BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

GUGUK KUŞU (CUCKOO) ALGORİTMASI İLE BULANIK SİSTEM OPTİMİZASYONU

Pınar ÖZKURT TUNA
Yüksek Lisans Tezi

Tez Danışmanı
Doç. Dr. Cihan KARAKUZU

BİLECİK, 2014
Ref.No:10043437

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

GUGUK KUŞU (CUCKOO) ALGORİTMASI İLE BULANIK SİSTEM OPTİMİZASYONU

Pınar ÖZKURT TUNA
Yüksek Lisans Tezi

Tez Danışmanı
Doç. Dr. Cihan KARAKUZU

BİLECİK, 2014

BILECIK SEYH EDEBALI UNIVERSITY
Graduate School of Sciences
Department of Computer Engineering

FUZZY SYSTEM OPTIMIZATION USING CUCKOO ALGORITHM

Pınar OZKURT TUNA
Master's Thesis

Thesis Advisor
Assoc. Prof. Dr. Cihan KARAKUZU

BILECIK, 2014

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ **BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**	**YÜKSEK LİSANS** **JÜRİ ONAY FORMU**

Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 05.06.2014 tarih ve 25/3 sayılı kararıyla oluşturulan jüri tarafından 19.06.2014 tarihinde tez savunma sınavı yapılan Pınar ÖZKURT TUNA "Guguk Kuşu (Cuckoo) Algoritması İle Bulanık Sistem Optimizasyonu" başlıklı tez çalışması Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak oy birliği/oy çokluğu ile kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI): Doç. Dr. Cihan KARAKUZU

ÜYE: Prof. Dr. İsmail ERTÜRK

ÜYE: Yrd. Doç. Dr. Mehmet YAKUT

ONAY

Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …/…/…… tarih ve ……/…… sayılı kararı.

ÖZET

Cuckoo Algoritması (COA), Ramin Rajabioun tarafından 2011 yılında geliştirilen yeni bir sezgisel arama algoritmasıdır. Bu tez çalışmasında COA başarımının incelenmesi ve değişik uygulamalara uyarlanması üzerinde durulmaktadır. Algoritmanın başarımı fonksiyon optimizasyonu ve bulanık mantık tabanlı dinamik sistem tanıma problemi çerçevesinde değerlendirilmiştir.

Çalışmada örnek dinamik sistemlerin modellemesi yapılırken COA kullanılmıştır. COA uyarlanarak iki yeni algoritma D-COA ve Ü-COA tanımlanarak belirlenen bazı dinamik sistem tanıma problemleri üzerinde bu üç farklı algoritma istatistiki olarak kıyaslanmıştır. Dinamik sistem modelleme aracı olarak ANFIS bulanık modelleme yapısı kullanılmıştır. COA'nın bulanık ağ yapısı üzerinde eniyileme başarımını daha detaylı inceleyebilmek için farklı veri setleri (test seti) kullanılarak her bir sistem için elde edilen ANFIS modeli değerlendirilmiştir.

Tez çalışmalarının ikinci evresinde ise her bir problem için COA, D-COA ve Ü-COA ile elde edilen sonuçlar ABC ve PSO algoritmalarıyla elde edilen sonuçlar ile karşılaştırılmaktadır. Yapılan değerlendirme sonuçlarına göre; D-COA'nın COA'dan ortalama %5 kadar daha hızlı çalıştığı tespit edilmiştir.

Anahtar Kelimeler: COA, ANFIS, Dinamik Sistem, Sezgisel Algoritma, Optimizasyon

ABSTRACT

Cuckoo Algorithm (COA) which has been improved by Ramin Rajabioun in 2011 is a new heuristic searching algorithm. In this thesis study, it has been studied on investigating performance of COA and adapting it to various applications. Performance of the algorithm has been evaluated in the framework of function optimization and dynamic system identification problem based on fuzzy logic.

In the study, COA has been used while modeling of sample dynamic systems are being done. Modifing COA algorithm, two new algorithms D-COA and Ü-COA have been defined and these three different COA algorithms have been compared statistically on dynamic system identification problems. ANFIS fuzzy model has been used as a modeling tool. Obtained ANFIS model for each system has been evaluated with different data set (test set) in order to investigate more detailed optimization performance of COA on fuzzy network structure.

At the second stage of this thesis study, the results obtained by COA, D-COA and Ü-COA have been compared with the result obtained by ABC and PSO for each problem. According to the evaluation results, it is identified that D-COA run approximately 5% more speed than COA.

Key words: COA, ANFIS, Dynamic System, Heuristic Algorithm, Optimization

TEŞEKKÜR

Tez çalışmamda bilgi birikimi ve tecrübesiyle bana yol gösteren ve yardımcı olan danışman hocam Sayın Doç. Dr. Cihan KARAKUZU'ya çok teşekkür ederim.

Her zaman yanımda olan ve beni destekleyen sevgili eşim Fuat TUNA'ya ve aileme saygı ve teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

1. GİRİŞ

En basit anlamı ile optimizasyon eldeki kısıtlı kaynakları en uygun biçimde kullanmak olarak tanımlanabilir (Bunday, 1984). Matematiksel olarak ifade etmek gerekirse optimizasyon kısaca bir fonksiyonun minimize veya maksimize edilmesi olarak tanımlanabilir (Kahaner, 1989). Diğer bir deyişle optimizasyon "en iyi amaç kriterinin en iyi değerini veren kısıtlardaki değişkenlerin değerini bulmaktır" (Edgar, 1989). Başka bir tanımlama ile "belirli amaçları gerçekleştirmek için en iyi kararları verme sanatı" veya "belirli koşullar altında herhangi bir şeyi en iyi yapma" (Kübat, 1983) olarak da tanımlanan optimizasyon kısaca "en iyi sonuçları içeren işlemler topluluğudur" (Rao, 1978). Optimizasyonda bir amaç da maksimum kar veya minimum maliyeti sağlayacak üretim miktarını kısıtlara bağlı olarak tespit etmektir. Günümüzün bilgisayar teknolojisi sayesinde güncel bir kavram olan optimizasyon kavramı çok çeşitli endüstri kesimlerinde uygulama olanağı bulmuştur.

Değişen teknolojilerin, sınırlı kaynakların, artan rekabetin, karmaşık hale gelen sistemlerin doğurduğu problemlerin klasik yöntemlerle (matematiksel veya matematiksel olmayan, analitik veya sayısal) çözümünün güçleşmesi optimizasyon kavramını güncelleştiren en önemli sebeptir. Bu yönüyle optimizasyonun kullanılmadığı bir bilim dalı hemen hemen yok gibidir.

Son yıllarda sezgisel yaklaşımlar, karmaşık problemlerin çözümünde önemli katkılar sunmaktadır. Optimizasyon problemlerini çözmek için farklı yöntemler vardır. Genetik Algoritma (GA), Parçacık Sürü Optimizasyonu (PSO), Karınca Koloni Optimizasyonu (ACO), Yapay Arı Kolonisi (ABC) algoritmaları sıklıkla tercih edilen algoritmalardır.

Bu iyi bilinen yöntemlerin yanı sıra araştırmalar hala devam etmekte ve uygulanabilir yöntemler geliştirilmektedir. Cuckoo (Guguk Kuşu) Algoritması (COA), Ramin Rajabioun tarafından 2011 yılında geliştirilen yeni bir sezgisel arama algoritmasıdır. Bu tez çalışmasında COA'nın başarımının incelenmesi, geliştirilmesi ve dinamik sistem modellenmesinde kullanımı üzerinde durulmaktadır.

1.1. Tezin Amacı ve Kapsamı

Tezin amacı, yeni bir sezgisel arama ve eniyileme algoritması olan COA'nın öğrenilmesi, çalışma mekanizmasının kavranması ve bu süreç boyunca elde edilen/gözlenen sonuçlara göre eksikliklerinin giderilmesi amacıyla geliştirilmesidir.

Bu amaç doğrultusunda, bu tez çalışması boyunca fonksiyon optimizasyonu ve bulanık sistem tabanlı dinamik sistem modelleme üzerinde çalışılmıştır. Fonksiyon optimizasyonu çalışmaları sırasında iki yeni COA geliştirilmiş (D-COA ve Ü-COA) ve bunların birbirleriyle yerel ve küresel minimuma takılma bakımından kıyaslaması yapılmıştır.

Tez çalışmalarının ikinci evresini oluşturan bulanık sistem tabanlı dinamik sistem modelleme çalışmalarında ise; COA ANFIS bulanık çıkarım sisteminin optimizasyonuna uyarlanarak literatürden alınan beş farklı eğrisel dinamik sistemin modellenmesinde işletilmiştir.

1.2. Literatür Taraması

Ramin Rajabioun tarafından (Rajabioun, 2011) 2011 yılında bilimsel camiaya tanıtılan COA ile ilgili olarak 2011 yılından itibaren yapılan literatür taraması sonucunda COA ile yapılan ve bu tez çalışması ile ilgili olan çalışmalar aşağıda kısaca özetlenmektedir.

S. Berrazouane ve K. Mohammedi (Berrazouane veMohammedi, 2014), bir hibrid elektrik sisteminde enerji yönetimi için ANFIS bulanık çıkarım sistemi kullanılarak COA'nın optimizasyon başarımını gerçekleştirmişlerdir. Haftalık güneş ışını, ortam sıcaklığı gibi parametreleri istenilen değerlerde ayarlamak için COA ile optimizasyon tercih edilmiştir.

K. Chandrasekaran ve Sishaj P. Simon (Chandrasekaran ve Simon, 2012), çok amaçlı problemleri çözmek için bulanık sistem destekli COA kullanmıştır. Bu çalışmada, ANFIS yapısı kullanılarak COA ile en iyi değerleri bulmak hedeflenmiştir. Bu yöntem hem tek hem de çok amaçlı optimizasyon problemleri için test edilmiş ve onaylanmıştır. Puja Dash, Lalit Chandra, Nidul Sinha (Dash, Chandra, Sinha, 2014), tarafından yapılan çalışmada, termal sistemlerin kontrolünün AGC algoritması (Automatic Generation Control) ve COA ile sağlanması gerçekleştirilmiştir.

Optimizasyonda kullanılan parametreler aynı anda Cuckoo arama algoritması denilen daha yeni ve güçlü bir evrimsel hesaplama tekniği kullanılarak optimize edilmiştir. Bu iki algoritmanın performansları karşılaştırılmıştır. Çalışma sonucunda COA'nın daha duyarlı olduğu tespit edilmiştir.

M. Basu ve A. Chowdhury (Basu ve Chowdhury, 2013) tarafından yapılan çalışmada yine ekonomi alanında COA'nın kullanılmasından bahsedilmektedir. Şebekelerin güç dağıtım problemi için COA kullanılmıştır. Bu problemin istenen şekilde çözümlenebilmesi için çalışılmıştır. Varolan diğer tekniklerle karşılaştırıldığında ve elde edilen çözümün kalitesi göz önüne alındığında, COA şebeke güç gönderme sorunlarının çözümü için umut verici bir alternatif yaklaşım olarak görülmüştür.

1.3. Tezin Katkısı

ANFIS bulanık çıkarım sistemi kullanılarak COA'nın dinamik sistem modellemede optimizasyon başarımının incelendiği bir çalışma bu zamana kadar henüz gerçekleşmemiştir. Daha çok ekonomik alanda karşılaşılan problemlerin çözümü için bu algoritma tercih edilmiştir. Tam olarak ANFIS sistemi kullanılarak COA ile optimizasyon gerçekleştiren literatürde herhangi bir çalışma bulunmamaktadır. Henüz yeni bir sezgisel arama algoritması olduğu için üzerinde çalışmalar devam etmektedir.

2. GUGUK KUŞU (CUCKOO) ALGORİTMASI

2.1. Giriş

Bu bölümde, guguk kuşlarının yaşam tarzından esinlenerek Rajabioun (2011) tarafından geliştirilen yeni bir sezgisel algoritma olan Cuckoo Optimizasyon Algoritması (COA) tanıtılacaktır. Optimizasyon algoritmasının temelini kuşların yerleşimi ve üremesi oluşturmaktadır. Bu modellemede kullanılan 2 tane form vardır. Bunlar; olgun kuşlar ve yumurtalardır. Olgun kuşlar, diğer kuşların yuvalarına yumurtalarını bırakırlar, ev sahibi kuşlar tarafından bu yumurtalar öldürülmez, büyürler ve olgun kuşa dönüşürler. Kuşlar çevresel özelliklerine uyum sağlamak için göç edebilecekleri en uygun yerleri bulmaya çalışırlar.

Bu optimizasyon algoritması, guguk kuşu ailesinin yaşam tarzından esinlenilerek oluşmuştur. Aşağıda guguk kuşlarının yaşam tarzından, ilk yetişme ortamlarından, guguk kuşlarının yumurtlama stilinden, guguk kuşlarının göçmenliklerinden bahsedilmiştir. Guguk kuşlarının yaşam tarzları, yumurtlama ve üreme özgürlükleri, bu sezgisel optimizasyon algoritmasının gelişmesi için temel güdü oluştururken, optimizasyon problemini çözmek için ilk olarak, ilk nüfusun hangi pozisyonlarda olduğunu bilmemiz gerekir. Algoritma, izleyen alt bölümlerde detaylı anlatılacaktır.

2.2. COA'nın Esin Kaynağı ve Çalışma Alt Yapısı

Guguk kuşları herhangi bir kuş yuvasının hakiki sahibi kuş uzaklaşır uzaklaşmaz gözetlediği yerden gelir ve bir yumurtasını ev sahibi kuşun yumurtaları arasına bırakır. Yuva sahibi kuşun yumurtalarından birisini de gagasıyla alarak yuvadan uzaklaşır. Daha sonra yavru guguk kuşu, yuva sahibi kuştan daha büyük hale gelmesine rağmen onun tarafından beslenmeye devam edilir. Guguk kuşu, ziyaret ettiği her yuvaya sadece bir yumurta bırakır. Yumurtadan yeni çıkan bir guguk yavrusu, henüz gözleri bile açılmadan sanki öğretilmiş gibi, ev sahibi kuşun yumurtalarını sırtını ve kanatlarını kullanarak yuvadan atar.

Guguk kuşları yaşam tarzları, yumurtlama ve üreme özgürlükleri ile COA'nın gelişmesi için temel güdü oluşturmuştur. Diğer metotlara benzer olarak, COA da belirli

bir nüfusla başlar. Farklı toplumlardaki kuş nüfusu, olgun kuşlardan ve yumurtalardan oluşur.

COA'nın akış şeması Şekil 2.1'de görülmektedir. Akış şeması incelendiğinde, diğer algoritmalarda olduğu gibi bu algoritma da başlangıçta belli bir nüfus ile başlar.

Başlangıçtaki bu guguk kuşları, bazı kuşların yuvalarına yumurtalarını bırakır. Bu yumurtaların büyümesine ev sahibi kuş tarafından izin verilir. Olgun kuş haline geldikten sonra, başka bir yaşam alanına doğru göç ederler. Göç eden kuşlar yine üremeye başlarlar. Aynı şekilde başka kuşların yuvalarına yumurtalarını bırakırlar. Bu yumurtalardan yavru kuşlar çıkar ve büyürler. Büyüyen kuşlar içinde en iyi grup belirlenir ve bu grup yeni yaşam alanı olarak seçilir. Yeni guguk kuşu nüfusu bu alana doğru göç eder. Eğer algoritma sonunda istenilen durum oluşmamışsa, bu durum tekrarlanır. Eğer istediğimiz sonuçları elde etmişsek, algoritma durdurulur ve sonuçlar alınır.

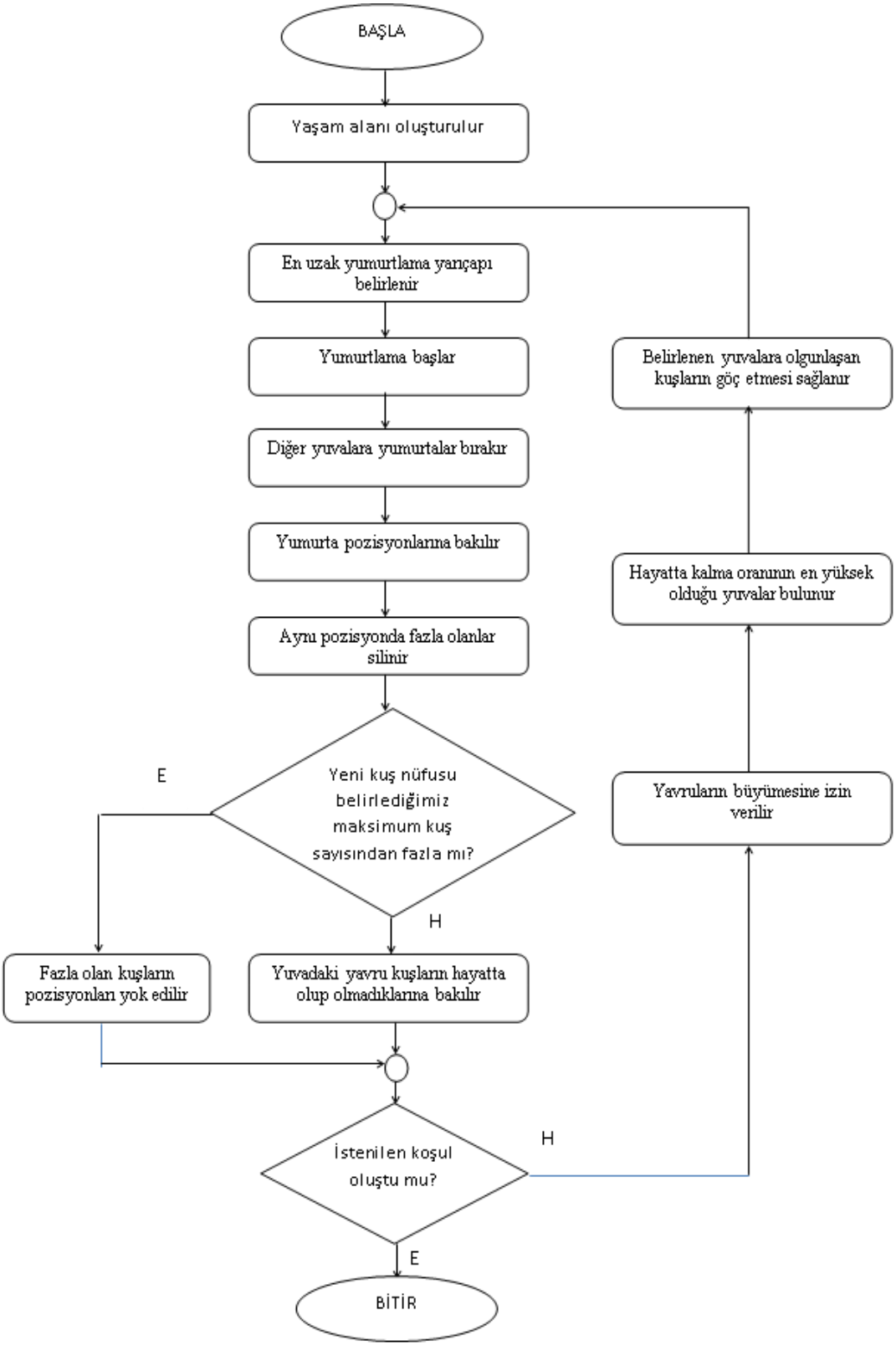
Optimizasyon problemini çözmek için ilk olarak, ilk nüfusun hangi pozisyonlarda olduğunu bilmemiz gerekir. Bu ilk pozisyonlar bir dizi içinde tutulurlar. Bu diziye "yetişme ortamı" denir. N boyutlu bir optimizasyon probleminde, mevcut yaşam pozisyonları şu şekilde tutulmaktadır:

$$habitat = [x_1, x_2, ..., x_N] \tag{2.1}$$

Ölçüt fonksiyonu olarak seçilen fonksiyon, bu yaşam alanına uyarlanır:

$$CostFunction = f(habitat) = f(x_1, x_2, ..., x_N) \tag{2.2}$$

Optimizasyon algoritmasını başlatmak için, rasgele değerlerle belirlenmiş adayları içinde tutan matris oluşturulur. Algoritmada en yüksek ve en düşük limitler belirlenerek her kuşa belirli sayıda yumurta tahsis edilir. Kuşlar, bu yumurtaları kendi doğal ortamlarından maksimum olabilecek uzaklığa bırakırlar.



Şekil 2.1. COA akış şeması (Rajabioun, 2011).

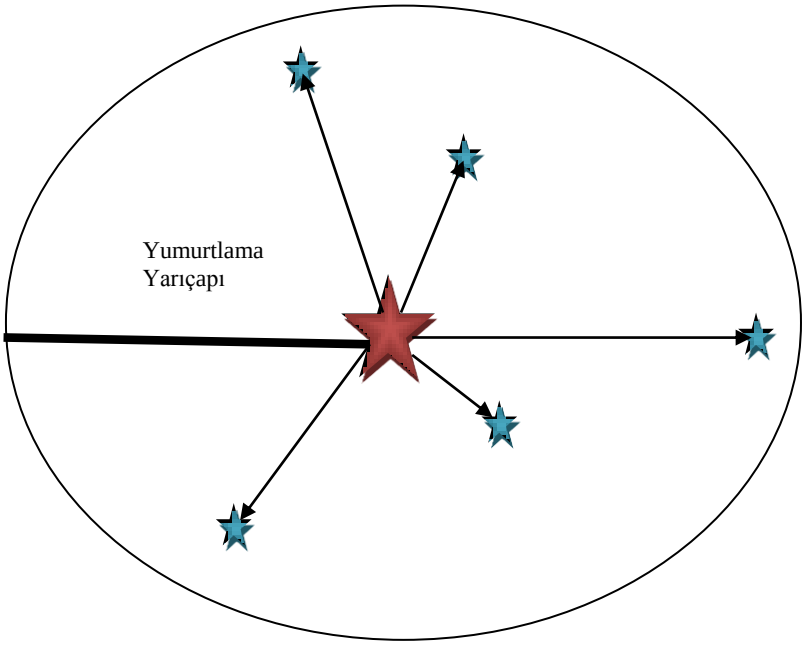
Bu maksimum uzaklığa "yumurtlama yarıçapı" denir. Problemde en düşük limit $yr_{\min}$, en yüksek limit $yr_{\max}$ olarak atanır.

Yumurtlama yarıçapı yr_i eşitlik (3)'teki gibi hesaplanır:

$$yr_i = \alpha \times \frac{y_n}{\sum_j^j y_n^j} \times (yr_{\max} - yr_{\min}) \tag{2.3}$$

Eşitlikte yr_i i. kuşun yumurtlama yarıçapını, α yarıçap oranını, y_n i. kuş için yumurta numarası, $yr_{\max}$ maksimum yumurtlama yarıçapı, $yr_{\min}$ minimum yumurtlama yarıçapını göstermektedir.

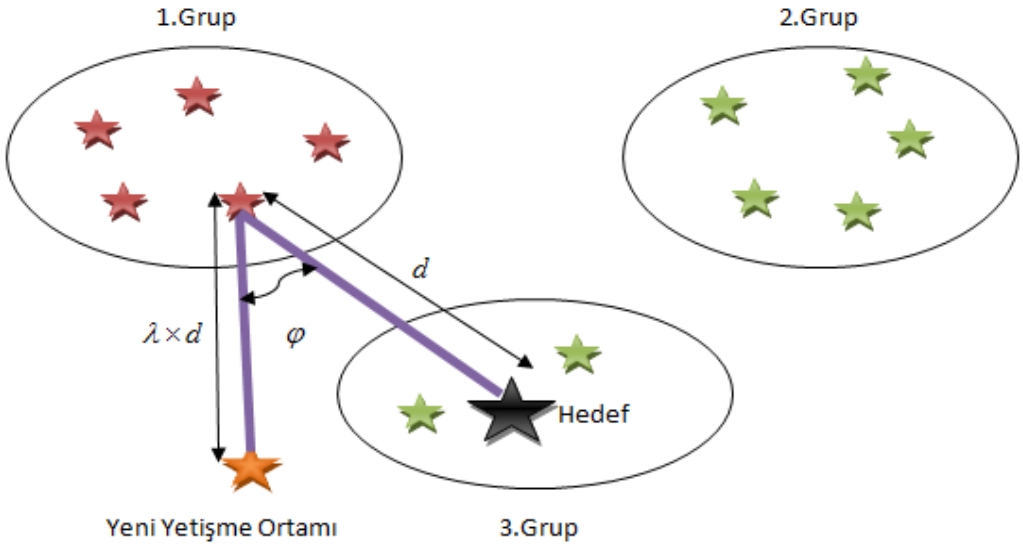
Her guguk kuşu Şekil 2.2'de görüldüğü gibi yumurtlama yarıçapı içinde başka kuşların yuvalarına yumurtalarını bırakırlar.



Şekil 2.2. Rasgele yumurtlama yarıçapı belirleme (Rajabioun, 2011).

Eğer ev sahibi kuşlar tarafından yumurtaların kendilerine benzemediği keşfedilirse, yuvadan yumurtalar ev sahibi kuşlar tarafından atılır. Genelde bu yuvadan atılan yumurtaların oranı %10 civarındadır. Bazı zamanlarda ev sahibi kuşların yavruları açlıktan ölür, sadece guguk kuşu yavruları yuvada kalır.

Genç yavru kuşlar büyüyüp olgunlaştıkları zaman kendilerine başka yeni yaşam alanı bulurlar. Bu alanlara doğru göç ederler. Şekil 2.3'te görüldüğü gibi her kuş φ değeri kadar sapmayla yeni yaşam alanlarına göç ederler.



Şekil 2.3. Guguk kuşlarının yaşam alanları (Rajabioun, 2011).

Göç esnasında λ ve φ değerleri, yeni pozisyonları hesaplamak için yardımcı parametrelerdir. λ ve φ parametreleri şu şekilde ifade edilir:

$$\lambda \sim U(0,1) \tag{2.4}$$

$$\varphi \sim U(-w,w) \tag{2.5}$$

$\lambda \sim U(0,1)$ ifadesinde λ parametresi, 0 ile 1 arasında rasgele bir sayıdır, w değeri ise hedef yaşam alanındaki sapmayı sınırlayan bir parametredir.

Bütün kuşlar, hedefledikleri alana doğru göç edip yeni nüfus oluşturduklarında her olgun kuşa bazı yumurtalar tahsis edilir. Her kuş için tekrar yumurtlama yarıçapı hesaplanır. Daha sonra yeni yumurtlama işlemi başlar.

2.3. COA Kaba Kodu

1. Mevcut uzayda rasgele belirlenen noktalar ile yaşam alanı oluşturulur.
2. Her kuşa bazı yumurtalar tahsis edilir.
3. Her kuş için en uzak yumurtlama yarıçapı belirlenir.
4. Belirlenen yumurtlama yarıçapı alanı içine yumurta bırakmasına izin verilir.
5. Ev sahibi kuşlar tarafından ayırt edilen yumurtalar yok edilir.
6. Civcivlerin yumurtadan çıkmasına ve büyümesine izin verilir.
7. Büyüyen her kuşun yaşam alanı değerlendirilir.
8. Alan içinde yaşayabilecek kuş sayısı limitlendirilir ve istenmeyen alandakiler yok edilir.
9. Kuşlar sınıflandırılır, en iyi kuş grubu tespit edilir ve hedef yaşam alanı seçilir.
10. Yeni guguk kuşu nüfusunun hedef yaşam alanına göç etmesine izin verilir.
11. Son koşuldan memnun olunursa optimizasyon durdurulur, eğer memnun olunmazsa 2. adımdan itibaren tekrarlanır.

2.4. Sonuç

COA, yeni bir sezgisel algoritma olup guguk kuşlarının yaşam tarzından esinlenerek geliştirilen bir çalışma mantığına sahiptir. Diğer algoritmalarda olduğu gibi COA da başlangıçta belli bir nüfusla başlar.

Guguk kuşlarının yumurtlamasında esas olan yumurtlama yarıçapı parametresidir. Bu parametre kullanılarak algoritma çalıştırılmıştır.

3. COA İLE FONKSİYON OPTİMİZASYONU

3.1. Giriş

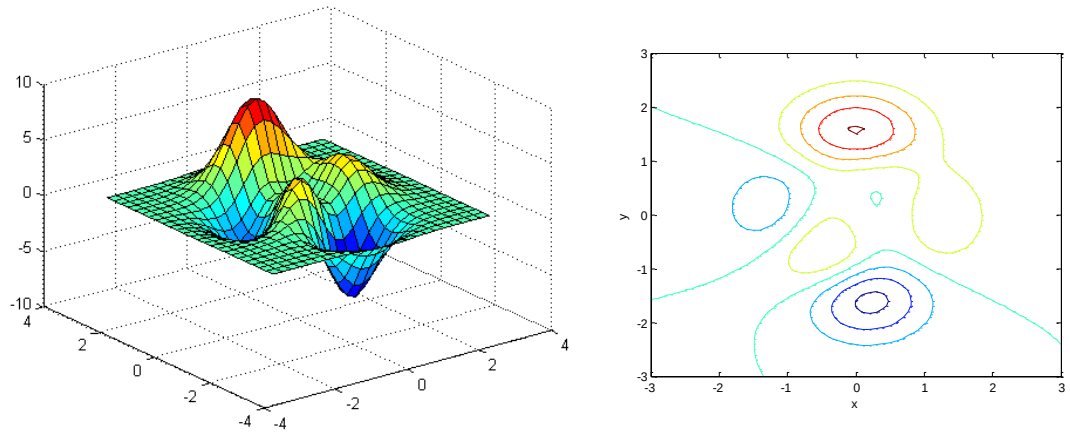
Standart COA (S-COA)'nın işleyişi, çalışması ve mantığının kavranması amacıyla öncelikle *peaks* fonksiyonu üzerinde çalışılmıştır. Bu çalışmalar sırasında algoritmanın başarımını etkileyen en önemli değişkenin yumurtlama yarıçapı olduğu gözlenmiştir. Bu gözlem sonucunda iki yeni yaklaşım geliştirilmiştir. Geliştirilen bu yaklaşımların kıyaslamalı başarımları için literatürden alınan beş farklı fonksiyon üzerinde de çalışılmıştır. Bu çalışmaların sonuçları aşağıda kısaca özetlenmiştir.

3.2. Örnek 1: F1 (Peaks)

Peaks fonksiyonu aşağıdaki gibi tanımlanır.

$$z = 3(1-x)^2 \times e^{(-x^2-(y+1)^2)} - 10 \times (x \div 5 - x^3 - y^5) \times e^{(x^2-y^2-1\div 3)} \times e^{(-(x+1)^2-y^2)} \qquad (3.1)$$

Fonksiyonun yüzeyi ve eş yükselti eğrileri Şekil 3.1'de verilmiştir. Görüleceği üzere, fonksiyon çeşitli yerel tepe ve çukurlar içermektedir. S-COA ile bu fonksiyonun küresel minimum noktasının bulunması üzerine çalışılmıştır.



(a) (b)

Şekil 3.1. *Peaks* fonksiyonu yüzeyi (a) ve eş yükselti eğrileri(b).

Peaks fonksiyonunun yüzey grafiği Şekil 3.1.a'daki gibidir. Bu şekilden görüleceği üzere, fonksiyon 3 adet tepe 3 adet çukura sahip fonksiyondur. Küresel minimum x= 0,3103 ve y=-1,552 koordinatlarında -6,54'tür.

COA'nın çalışmasını anlamak ve başarısını test edebilmek için ilk olarak matematiksel ifadesi (3.1)'de verilen, Şekil 3.1'den de görüleceği üzere birkaç yerel minimumu ve maksimumu bulunan "*peaks*" fonksiyonu üzerinde çalışılmış, algoritmaya küresel minimum buldurulmuştur. Algoritmanın çalışmasını anlamaya çalışırken, algoritmada kullanılan yarıçap parametresinin değiştirilerek daha iyi sonuçlar alınabileceği keşfedilmiştir.

Yarıçap parametresi, bu tez çalışmasında geliştirilen algoritmalarda kuşların yumurtalarını bırakacakları alanı sınırlayarak, istenilen sonuçlara ulaşmada daha çok yardımcı olmaktadır.

Yarıçapı belirlerken, iki yeni yaklaşım geliştirilmiştir. Bunlar; yumurtlama yarıçapının tekrarla doğrusal olarak azaltıldığı doğrusal yaklaşım (D-COA) ve yumurtlama yarıçapının tekrarla üstel olarak azaltıldığı üstel yaklaşım (Ü-COA)'dır.

D-COA yaklaşımı için yumurtlama yarıçapı (3.2) eşitliği ile hesaplanır.

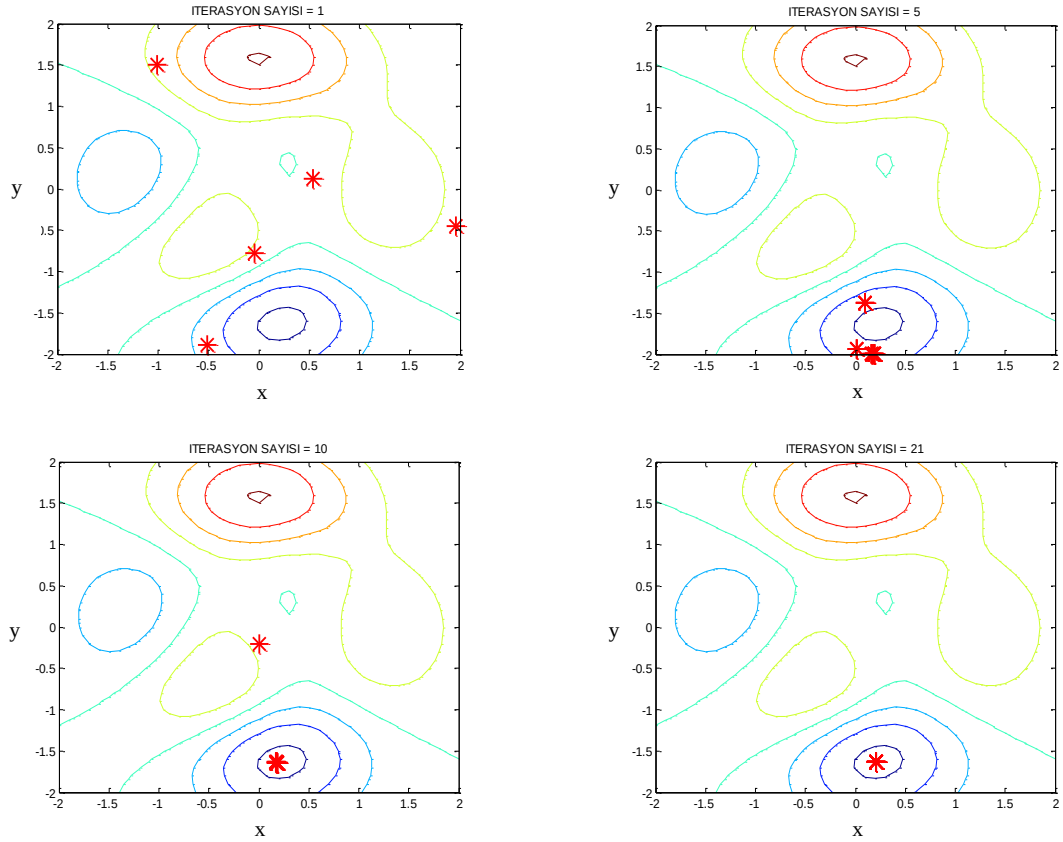
$$yr = r_{\max} - \left[\left(r_{\max} - r_{\min} / G_{\max}\right)\right] \times n \tag{3.2}$$

Ü-COA için ise yumurtlama yarıçapı (3.3)'de verilen eşitlikle belirlenmektedir.

$$yr = 0{,}95 \times e^{(-0{,}05 \times n)} \tag{3.3}$$

Matematiksel ifadelerde yer alan terimlerden yr yarıçap, $r_{\max}$ maksimum yarıçap, $r_{\min}$ minimum yarıçap, n tekrar ve $G_{\max}$ maksimum tekrar sayısı anlamlarına gelmektedir.

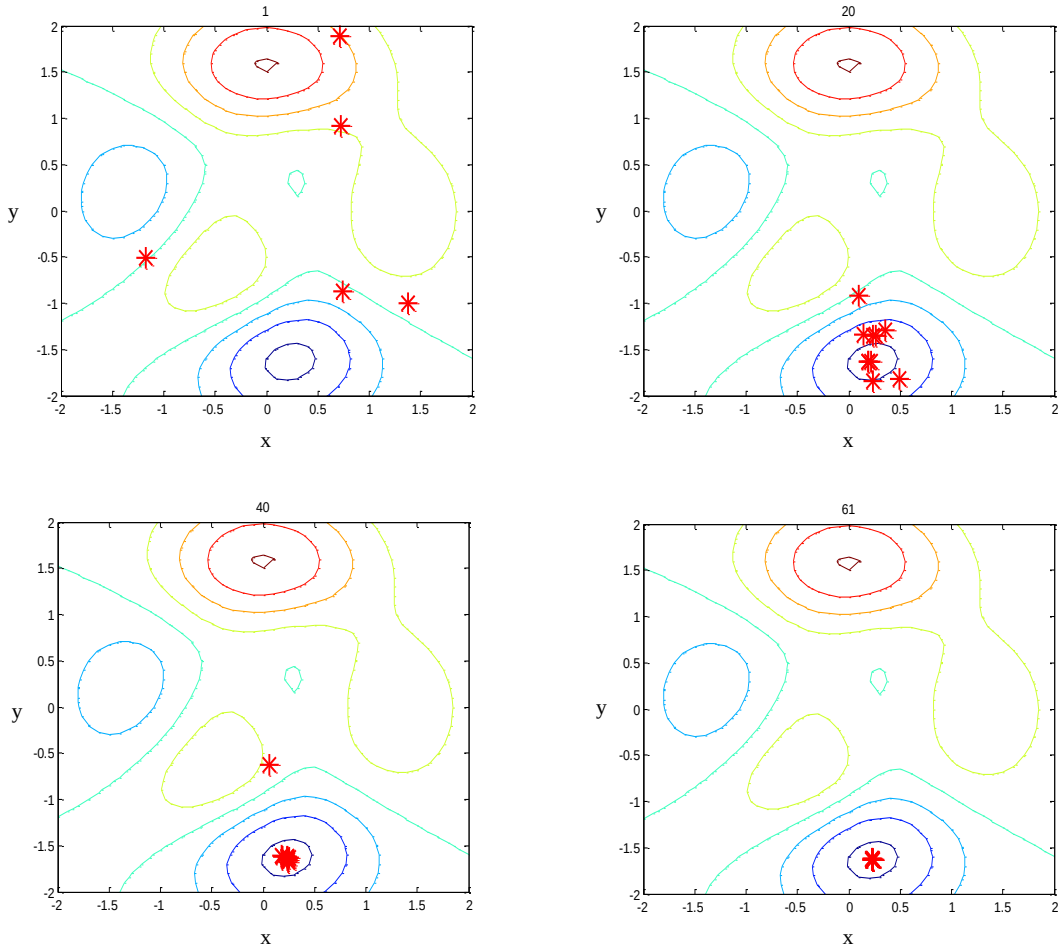
Standart yaklaşımda, yr sabit ve 0,01 olarak seçilmiştir. Bu yaklaşım kullanılarak algoritmamızı inceleyecek olursak, ilk andaki kuş nüfusu pozisyonları dağınık olmasına rağmen en son tekrar adımında kuşların küresel minimuma çok yakın civarda toplandıkları Şekil 3.2'de görülmektedir.



Şekil 3.2. Standart COA ile "*peaks*" fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Doğrusal yaklaşımda, *yr* doğrusal olarak azalmaktadır. $r_{\max}$ 10 ve $r_{\min}$ 5e-3 olarak belirlenmiş ve ***yaricapOrani*** isimli dizi oluşturularak her azalan değer buradan alınarak kuşların pozisyonları bulunmuştur. Bu yaklaşım kullanılarak "*peaks*" fonksiyonu optimizasyonundaki bazı tekrarlarda bireylerin durumu Şekil 3.3'de görülmektedir.

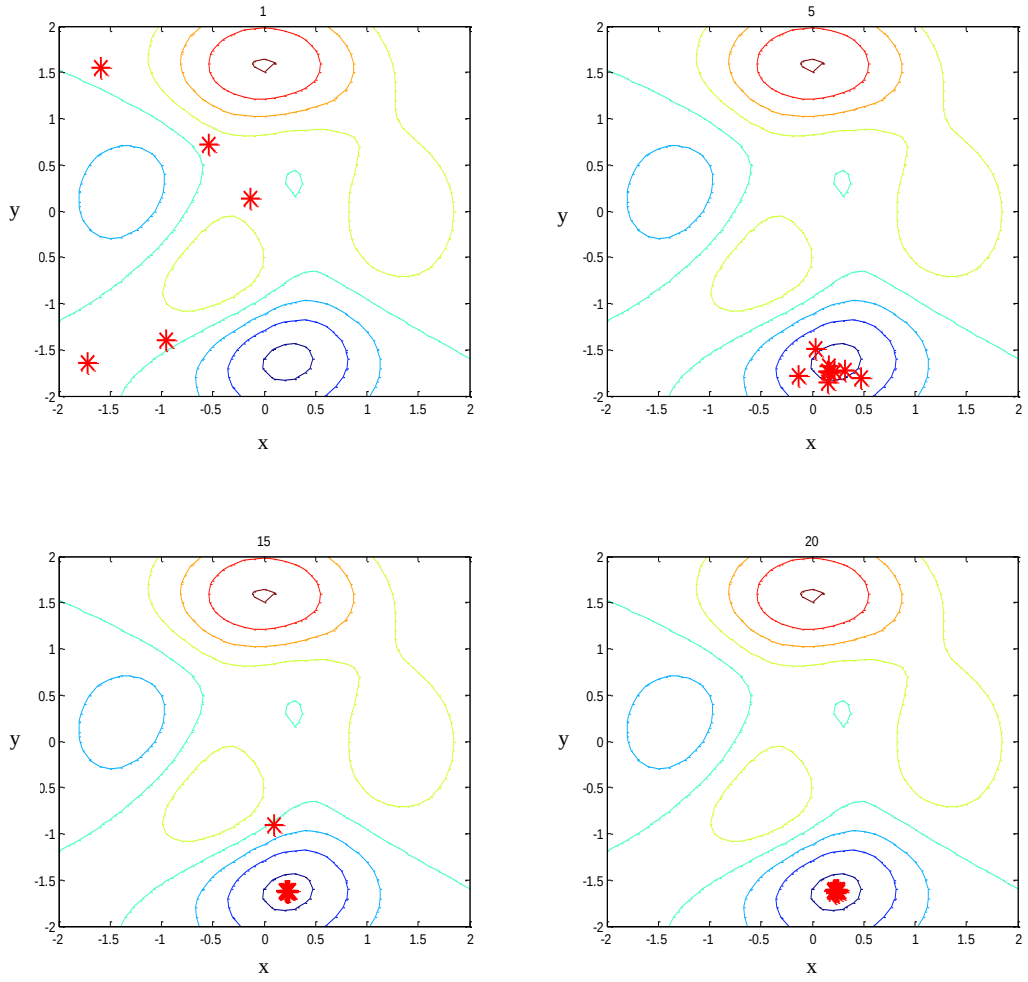
Üstel yaklaşımda ise, yarıçap üstel olarak azalmaktadır. Bu yaklaşımda da ***yaricapOrani*** isimli dizi oluşturularak her tekrardaki elde edilen pozisyonlar burada tutulmaktadır.



Şekil 3.3. D-COA ile "*peaks*" fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Üstel yaklaşım kullanılarak, "*peaks*" fonksiyonundaki tekrar adımları Şekil 3.4'de görülmektedir.

Yukarıda kısaca anlatılan iki yeni (D-COA, Ü-COA) yaklaşımın standart COA'ya göre başarımlarını istatiksel olarak belirlemek amacıyla her bir algoritma *peaks* fonksiyonu optimizasyonu problemi üzerinde 100'er nesil koşturulmuştur.



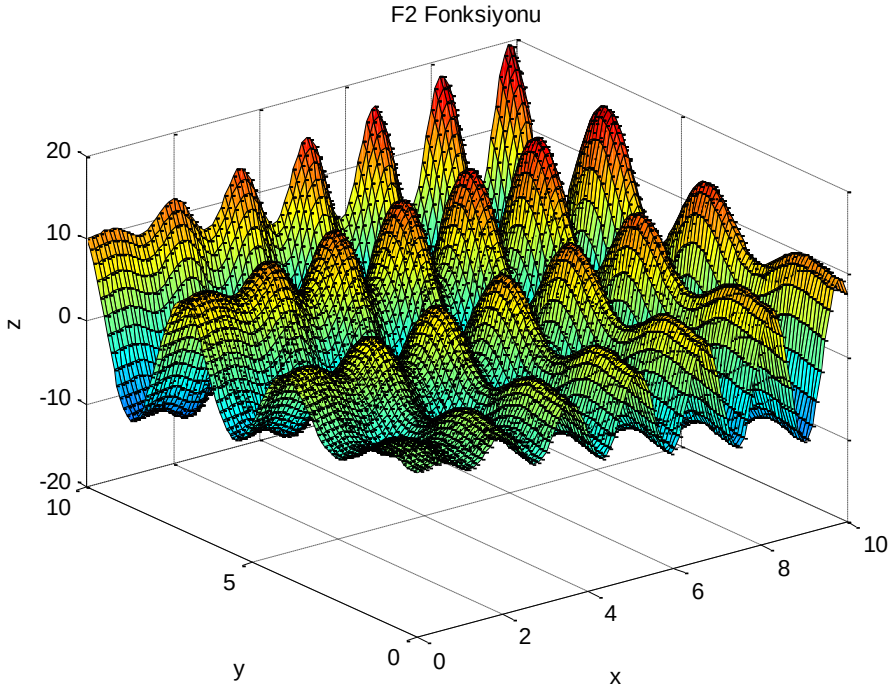
Şekil 3.4. Ü-COA algoritması ile "*peaks*" fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

3.3. Örnek 2: F2

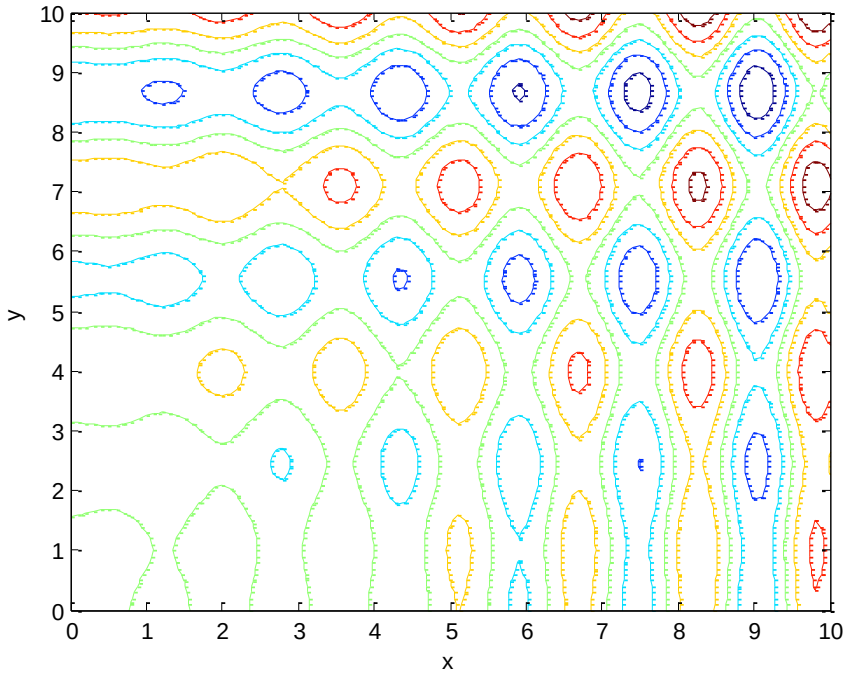
COA'nın çalışmasını anlamak ve başarısını test edebilmek için kullandığımız ikinci fonksiyonun matematiksel ifadesi (3.4)'de görülmektedir (Rajabioun, 2011). Şekil 3.5'den de anlaşıldığı üzere F2 fonksiyonu birçok tepe ve çukurdan oluşmaktadır. Bu fonksiyon kullanılarak, geliştirmiş olduğumuz D-COA ve Ü-COA algoritmalarının başarımı incelenmektedir.

$$z = x\sin(4x) + 1{,}1y\sin(2y) \tag{3.4}$$

Küresel minimum, x=9,039 ve y=8,668 koordinatlarında -18,5547'dir.

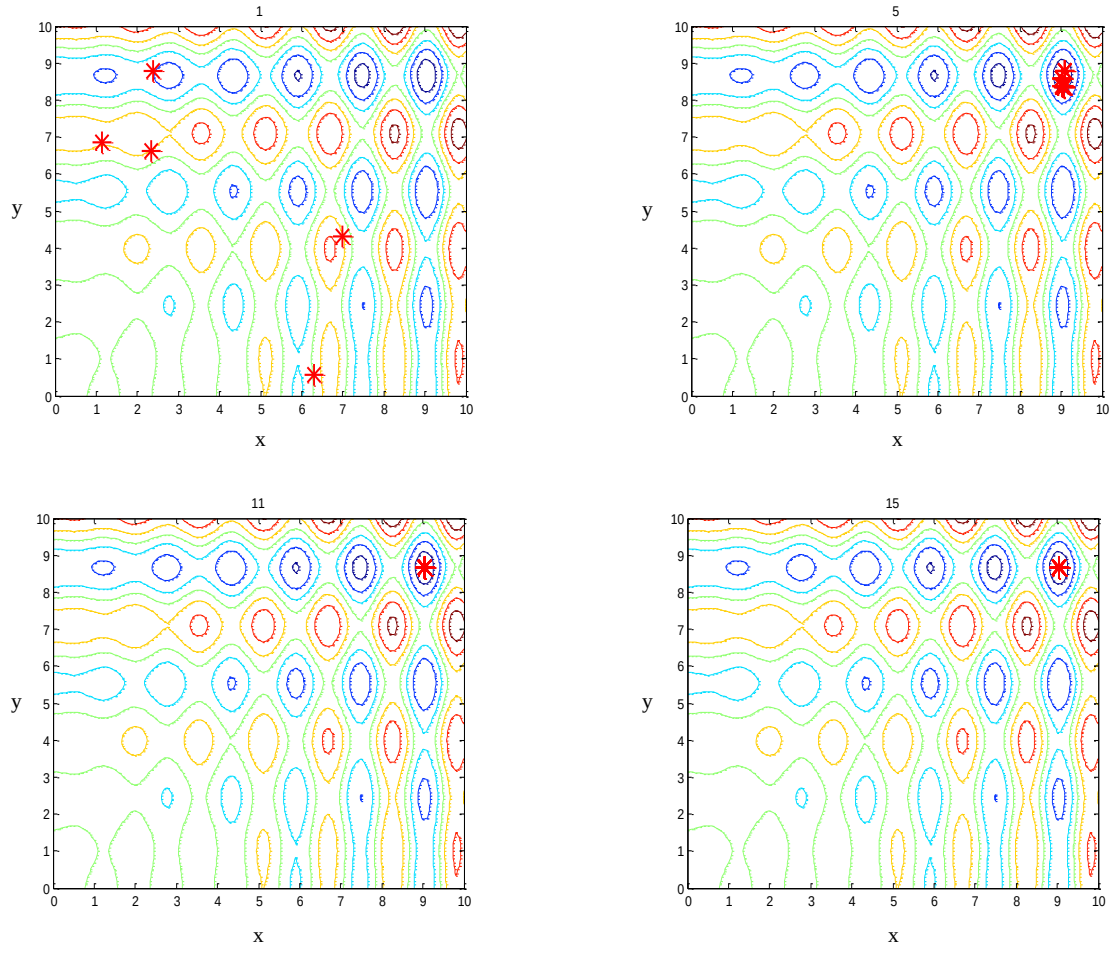


Şekil 3.5. Örnek 2 (F2) fonksiyonu.



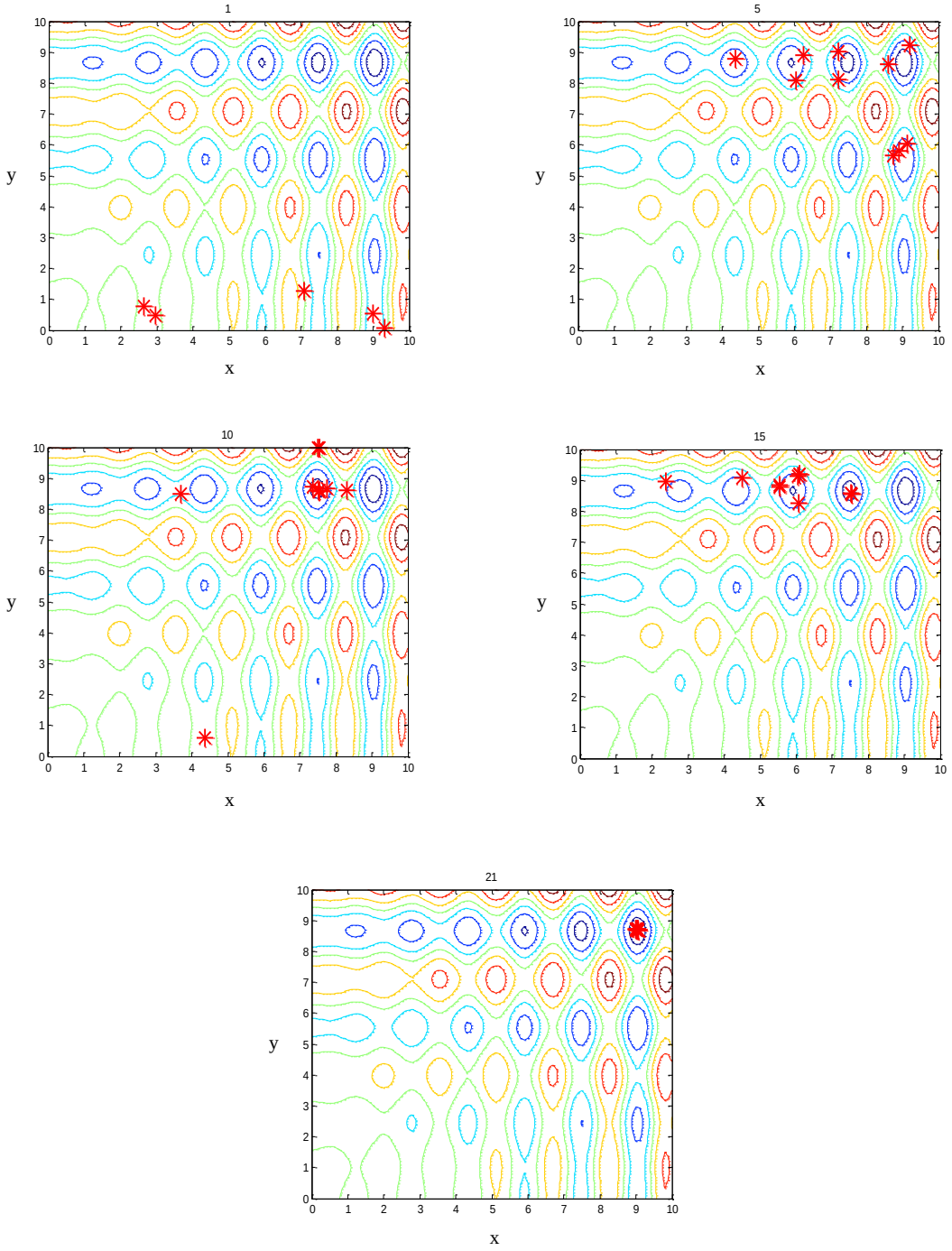
Şekil 3.6. Örnek 2 (F2) fonksiyonu eş yükselti eğrileri.

Standart COA kullanılarak, Şekil 3.7'deki sonuçlar elde edilmiştir.



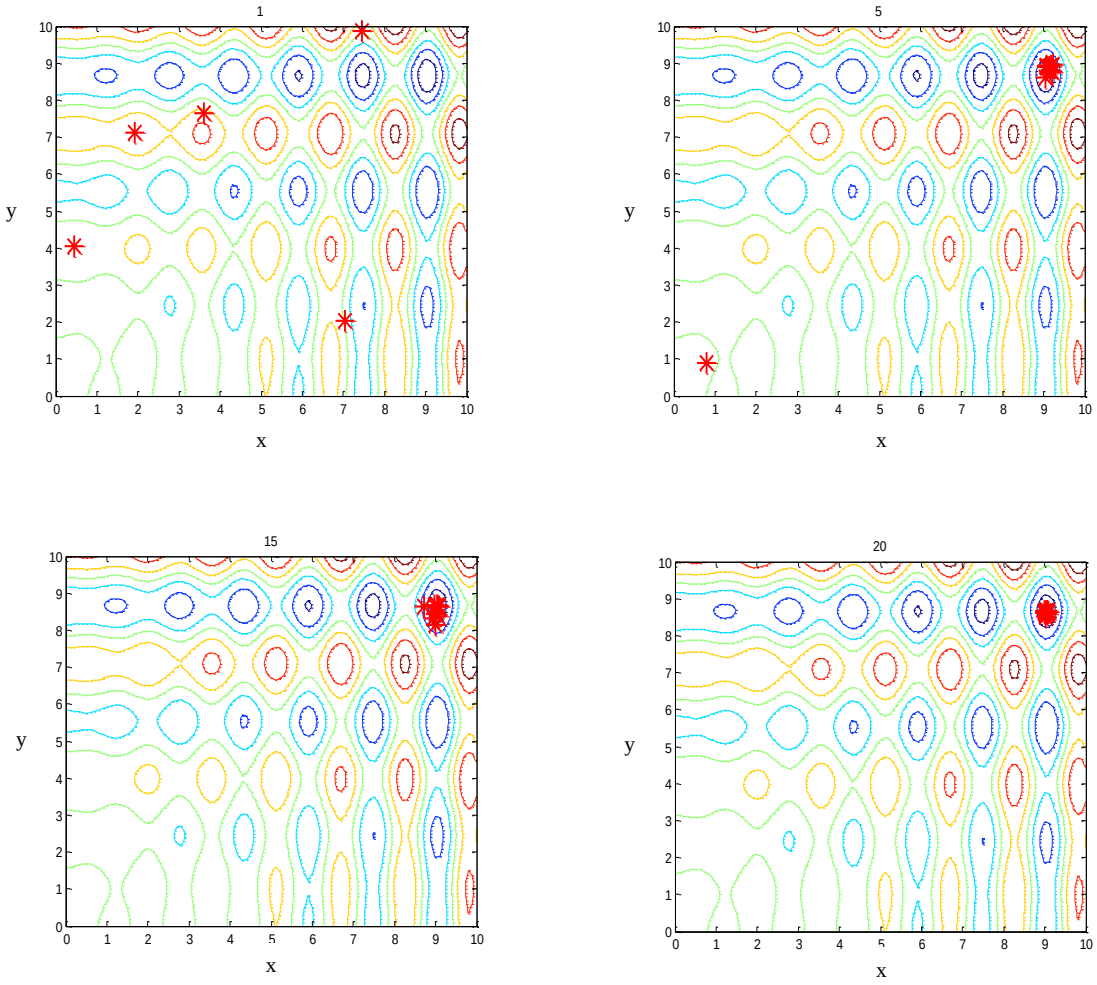
Şekil 3.7. S-COA ile F2 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

F2 fonksiyonu için standart yaklaşım kullanılarak çalıştırıldığında elde edilen Şekil 3.7'deki sonuçlara bakılacak olursa 11. tekrarda istenilen global minimum noktasına ulaşıldığı görülmüştür. Fakat F2 fonksiyonunda birçok tepe ve çukura sahip olduğu için lokal minimuma gitme olasılığı her zaman daha yüksektir. Aşağıda Şekil 3.8'de D-COA ile F2 fonksiyonu optimizasyonunda elde edilen sonuçlar verilmiştir:



Şekil 3.8. D-COA ile F2 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Üstel yaklaşım kullanılarak elde edilen sonuçlar Şekil 3.9'da verilmiştir.



Şekil 3.9. Ü-COA ile F2 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

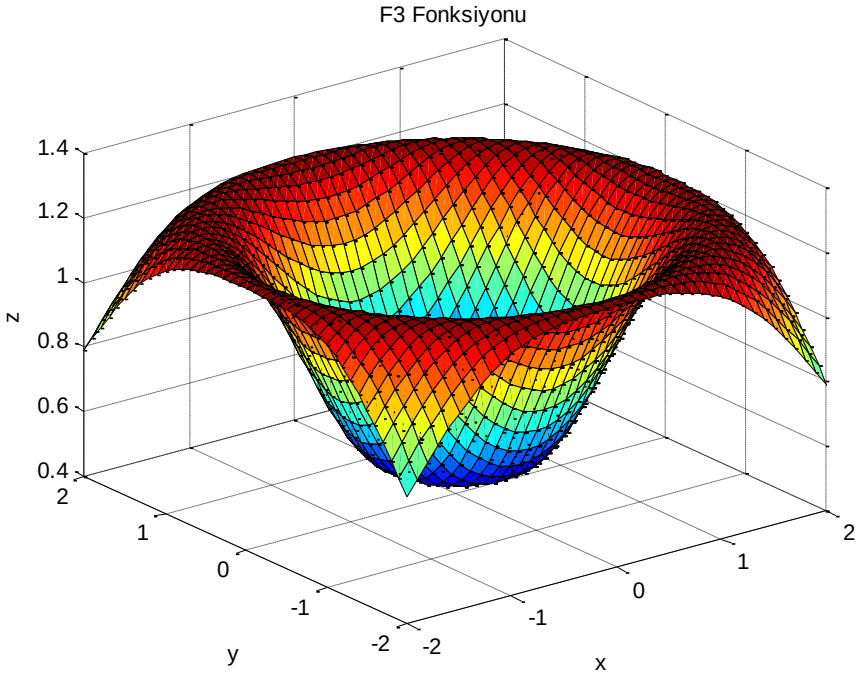
3.4. Örnek 3: F3

F3 fonksiyonu (Rajabioun, 2011) aşağıdaki gibi tanımlanır :

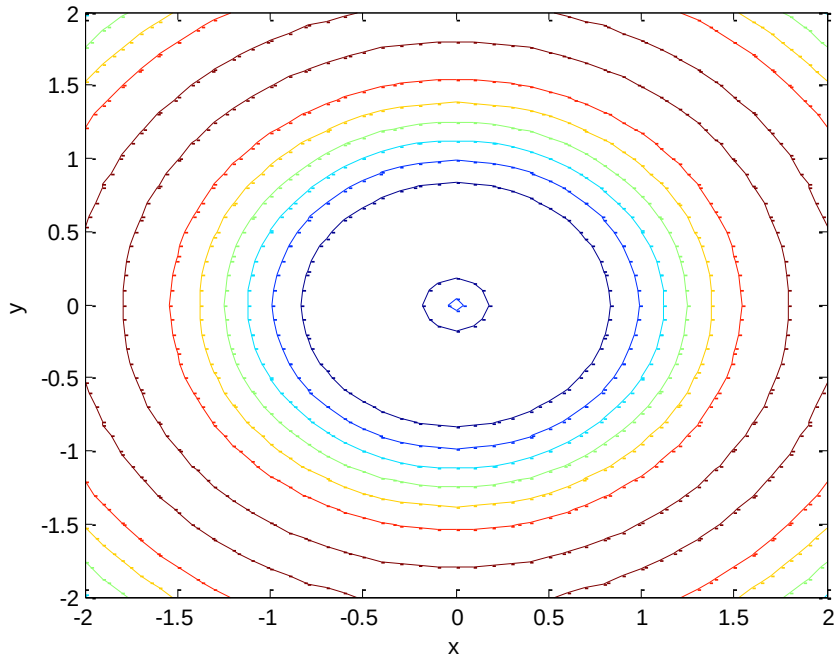
$$z = 0{,}5 + \frac{\sin^2(\sqrt{x^2 + y^2} - 0{,}5}{1 + 0{,}1(x^2 + y^2)} \tag{3.5}$$

Küresel minimum, x=0 ve y=0,5 koordinatlarında 0,5'tir.

F3 fonksiyonu yüzeyi Şekil 3.10'da, eş yükselti eğrileri ise Şekil 3.11'de gösterilmiştir:

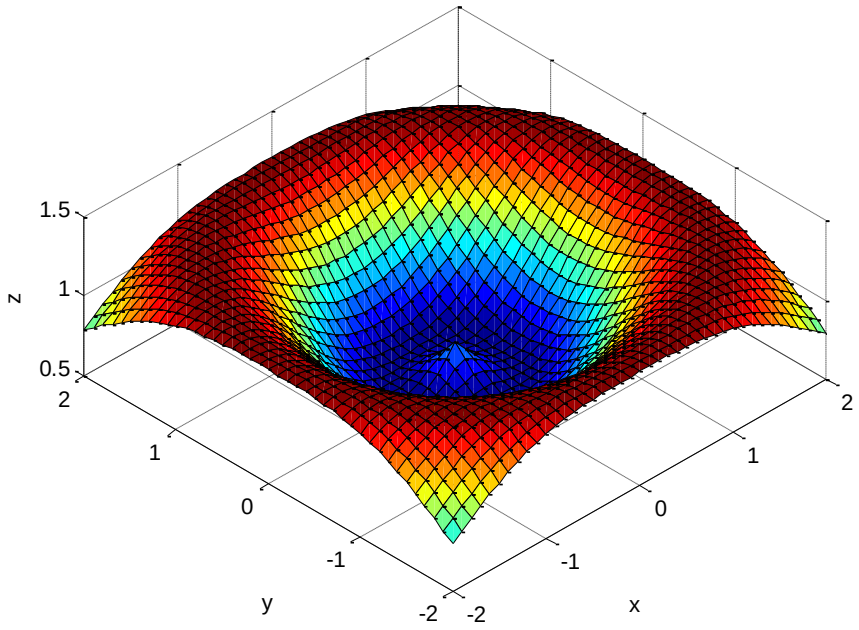


Şekil 3.10. Örnek 3 (F3) fonksiyonu yüzeyi.



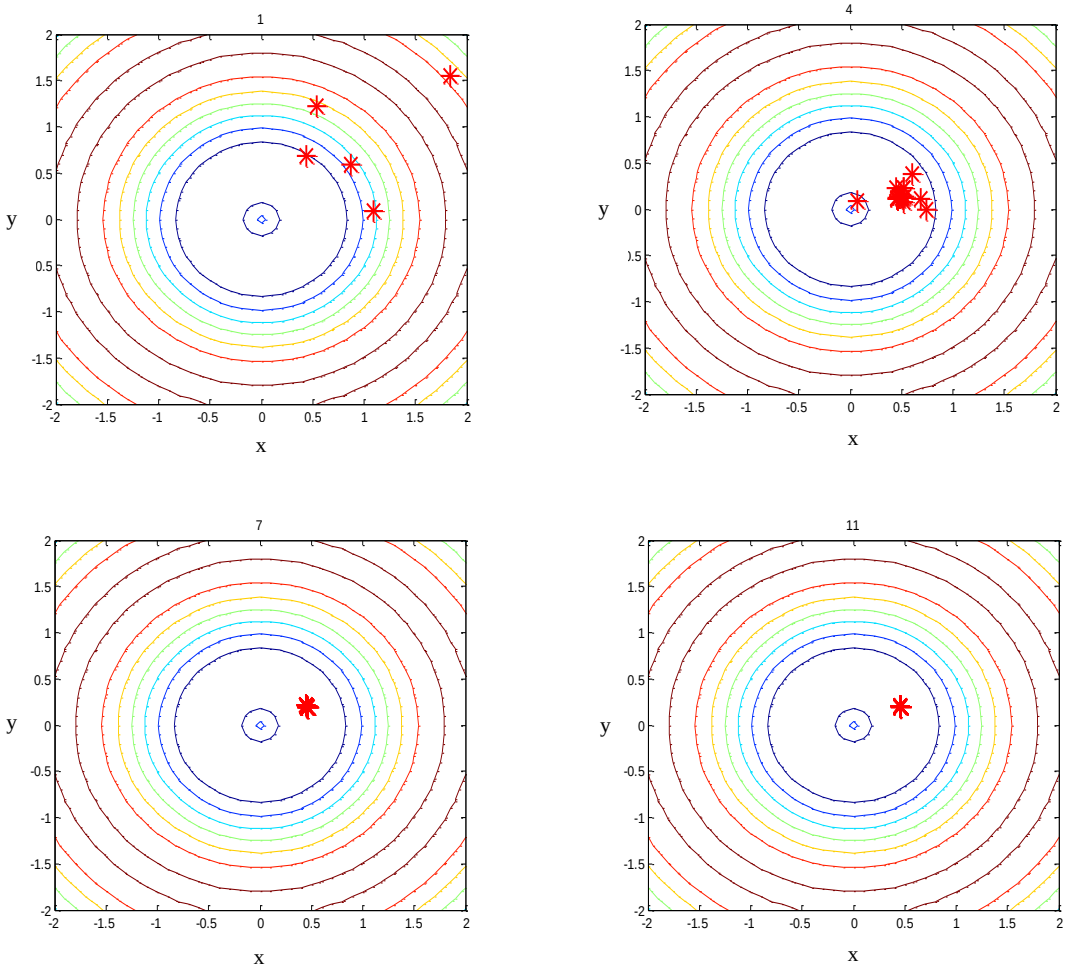
Şekil 3.11. Örnek 3 (F3) fonksiyonu eş yükselti eğrileri.

Şekil 3.10 ve Şekil 3.11'den de anlaşılacağı gibi bu fonksiyon diğer fonksiyonlardan daha farklıdır. Aslında Şekil 3.12'de görüldüğü gibi fonksiyonda hem tepe hem de çukur noktası vardır.



Şekil 3.12. F3 fonksiyonun detaylı global minimum noktası.

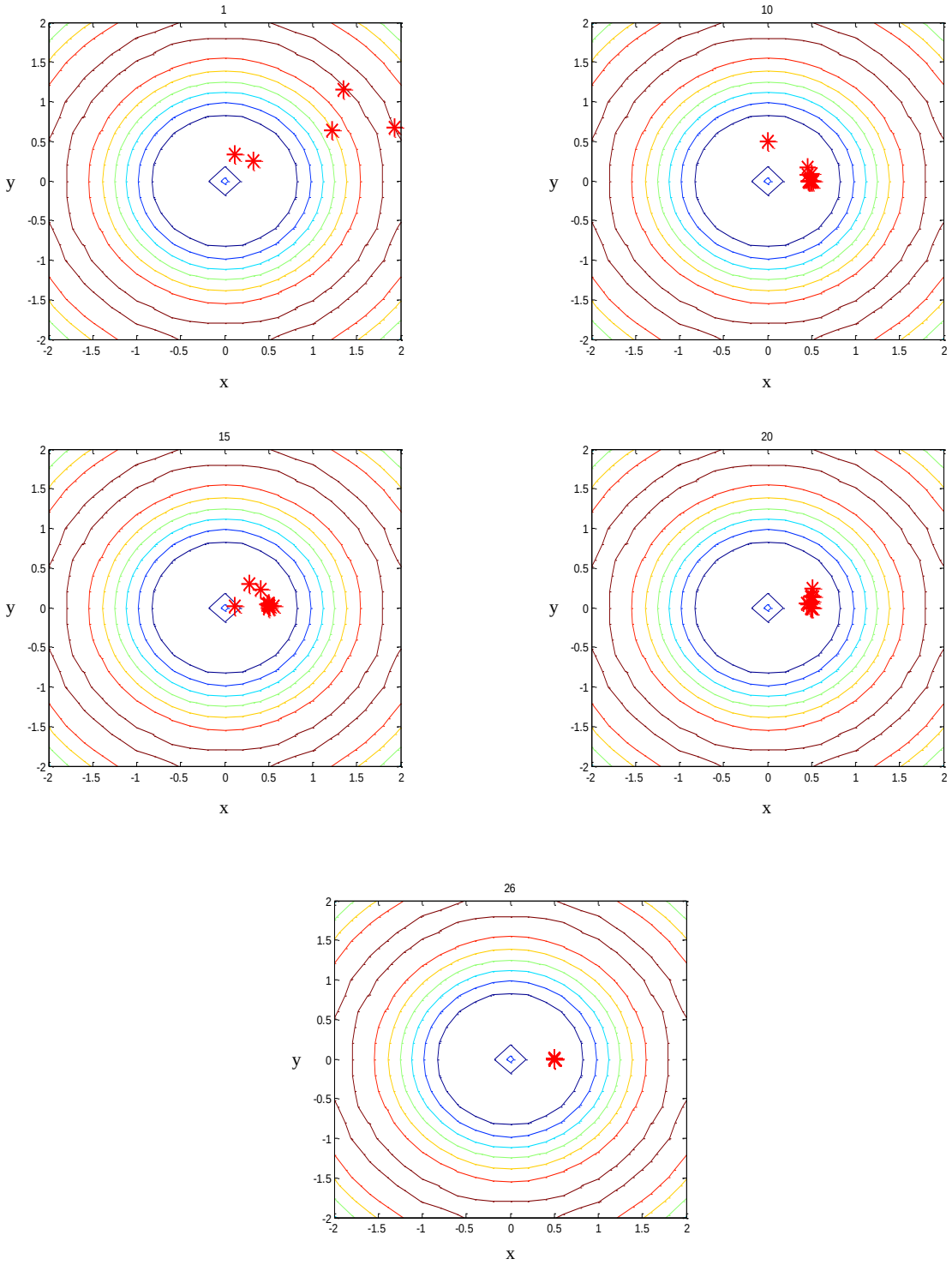
Standart yaklaşımla F3 fonksiyonu için S-COA çalıştırıldığında elde edilen sonuçlar Şekil 3.13'de gösterilmektedir.



Şekil 3.13. S-COA ile F3 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

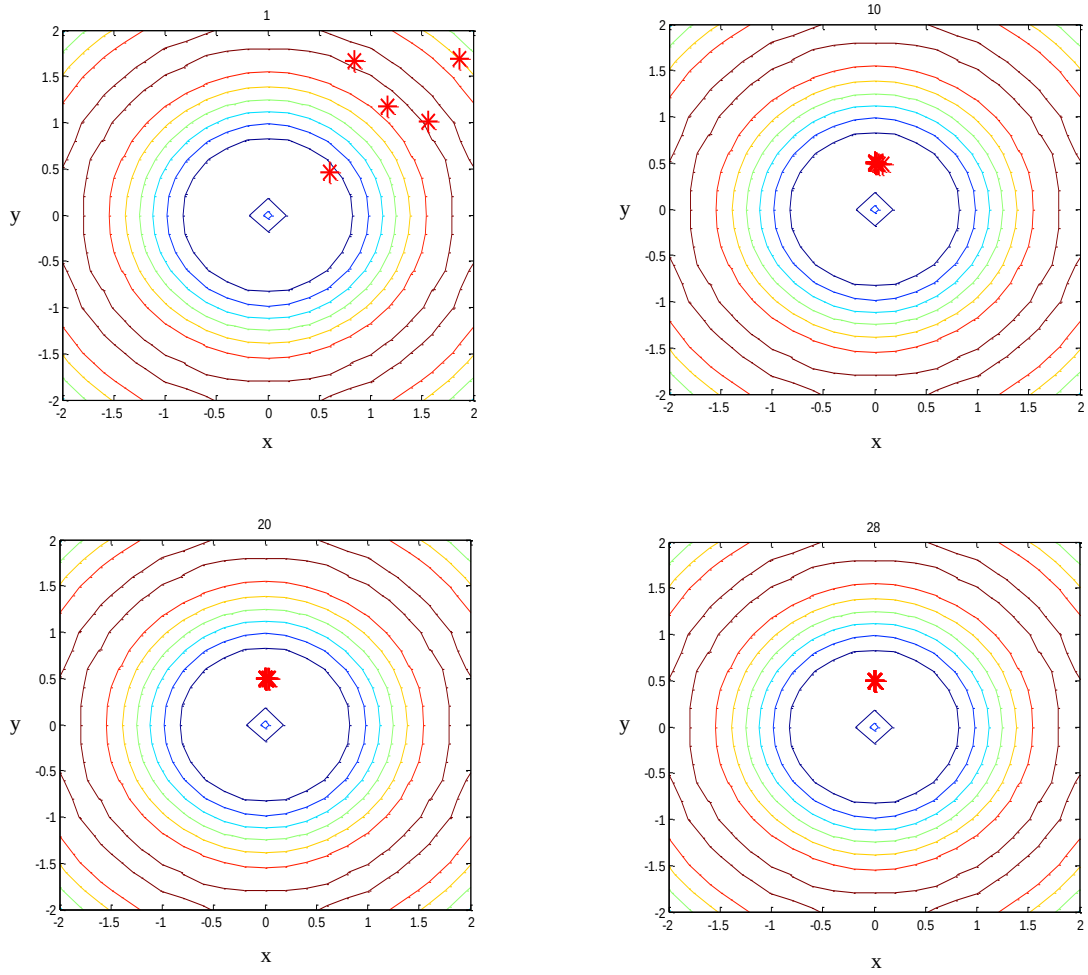
Şekil 3.13'den de görüleceği gibi F3 fonksiyonu için algoritma çalıştırıldığında çok kısa sürede istenilen yerel minimum noktasına gelinmektedir. Tam {0,0} noktasına gelmemesinin nedeni o noktanın aslında tam orta noktada bulunan bir tepe noktası olmasıdır. Bu nedenle, bireyler bu nokta etrafındaki çukurda toplanmaktadır.

Doğrusal yaklaşım kullanılarak çalıştırılan F3 fonksiyonundan elde edilen sonuçlar ve grafikler ise Şekil 3.14'de gösterilmektedir.



Şekil 3.14. D-COA ile F3 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Üstel yaklaşım kullanılarak çalıştırılan F3 fonksiyonundan elde edilen sonuçlar ve grafikler ise Şekil 3.15'de gösterilmektedir.



Şekil 3.15. Ü-COA ile F3 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

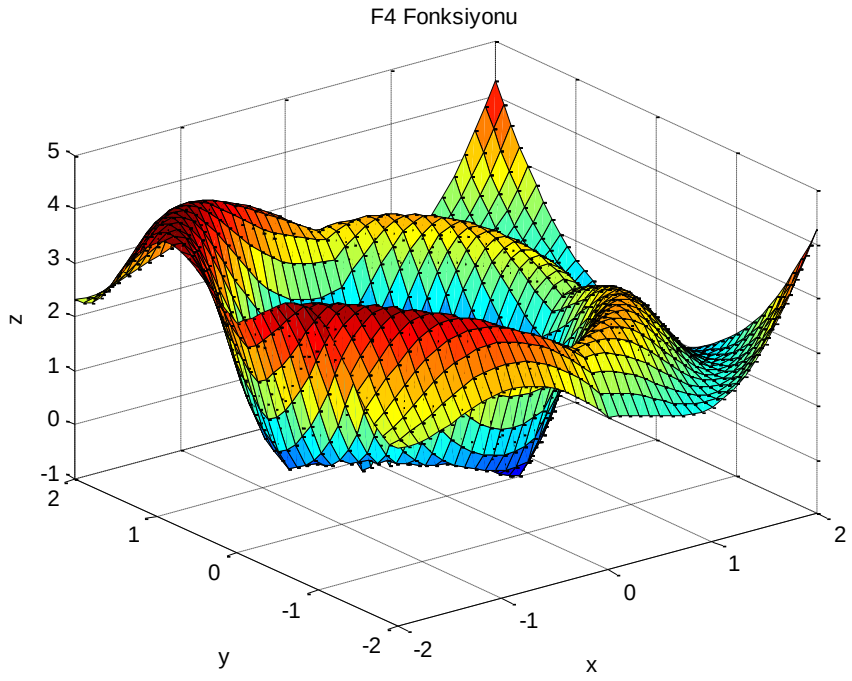
3.5. Örnek 4: F4

F4 fonksiyonu (Rajabioun, 2011), aşağıdaki gibi tanımlanmaktadır:

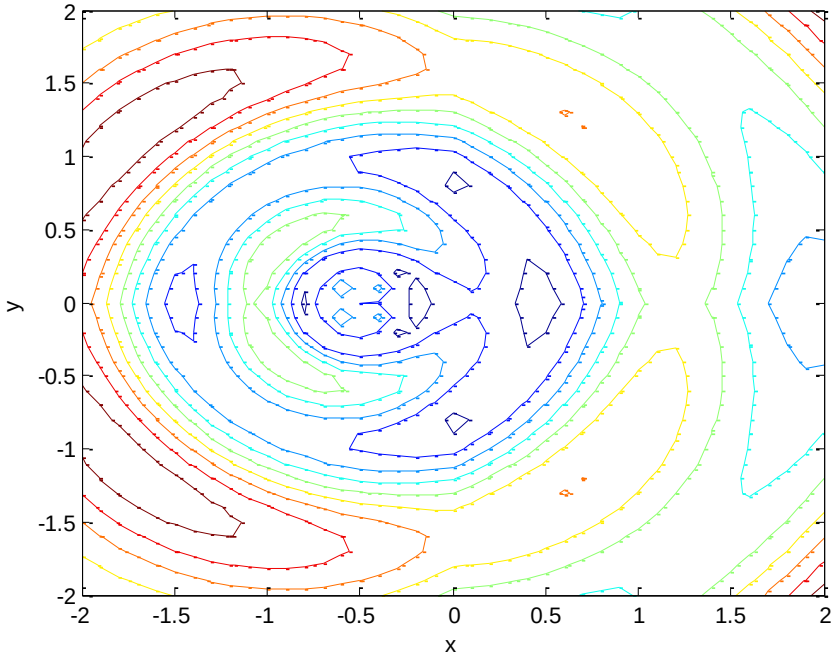
$$z = (x^2 + y^2)^{0,25} * \sin 30\left[(x+0,5)^2 + y^2\right]^{0,1} + |x| + |y| \qquad (3.6)$$

F4 fonksiyonu yüzeyi Şekil 3.16'da, eş yükselti eğrileri ise Şekil 3.17'de gösterilmiştir:

Küresel minimum, x=-0,2 ve y=0 koordinatlarında -0,2471' dir.



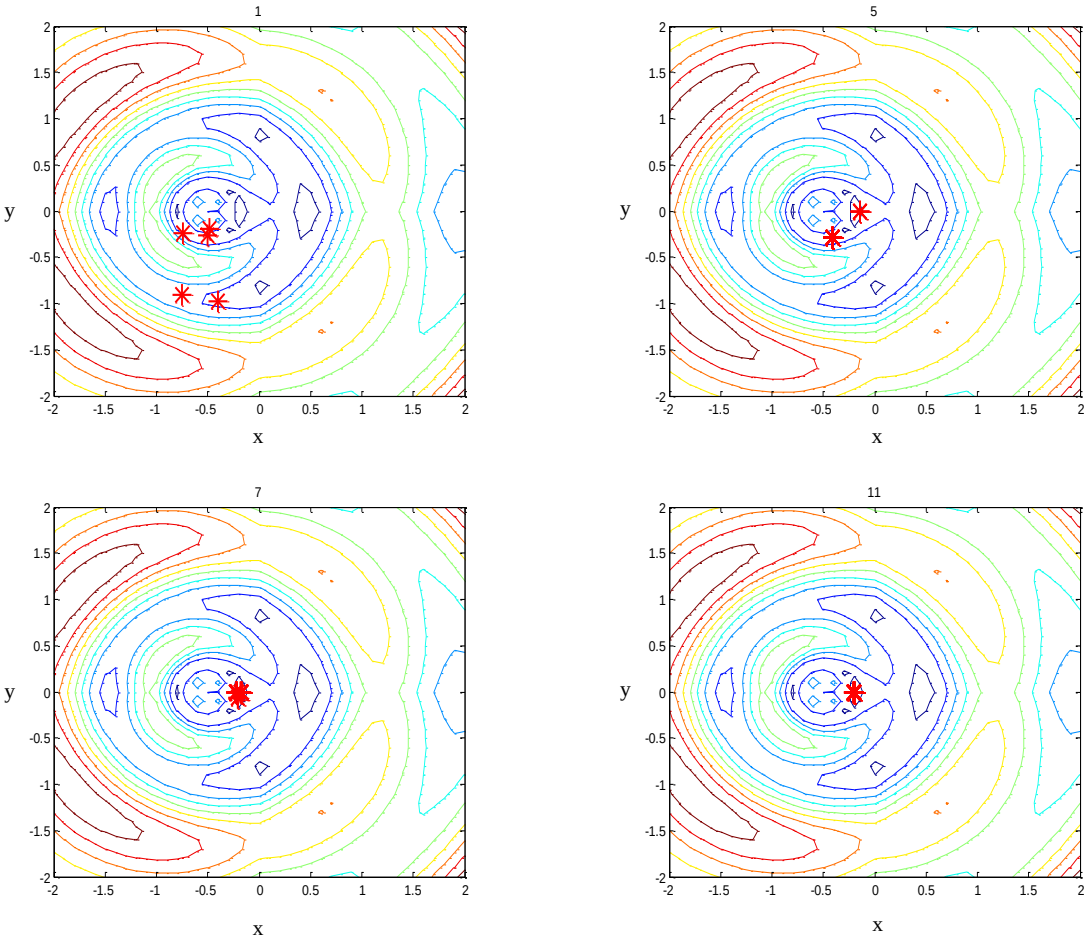
Şekil 3.16. F4 fonksiyonu yüzeyi.



Şekil 3.17. F4 fonksiyonu eş yükselti eğrileri.

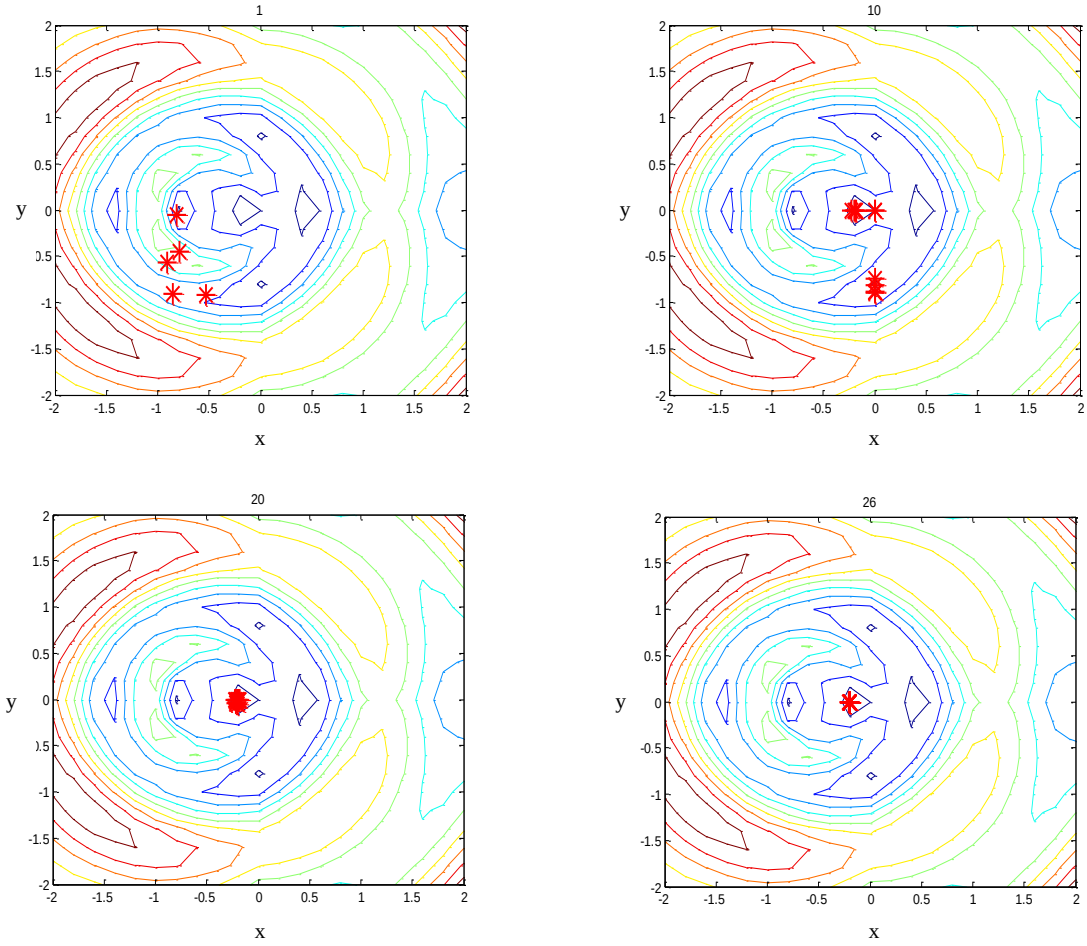
Şekil 3.16 ve Şekil 3.17'den görüleceği üzere F4 fonksiyonu birden fazla tepe ve çukur noktasına sahiptir. Eş yükselti eğrilerinde bu noktalar görülmektedir.

S-COA kullanılarak elde edilen sonuçlar Şekil 3.18'de gösterilmektedir.



Şekil 3.18. S-COA ile F4 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

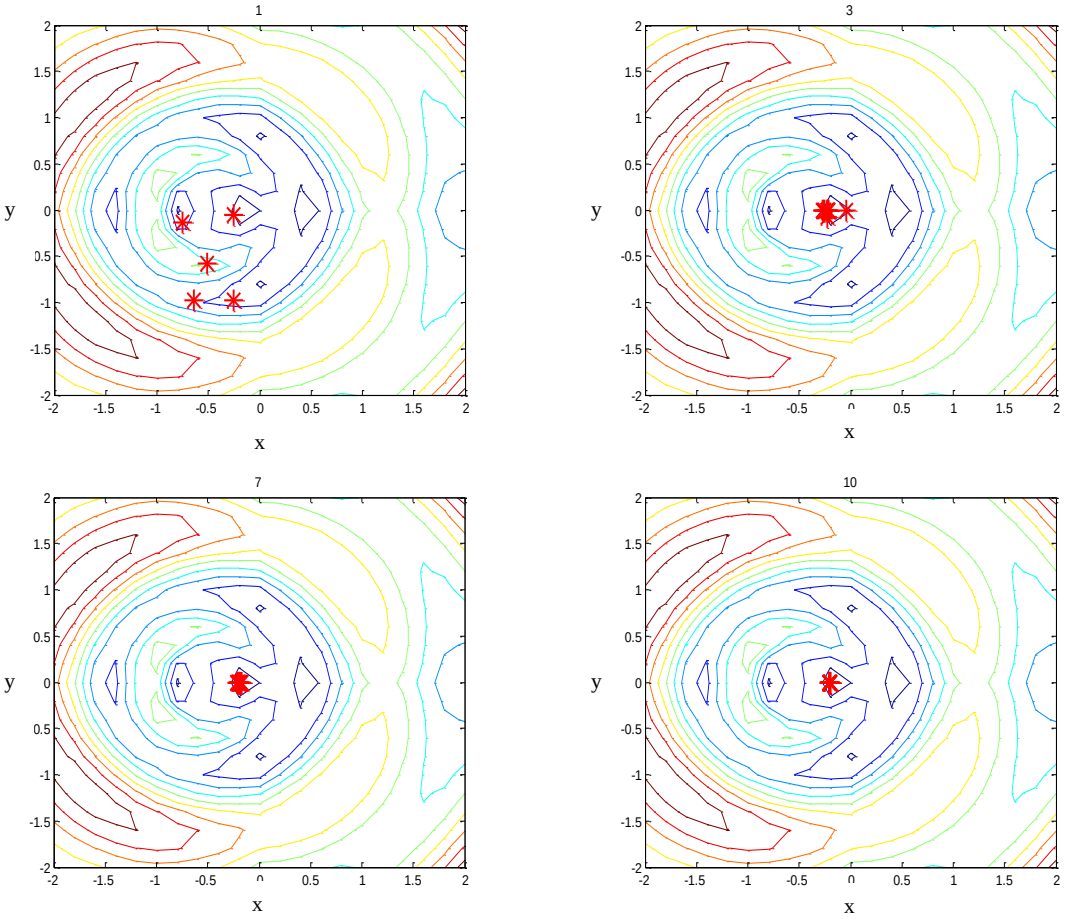
Doğrusal yaklaşım kullanılarak elde edilen muhtelif adımlardaki tekrar sonuçları ise Şekil 3.19'da gösterilmektedir.



Şekil 3.19. D-COA ile F4 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Standart yaklaşımda F4 fonksiyonu 11 tekrarda istenilen yerel minimum noktasına ulaşırken, doğrusal yaklaşımda 26. tekrarda istenilen minimum noktasına ulaşmaktadır. Üstel yaklaşımda ise yarıçap parametresinin üstel olarak azalmasından dolayı sonucu daha hızlı yerel minimum noktasına ulaştırmıştır.

Üstel yaklaşım kullanılarak elde edilen muhtelif adımlardaki tekrar sonuçları da Şekil 3.20'de gösterilmektedir.

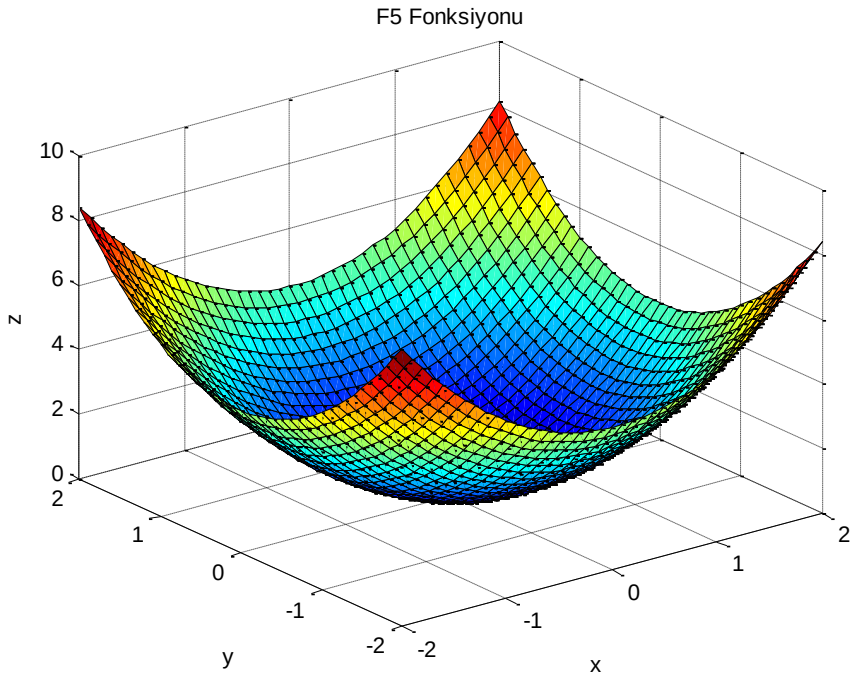


Şekil 3. 20. Ü-COA ile F4 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

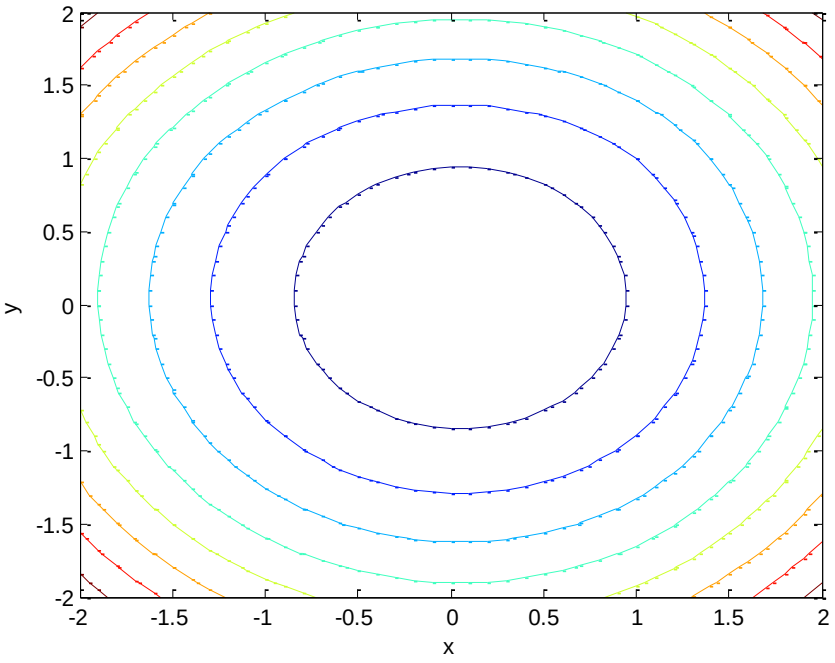
3.6. Örnek 5: F5

F5 fonksiyonu (Rajabioun, 2011) aşağıdaki gibi tanımlanmaktadır:

$$z = -0{,}1069\left(x^2 + y^2\right) + 0{,}1|1 - x| + 0{,}1|1 - y| \tag{3.7}$$

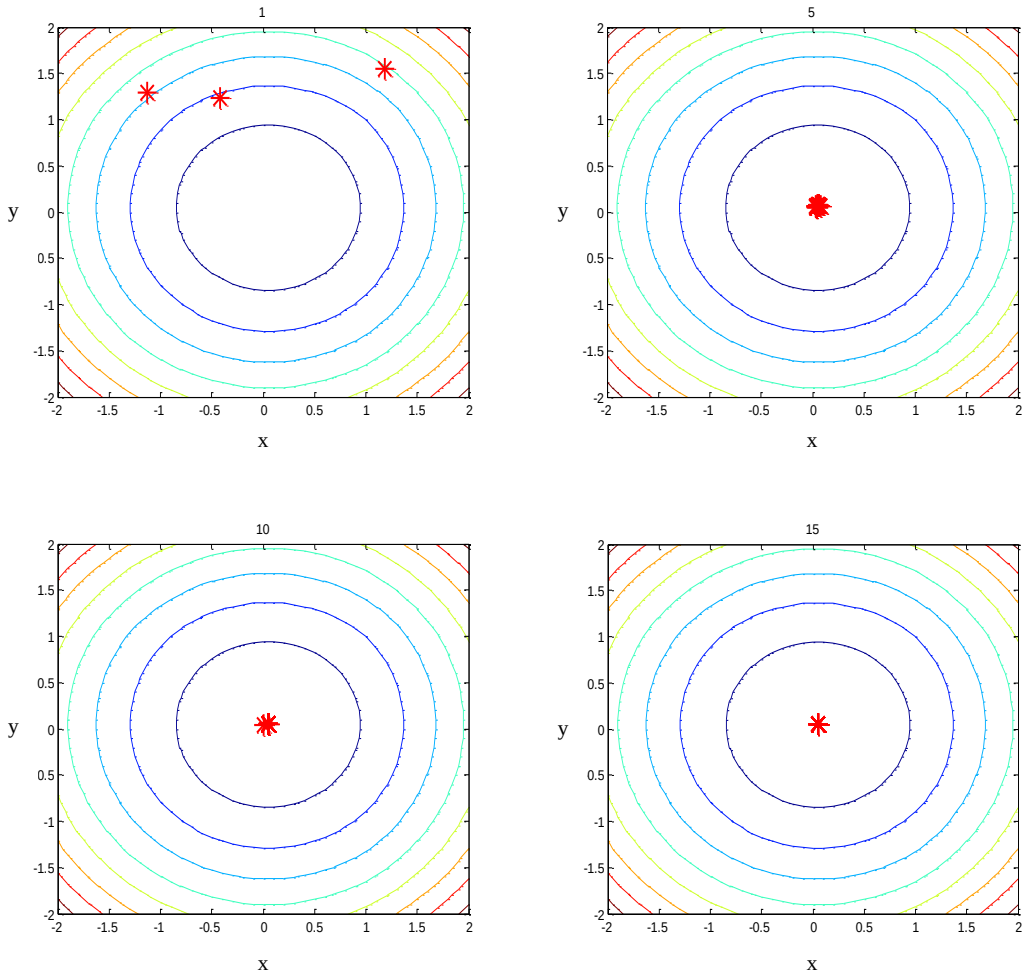


Şekil 3.21. F5 Fonksiyonu yüzeyi.



Şekil 3.22. F5 fonksiyonu eş yükselti eğrileri.

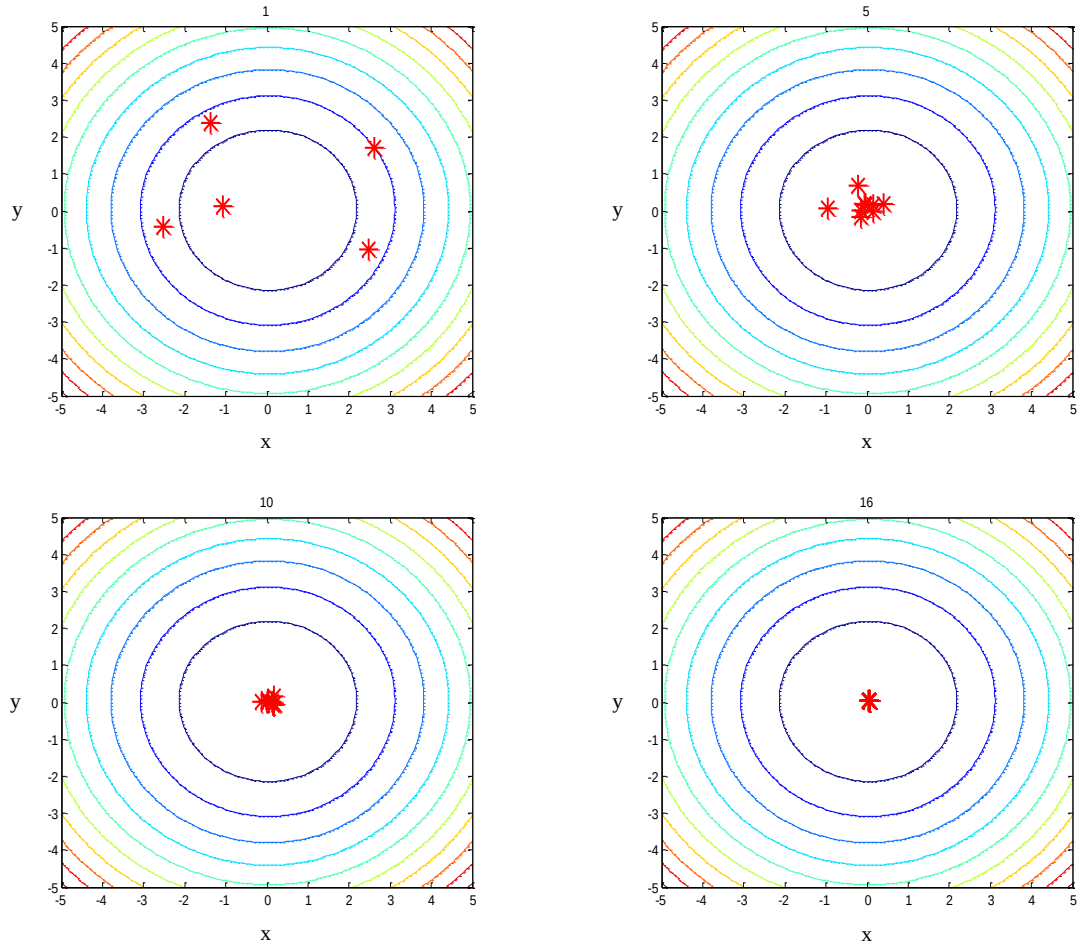
Şekil 3.23'de standart yaklaşım kullanılarak elde edilen sonuçlar bulunmaktadır.



Şekil 3.23. S-COA ile F5 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Doğrusal yaklaşım kullanılarak elde edilen pozisyonlar Şekil 3.24'de gösterilmektedir.

Üstel yaklaşımla elde edilen birey pozisyonları da Şekil 3.25'de gösterilmektedir:

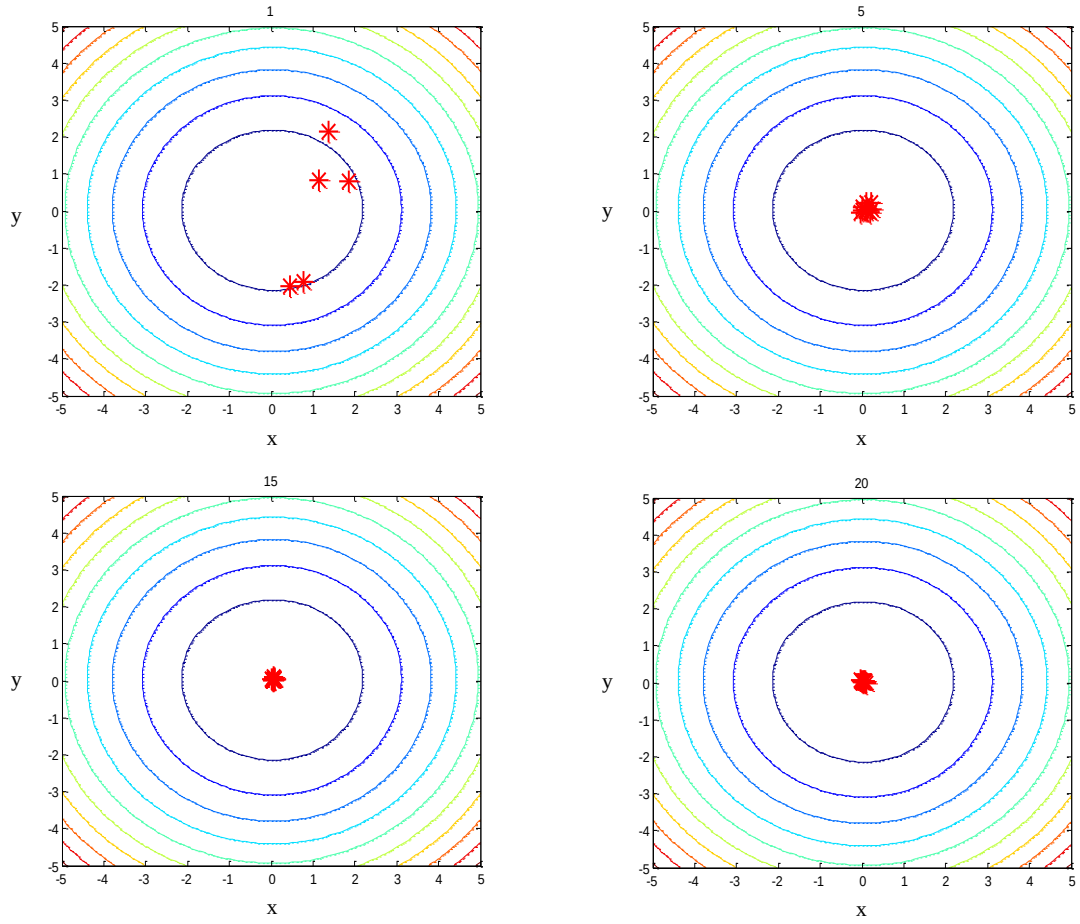


Şekil 3.24. D-COA ile F5 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

S-COA ile F5 fonksiyonu optimizasyonunda, Şekil 3.23'de görüldüğü gibi 15 adımda istenilen küresel minimum noktasına ulaşılmıştır.

D-COA ile F5 fonksiyon optimizasyonunda da Şekil 3.24'den görüleceği üzere 15 adımda istenilen küresel minimum noktasına ulaşılmıştır.

Şekil 3.25'e bakıldığında ise Ü-COA ile F5 fonksiyon optimizasyonu, 20 adımda küresel minimum noktasına ulaşılarak tamamlanmıştır.



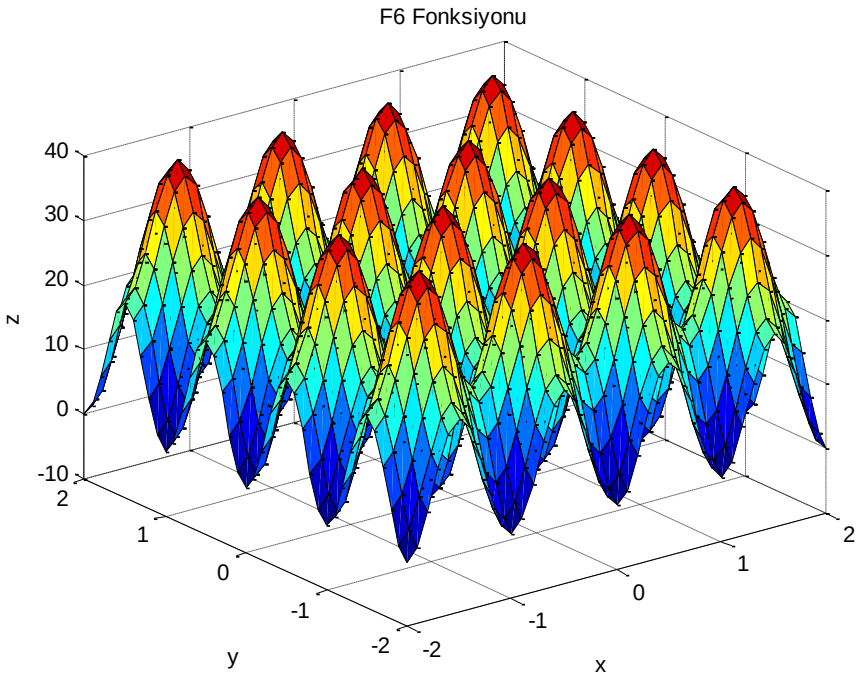
Şekil 3.25. Ü-COA ile F5 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

3.7. Örnek 6: F6

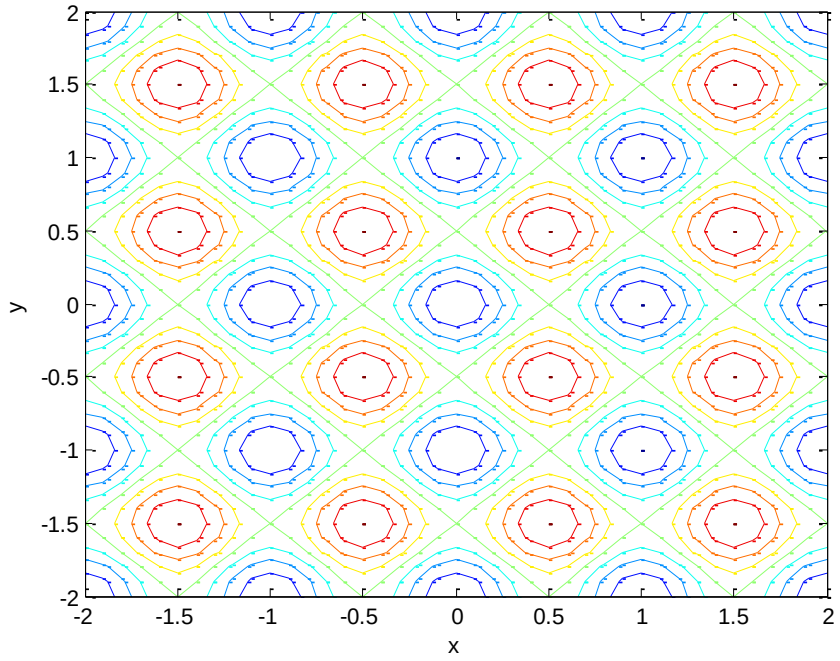
F6 fonksiyonu (Rajabioun, 2011), aşağıdaki gibi tanımlanmaktadır:

$$z = 10n + \sum_{i=1}^{n} (x_i^{\,2} - 10\cos(2\Pi x_i)) \tag{3.8}$$

Şekil 3.26'dan da görüleceği gibi bu fonksiyon birden fazla çukur ve tümseğe sahiptir. Bu fonksiyon kullanılarak, geliştirmiş olduğumuz 3 yöntemde algoritmanın çalışması aşağıda verildiği gibi gözlemlenmiştir. Küresel minimum, 10 boyutlu fonksiyon için $x_0,\ldots,x_9=0$ koordinatlarında 0'dır.

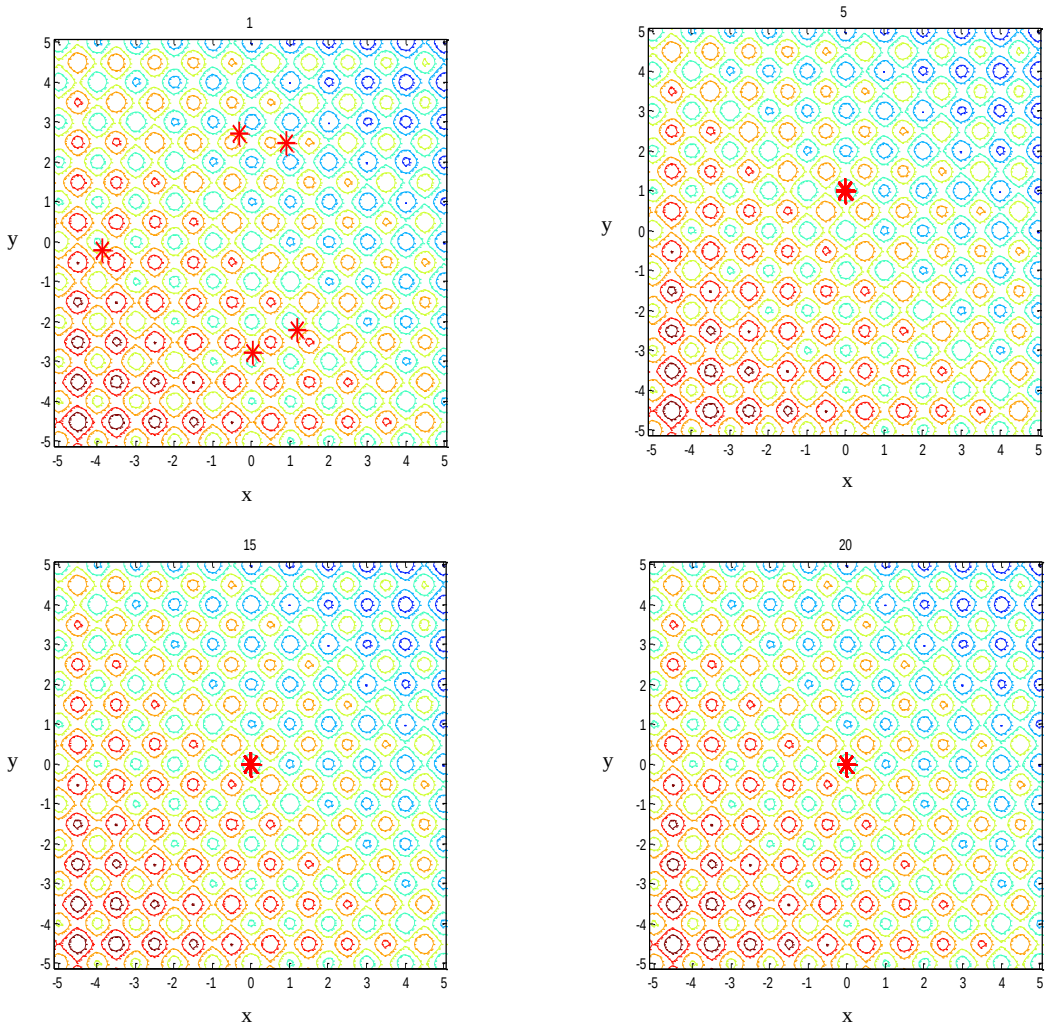


Şekil 3.26. F6 fonksiyonu yüzeyi.



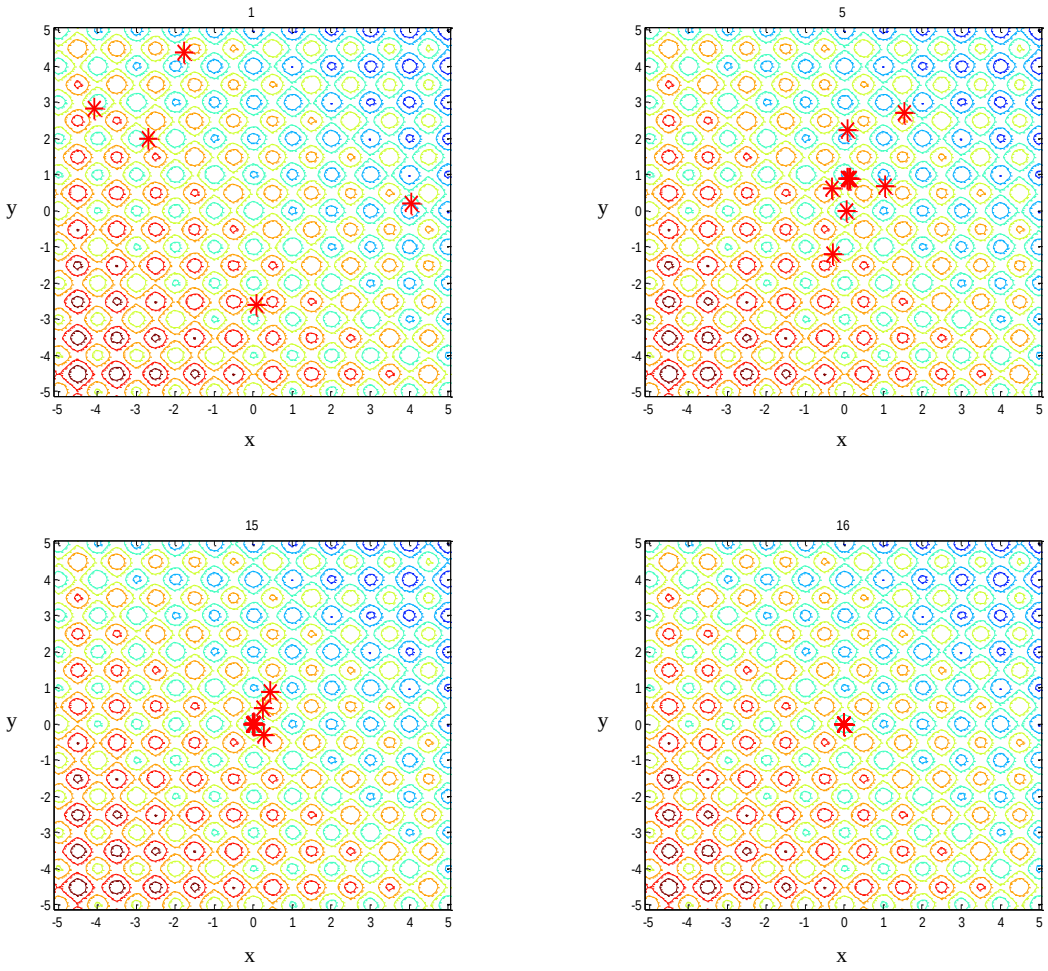
Şekil 3.27. F6 fonksiyonu eş yükselti eğrileri.

Standart yaklaşım kullanılarak elde edilen sonuçlar, Şekil 3.28'de gösterilmektedir.



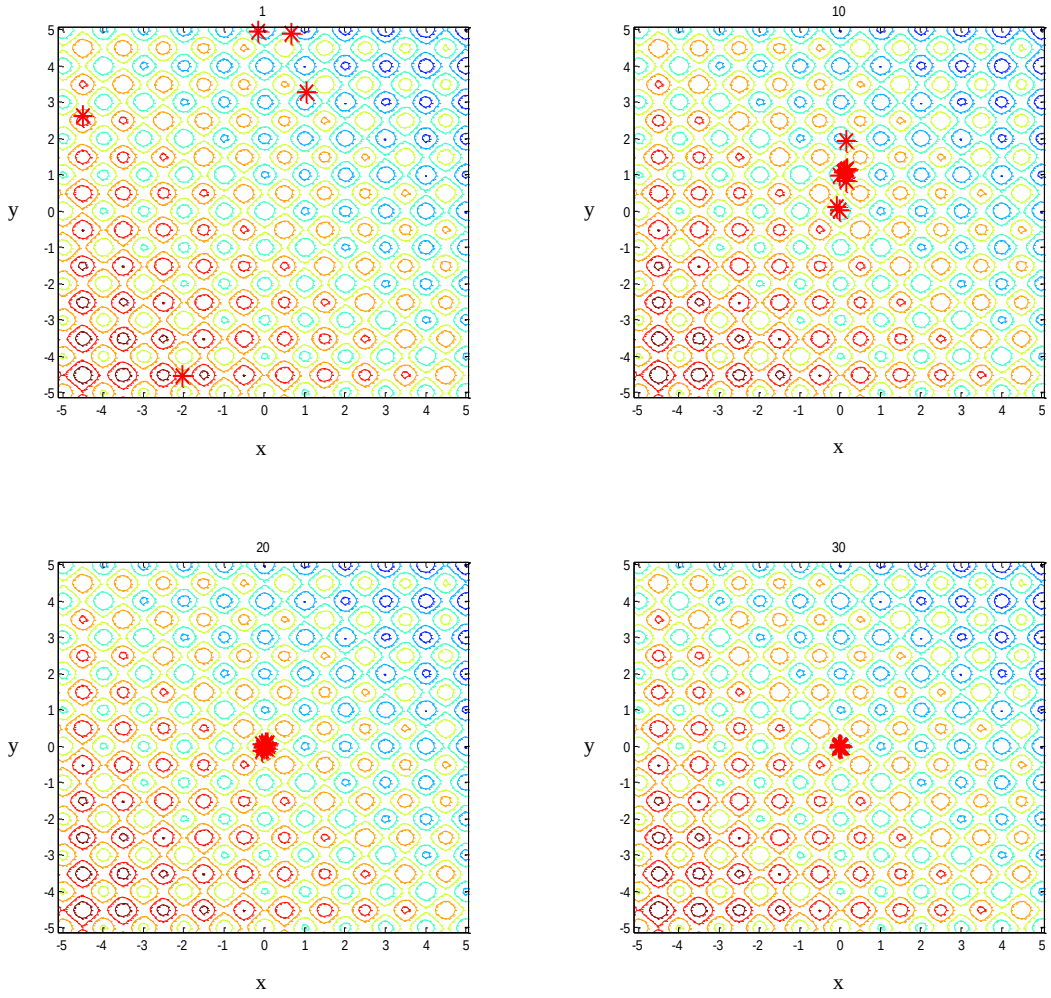
Şekil 3.28. S-COA ile F6 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Doğrusal yaklaşım kullanılarak elde edilen bireylerin pozisyonları Şekil 3.29'da gösterilmektedir:



Şekil 3.29. D-COA ile F6 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

Üstel yaklaşım kullanılarak elde edilen bireylerin pozisyonları ise Şekil 3.30'da gösterilmektedir:



Şekil 3.30. Ü-COA ile F6 fonksiyonu optimizasyonunun muhtelif adımlardaki birey pozisyonları.

3.8. Geliştirilen D-COA ve Ü-COA Başarım Değerlendirmesi

Bu çalışmada standart COA yanında iki algoritma geliştirilmiştir. Bunlar Doğrusal COA ve Üstel COA'dır. Doğrusal COA (D-COA), yumurtlama yarıçapının tekrarla doğrusal olarak azaltıldığı doğrusal yaklaşımdır. Üstel COA (Ü-COA) ise, yumurtlama yarıçapının tekrarla üstel olarak azaltıldığı yaklaşımdır. Geliştirilen COA yaklaşımların, standart COA ile yapılan başarım kıyaslaması Çizelge 3.1'de gösterilmiştir.

S-COA, D-COA ve Ü-COA kullanılarak yapılan fonksiyon optimizasyonunda, popülasyon büyüklüğü 20, tekrar sayısı 100 alınarak 100'er kez tekrar yapılmıştır. Elde edilen sonuçlar, Çizelge 3.1'de özetlenmiştir.

Çizelge 3.1. Geliştirilen yeni algoritmaların, standart COA ile yapılan başarım kıyaslamaları.

Fonksiyonlar	Yaklaşımlar	Global Minimum Bulma Sayısı	Lokal Minimuma Takılma Sayısı
F1	S-COA	65	35
	D-COA	65	35
	Ü-COA	72	28
F2	S-COA	27	73
	D-COA	55	45
	Ü-COA	22	78
F3	S-COA	100	0
	D-COA	100	0
	Ü-COA	100	0
F4	S-COA	67	33
	D-COA	89	11
	Ü-COA	35	65
F5	S-COA	100	0
	D-COA	100	0
	Ü-COA	100	0
F6	S-COA	47	53
	D-COA	65	35
	Ü-COA	69	31

3.9. Sonuç

Tez çalışmasında, algoritmanın çalışmasının anlaşılması bakımından kolay bir fonksiyon olduğu için başlangıç olarak "Peaks" fonksiyonu seçilmiştir. Diğer beş fonksiyon ise, Ramin Rajabioun'un makalesinde yer aldığı için karşılaştırma amacıyla tercih edilmiştir.

Çizelge 3.1'i inceleyecek olursak, burada geliştirilen algoritmalardan en zor olanların F2 ve F6 olduğu görülmektedir. F2 fonksiyonu zor bir fonksiyon olmasına rağmen geliştirilen D-COA'nın global minimumu bulma sayısı diğer algoritmalarına göre daha fazladır. F6 fonksiyonunda da yine geliştirilen algoritmaların ikisi de (D-COA ve Ü-COA) global minimumu bulma sayısı olarak standart COA'dan daha başarılıdır. Bu sonuçlara göre, geliştirilen D-COA ve Ü-COA zor olan fonksiyonlarda standart COA'ya göre daha iyi çalışmakta ve istenilen sonuçlar vermektedir.

4. COA İLE BULANIK SİSTEM OPTİMİZASYONU

4.1. Giriş

Bu bölümde, bu tez çalışmasında COA'nın bulanık sistem modellemede başarımını incelemek için kullanılan örnek dinamik sistemler, bulanık çıkarım sistemi ve sezgisel algoritmalardan bahsedilmiştir.

Literatürden alınan beş örnek dinamik sistemin (ÖDS) bulanık mantık tabanlı modellenmesinde COA ve bu çalışmada geliştirilen iki adet (D-COA, Ü-COA) türevinin başarımı incelenmiştir. Bu incelemede bulanık sistem olarak ANFIS bulanık modeli tercih edilmiştir. COA'nın başarımı örnek dinamik sistemlerin modellenmesinde ANFIS parametrelerinin optimizasyonu problemi üzerinde irdelenmiştir.

Şekil 4.1'de ÖDS sistemleri için sistem tanımanın eğitim fazında kullanılan giriş (u(k)) dizileri, Şekil 4.2'de ÖDS sistemleri için sistem tanımanın test fazında kullanılan giriş (u(k)) dizileri gösterilmektedir. Yukarıda tanımlanan sistem ve yapılar başka tez çalışmasında da kullanılmış olup bu çalışmada kıyaslama olması açısından aynı sistem ve yapıların kullanılması tercih edilmiştir(Yıldırım, 2013).

Çizelge 4.1'de bu çalışmada kullanılan ÖDS'ler tanımlanmış olup, bu sistemlerin ANFIS ile modellenmesinde kullanılan girişler, üyelik fonksiyonu sayıları, kural sayıları ve optimize edilecek parametre sayısı Çizelge 4.2'de listelenmiştir.

4.2. Örnek Dinamik Sistemler

Bu çalışmada COA'nın bulanık sistem modellemede başarımını incelemek amaçlı kullanılmak üzere Çizelge 4.1'de verilen ve literatürde sıkça kullanılan dinamik sistemler kullanılmıştır.

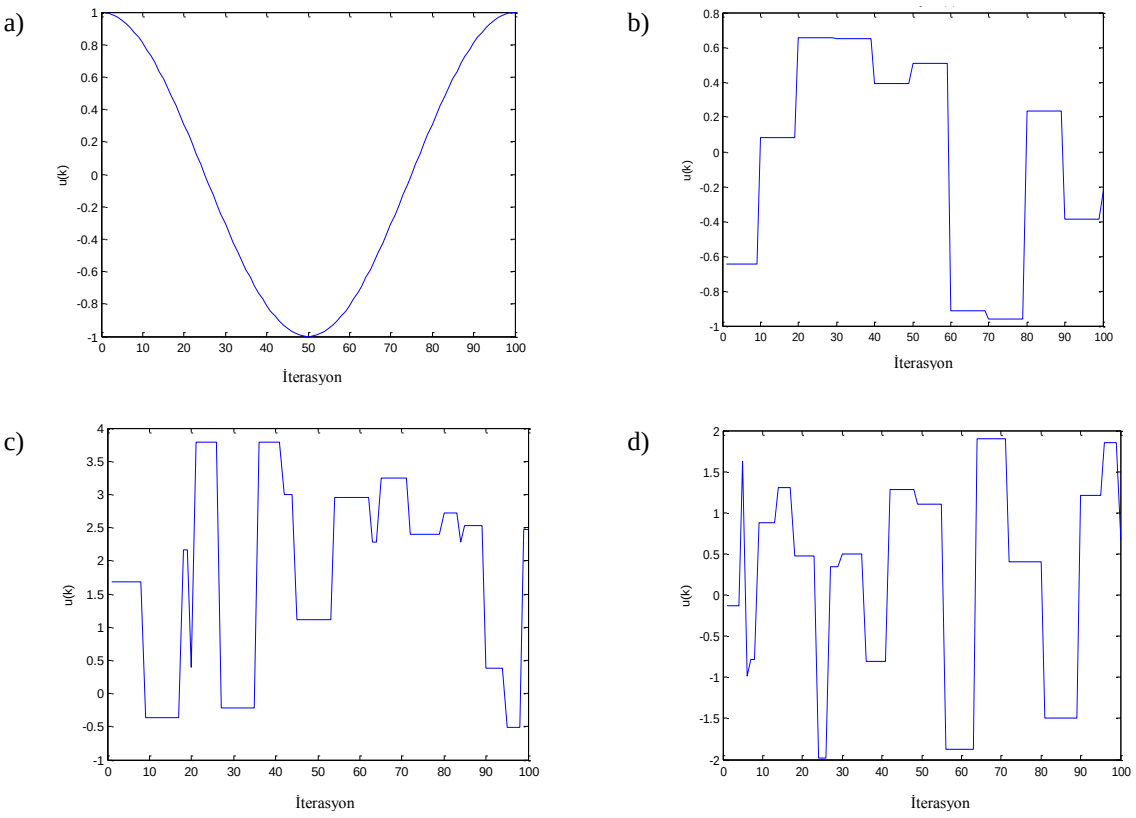
Çizelge 4.1. COA öğrenmeli bulanık mantık tabanlı dinamik sistem tanıma/modelleme için literatürden seçilen örnek dinamik sistemler (ÖDS).

	Örnek Dinamik Sistemler	**Eğitim Seti**	**Test Seti**
1	$y(k)=\frac{y(k-1)\cdot y(k-2)\cdot\left(y(k-1)+2.5\right)}{1+y^2(k-1)+y^2(k-2)}+u(k)$, (Narendra,1990)	$u(k)=\cos\frac{2\pi k}{100}$	$u(k)=\sin\frac{2\pi k}{25}$
2	$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k)$, (Narendra,1990)	$u(k)=\cos\frac{2\pi k}{100}$	$u(k)=\sin\frac{2\pi k}{25}$
3	$y(k+1)=y(k)+u(k)e^{-3\lvert y(k)\rvert}$, (Babuska,2012)	[-1 1] Aralığında rasgele genlikli	[-1 1] Aralığında rasgele genlikli
4	$y(k+1)=\frac{24+y(k)}{30}y(k)-0.8\frac{u(k)^2}{1+u(k)^2}y(k-1)+0.5u(k)$, (Oussar,1998)	[-5 5] Aralığında rasgele genlikli	[-5 5] Aralığında rasgele genlikli
5	$y(k+1)=0.5*(\frac{y(k)}{1+y^2(k)}+(1+u(k))u(k)(1-u(k)))$ (Sastry,1994)	[-2 2] Aralığında rasgele genlikli	[-2 2] Aralığında rasgele genlikli

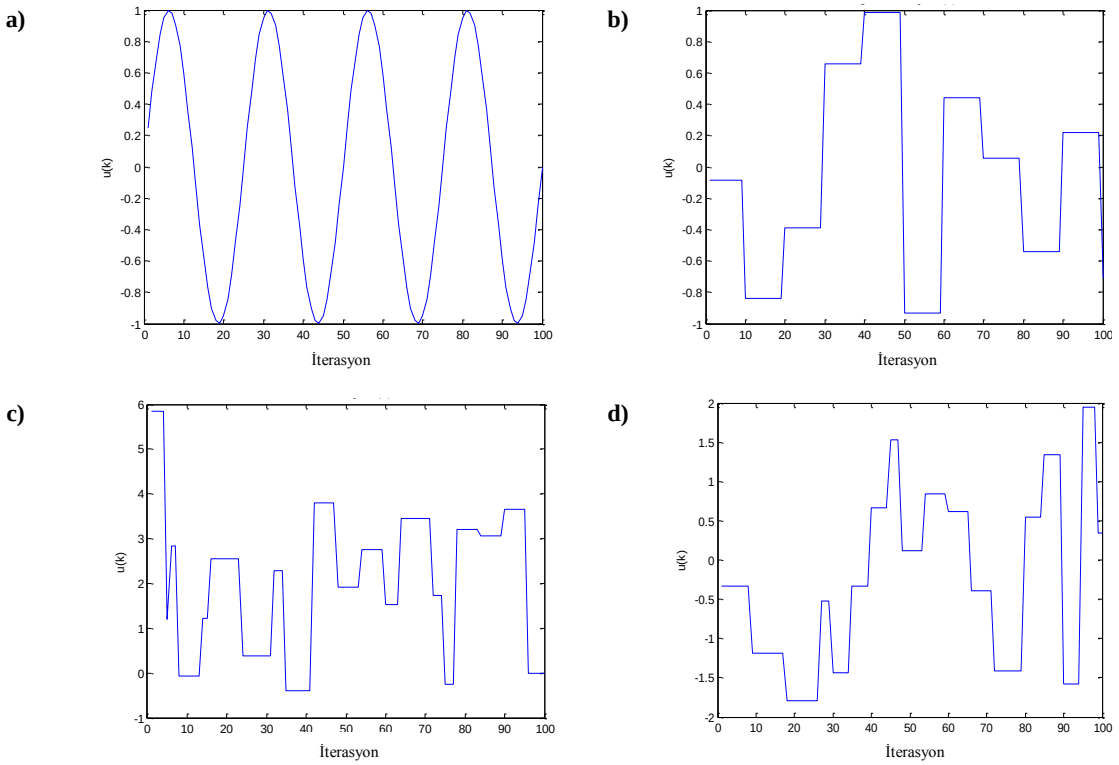
Şekil 4.1'de, uygulamada kullanılacak olan örnek dinamik sistemlerin girişlerinde kullanılan u(k) dizileri verilmiştir. Uygulamada örnek dinamik sistemler modellenirken Şekil 4.3'de verilen ANFIS yapısı kullanılmıştır. Her bir örnek dinamik sistem için ANFIS yapısı oluşturulurken Çizelge 4.2'de tanımlanan ANFIS dikkate alınmıştır. Her bir sistemin girişleri belirlendikten sonra, her girişe ait "Gauss" üyelik fonksiyonu (ÜF) ve kural sayıları tanımlanmıştır.

Çizelge 4.2. Her bir ÖDS için kullanılan ANFIS yapısı.

ÖDS No	Girişler	Giriş ÜF sayıları	Kural sayısı	Ayarlanacak parametre sayısı (D)
1	*u(k), y(k-2), y(k-1)*	2, 2, 2	8	36
2	*u(k), y(k), y(k-1)*	2, 2, 2	8	36
3	*u(k), y(k)*	2, 2	4	20
4	*u(k), y(k), y(k-1)*	2, 2, 2	8	36
5	*u(k), y(k)*	2, 2	4	20



Şekil 4.1. ÖDS sistemleri için sistem tanımanın eğitim fazında kullanılan giriş (u(k)) dizileri: (a) ÖDS 1 ve 2 için, (b) ÖDS 3 için [-1 1] aralığında rasgele genlikli 10 örnekleme periyotlu darbe (c) ÖDS 4 için [-5 5] aralığında rasgele genlikli ve rasgele örnekleme periyotlu darbe (d) ÖDS 5 için [-2 2] aralığında rasgele genlikli ve rasgele örnekleme periyotlu darbe.



Şekil 4.2. ÖDS sistemleri için sistem tanımanın test fazında kullanılan giriş (u(k)) dizileri: (a) ÖDS 1 ve 2 için, (b) ÖDS 3 için [-1 1] aralığında rasgele genlikli 10 örnekleme periyotlu darbe (c) ÖDS 4 için [-5 5] aralığında rasgele genlikli ve rasgele örnekleme periyotlu darbe (d) ÖDS 5 için [-2 2] aralığında rasgele genlikli ve rasgele örnekleme periyotlu darbe.

4.3. ANFIS Bulanık Çıkarım Sistemi

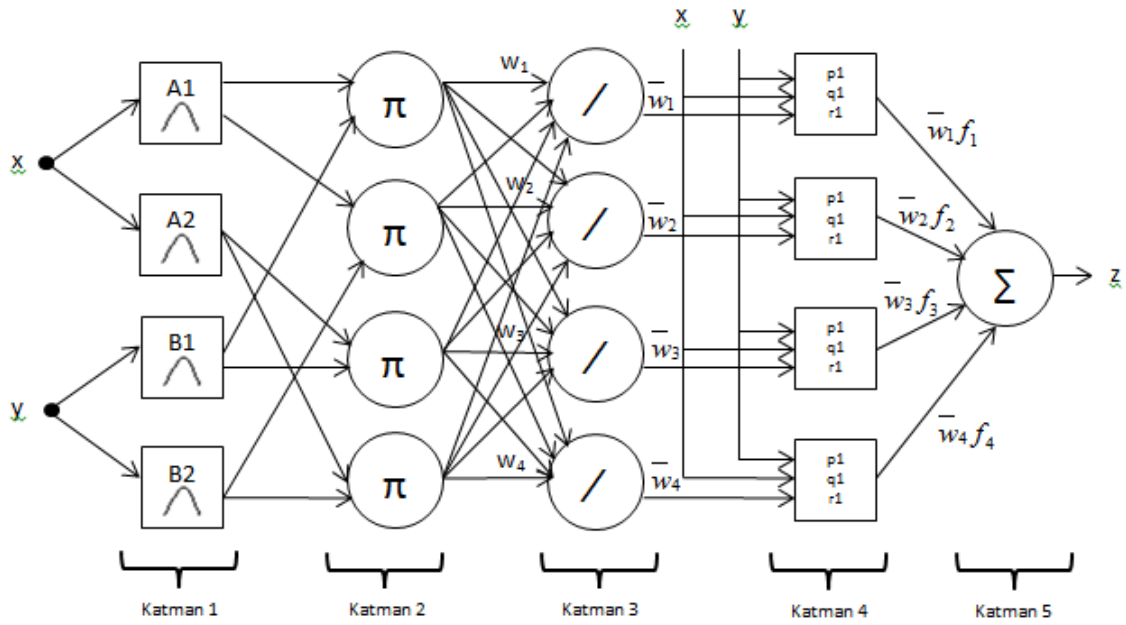
Bulanık mantık küme teorisine dayanan matematiksel bir disiplindir ve klasik küme gösteriminin genişletilmesidir. Bulanık mantığın ve bu mantık kurallarını kullanan bulanık küme teorisinin Lotfi A. Zadeh tarafından geliştirilip 1965 tarihli orijinal makalesinde yayınlanmasından sonra birçok mühendislik disiplininde kullanılagelmiştir(Zadeh, 1965).

Bulanık sistemler, bulanık küme teorisi, bulanık if-then kuralları ve bulanık mantığa dayalı popüler bir hesaplama yapısına sahiptir. Bulanık mantık ile sistem modellemede; bulanıklaştırma, bulanık çıkarım ve durulama olmak üzere üç temel adım vardır. Bulanıklaştırma aşamasında, girişler bulanıklaştırılır, yani sistem girişlerine uygun giriş üyelik fonksiyonları belirlenir, sonraki aşamada "if-then" kuralları kullanılarak bulanık kümelerin diğerleriyle ilişkilendirilmesi sağlanır. Durulama ile

bulanık çıkış değerlerinin kesin bir çıkış değerine dönüştürülmesi sağlanır (Şenol ve Yıldırım, 2008).

Sistem modellemek için bulanık girişimli çeşitli bulanık modeller vardır. Bunlardan en çok kullanılanları; Mamdani, Sugeno ve Tsukamoto bulanık modelleridir. Mamdani bulanık sistem modeli en basit model olup diğer modellerin temelini oluşturmaktadır. Mamdani bulanık mantık modelinin bir uyarlaması olan Sugeno bulanık mantık modeli ilk kez 1985 yılında kullanılmaya başlanmıştır.

Bulanık sistemlerin etkinliğinin artırılmasında, uyarlama tekniğinin katkısını sağlamaya yönelik çeşitli yöntemler geliştirilmiştir. Bunlardan biri de 1993'de Jang tarafından geliştirilmiştir (Jang, 1993). ANFIS Sugeno tipi bulanık sistemlerin yapısına sahip, her biri belli bir işlevi gerçekleştirmek üzere tasarlanmış 5 katmanlı bir ağ yapısına sahiptir. Şekil 4.3'de iki giriş, iki kural, tek çıkışlı Sugeno bulanık çıkarımına eşdeğer ANFIS mimarisi görülmektedir.



Şekil 4.3. İki girişli-tek çıkışlı birinci dereceden kural polinomlu ANFIS mimarisi.

Bu sistemin parametreleri 1.katmandaki üyelik fonksiyonlarının ve 4.katmanda imlenen kuralların parametreleridir. Bu parametreler uygun değerlere getirildiğinde sistem istenen (x,y)→z eşleşmesini gerçekleştirir.

COA'nın ve bu çalışmada geliştirilen iki türevinin bulanık sistemi eğitme başarımını ölçebilmek için yapılan deneysel çalışmalar ve erişilen bulgulara sonraki bölümlerde yer verilmiştir.

4.4. COA ve Geliştirilen Yaklaşımlar İle Modelleme

Çalışmada örnek dinamik sistemlerin tanıma/modellemesi yapılırken sadece standart COA kullanılmamıştır. Uygulama için standart COA modifiye edilerek iki adet geliştirilmiş COA (D-COA, Ü-COA) tanımlanmış ve belirlenen dinamik sistem tanıma problemleri üzerinde üç farklı COA ile çalışılmıştır. Bu algoritmaların kısaca tanımı aşağıda verilmiştir:

- **Standart COA (S-COA):** COA üzerinde herhangi bir değişiklik yapılmamış halidir. Sistem tanıma ve modelleme için kullanılan standart COA için yazılım kodları EK-1'de verilmiştir.

- **Geliştirilen D-COA:** Yumurtlama yarıçapının (yr) tekrarla doğrusal olarak azaltıldığı doğrusal yaklaşımdır. Geliştirilen bu yaklaşımda, yr doğrusal olarak azalmaktadır. $r_{\max}$ 10 ve $r_{\min}$ 0,005 olarak belirlenmiş ve 4.1'deki gibi tanımlanmıştır.

$$yr = r_{\max} - \left(\left(r_{\max} - r_{\min} \right) / G_{\max} \right) \times n \tag{4.1}$$

- **Geliştirilen Ü-COA:** Yumurtlama yarıçapının tekrarla üstel olarak azaltıldığı doğrusal yaklaşımdır. Bu yaklaşımda yumurtlama yarıçapı (yr) 4.2'deki gibi tanımlanmıştır.

$$yr = 0{,}95 \times e^{(-0{,}05 \times n)} \tag{4.2}$$

Matematiksel ifadelerde yer alan terimlerden yr yarıçap, $r_{\max}$ maksimum yarıçap, $r_{\min}$ minimum yarıçap, n tekrar no ve $G_{\max}$ maksimum nesil/tekrar sayısı anlamlarına gelmektedir.

4.5. Bireylerin Yapısı

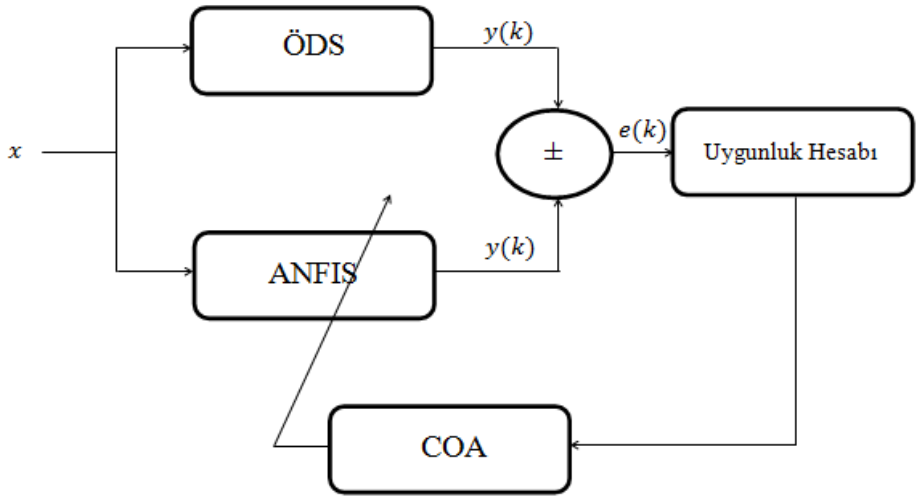
Çalışmada dinamik sistem modelleme aracı olarak Şekil 4.1'de verilen ANFIS yapısı kullanılmıştır. Çizelge 4.1'de verilen her bir ÖDS için Çizelge 4.2'de görüldüğü gibi girişler, giriş üyelik fonksiyon sayıları, belirlenen üyelik fonksiyonuna göre kural ve parametre sayısı belirlenmiştir. ANFIS bulanık mantık modellemesi yapılırken sistemler durumuna göre belirlenen girişler için ikişer adet üyelik fonksiyonu tanımlanmış ve bunlara ait kural sayıları Çizelge 4.2'de verilmiştir. ANFIS bulanık mantık modelinde sistemleri modelleyebilmek için Sugeno yöntemi ve üyelik fonksiyonu olarak yaygın olarak kullanılan Gauss fonksiyonu seçilmiştir. Buna göre ANFIS optimizasyonu için kullanılan bireylerin yapısı aşağıdaki gibidir.

$$b_i = [c_{11}c_{12}c_{13}...\sigma_{11}\sigma_{12}\sigma_{13}...p_1q_1r_1...p_Rq_Rr_R] \tag{4.3}$$

Eşitlikteki; c_{ij}, σ_{ij} ve $p_K\ q_K\ r_K$ sırasıyla i. girişin j. ÜF'sinin merkezini, i. Girişin j. ÜF'sinin standart sapmasını, k. kuralın sonuç (kural) parametrelerini göstermektedir.

4.6. COA İle ANFIS Parametre Optimizasyonu

Şekil 4.4'te COA ile ANFIS optimizasyonu blok yapısı görülmektedir. Blok yapısı incelendiğinde örnek dinamik sistemlerin ANFIS ile modellenmesi sonucunda elde edilen hatanın $f_i = \frac{1}{k_{max}}\sum_{k=1}^{k_{max}} e(k^2)$ ölçüt fonksiyonu ile COA ve türevleri kullanılarak en aza indirgenmesi sağlanmaya çalışılmaktadır.



Şekil 4.4. COA ile ANFIS optimizasyonu blok yapısı.

5. DİNAMİK SİSTEM MODELLEMEDE COA İLE BULANIK SİSTEM OPTİMİZASYONU

5.1. Giriş

Bu bölümde, COA, D-COA ve Ü-COA'nın optimizasyon başarımı bulanık mantık tabanlı dinamik sistem modelleme problemi üzerinde kıyaslamalı olarak incelenmiştir. Kıyaslama Bölüm 4 Çizelge 4.1'de verilen beş adet dinamik sistemin bulanık modellenmesi üzerine yapılmıştır.

Değerlendirilen üç algoritma her bir dinamik sistem için rasgele oluşturulmuş başlangıç nüfusu ile 50'şer kez ayrı ayrı koşturulmuştur. Her bir algoritmanın 50'şer kez koşmasıyla elde edilen sonuçlar istatistiki olarak irdelenmiştir.

Çizelge 4.1'de verilen örnek dinamik sistemler sırasıyla COA, D-COA ve Ü-COA kullanılarak ANFIS ile modellenmiş ve her bir sistem için elde edilen model sonuçları en iyi, en kötü, başlangıç durum ve son durum şeklinde kategorize edilerek aşağıdaki alt bölümlerde grafiksel olarak verilmiştir. Verilen şekiller 50'şer kez koşma sonrasında belirlenen en iyi koşma içindir. Verilen şekillerde mavi düz çizgi ile verilenler istenen sonucu, kırmızı kesikli çizgiyle verilen ise elde edilen sonucu göstermektedir.

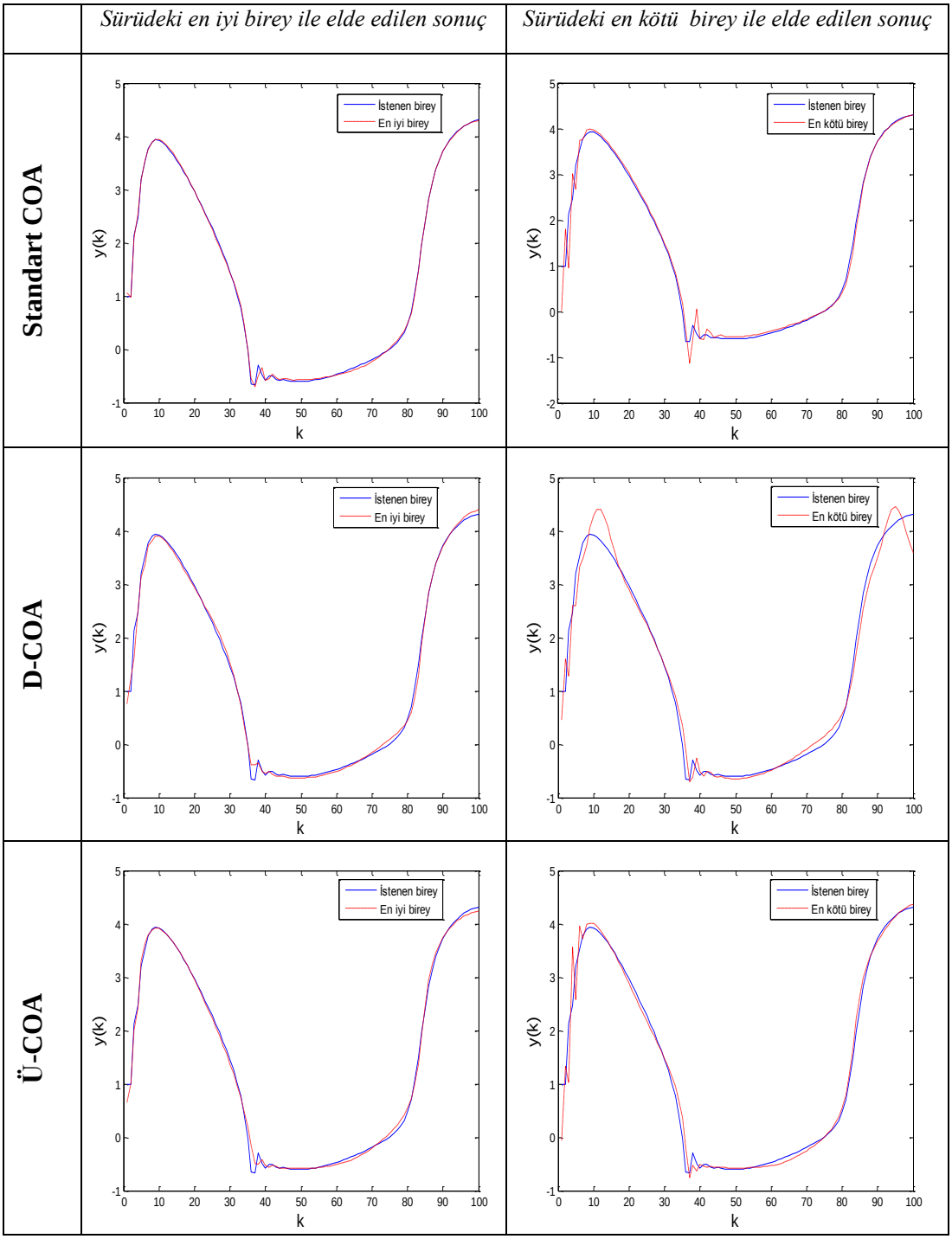
5.2. ÖDS 1'in Tanınması/Modellenmesi

5.2.1. ÖDS 1 için eğitim aşaması ve sonuçları

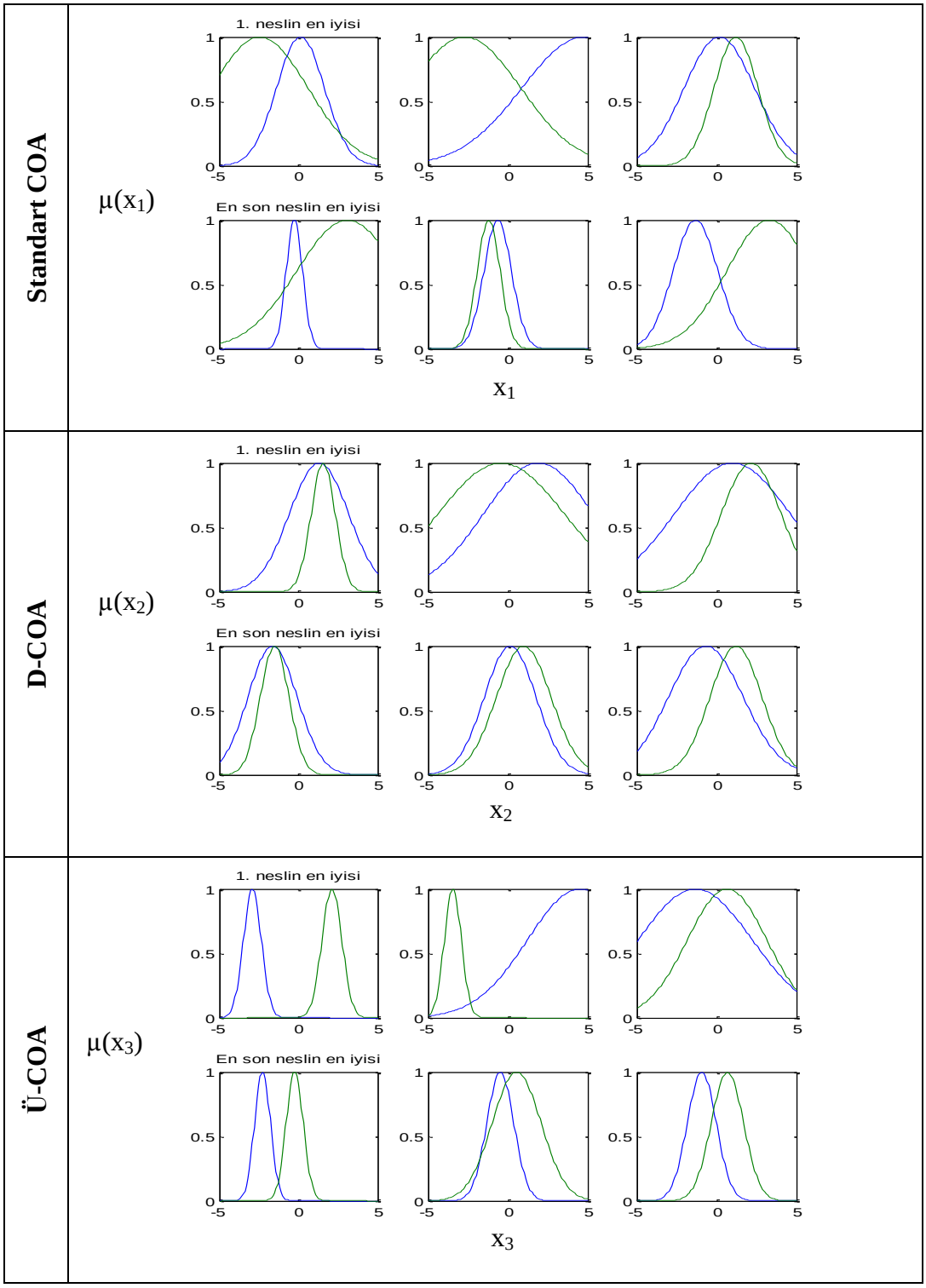
ÖDS 1 için Şekil 4.1(a)'da verilen u(k) kullanılarak oluşturulan eğitim seti dizileri kullanılarak koşturulan COA, D-COA ve Ü-COA'nın başarım sonuçları sırasıyla aşağıda verilmiştir. Bu uygulama sonucunda, COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 1 eğitim fazı için en iyi birey ile elde edilen en iyi ve en kötü birey ile elde edilen en kötü sonuç grafikleri Şekil 5.1'de verilmiştir.

Bu grafikler detaylı bir şekilde incelendiğinde tüm algoritmaların ÖDS1'in eğitim setine göre birbirlerine yakın sonuçlar elde ettikleri görülmüştür. Şekil 5.2'de standart COA, D-COA, Ü-COA'nın ÖDS 1 için bulanık modelleme eğitim fazında ÜF'lerin başlangıç ve son durum grafikleri verilmiştir. Bu şekillerde verilen grafikler ile

her bir algoritmanın üyelik fonksiyonları üzerinde yaptığı değişimler görülebilir. Şekillerde birinci sütunda verilen grafikler 1. girişe, 2. sütunda verilenler 2. girişe, 3. sütunda verilenler ise 3. girişe ait ÜF'lerin ilk ve son durumunu göstermektedir.

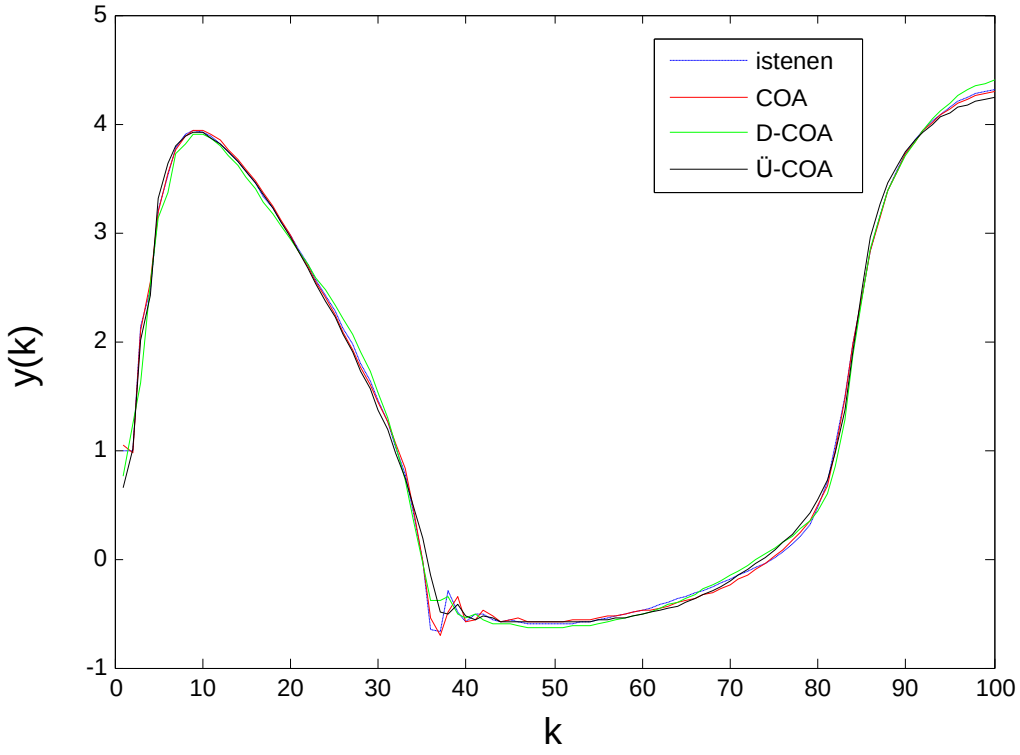


Şekil 5.1. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 1 için eğitim fazı sistem tanıma sonuçları.



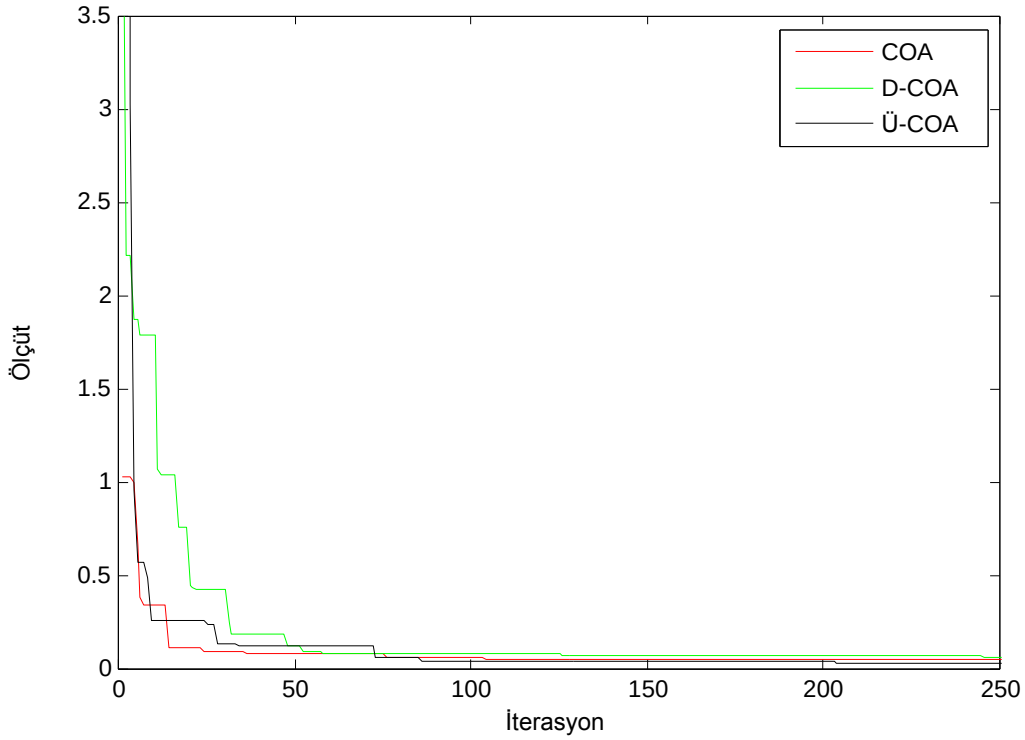
Şekil 5.2. COA öğrenmeli ANFIS ile ÖDS 1'in bulanık modelleme eğitim fazında ÜF'lerin başlangıç ve son durumları.

Şekil 5.3'de algoritmaların ÖDS 1 için elde ettikleri başarım görülmektedir. Buna göre COA ve Ü-COA istenene yakın sonuçlar elde ederken; D-COA başarımının ise diğerlerine göre daha kötü olduğu görülmektedir.



Şekil 5.3. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 1 için elde edilen modellerin karşılaştırılması.

ÖDS 1 için algoritmaların en iyi ölçüt değerlerinin karşılaştırılması Şekil 5.4'de verilmiştir. Verilen bu eğitim seyirlerini gösteren ölçüt değişimleri 50 ayrı koşmada her bir tekrarda elde edilen ölçüt değerlerinin ortalaması olarak bulunmuştur.



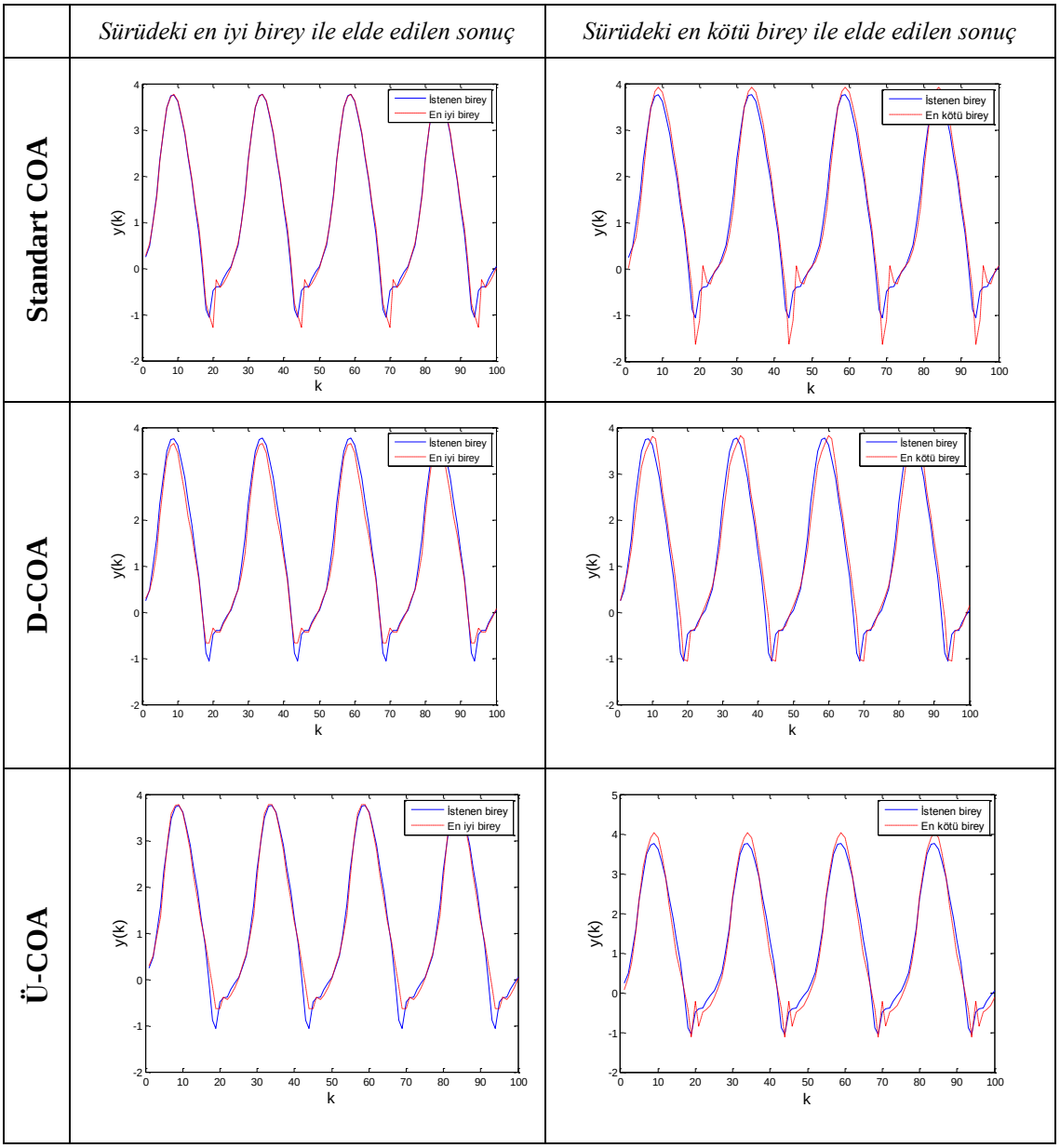
Şekil 5.4. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 1 modelleme eğitim seyri.

5.2.2. ÖDS 1'in test aşaması ve sonuçları

ÖDS 1 için Şekil 4.2(a)'da verilen u(k) kullanılarak oluşturulan veri seti (test seti) ile COA, D-COA ve Ü-COA tarafından eğitilen ANFIS modellerinin başarımı incelenmiştir. Çalışma ile COA'nın bulanık sistem optimizasyonundaki başarısı değerlendirilmiştir.

Şekil 4.1(a)'da verilen ÖDS 1 için girişlerden biri olan u(k) dizisi için eğitim fazında kullanılırken, test fazı için Şekil 4.2(b)'den de görüleceği üzere olarak kullanılmıştır. ÖDS 1 için giriş dizisi hazırlandıktan sonra, Çizelge 4.2'de belirlenen diğer girişlerle beraber test veri seti oluşturulmuştur. Oluşturulan test seti ile COA, D-COA ve Ü-COA ile elde edilen ÖDS 1 ANFIS modellerinin sonuçları Şekil 5.5 ve Şekil 5.6'da grafiksel olarak verilmiştir.

Şekil 5.5 ve Şekil 5.6'da verilen grafikler detaylı olarak incelendiğinde test fazında en iyi sonucun D-COA ile alındığı görülmüştür.

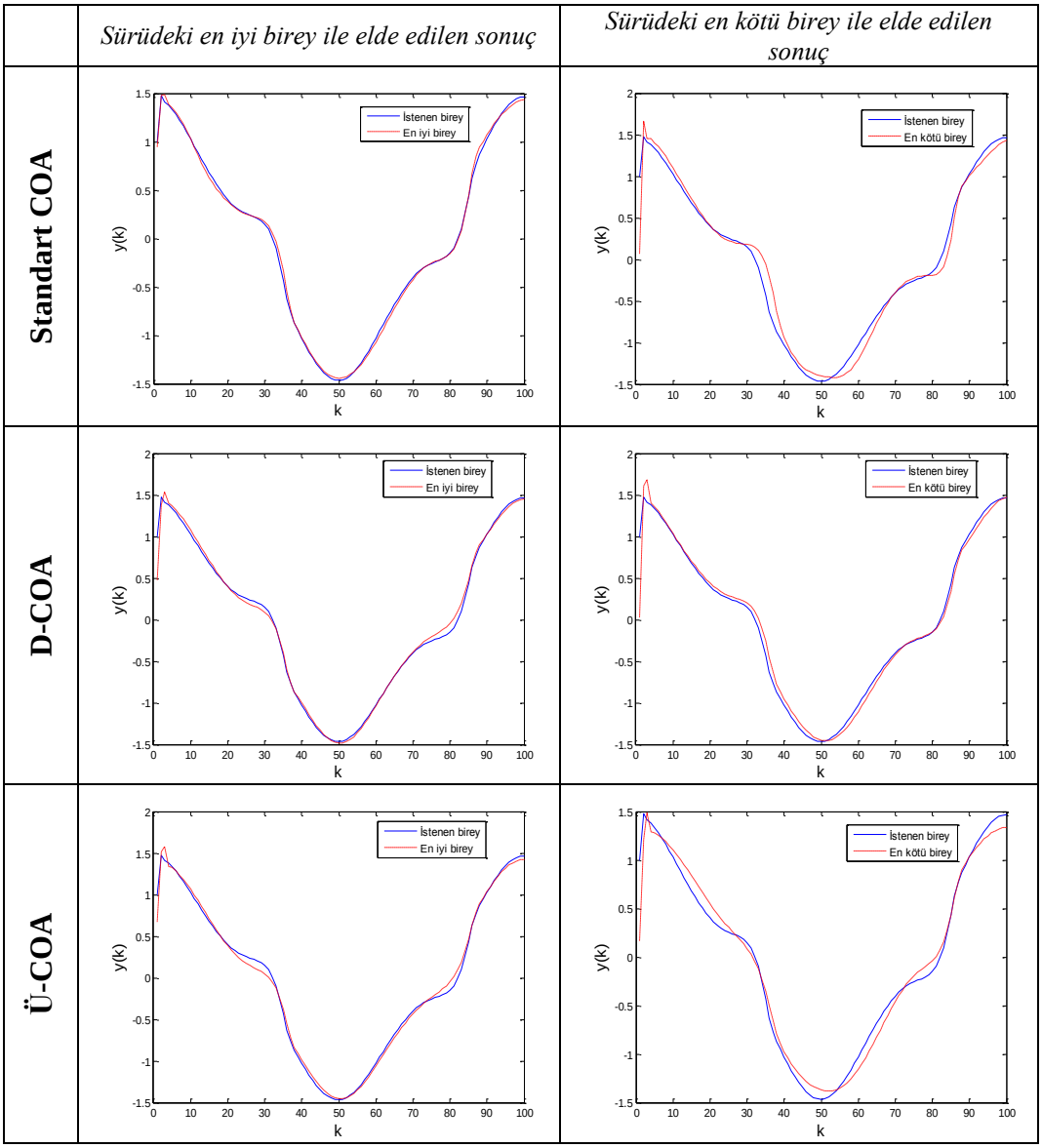


Şekil 5.5. COA öğrenmeli ANFIS ile ÖDS 1 için test fazı sistem tanıma sonuçları.

5.3. ÖDS 2'nin Tanınması/Modellenmesi

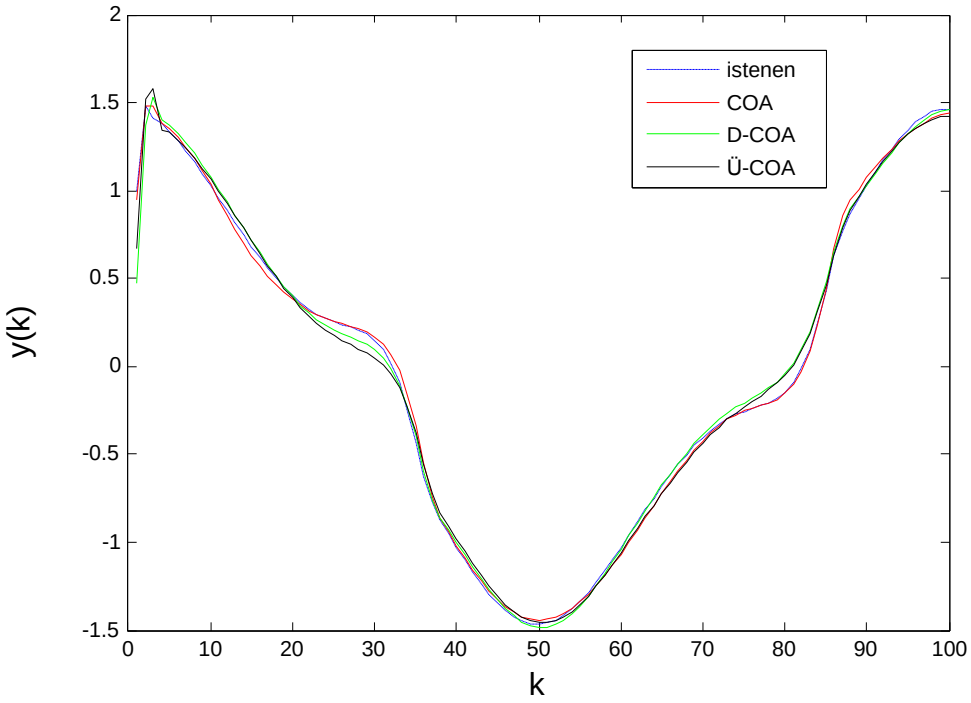
5.3.1. ÖDS 2 için eğitim aşaması ve sonuçları

ÖDS 2 için, Bölüm 4 Çizelge 4.2'de verilen giriş değişkenleri ve Şekil 4.2(a)'da verilen u(k) giriş dizisi ile eğitim veri seti hazırlanmıştır. ÖDS 2 için hazırlanan eğitim veri seti kullanılarak sırasıyla COA, D-COA ve Ü-COA ANFIS optimizasyonu için koşturulmuştur. Uygulama sonucunda COA, D-COA ve Ü-COA'nın sistem tanıma sonuçları grafiksel olarak Şekil 5.6 ve Şekil 5.7'de özetlenmiştir.

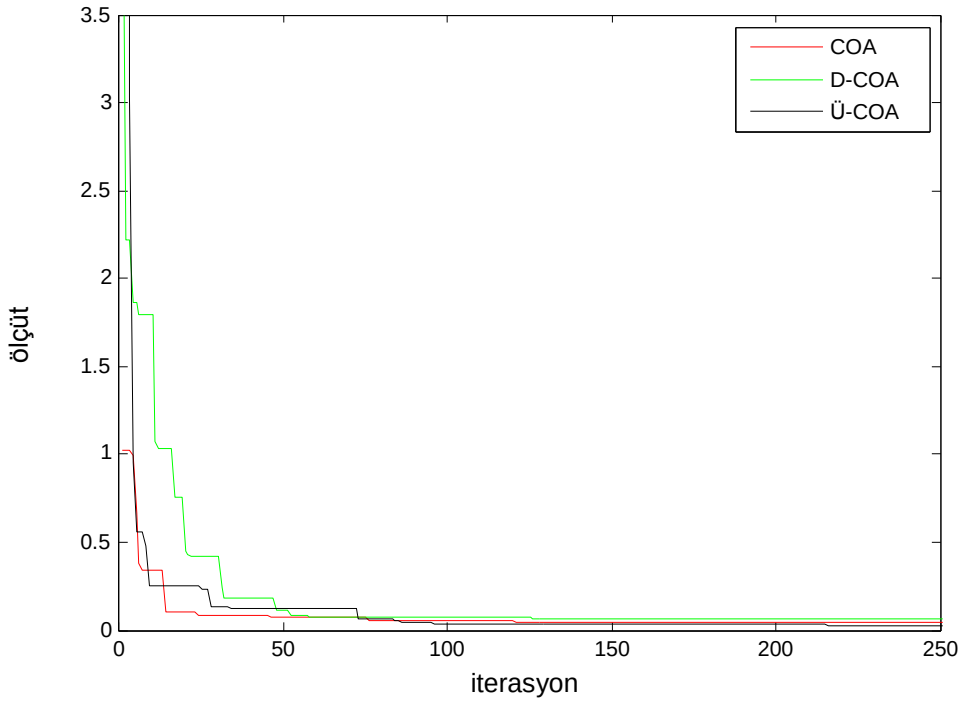


Şekil 5.6. COA öğrenmeli ANFIS ile ÖDS 2 için eğitim fazı sistem tanıma sonuçları.

ÖDS 2'nin ANFIS ile modellenmesi için Standart COA, D-COA, Ü-COA ile ayrı ayrı koşturulması sonucunda elde edilen en iyi ölçüt değerine sahip koşmanın en iyi bireyi ile elde edilen modelleme sonuçları kıyaslanmış ve grafiksel olarak Şekil 5.7'de verilmiştir.



Şekil 5.7. Standart COA, D-COA, Ü-COA öğrenmeli ANFIS ile ÖDS 2 elde edilen modellerin karşılaştırılması.



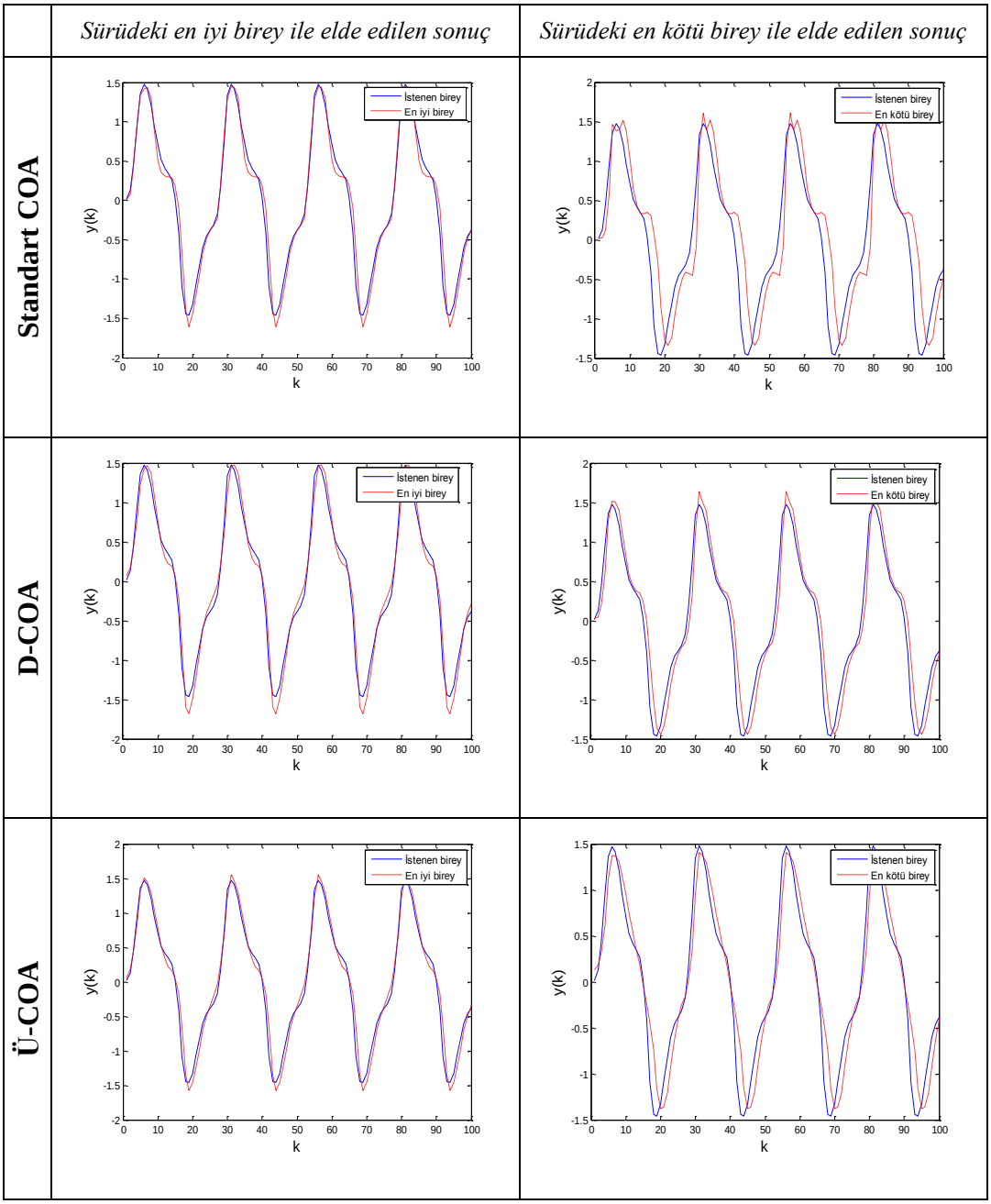
Şekil 5.8. COA, D-COA, Ü-COA öğrenmeli ANFIS ile ÖDS 2 modelleme eğitim seyri.

Buna göre standart COA ile istenene yakın sonuçlar elde edilirken diğer algoritmalarda ise daha uzak sonuçlar elde edilmiştir. Ancak genel olarak başarım iyi olarak görülmektedir. Şekil 5.8'de ise her bir algoritma için en iyi koşmaların en iyi bireyleri ile elde edilen modelleme eğitim seyri verilmiştir.

5.3.2. ÖDS 2'nin test aşaması ve sonuçları

ÖDS 2 için Şekil 4.2(a)'da verilen u(k) kullanılarak oluşturulan veri seti (test seti) aracılığı ile koşturulan COA, D-COA ve Ü-COA'nın ANFIS modelindeki çıkış cevapları ve başarımı incelenmiştir. COA kullanılarak elde edilen çıkış cevapları D-COA ve Ü-COA ile elde edilen cevaplarla karşılaştırılmıştır. Çalışma ile COA'nın bulanık sistem optimizasyonundaki başarısı değerlendirilmiştir.

Bölüm 4 Şekil 4.1(a)'da verilen ÖDS 2 için girişlerden biri olan u(k) dizisi için eğitim fazında; $u(k) = \cos(2\pi k/100)$ kullanılırken, test fazı için Şekil 4.2(a)'dan da görüleceği üzere $u(k) = \sin(2\pi k/25)$ kullanılmıştır. ÖDS 2 için $u(k)$ giriş dizisi hazırlandıktan sonra, Çizelge 4.2'de belirlenen diğer girişlerle beraber test veri seti oluşturulmuştur. Oluşturulan test seti ile COA, D-COA ve Ü-COA ile elde edilen ÖDS 2 ANFIS modellerinin sonuçları Şekil 5.9'da grafiksel olarak verilmiştir.

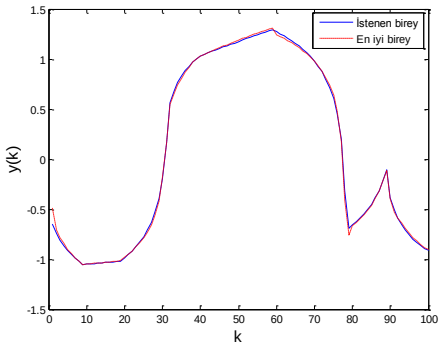
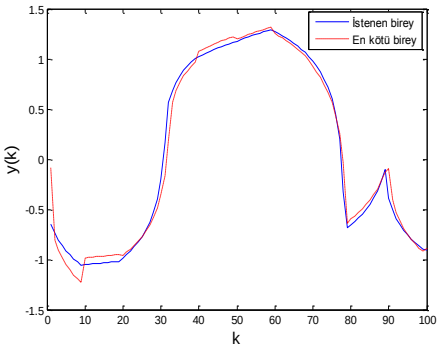
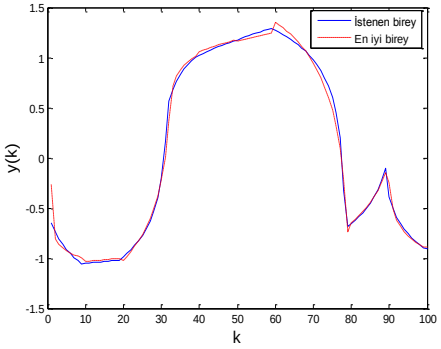
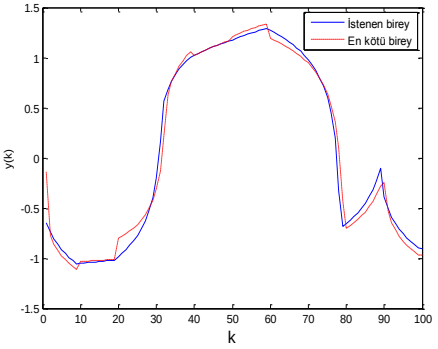
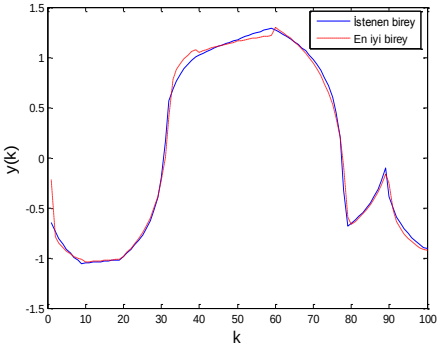
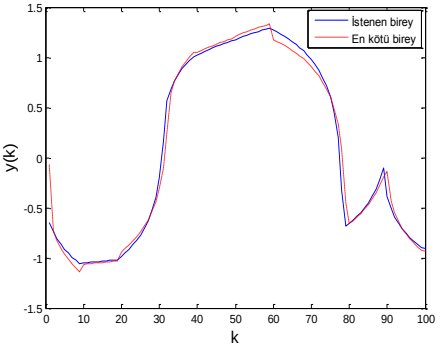


Şekil 5.9. COA öğrenmeli ANFIS ile ÖDS 2 için test fazı sistem tanıma sonuçları.

5.4. ÖDS 3 ‘ün Tanınması/Modellenmesi

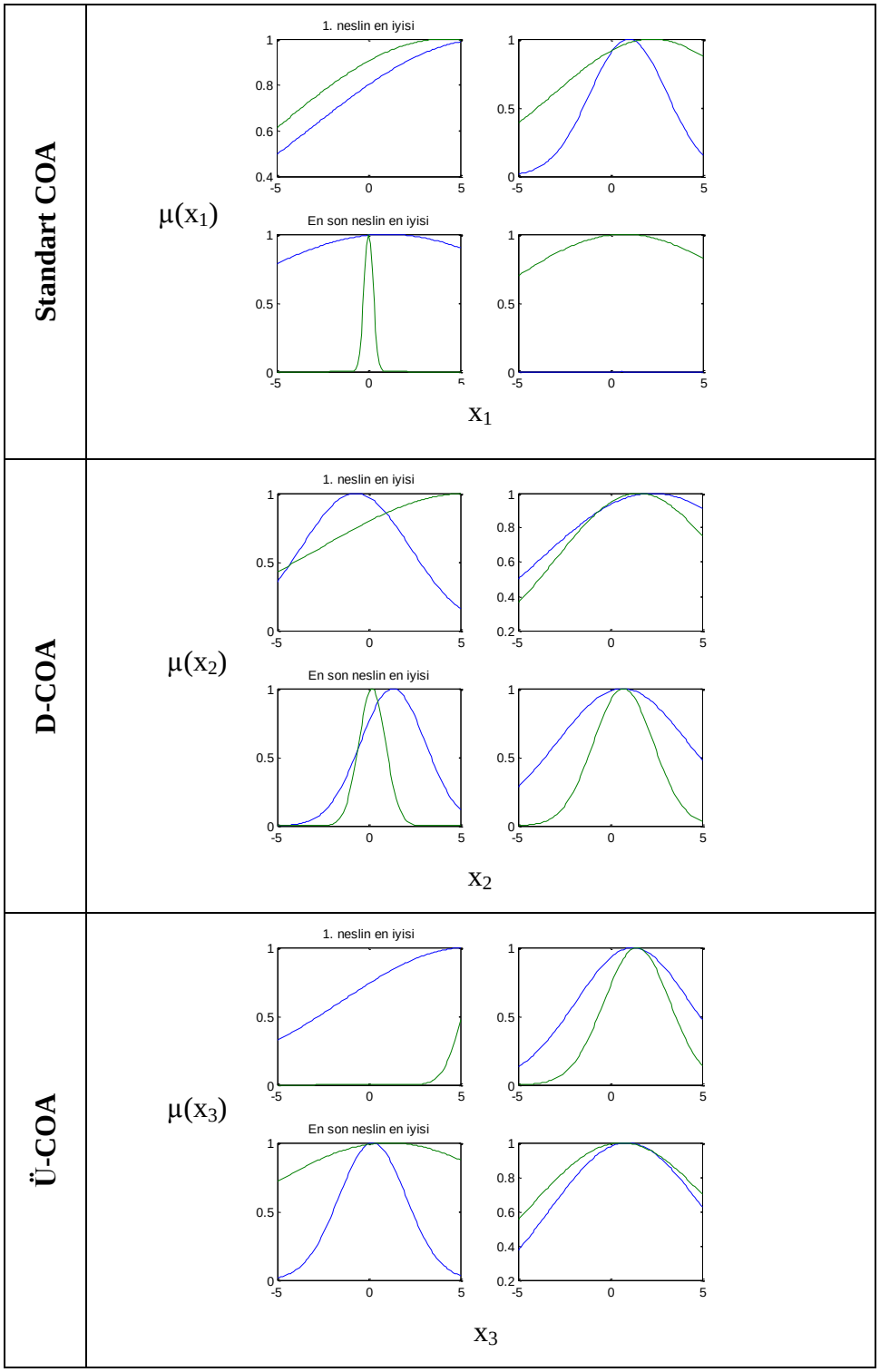
5.4.1. ÖDS 3 için eğitim aşaması ve sonuçları

ÖDS 3 için Çizelge 4.1’de verilen fonksiyona uygun olarak ANFIS yapısında kullanılmak üzere Çizelge 4.2’deki girişler belirlenmiştir.

	Sürüdeki en iyi birey ile elde edilen sonuç	*Sürüdeki en kötü birey ile elde edilen sonuç*
Standart COA		
D-COA		
Ü-COA		

Şekil 5.10. COA öğrenmeli ANFIS ile ÖDS 3 için eğitim fazı sistem tanıma sonuçları.

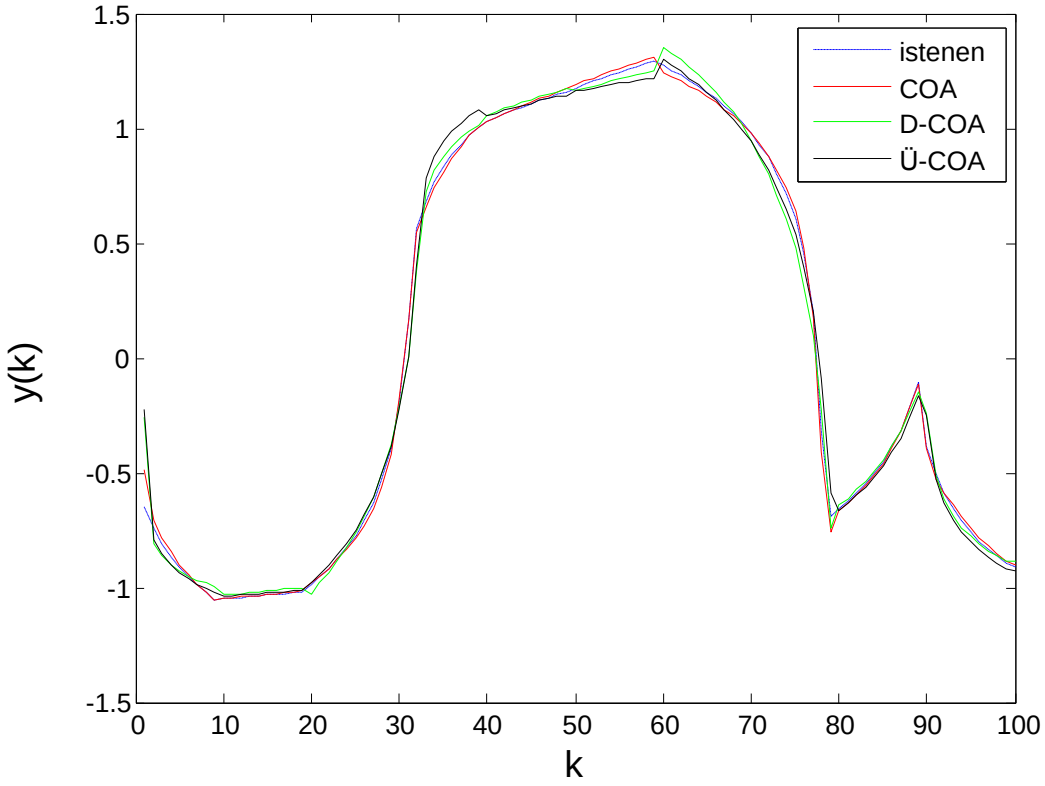
ÖDS 3'ün ANFIS modelleri için elde edilen en iyi birey için en iyi ve en kötü birey için en kötü sonuçlar grafiksel olarak Şekil 5.10'da verilmiştir.



Şekil 5.11. ÖDS 3 için COA öğrenmeli ANFIS ile eğitim fazı için kullanılan ÜF'lerin başlangıç ve son durumları.

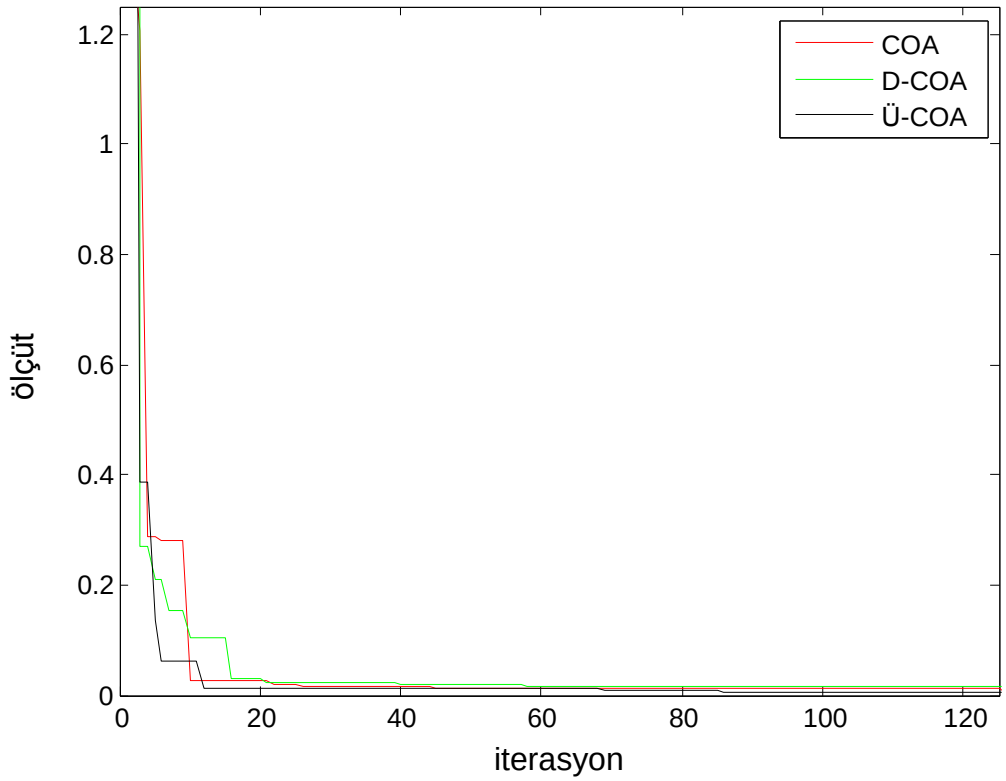
Şekil 5.11'de verilen grafikler ile her bir algoritmanın üyelik fonksiyonları üzerinde uyguladığı değişimler görülebilir. Şekillerde birinci sütunda verilen grafikler 1. girişe, 2. sütunda verilenler ise 2. girişe ait ÜF'lerin ilk ve son durumunu göstermektedir.

Şekil 5.12'de COA, D-COA ve Ü-COA için elde edilen en iyilerin başarımların karşılaştırılması verilmiştir. Şekil 5.12'de algoritmaların ÖDS 3 için elde ettikleri başarım incelendiğinde tüm algoritmaların istenene yakın cevaplar verdiği görülmektedir.



Şekil 5.12. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 3 için elde edilen modellerin karşılaştırılması.

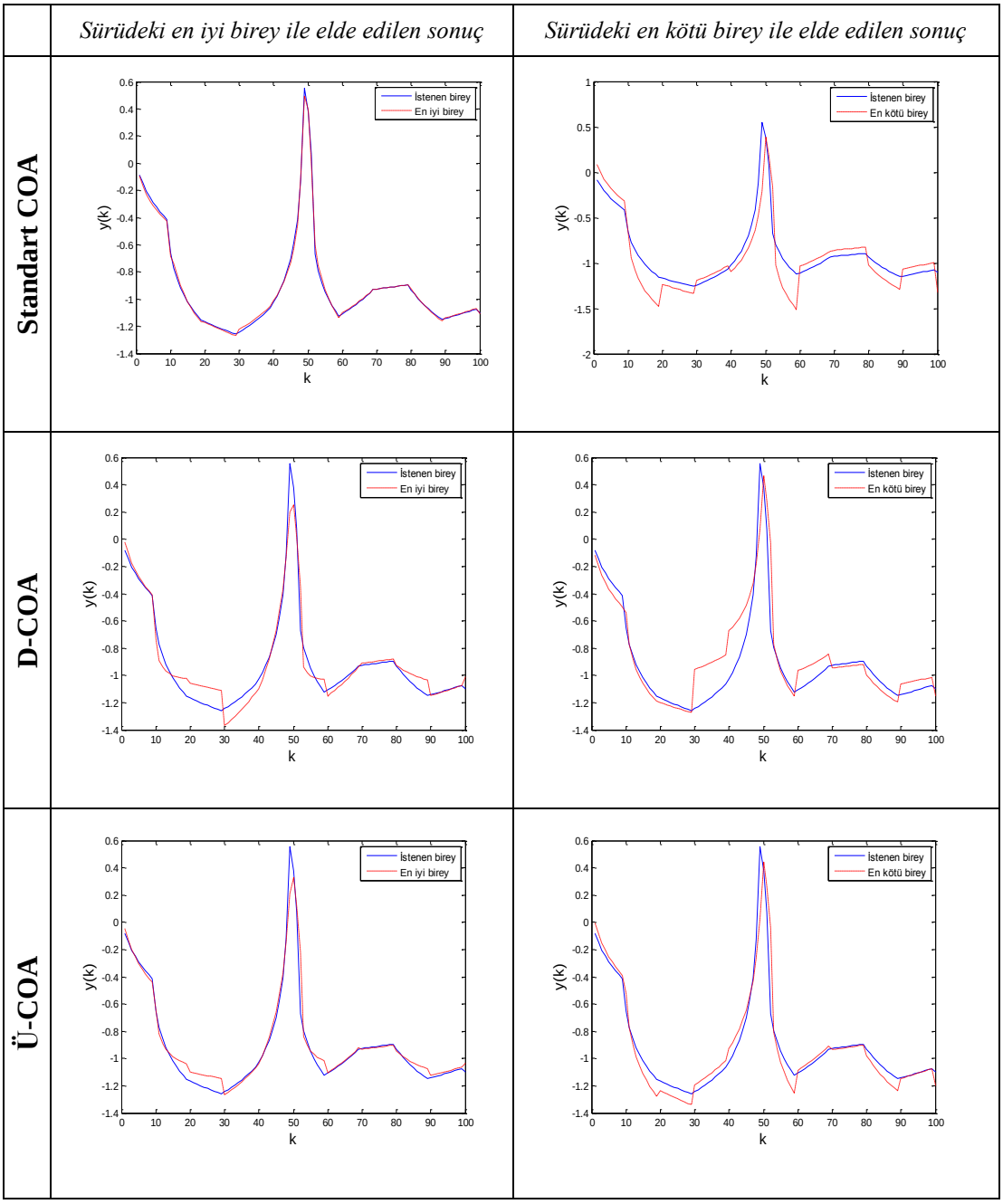
Şekil 5.13'de verilen eğitim seyri, her bir algoritma için ayrı ayrı 50'şer koşmanın (en iyi bireyler için) ortalaması alınarak elde edilmiştir.



Şekil 5.13. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 3 için modelleme eğitim seyri.

5.4.2. ÖDS 3'ün test aşaması ve sonuçları

ÖDS 3 için Çizelge 4.1'deki ÖDS 3 sisteminde kullanılan $u(k)$ dizisinin rasgele değer atanması sonucunda eğitim aşamasında kullanılan veri setinden daha farklı bir veri seti elde edilmiş ve elde edilen yeni veri seti test için kullanılmıştır. Şekil 5.14'de uygulamada kullanılan algoritmaların belirlediği parametrelere sahip ANFIS modellerinin test veri seti için elde edilen sonuçlar verilmiştir.



Şekil 5.14. COA öğrenmeli ANFIS ile ÖDS 3 için test fazı sistem tanıma sonuçları.

5.5. ÖDS 4'ün Tanınması/Modellenmesi

5.5.1. ÖDS 4 için eğitim aşaması ve sonuçları

ÖDS 4 için veri seti oluşturulmadan önce sistemin davranışında önemli bir yere sahip olan $u(k)$ dizisi oluşturulmuştur. $u(k)$ dizisinin genliği rasgele [-5 5] aralığında ve darbe genişliği [1 20] aralığında rasgele alınarak, Şekil 4.1(c)'deki gibi elde edilmiştir.

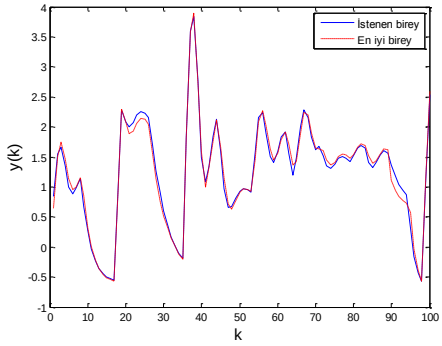
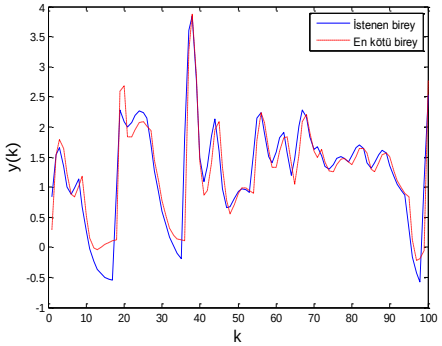
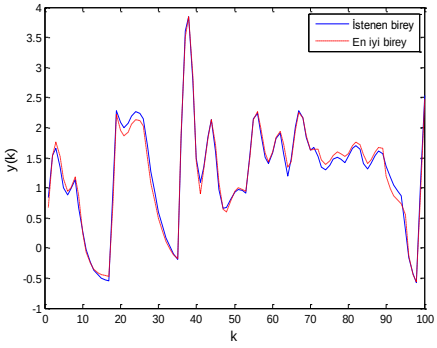
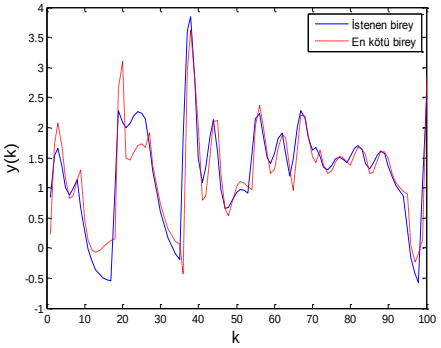
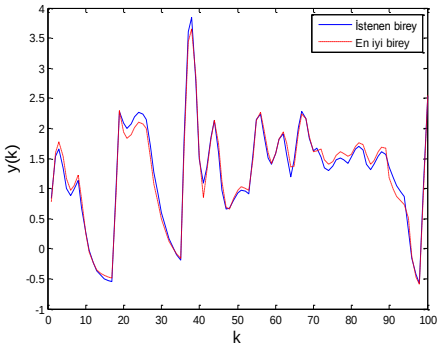
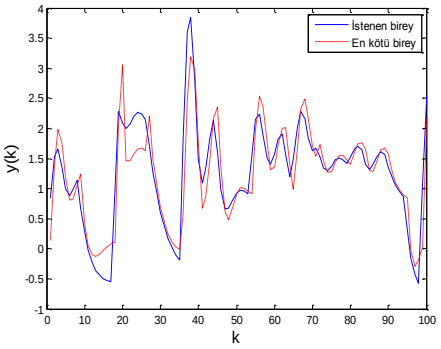
ÖDS 4 için eğitim veri seti ile sırasıyla COA, D-COA ve Ü-COA 50'şer kez koşturulmuştur. Uygulama sonucunda sistem tanıma sonuçlarına göre sürüdeki en iyi birey için en iyi ve en kötü birey için en kötü sonuç grafikleri Şekil 5.15'de verilmiştir.

COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 4'ün eğitim fazının sistem tanıma sonuçlarına göre elde edilen modellerin karşılaştırılması Şekil 5.16'da verilmiştir. ÖDS 4 için tüm algoritmaların koşturulması sonucunda ANFIS öğrenmeli modelleme eğitim seyri Şekil 5.17'de verilmiştir.

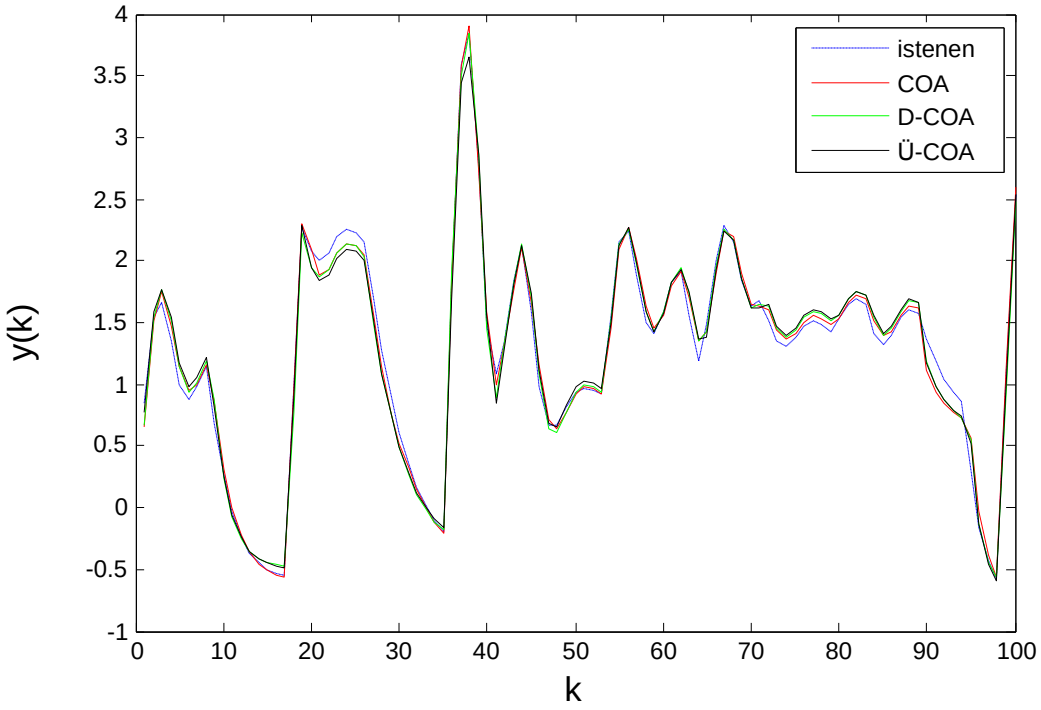
5.5.2. ÖDS 4'ün test aşaması ve sonuçları

ÖDS 4 için test veri seti hazırlanabilmesi amacıyla $u(k)$ dizisi için rasgele değer ataması ile eğitim aşamasında kullanılan örneklerden farklı örnekler elde edilmiştir. Test veri seti kullanılarak standart COA, D-COA ve Ü-COA ile ÖDS 4 için elde edilen ANFIS modellerinin başarımı Şekil 5.18'de verilmiştir.

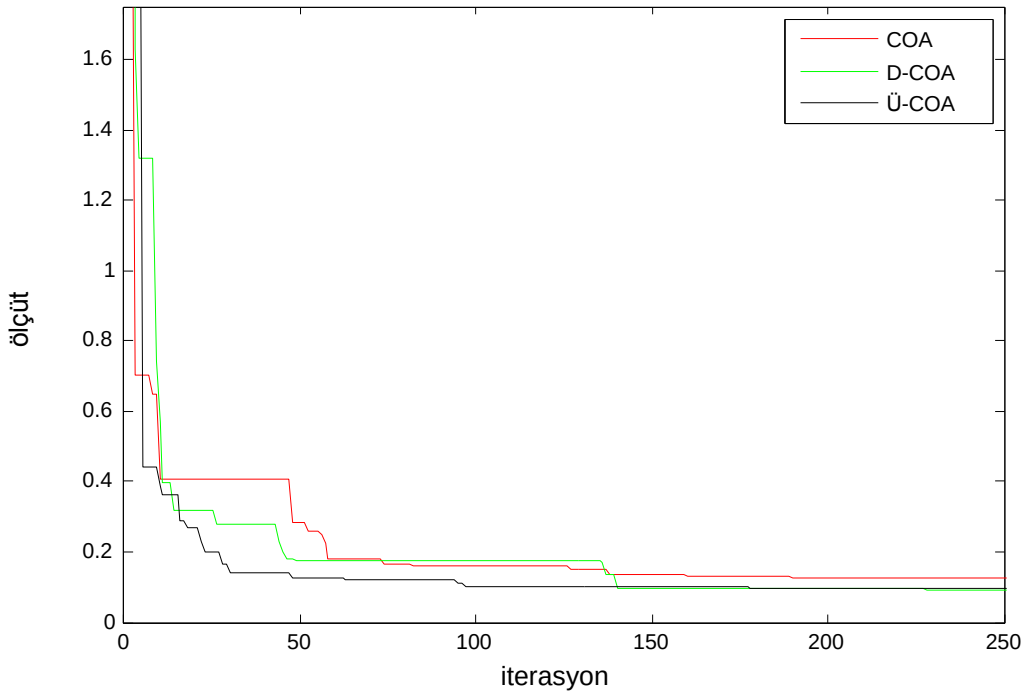
Şekil 5.18 detaylı olarak incelendiğinde; COA, D-COA ve Ü-COA'nın ANFIS öğrenmeli test fazı sonuçlarında en yakın sonucun D-COA ile elde edildiği görülmektedir.

	Sürüdeki en iyi birey ile elde edilen sonuç	*Sürüdeki en kötü birey ile elde edilen sonuç*
Standart COA		
D-COA		
Ü-COA		

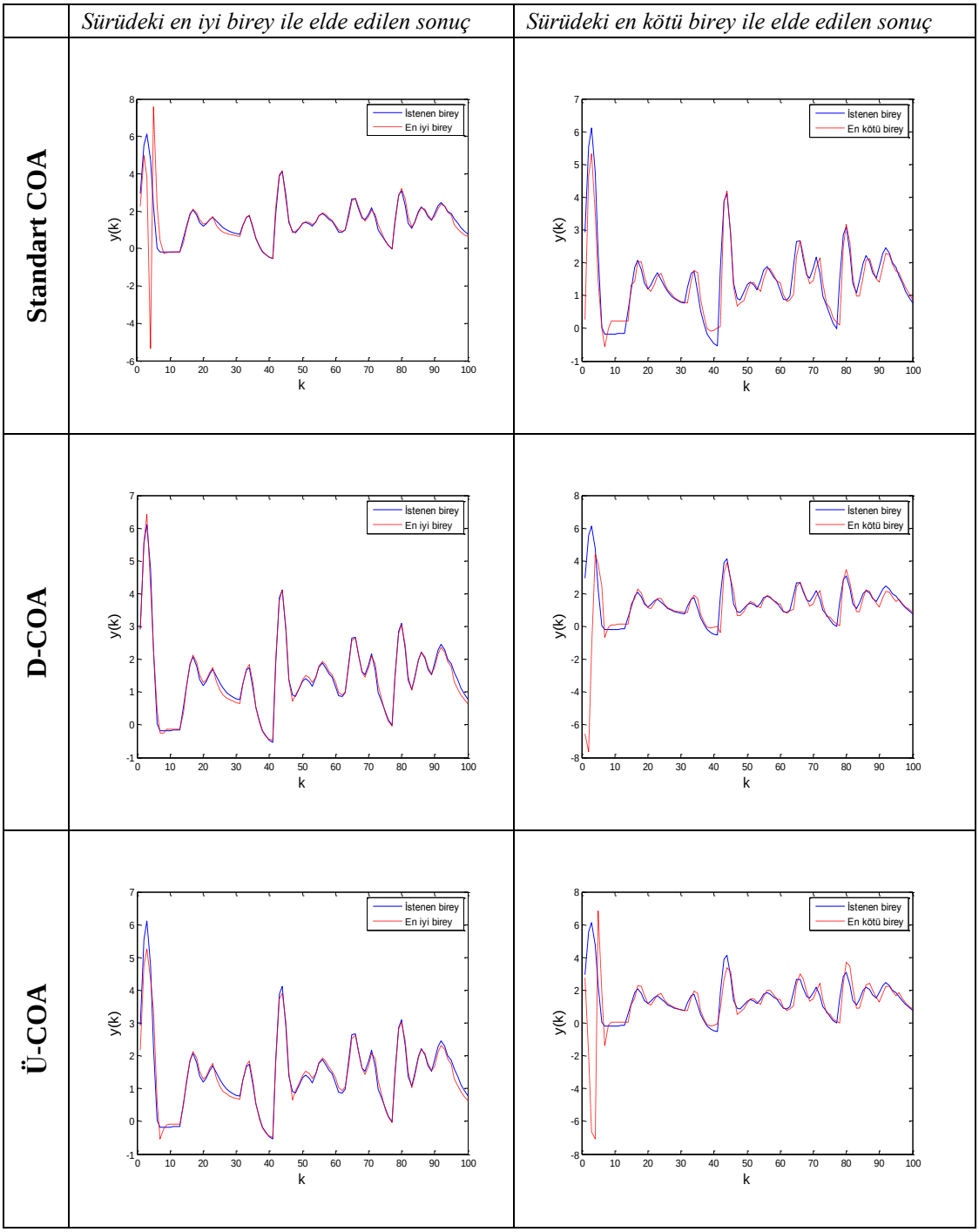
Şekil 5.15. COA öğrenmeli ANFIS ile ÖDS 4 için eğitim fazı sistem tanıma sonuçları.



Şekil 5.16. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 4 için elde edilen modellerin karşılaştırılması.



Şekil 5.17. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 4 modelleme eğitim seyri.

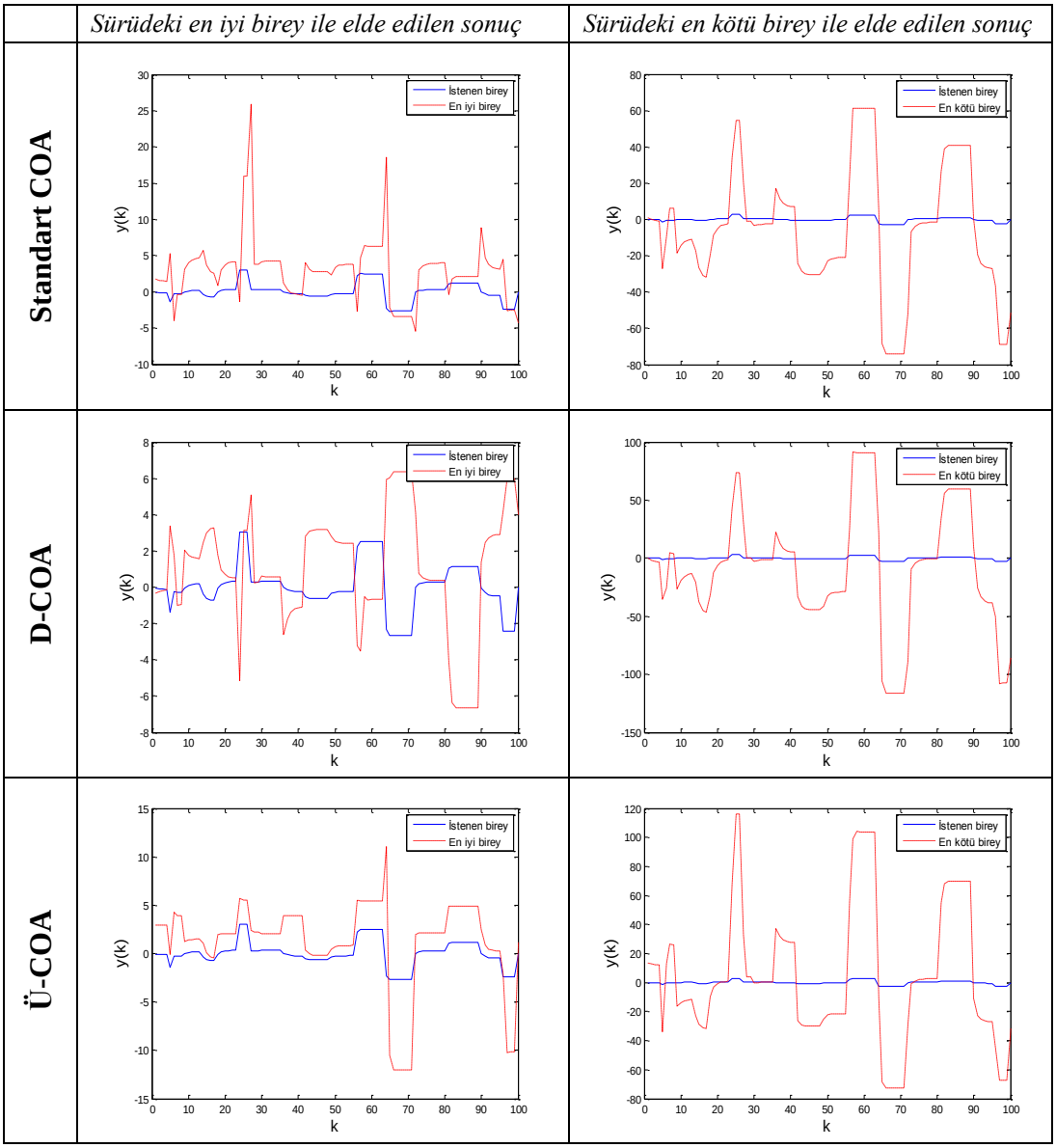


Şekil 5.18. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 4 için test fazı sistem tanıma sonuçları.

5.6. ÖDS 5'in Tanınması/Modellenmesi

5.6.1. ÖDS 5 için eğitim aşaması ve sonuçları

ÖDS 5 için, Çizelge 4.1'de verilen giriş değişkenleri ve Şekil 4.1(d)'de verilen $u(k)$ dizisi kullanılarak eğitim veri seti hazırlanmıştır. Buna göre $u(k)$ dizisi oluşturulurken genliğin rasgele [-5 5] aralığında ve genişliğin [1 20] aralığında rasgele olması dikkate alınarak Şekil 4.1(d)'deki $u(k)$ giriş dizisi elde edilmiştir.

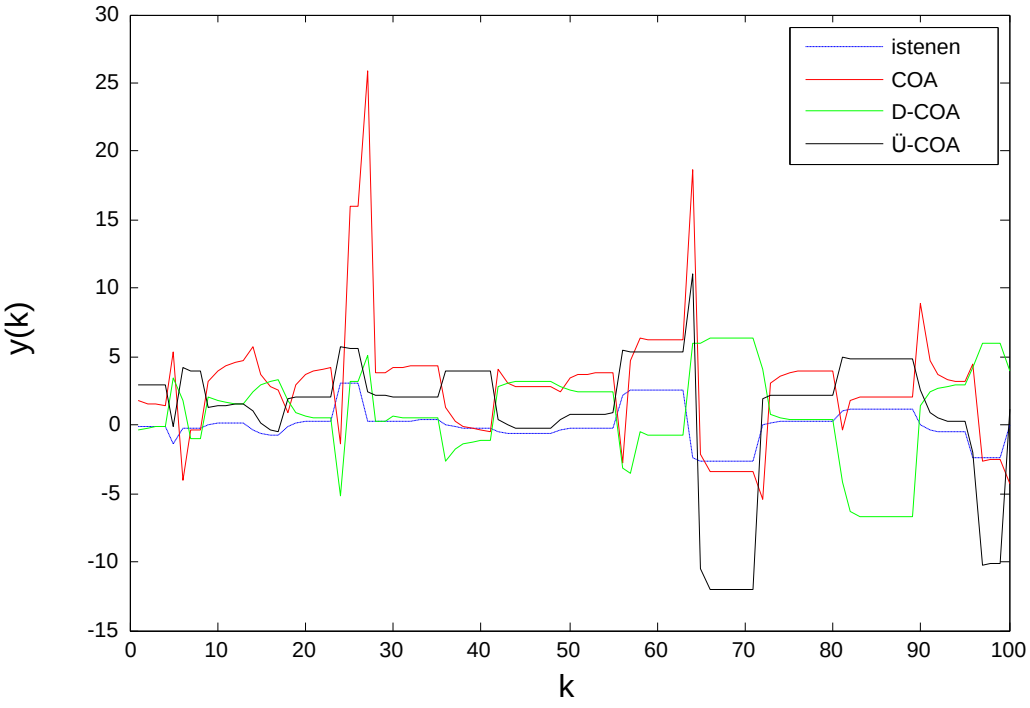


Şekil 5.19. COA öğrenmeli ANFIS ile ÖDS 5 için eğitim fazı sistem tanıma sonuçları.

ÖDS 5 için eğitim veri seti kullanılarak sırasıyla COA, D-COA ve Ü-COA ANFIS optimizasyonu için 50'şer kez koşturulmuştur.

COA, D-COA ve Ü-COA'nın sistem tanıma sonuçları Şekil 5.19'da elde edilen en iyi birey ve en kötü birey için grafiksel olarak verilmiştir. Sistem tanıma sonucuna ait grafikler sırasıyla incelendiğinde istenen çıkışa göre en yakın cevabı standart COA'nın verdiği diğer algoritmaların ise kötü olduğu görülmektedir. Ancak yine de 3 algoritmanın da istenilen sonucu vermediği görülmektedir.

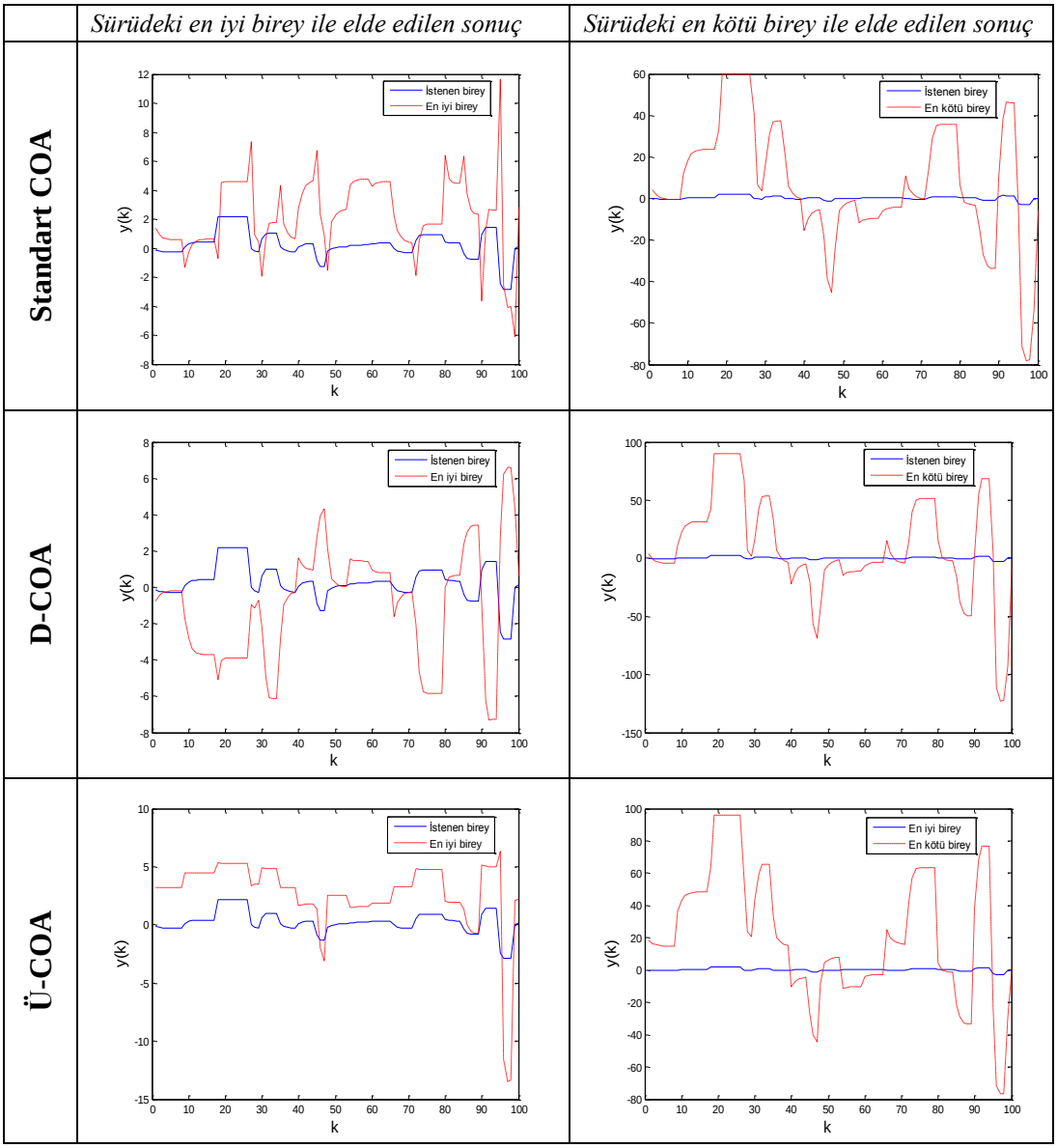
ÖDS 5'in ANFIS modellenmesi için 50'şer kez koşturulan standart COA, D-COA ve Ü-COA ile elde edilen en iyi ölçüt değerine sahip koşmanın en iyi bireyi ile elde edilen modelleme sonuçları kıyaslanmıştır. İstenen çıkışa bakıldığında COA ile elde edilen sonuçların D-COA ve Ü-COA'ya göre istenene daha yakın olduğu görülmektedir. Ayrıca her bir algoritmanın ANFIS yapısının optimizasyonu amaçlı koşturulması sonucunda elde edilen sonuçlara göre ÖDS 5 için algoritmaların başarılı olmadığı görülmüştür.



Şekil 5.20. COA, D-COA ve Ü-COA öğrenmeli ANFIS ile ÖDS 5 için elde edilen modellerin karşılaştırılması.

5.6.2. ÖDS 5'in test aşaması ve sonuçları

ÖDS 5 için test amaçlı [-2 2] aralığında kullanılan rasgele genlikli ve rasgele örnekleme periyotlu $u(k)$ dizisi Şekil 4.2(d)'de verilmiştir. Oluşturulan veri (test seti) seti aracılığı ile ÖDS 5'in COA, D-COA ve Ü-COA öğrenmeli ANFIS modellerinin çıkış cevapları ve performansı incelenmiştir. Oluşturulan test seti ile COA, D-COA ve Ü-COA ile elde edilen ÖDS 5 ANFIS modellerinin sonuçları Şekil 5.22'de verilmiştir. Verilen şekillere bakıldığında D-COA'nın test fazı için performansı diğerlerine göre daha iyi sonuç vermiştir. Ü-COA ise istenen değerlerden uzak performans sergilemiştir.



Şekil 5.21. COA öğrenmeli ANFIS ile ÖDS 5 için test fazı sistem tanıma sonuçları.

5.7. COA, D-COA ve Ü-COA Başarım Değerlendirmesi

COA'nın bulanık mantık tabanlı dinamik sistem modellemede başarımını değerlendirmek için Çizelge 4.1'de verilen beş farklı sistem kullanılmıştır. Çizelge 4.1'de verilen her bir sistem için standart COA 50'şer kez koşturulmuştur. Bunun yanında standart COA'da değişiklikler yapılarak elde edilen D-COA ve Ü-COA da kullanılmıştır. COA ile elde edilen sonuçlar, literatürde çok kullanılan ABC ve PSO algoritmalarına ait (Karakuzu ve diğ., 2013) ve (Yıldırım, 2013)'den alınan sonuçlarla kıyaslanmıştır. Başarım kıyaslamasında her bir sistem için eğitim fazında kullanılan algoritmaların popülasyon büyüklükleri ve maksimum nesil sayıları aynı alınmıştır. Koşturulan algoritmaların eğitim seti için başarım kıyaslaması Çizelge 5.1'de, test seti için başarım kıyaslaması Çizelge 5.2'de verilmiştir.

Çizelge 5.1 ile yapılan kıyaslamada her bir sistem için sırasıyla ortalama, standart sapma, en iyi, en kötü, 1 nesil süresi, başarım sayısı olmak üzere sonuçlar 6 ayrı kategoride istatistiki olarak irdelenmiştir. Bu kategorileri sırasıyla aşağıdaki gibi tanımlayabiliriz:

- Ortalama: 50 ayrı koşma sonunda bulunan eğitim ölçüt değerlerinin ortalaması.
- Standart sapma: 50 ayrı koşma sonunda bulunan eğitim ölçüt değerlerinin standart sapması.
- En iyi: 50 ayrı koşma sonucu elde edilen ölçütler içinde en iyi ölçüt değeri.
- En kötü: 50 ayrı koşma sonucu elde edilen ölçütler içinde en kötü ölçüt değeri.
- 1 nesil süresi: 50 ayrı koşma sonunda bulunan ortalama 1 nesil süresi.

Çizelge 5.1'deki ortalama başarım sayıları incelendiğinde, COA'nın D-COA ve Ü-COA'ya göre daha iyi başarım sağladığı görülmüştür. Geliştirilen bu yeni algoritmalar da COA'ya yakın ortalama başarım sayısına sahip olduğundan, algoritmanın istenen değerlere yakın sonuçlar verdiği tespit edilmiştir.

Çizelge 5.3'e bakılarak ortalama başarım sayıları incelendiğinde de, COA'nın ABC ve PSO algoritmalarına göre daha iyi başarım sağladığı görülmektedir. Sadece ÖDS 5 için ABC algoritması COA ve PSO'ya göre daha başarılıdır.

Çalışmada COA'nın bulanık ağ yapısı üzerinde başarımını daha iyi inceleyebilmek için farklı veri setleri (test seti) kullanılarak her bir sistem için elde edilen ANFIS modeli test edilmiştir. Bu veri setleri kullanılarak elde edilen karşılaştırmalar Çizelge 5.2 ve 5.4'de görülmektedir.

Çizelge 5.1. COA, D-COA ve Ü-COA'nın ÖDS modellemede *EĞİTİM* fazı başarım kıyaslaması.

Algoritma	Ölçüt	ÖDS 1 (Maksimum nesil =1000)	ÖDS 2 (Maksimum nesil =1000)	ÖDS 3 (Maksimum nesil =1000)	ÖDS 4 (Maksimum nesil =1000)	ÖDS 5 (Maksimum nesil =1000)
COA	Ortalama	**0.017203**	**0.008346**	**0.006750**	0.044761	**3.714477**
	Std sapma	0.012694	0.004093	0.002377	0.042965	**2.945203**
	En iyi	**0.001364**	**0.000301**	**0.000615**	**0.008315**	25.37373
	En kötü	**0.042002**	0.018646	**0.009954**	0.145253	**1171.587**
	1 nesil süresi (s)	0.110193	0.109592	0.094342	**0.010540**	0.095543
	Başarım sayısı	3	2	3	2	3
	Ortalama Başarım Sayısı	13/25=0.52				
D-COA	Ortalama	0.028207	0.009420	0.008736	**0.041674**	3.861290
	Std sapma	**0.010658**	0.003056	**0.001361**	**0.034728**	3.29978
	En iyi	0.010503	0.001819	0.001705	0.010562	12.36338
	En kötü	0.055234	0.017436	0.011648	**0.125969**	1424.101
	1 nesil süresi (s)	**0.109283**	**0.107999**	**0.093806**	0.011080	**0.078887**
	Başarım sayısı	2	1	2	3	1
	Ortalama Başarım Sayısı	9/25=0.36				
Ü-COA	Ortalama	0.028579	0.009920	0.008759	0.041675	591.4806
	Std sapma	0.011911	**0.002582**	0.001527	0.042072	515.9679
	En iyi	0.009382	0.001581	0.001090	0.012526	**7.239711**
	En kötü	0.053710	**0.014873**	0.010927	0.136253	2115.368
	1 nesil süresi (s)	0.109519	0.117842	0.094386	0.110911	0.09068
	Başarım sayısı	0	2	0	0	1
	Ortalama Başarım Sayısı	3/25=0.12				

Çizelge 5.2. COA, D-COA ve Ü-COA'nın ÖDS modellemede *TEST* fazı başarım kıyaslaması.

Algoritma	Ölçüt	ÖDS 1 (Maksimum nesil =1000)	ÖDS 2 (Maksimum nesil =1000)	ÖDS 3 (Maksimum nesil =1000)	ÖDS 4 (Maksimum nesil =1000)	ÖDS 5 (Maksimum nesil =1000)
COA	En iyi	**0.0096**	**0.0191**	0.0051	1.5419	10.4809
	En kötü	0.6013	0.1642	0.0134	3.2664	**834.2285**
	Başarım Sayısı	1	1	0	0	1
	Ortalama Başarım Sayısı	3/10=0.3				
D-COA	En iyi	0.0485	0.0322	0.0028	1.4356	7.5879
	En kötü	**0.1316**	**0.0344**	**0.0104**	**2.3653**	1029.2
	Başarım Sayısı	1	1	1	1	0
	Ortalama Başarım Sayısı	4/10=0.4				
Ü-COA	En iyi	0.0126	0.0607	**0.0018**	**1.2817**	**7.2397**
	En kötü	0.1401	0.1251	0.0202	9.0760	1694.64
	Başarım Sayısı	0	0	1	1	1
	Ortalama Başarım Sayısı	3/10=0.3				

Her bir sistem için elde edilen ANFIS modeli test sonuçlarına göre Çizelge 5.2'den de görüleceği üzere en iyi başarımı D-COA sağlamaktadır. Bu algoritmanın başarımını sırasıyla COA ve Ü-COA takip etmektedir.

Çizelge 5.3. COA, ABC ve PSO algoritmalarının ÖDS modellemede EĞİTİM fazı başarım kıyaslaması.

Algoritma	Ölçüt	ÖDS 1 (Maksimum nesil =1000)	ÖDS 2 (Maksimum nesil =1000)	ÖDS 3 (Maksimum nesil =1000)	ÖDS 4 (Maksimum nesil =1000)	ÖDS 5 (Maksimum nesil =1000)
COA	Ortalama	**0.017203**	**0.008346**	**0.006750**	**0.044761**	3.714477
	Std sapma	0.012694	**0.004093**	**0.002377**	0.042965	2.945203
	En iyi	**0.001364**	**0.000301**	**0.000615**	0.008315	25.37373
	En kötü	**0.042002**	**0.018646**	**0.009954**	0.145253	1171.587
	1 nesil süresi (s)	0.110193	0.109592	0.094342	**0.010540**	0.095543
	Başarım sayısı	3	4	4	2	0
	Ortalama Başarım Sayısı	13/25=0.52				
ABC*	Ortalama	0.036994	0.045392	0.012165	0.050564	**0.195731**
	Std sapma	**0.012073**	0.032781	0.013409	**0.019603**	**0.038999**
	En iyi	0.013664	0.003924	0.004422	**0.01568**	**0.171836**
	En kötü	0.058886	0.145957	0.056209	**0.092783**	**0.453153**
	1 nesil süresi (s)	**0.026451**	**0.02701**	**0.015753**	0.024448	0.064086
	Başarım sayısı	2	1	1	3	4
	Ortalama Başarım Sayısı	11/25=0.44				
PSO*	Ortalama	0.079258	0.030886	0.212148	0.120247	0.413184
	Std sapma	0.063467	0.027827	0.568219	0.043797	0.044641
	En iyi	0.027028	0.008052	0.005427	0.03366	0.319359
	En kötü	0.437995	0.197535	3.759978	0.220542	0.512254
	1 nesil süresi (s)	0.047024	0.045804	0.02863	0.047661	**0.031011**
	Başarım sayısı	0	0	0	0	1
	Ortalama Başarım Sayısı	1/25=0.04				

Çizelge 5.4. COA, ABC ve PSO algoritmalarının ÖDS modellemede TEST fazı başarım kıyaslaması.

Algoritma	Ölçüt	ÖDS 1 (Maksimum nesil =1000)	ÖDS 2 (Maksimum nesil =1000)	ÖDS 3 (Maksimum nesil =1000)	ÖDS 4 (Maksimum nesil =1000)	ÖDS 5 (Maksimum nesil =1000)
COA	En iyi	**0.0096**	**0.0191**	**0.0051**	1.5419	10.4809
	En kötü	**0.6013**	**0.1642**	**0.0134**	3.2664	834.2285
	Başarım Sayısı	2	2	2	0	1
	Ortalama Başarım Sayısı	7/10=0.7				
ABC*	En iyi	0.027788	0.020664	0.025316	0.094835	**0.203715**
	En kötü	1.97251	0.801725	547.1167	5.72806	1322.705
	Başarım Sayısı	0	1	0	0	1
	Ortalama Başarım Sayısı	2/10=0.2				
PSO*	En iyi	0.054068	0.025113	0.023815	**0.083096**	0.324227
	En kötü	1.278393	1.094454	51.89824	**0.732723**	**0.931145**
	Başarım Sayısı	0	0	0	2	1
	Ortalama Başarım Sayısı	3/10=0.3				

6. SONUÇLAR

Bu tez çalışmasında, yeni bir sezgisel arama algoritması olan COA'nın optimizasyon başarımı hem fonksiyon hem de bulanık ağ yapısı optimizasyonunda detaylı bir şekilde incelenmiş ve elde edilen sonuçlar istatistiksel olarak kıyaslanmıştır. Bu amaçla literatürde bilinen 6 adet fonksiyon ve 5 farklı örnek dinamik sistem (ÖDS) kullanılmıştır.

Başlangıç olarak standart COA (S-COA)'nın işleyişi, çalışması ve mantığının kavranması amacıyla öncelikle "*peaks*" fonksiyonu üzerinde çalışılmıştır. Elde edilen gözlemler sonucunda iki yeni yaklaşım geliştirilmiştir. Geliştirilen bu yaklaşımların kıyaslamalı başarım değerlendirmesi için literatürden alınan altı farklı fonksiyon üzerinde çalışılmış ve bu çalışmaların sonuçları Bölüm 3'te kısaca özetlenmiştir.

S-COA ile altı farklı fonksiyon optimizasyonu için muhtelif adımlardaki birey pozisyonları incelendiğinde, bu çalışmalar sırasında algoritmanın başarımını etkileyen en önemli değişkenin yumurtlama yarıçapı olduğu gözlenmiştir. Bu değişken değeri x-y aralığında değiştirilerek istenen sonuçlar elde edilmeye çalışılmıştır.

Geliştirilen algoritmaların standart COA ile yapılan başarım değerlendirmesi sonucunda, F1 fonksiyonu için en iyi sonucu Ü-COA vermiştir. F2 fonksiyonu için standart COA ve Ü-COA'nın lokal minimuma takılma sayısı global minimum bulma sayısından daha fazla olduğu için bu fonksiyon için D-COA en iyi sonucu vermiştir. F3 fonksiyonu için algoritmaların tümü başarılıdırlar. F4 fonksiyonu için en iyi sonucu D-COA vermektedir. F5 fonksiyonunda F3 fonksiyonundaki gibi istenilen sonucu her algoritma sağlamaktadır. F6 fonksiyonu için Ü-COA ile global minimumu en çok sayıda bulan dolayısıyla en başarılı algoritmadır.

COA'nın bulanık sistem modellemede başarımını incelemek için beş farklı örnek dinamik sistem (ÖDS) kullanılmıştır. Bu sistemlerin her biri COA ve geliştirilen D-COA, Ü-COA kullanılarak ANFIS ile modellenerek her bir dinamik sistem için ANFIS yapısının parametreleri için en uygun değerler bulunmuştur.

Örnek dinamik sistemler için eğitim veri seti kullanılarak sırasıyla COA, D-COA ve Ü-COA ANFIS optimizasyonu için kullanılarak 50'şer kez koşturulmuştur.

Algoritmaların sistem tanıma sonuçları elde edilen en iyi birey ve en kötü birey için sonuç grafikleri verilmiştir.

Örnek dinamik sistemleri modelleme amacıyla ANFIS optimizasyon başarımları incelendiğinde, sadece ÖDS 5 için istenilen sonuçlara yakın değerler elde edilememiştir. Diğer dört sisteme bakıldığında ise istenen sonuçlara yakın değerler elde edildiği görülmüştür.

Her bir ÖDS için COA, D-COA ve Ü-COA ile elde edilen bulanık modeller ABC ve PSO algoritmaları elde edilen modeller ile karşılaştırılarak değerlendirme yapılmıştır. Ortalama başarım sayılarına göre COA'nın yeni geliştirilen D-COA'ya göre %16, Ü-COA'ya göre ise %40 daha başarılı olduğu sonucuna varılmıştır. Yeni geliştirilen COA türevleri ile literatürde kullanılan ABC ve PSO algoritmaları karşılaştırıldığında ise D-COA'nın başarımı %36, Ü-COA'nın başarımı %12 PSO algoritmasından daha yüksektir.

Tez çalışmasında yapılan incelemeler sonucunda yeni bir algoritma olan COA'nın bulanık sistem parametreleri optimizasyonunda başarılı sonuçlar verdiği görülmüştür. Kullanılan örnek dinamik sistemlerin başarımı, literatürde bulunan diğer algoritmalarla karşılaştırıldığında elde edilen sonuçlara göre COA ve geliştirilen türevlerinin daha etkin olarak çalıştığı ve istenilen sonuçları verdiği görülmüştür.

7.KAYNAKLAR

Babuska R., "Fuzzy System, Modeling and Identification", http://www.dcsc.tudelft.nl/~babuska/transp/fuzzmod.pdf ,2012.

Basu M, Chowdhury A.,"Cuckoo search algorithm for economic dispatch", ***Elsevier***, (60):99-108(2013).

Bhandari, A.K., Singh, V.K., Kumar, A., and Singh, G.K. "Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy", ***Expert Systems With Applications***,(Accepted) (ISI-Cited Publication),2013.

Bunday,B.D., "Basic Optimization Methods", ***Edward Arnold Ltd***, London, 1984.

Chandrasekaran K., Simon P, "Multi-objective scheduling problem: Hybrid approach using fuzzy assisted cuckoo search algorithm", ***Swarm and Evolutionary Computation***, 1-16(2012).

Chang, Ping-Teng, Lee Jung-Hua, "A fuzzy DEA and knapsack formulation integrated model for project selection", ***Computers & Operatıons Research***, 1(39):112-125(2012).

Xin-She Yang and Amir H. Gandomi, "Bat Algorithm: A Novel Approach for Global Engineering Optimization", ***Engineering Computations***, 5(29):464-483(2012).

Himmelblau,D.M., Edgar,T.F.,Optimization of Chemical Processes, ***McGraw-Hill***, Inc.NY,1989.

Jang S. - R. J.,"ANFIS: adaptive-network-based fuzzy inference system", ***IEEE Trans. on Systems***, 3***(***23):665-684(1993).

Karakuzu C., Yıldırım Ö., Yüzgeç U., "Bulanık Mantık Tabanlı Sistem Modellemede ABC Algoritmasının Kıyaslamalı Optimizasyon Başarımı", ***TOK 2013 Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı Bildiri Kitabı,*** Malatya, Türkiye, 202-207(2013).

Kahaner , D.,Moler, C., Nash, S., "Numerical Methods and Software, Prentice Hall", ***Inc.Englewood Cliffs***, NJ, (1989).

L.A. Zadeh, "Fuzzy sets", ***Information and Control 8***, 338-353,(1965).

Marichelvam M., Prabaharan T., Yang X., "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan", ***Applied Soft Computing Journal,*** (19):9-93(2014).

Özçalık H.R., Uygur A.F., "Dinamik Sistemlerin Uyumlu Sinirsel-Bulanık Ağ Yapısına Dayalı Etkin Modellenmesi", ***KSÜ Fen ve Mühendislik Dergisi***, 6(1):36-46(2003).

Passino, K. M., Yurkovıch, S. ,"Fuzzy Control", ***Addison Wesley***, California, 1997.

Rajabioun R., "Cuckoo Optimization Algorithm", ***Applied soft computing*** 11(8):5508-5518(2011).

Rao,S.S.,"Optimization Theory and Applications", 2nd.Edition, ***Halsted***, 1978.

Su Zhi-gang, Wang Pei-hong, Shen Jiong, "Convenient T-S fuzzy model with enhanced performance using a novel swarm intelligent fuzzy clustering technique", ***Journal of Process Control***, 1(22):108-124(2012).

Şen, Z., "Mühendislikte bulanık (fuzzy) mantık ile modelleme prensipleri", ***Su Vakfı Yayınları, Bilge Yayıncılık,*** İstanbul, 2004.

Şenol, C., Yıldırım T., "Fuzzy – Neural Networks for Medical Diagnosis", ***International Journal of Reasoning-based Intelligent Systems,*** 3/4(2):265-274(2010).

Yang X., Deb S., "Multiobjective cuckoo search for design optimization", 6(40):1616-1624(2013).

Yıldırım Ö., "Sezgisel Arama Algoritma Tabanlı Bulanık Sistem Optimizasyonu", Yüksek Lisans Tezi, ***Bilecik Şeyh Edebali Üniversitesi. F.B.E***, 2013.

8. ÖZGEÇMİŞ

Kişisel Bilgiler

Ad Soyad: PINAR ÖZKURT TUNA

Doğum Yeri ve Tarihi: İzmir – 01.07.1985

Eğitim Durumu

2003-2007 Süleyman Demirel Üniversitesi, ISPARTA
Teknik Eğitim Fakültesi
Bilgisayar Sistemleri Öğretmenliği

1999-2003 İsmet İnönü Anadolu Meslek Lisesi, MANİSA
Devlet Parasız Yatılı-Bilgisayar Bölümü

1996-1999 Karabağlar Cumhuriyet Lisesi, İZMİR
Ortaokul Bölümü

1991-1996 Ali Akatlar İlköğretim Okulu, İZMİR

İş Deneyimi

2002-2003 Öğretim Yılı VALF Armatür Sanayi – MANİSA
Bilgi İşlem Bölümü-Stajyer
Yapılan İşler:
-Server yönetimi
-İşletim Sistemi Kurumu
-İnterneti yararlı bir şekilde kullanma yöntemleri

27.06.2005-22.07.2005 Diana Bilgisayar İletişim Yazılım Servis ve Danışmanlık Hizmetleri San. ve Tic. A.Ş. – İZMİR
Yazılım Bölümü-Stajyer
Yapılan İşler:
-C# yazılım dilinde program geliştirme
-Şirketin internet sitesi için "Sıkça Sorulan Sorular" linki hazırlanması

25.07.2005-19.08.2005 Gama Bilgisayar Tic. Ltd. Şti. – İZMİR
Teknik Servis Bölümü – Stajyer

Yapılan İşler:

-PC Toplanması

-Bilgisayarda meydana gelebilecek arızaların tespiti

-Bazı donanım araçlarının püf noktalarını kavrama

-Bilgisayara dahili veya harici olarak bağlanan tüm donanım birimlerinin birbirleriyle uyumlarının kavranması

2007-2012 Öğretim Yılı Yenişehir Akçeşme İlköğretim Okulu
Bilişim Teknolojileri Öğretmeni

2012-2014 Öğretim Yılı Yenişehir Teknik ve Endüstri Meslek Lisesi
(Halen) Bilişim Teknolojileri Alan Öğretmeni

İletişim

Adres: Yenişehir Teknik ve Endüstri Meslek Lisesi Yenişehir/Bursa
E-posta:pinarozkurtt@gmail.com

Diğer

Kişisel Özellikler :

- ✓ Yeniliklere açık ve kolay uyum sağlayabilen,
- ✓ Araştırmacı,
- ✓ Grup çalışmalarında başarılı,
- ✓ Yeni bir şeyler üretmekten zevk alan,
- ✓ İnancını hiçbir zaman kaybetmeyen, azimli

EK-1: Standart COA ile fonksiyon optimizasyonunda kullanılan kodlar

```
function deger= standartCuckoo()
clc, clear all, close all
costFunction = 'peaks_fonk';
parametre_sayisi = 2;
varLo = -2;
varHi = 2;
ilk_nufus = 5;
minYumurtaSayisi = 2;
maxYumurtaSayisi = 7
maxIterasyonSayisi = 10
kumeSayisi =1;
hareketKatsayisi = 2;
maxYasayabilecekKusSayisi = 20;
yaricapOrani = 0.01;
optimizasyonuKesenVaryans = 1e-10;
kusNufusu = cell(ilk_nufus,1
for kusNumarasi = 1:ilk_nufus
    kusNufusu{kusNumarasi}.center = ( varHi-varLo )*
    rand(1,parametre_sayisi) + varLo;
end
iterasyon = 0;
baslangic_degeri = 0.5;
hedefNoktasi = (varHi - varLo)*rand(1,parametre_sayisi) + varLo;
globalBestCuckoo = hedefNoktasi;
globalMaxProfit = baslangic_degeri;
uygunluk_vektoru = [];
while  (iterasyon <= maxIterasyonSayisi)
    iterasyon = iterasyon + 1
for kusNumarasi = 1:ilk_nufus
        kusNufusu{kusNumarasi}.yumurtaNumarasi = floor(
        (maxYumurtaSayisi - minYumurtaSayisi) * rand +
        minYumurtaSayisi );
        kusNufusu{kusNumarasi}.yumurtaNumarasi
end
topla = 0;
for kusNumarasi = 1:ilk_nufus
topla = topla + kusNufusu{kusNumarasi}.yumurtaNumarasi;
end
for kusNumarasi = 1:ilk_nufus
        kusNufusu{kusNumarasi}.eggLayingRadius =
        kusNufusu{kusNumarasi}.yumurtaNumarasi/topla * (
        yaricapOrani * (varHi-varLo) );
end
for kusNumarasi = 1:ilk_nufus
        kusNufusu{kusNumarasi}.yumurtaninBirakildigiYaricapUzakligi
        = kusNufusu{kusNumarasi}.eggLayingRadius *
        rand(kusNufusu{kusNumarasi}.yumurtaNumarasi,1);
end
for kusNumarasi = 1:ilk_nufus
parametreler = kusNufusu{kusNumarasi}.center
        gecici_yaricaplar = kusNufusu{kusNumarasi}.
        yumurtaninBirakildigiYaricapUzakligi;
        yaricapSayisi = numel(gecici_yaricaplar);
        angles = linspace(0,2*pi,yaricapSayisi);
        yeniParametreler = [];
```

```
for cnt = 1:yaricapSayisi
         degerEklemeDizisi = zeros(1,parametre_sayisi);
for iii = 1:parametre_sayisi
         randNum = floor(2*rand)+1;
         degerEklemeDizisi(iii) = (-1)^randNum*
         gecici_yaricaplar(cnt)* cos(angles(cnt)) +
         gecici_yaricaplar(cnt)*sin(angles(cnt));
end
         yeniParametreler = [yeniParametreler; parametreler +
         degerEklemeDizisi ];
end
         yeniParametreler(find(yeniParametreler>varHi)) = varHi;
         yeniParametreler(find(yeniParametreler<varLo)) = varLo;
         kusNufusu{kusNumarasi}.newPosition4Egg = yeniParametreler;
end

for kusNumarasi = 1:ilk_nufus
        geciciPop = kusNufusu{kusNumarasi}.newPosition4Egg;
        geciciPop = floor(geciciPop * 1e4)/1e4;
        ii = 2;
        cntt = 1;
while ii <= size(geciciPop,1) || cntt <= size(geciciPop,1)
if all((geciciPop(cntt,:) == geciciPop(ii,:)))
                geciciPop(ii,:) = [];
end
            ii = ii + 1;
if ii > size(geciciPop,1) && cntt <= size(geciciPop,1)
                cntt = cntt + 1;
                ii = cntt + 1;
if ii > size(geciciPop,1)
break
end
end
end
       kusNufusu{kusNumarasi}.newPosition4Egg = geciciPop;
end
for kusNumarasi = 1:ilk_nufus
       kusNufusu{kusNumarasi}.profitValues =
       feval(costFunction,[kusNufusu{kusNumarasi}.center ;
       kusNufusu{kusNumarasi}.newPosition4Egg]);
end
     allPositions = [];
     whichCuckooPopTheEggBelongs = [];
     gecici_uygunluk_degeri = [];
if ilk_nufus > maxYasayabilecekKusSayisi
for kusNumarasi = 1:ilk_nufus
            gecici_uygunluk_degeri = [gecici_uygunluk_degeri;
            kusNufusu{kusNumarasi}.profitValues];
            allPositions = [allPositions;
[kusNufusu{kusNumarasi}.center;
            kusNufusu{kusNumarasi}.newPosition4Egg
           (:,1:parametre_sayisi)]];
           whichCuckooPopTheEggBelongs=[whichCuckooPopTheEggBelongs;
           kusNumarasi*ones(size(kusNufusu{kusNumarasi}.
           newPosition4Egg (:,1:parametre_sayisi),1),1) ];
end

[sortedProfits, sortingIndex= sort
        (gecici_uygunluk_degeri,'ascend');
```

```
      bestCuckooCenter =
      allPositions(sortingIndex(1),1:parametre_sayisi);
      sortedProfits = sortedProfits(1:maxYasayabilecekKusSayisi);
      allPositions = allPositions(sortingIndex
                     (1:maxYasayabilecekKusSayisi),:);
      clear kusNufusu
for ii = 1:maxYasayabilecekKusSayisi
            kusNufusu{ii}.newPosition4Egg = allPositions(ii,:);
            kusNufusu{ii}.center = allPositions(ii,:);
            kusNufusu{ii}.profitValues = sortedProfits(ii);
end
ilk_nufus = maxYasayabilecekKusSayisi;
else
for kusNumarasi = 1:ilk_nufus
            gecici_uygunluk_degeri = [gecici_uygunluk_degeri;
            kusNufusu{kusNumarasi}.profitValues];
            allPositions = [allPositions;
[kusNufusu{kusNumarasi}.center;
            kusNufusu{kusNumarasi}.newPosition4Egg
            (:,1:parametre_sayisi)] ];
            whichCuckooPopTheEggBelongs =
[whichCuckooPopTheEggBelongs;
            kusNumarasi*ones(size(kusNufusu{kusNumarasi}.
            newPosition4Egg(:,1:parametre_sayisi),1),1) ];
end
      [sortedProfits, sortingIndex] =
      sort(gecici_uygunluk_degeri,'ascend');
      bestCuckooCenter=allPositions(sortingIndex(1),
                                    1:parametre_sayisi);

end
x=-2:0.1:2;
     [X1,Y1]=meshgrid(x,x);
      Z1=peaks(X1,Y1);
      contour(X1,Y1,Z1); hold on
for kusNumarasi = 1:ilk_nufus
x=kusNufusu{kusNumarasi}.center(1)
y=kusNufusu{kusNumarasi}.center(2)
      plot(x,y,'r*','LineWidth',2,'MarkerSize',15);
      title(num2str(iterasyon))
end
      drawnow; pause(0.03),
      hold off
     baslangic_degeri  = sortedProfits(1);
      currentBestCuckoo = bestCuckooCenter;
      currentMaxProfit = feval(costFunction,currentBestCuckoo);
if currentMaxProfit < globalMaxProfit
      globalBestCuckoo = currentBestCuckoo;
      globalMaxProfit = currentMaxProfit;
end
uygunluk_vektoru = [uygunluk_vektoru globalMaxProfit];
      allPositions = [];
      whichCuckooPopTheEggBelongs = [];
for kusNumarasi = 1:ilk_nufus
      allPositions = [allPositions;
      kusNufusu{kusNumarasi}.newPosition4Egg(:,1:parametre_sayisi)];
      whichCuckooPopTheEggBelongs = [whichCuckooPopTheEggBelongs;
      kusNumarasi*ones(size(kusNufusu{kusNumarasi}.newPosition4Egg
```

```
      (:,1:parametre_sayisi),1),1) ];
end
if sum(var(allPositions)) < optimizasyonuKesenVaryans
break
else
       [clusterNumbers, clusterCenters] =
       kmeans(allPositions,kumeSayisi);
end
       cluster = cell(kumeSayisi,1);
for ii = 1:kumeSayisi
       cluster{ii}.positions = [];
       cluster{ii}.profits = [];
end
       pointer = 1;
for cnt = 1:length(clusterNumbers)
if cnt < length(clusterNumbers)
if clusterNumbers(cnt) == clusterNumbers(cnt+1)  &&...
                    whichCuckooPopTheEggBelongs(cnt) ==
                    whichCuckooPopTheEggBelongs(cnt+1)
                pointer = pointer + 1;
else
                pointer = 1;
end
       cluster{clusterNumbers(cnt)}.positions =
[cluster{clusterNumbers(cnt)}.positions;
       kusNufusu{whichCuckooPopTheEggBelongs(cnt)}.
       newPosition4Egg(pointer,1:parametre_sayisi)];
       cluster{clusterNumbers(cnt)}.profits   =
[cluster{clusterNumbers(cnt)}.profits;
       kusNufusu{whichCuckooPopTheEggBelongs(cnt)}.
       profitValues(pointer)];
else
       cluster{clusterNumbers(cnt)}.positions =
[cluster{clusterNumbers(cnt)}.positions;
       kusNufusu{whichCuckooPopTheEggBelongs(cnt)}.
       newPosition4Egg(pointer,1:parametre_sayisi)];
       cluster{clusterNumbers(cnt)}.profits   =
[cluster{clusterNumbers(cnt)}.profits;
       kusNufusu{whichCuckooPopTheEggBelongs(cnt)}.
       profitValues(pointer)];
end
end
f_mean = zeros(kumeSayisi,1);
for cnt = 1:kumeSayisi
f_mean(cnt) = mean(cluster{cnt}.profits
end
  [sorted_f_mean, sortingIndex_f_mean] = sort(f_mean,'ascend');
    minFmean = sorted_f_mean(1);   indexOfBestCluster =
    sortingIndex_f_mean(1);
    [minProfitInBestCluster, indexOfBestEggPosition] =
    min(cluster{indexOfBestCluster}.profits);
    hedefNoktasi  =  cluster{indexOfBestCluster}.positions
   (indexOfBestEggPosition,1:parametre_sayisi);
    numNewCuckooS = 0;
for cntClstr = 1:size(cluster,1)
        tmpCluster = cluster{cntClstr};
        tmpPositions = tmpCluster.positions;
for cntPosition = 1:size(tmpPositions,1)
```

```
            tmpPositions(cntPosition,:) = tmpPositions(cntPosition,:)
+ ...
hareketKatsayisi * rand(1,parametre_sayisi) .*  (hedefNoktasi  -
tmpPositions(cntPosition,:));
end
tmpPositions(find( tmpPositions>varHi )) = varHi;
tmpPositions(find( tmpPositions<varLo )) = varLo;
cluster{cntClstr}.positions = tmpPositions;
cluster{cntClstr}.center = mean(tmpPositions);
numNewCuckooS = numNewCuckooS + size(cluster{cntClstr}.positions,1);
end
ilk_nufus = numNewCuckooS;
clear cuckooPop
kusNufusu = cell(ilk_nufus,1);
cntNumCuckooS = 1;
for cnt = 1:size(cluster,1)
tmpCluster = cluster{cnt};
tmpPositions = tmpCluster.positions;
for cntPosition = 1:size(tmpPositions,1)
kusNufusu{cntNumCuckooS}.center =
cluster{cnt}.positions(cntPosition,1:parametre_sayisi);
cntNumCuckooS = cntNumCuckooS + 1;
end
end
currentBestCuckoo = bestCuckooCenter;
currentMaxProfit = feval(costFunction,currentBestCuckoo);
if currentMaxProfit > globalMaxProfit
globalBestCuckoo = currentBestCuckoo;
globalMaxProfit = currentMaxProfit;
end
kusNufusu{end}.center = globalBestCuckoo;
kusNufusu{end}.profitValues = -
feval(costFunction,kusNufusu{end}.center);
tmp = rand(1,parametre_sayisi).*globalBestCuckoo;
tmp(find( tmp>varHi )) = varHi;
tmp(find( tmp<varLo )) = varLo;
kusNufusu{end-1}.center = tmp;
kusNufusu{end-1}.profitValues = feval(costFunction,kusNufusu
{end-1}.center);
end
disp('Best Params = ')
disp(globalBestCuckoo')
fprintf('\n')
disp('Cost = ')
disp(globalMaxProfit)
costVector =  uygunluk_vektoru;
plot(costVector,'-
ks','linewidth',3,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',15)
xlabel 'Cuckoo iterasyon'
ylabel 'Cost Value'
end
```