 July, Amsterdam, 1703-1706.

Fister, I., Fister, D., Yang X.S., 2013. A hybrid bat algorithm. Elektrotehniski vestnik, 80(1-2), 1-7.

Fu, J.F., Fenton, R.G., Cleghorn, W.L., 1991. A Mixed Integer-Discrete-Continuous Programming Method and its Application to Engineering Design Optimization. Engineering Optimization, 17(4), 263-280.

Gandomi, A.H., Yang, X.S., Alavi, A.H., 2011. Mixed variable structural optimization using Firefly Algorithm. Computers and Structures, 89(23-24), 2325-2336.

Gandomi, A.H., Yang, X.S., Alavi, A.H., Siamak T., 2013. Bat algorithm for constrained optimization tasks. Neural Computing and Applications, 22(6), 1239-1255.

Gao, W.F., Liu, S.Y., 2012. A modified artificial bee colony algorithm. Computers and Operations Research, 39(3), 687-697.

Glover, F., Laguna, M. (Ed.), 1997. Tabu Search. Kluwer Academic, 382s.

Goldberg, D.E. (Ed.), 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Professional, 432s.

Hagen, E., 2009. ASU - Ask A Biologist. Erişim Tarihi: 29.8.2013. http://askabiologist.asu.edu/echolocation.

He, Q., Wang, L., 2007. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Applications of Artificial Intelligence, 20(1), 89-99.

He, S., Prempain, E., Wu, Q.H., 2004. An improved particle swarm optimizer for mechanical design optimization problems. Engineering Optimization, 36(5), 585-605.

Hedar, A.R., Fukushima, M., 2006. Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization. Journal of Global Optimization, 35(4), 521-549.

Homaifar, A., Charlene, Q.X., Steven, L.H., 1994. Constrained Optimization via Genetic Algorithms. Transactions of The Society for Modeling and Simulation International, 62(4), 242-253.

Hsu, Y.L., Liu, T.C., 2007. Developing a fuzzy proportional-derivative controller optimization engine for engineering design optimization problems. Engineering Optimization, 39(6), 679-700.

Hu, X.E., Eberhart, R.C., Shi, Y., 2003. Engineering optimization with particle swarm. IEEE swarm intelligence symposium, 24-26 April, Indianapolis, 53-57.

Huang, F.Z., Wang, L., He, Q., 2007. An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and Computation, 186(1), 340-356.

Hwang, S.F., He, R.S., 2006. A hybrid real-parameter genetic algorithm for function optimization. Advanced Engineering Informatics, 20(1), 7-21.

Joines, J.A., Houck, C.R., 1994. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. Evolutionary Computation: IEEE World Congress on Computational Intelligence, 27-29 June, Orlando, 579-584.

Kannan, B.K., Kramer, S.N., 1994. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design. Journal of Mechanical Design, 116(2), 405-411.

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

Karaboga, D., Akay, B., 2009. A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation, 214(1), 108–132.

Kaveh, A., Talatahari, S., 2010. A novel heuristic optimization method: charged system search. Acta Mechanica, 213(3-4), 267-289.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. Neural Networks, 1995. Proceedings., IEEE International Conference on, 27 November - 1 December, Perth, 1942-1948.

Khan, K., Sahai, A., 2012. A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context. International Journal of Intelligent Systems and Applications, 4(7), 23-29.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by Simulated Annealing. Science, 220(4598), 671-680.

Komarasamy, G., Wahi, A., 2012. An Optimized K-Means Clustering Technique using Bat Algorithm. European Journal of Scientific Research, 84(2), 263 - 273.

Lee, K.S., Geem, Z.W., 2005. A new meta-heuristic algorithm for continuous engineering optimization:harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering, 194(36-38), 3902-3933.

Leite, J.P.B., Topping, B.H.V., 1998. Improved genetic operators for structural engineering optimization. Advances in Engineering Software, 29(7-9), 529 - 562.

Lemonge, A.C.C., Barbosa, H.J.C., 2004. An adaptive penalty scheme for genetic algorithms in structural optimization. International Journal for Numerical Methods in Engineering, 59(5), 703-736.

Levenberg, K., 1944. A method for the solution of certain non-linear problems in Least Squares. Quarterly Journal of Applied Mathmatics, 2(2), 164-168.

Li, H.L., Chou, C.T., 1993. A Global Approach for Nonlinear Mixed Discrete Programming in Design Optimization. Engineering Optimization, 22(2), 109-122.

Li, H.L., Chang, C.T., 1998. An approximate approach of global optimization for polynomial programming problems. European Journal of Operational Research, 107(3), 625-632.

Lin, J.H., Chou, C.W., Yang, C.H., Tsai H.L., 2012. A Chaotic Levy Flight Bat Algorithm for Parameter Estimation in Nonlinear Dynamic Biological Systems. Journal of Computer and Information Technology, 2(2), 56-63.

Litinetski, V.V., Abramzon, B.M., 1998. Mars - A Multistart Adaptive Random Search Method for Global Constrained Optimization in Engineering Applications. Engineering Optimization, 30(2), 125-154.

Liu, J.L., 2005. Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems. Engineering Optimization 37(5), 499-519.

Mahdavi, M., Fesanghary, M., Damangir, E., 2007. An improved harmony search algorithm for solving optimization problems. Applied Mathematics and Computation, 188(2), 1567-1579.

Marichelvam, M.K., Prabaharan, T., Yang, X.S., Geetha, M., 2013. Solving hybrid flow shop scheduling problems using bat algorithm. International Journal of Logistics Economics and Globalisation, 5(1), 15-29.

Mashinchi, H.M., Mehmet, A.O., Witold, P., 2011. Hybrid optimization with improved tabu search. Applied Soft Computing, 11(2), 1993-2006.

Mehrabian, A.R., Lucas, C., 2006. A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics, 1(4), 355-366.

Michalewicz, Z., Attia, N., 1994. Evolutionary optimization of constrained problems. Proceedings of the 3rd annual conf on evolutionary programming, 24-26 Feb., San Diego, 98-108.

Montes, M.E., Coello, C.A.C., Velazquez, R., 2007. Multiple trial vectors in differential evolution for engineering design. Engineering Optimization, 39(5), 567-589.

Montes, M.E., Coello, C.A.C., 2008. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. International Journal of General Systems, 37(4), 443-473.

Montes, E.M., Ocana, B.H., 2008. Bacterial Foraging for Engineering Design Problems: Preliminary Results. Proceedings of the 4th Mexican congress on evolutionary computation (COMCEV'2008), 8-10 October, Guanajuato, 33-38.

Musikapun, P., Pongcharoen, P., 2012. Solving Multi-Stage Multi-Machine Multi-Product Scheduling Problem Using Bat Algorithm. International Conference on Management and Artificial Intelligence, 7-8 April, Bangkok, 98-102.

Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P., Yang, X.S., 2012. BBA: A Binary Bat Algorithm for Feature Selection. Graphics Patterns and Images (SIBGRAPI): 25th Sibgrapi Conference on, 22-25 Aug., Ouro Preto, 291-297.

Nickabadi, A., Ebadzadeh M.M., Safabakhsh, R., 2011. A novel particle swarm optimization algorithm with adaptive inertia weight. Applied Soft Computing, 11(4), 3658 - 3670.

Noel, M.M., 2012. A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. Applied Soft Computing, 12(1), 353-359.

Parsopoulos, K.E., Vrahatis, M.N., 2005. Unified particle swarm optimization for solving constrained engineering optimization problems. Wang, L., Chen, K., Yew S.O. (Ed.), Advances in Natural Computation (582-591), Springer Berlin Heidelberg, 1311s.

Ragsdell, K.M., Phillips, D.T., 1976. Optimal Design of a Class of Welded Structures Using Geometric Programming. Journal of Engineering for Industry, 98(3), 1021-1025.

Rao, S.S., 2009. Engineering Optimization: Theory and Practice. John Wiley and Sons Inc, 840s.

Rashedi, E., Hossein, N.P., Saryazdi S., 2009. GSA: A Gravitational Search Algorithm. Information Sciences, 179(13), 2232-2248.

Ray, T., Saini, P., 2001. Engineering Design Optimization Usıng A Swarm With An Intelligent Information Sharing Among Individuals, Engineering Optimization, 33(6), 735-748.

Ray, T., Liew, K.M., 2003. Society and civilization: An optimization algorithm based on the simulation of social behavior. Evolutionary Computation, IEEE Transactions on, 7(4), 386-396.

Runarsson, T.P., Yao, X., 2000. Stochastic ranking for constrained evolutionary optimization. Evolutionary Computation, IEEE Transactions on, 4(3), 284-294.

Sadollah, A., Ardeshir, B., Hadi, E., Mohd, H., 2013. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. Applied Soft Computing, 13(5), 2592 - 2612.

Sakthivel, S., Natarajan, R., Gurusamy, P., 2013. Application of Bat Optimization Algorithm for Economic Load Dispatch Considering Valve Point Effects. International Journal of Computer Applications, 67(11), 35-39.

Sandgren, E., 1988. Nonlinear integer and discrete programming in mechanical design. Proc ASME design technology conf., Kissimine, 95–105.

Sandgren, E., 1990. Nonlinear Integer and Discrete Programming in Mechanical Design Optimization. Journal of Mechanical Design, 112(2), 223-229.

Schnitzler, H.U., Kalko, E.K.V., 2001. Echolocation by Insect-Eating Bats. BioScience, 51(7), 557-569.

Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. Evolutionary Computation Proceedings: IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on, 4-9 May, Anchorage, 69-73.

Shih, C.J., Lai, T.K., 1995. Mixed-Discrete Fuzzy Programming for Nonlinear Engineering Optimization. Engineering Optimization, 23(3), 187-199.

Siddall, J.N., 1972. Analytical decision-making in engineering design. Prentice-Hall, 416s.

Sood, M., Bansal, S., 2013. K-Medoids Clustering Technique using Bat Algorithm. International Journal of Applied Information Systems 5(8), 20-22.

Storn, R., Price, K., 1997. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4), 341-359.

Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S., 2005. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Rapor No: 2005005, 50s.

Taha, A.M., Tang, A., 2013. Bat algorithm for rough set attribute reduction. Journal of Theoretical and Applied Information Technology, 51(1), 1-8.

Tan, K.C., Chiam, S.C., Mamun, A.A., Goh, C.K., 2009. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. European Journal of Operational Research, 197(2), 701-713.

Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T., 2010. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization. Nature Inspired Computation and Applications Laboratory, 18-23 July, Barcelona, 1-23.

Tang, K.S., Man, K.F., Kwong S., He, Q., 1996. Genetic algorithms and their applications. IEEE Signal Processing Magazine, 13(6), 22-37.

Tsai, J.F., Li, H.L., Hu, N.Z., 2002. Global optimization for signomial discrete programming problems in engineering design. Engineering Optimization, 34(6), 613-622.

Tsai, P.W., Jeng, S.P., Bin, Y.L., Ming, J.T., Vaci, I., 2012. Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems. Applied Mechanics and Materials, 148-149(1), 134-137.

Wang, G., Guo, L., 2013. A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization. Journal of Applied Mathematics, 2013(1), 1-21.

Wu, S.J., Chow, P.T., 1995. Genetic Algorithms for Nonlinear Mixed Discrete-Integer Optimization Problems via Meta-Genetic Parameter Optimization. Engineering Optimization, 24(2), 137-159.

Xie, J., Zhou, Y., Chen, H., 2013. A Novel Bat Algorithm Based on Differential Operator and Lévy Flights Trajectory. Computational Intelligence and Neuroscience, 2013(1), 1-13.

Yang, X.S., 2009. Harmony Search as a Metaheuristic Algorithm. Geem, Z.W.(Ed.), Studies in Computational Intelligence (1-14), Springer Berlin Heidelberg, 203s.

Yang, X.S., 2010. A New Metaheuristic Bat-Inspired Algorithm. Cruz, C., González, J.R., Krasnogor, N., Pelta, D.A., Terrazas, G. (Ed.), Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) (65-74), Springer Berlin Heidelberg, 420s.

Yang, X.S., 2010. Nature-Inspired Metaheuristic Algorithms: Second Edition. Luniver Press, 160s.

Yang, X.S., 2011. Bat algorithm for multi-objective optimisation. International Journal of Bio-Inspired Computation, 3(5), 267-274.

Yang, X.S., Gandomi, A.H., 2012. Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 29(5), 464 - 483.

Yang, X.S., 2013. Bat Algorithm and Cuckoo Search: A Tutorial. Yang, X.S. (Ed.), Artificial Intelligence, Evolutionary Computing and Metaheuristics (421-434), Springer Berlin Heidelberg, 796s.

Yang, X.S., 2013. Optimization and Metaheuristic Algorithms in Engineering. Yang, X.S., Gandomi, A.H., Talatahari, S., Alavi, A.H. (Ed.), Metaheuristics in Water, Geotechnical and Transport Engineering (1-23), Elsevier, 496s.

Yilmaz, S., Kucuksille, E.U., 2013. Improved Bat Algorithm (IBA) on Continuous Optimization Problems. Lecture Notes on Software Engineering, 1(3), 279-283.

Yuchi, M., Kim, J.H., 2005. Ecology-inspired evolutionary algorithm using feasibility-based grouping for constrained optimization. Evolutionary Computation: The 2005 IEEE Congress on, 2-5 September, Edinburgh, 1455-1461.

Yun, Y.S., 2005. Study on Adaptive hybrid genetic algorithm and its applications to engineering design problems. Waseda University, Ph.D. Thesis, 115s, Tokyo.

Zahara, E., Kao, Y.T., 2009. Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Systems with Applications, 36(2), 3880-3886.

Zhang, C., Wang, H.P., 1993. Mixed-Discrete Nonlinear Optimization with Simulated Annealing. Engineering Optimization, 21(4), 277-291.

Zhang, J., Changyong, L., Yongqing, H., Jian W., Shanlin Y., 2009. An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. Applied Mathematics and Computation, 111(2), 392-416.

Zhang, M., Luo, W. Wang, X., 2008. Differential evolution with dynamic stochastic selection for constrained optimization. Information Sciences, 178(15), 3043-3074.

ÖZGEÇMİŞ

Adı Soyadı : Selim YILMAZ

Doğum Yeri ve Yılı : Aksaray, 1988

Medeni Hali : Evli

Yabancı Dili : İngilizce

E-posta : selimy@pau.edu.tr

Eğitim Durumu

Lise : Aksaray Teknik ve Endüstri Meslek Lisesi, 2006

Lisans : SÜ, Teknik Eğitim Fakültesi, Bilgisayar Öğretmenliği

Yüksek Lisans : SDÜ, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği

Mesleki Deneyim

MEB (Teknik Öğretmen)	2010-2012
PAÜ Teknoloji Fakültesi	2012-…….. (halen)

Yayınları

Yilmaz, S., Kucuksille, E.U., 2013. Improved Bat Algorithm (IBA) on Continuous Optimization Problems. Lecturer Notes on Software Engineering, 1(3), 279-283.

Dilmen, E., Yilmaz, S., Beyhan, S., 2013. Cascaded ABC-LM Algorithm Based Optimization and Nonlinear System Identification. International Conference on Electronics Computer and Computation (ICECCO), 7-9 Kasım, Ankara, 247-250.