Güncel Optimizasyon Tekniklerinin Matematiksel Problemlerin Çözümündeki Performanslarının Kıyaslanması

Aybike ÖZYÜKSEL ÇİFTÇİOĞLU*, Erkan DOĞAN[2]

[1]Celal Bayar Üniversitesi, Mühendislik Fakültesi, İnşaat Mühendisliği Bölümü, Tel: +90 236 2012389, aybike.ozyuksel@cbu.edu.tr
[2] Celal Bayar Üniversitesi, Mühendislik Fakültesi, İnşaat Mühendisliği Bölümü, Tel: +90 236 2012311, erkan.dogan@cbu.edu.tr
*İletişimden sorumlu yazar/Corresponding Author

Geliş/Recived: 17 Aralık (December) 2016
Kabul/Accepted: 20 Nisan (April) 2017
DOI: 10.18466/cbayarfbe.324006

Özet

Doğada yer alan böceklerin problem çözmede izlediği başarılı yol (besin kaynağına giden en kestirme yolu bulabilmeleri gibi) araştırmacılar tarafından incelenmekte, sürü içindeki davranışları taklit edilmeye çalışılarak optimizasyon teknikleri oluşturulmaktadır. Av arama, parçacık küme, karınca kolonisi, yapay arı kolonisi, yarasa algoritması, ateş böceği algoritması ve kandil böceği algoritması bu tekniklerden bazılarıdır. Bu çalışmada, bahsedilen yedi teknik ayrı ayrı incelenmiş olup, beş adet en küçükleme ve bir adet büyüğe yaklaştırma olmak üzere toplam altı adet optimizasyon problemi bu yedi algoritma ile ayrı ayrı çözülmüştür. Her bir problemin kendi içerisinde kısıtlayıcıları ve değişkenleri vardır. Her problem için en az on beş bin iterasyon yapılmış, problemin tipine göre bu sayı yirmi bine kadar çıkmıştır. Optimizasyon teknikleri her problem içinde ayrı ayrı karşılaştırılmıştır. Ayrıca bu altı problem kendi arasında da kıyaslanmıştır. Problemler için optimizasyon tekniklerinin birbiri ile karşılaştırılmış grafikleri çizilerek algoritmaların yakınsamalar üzerinden performans kıyaslamaları da yapılmıştır.

Anahtar Kelimeler: Av Arama Algoritması, Parçacık Küme Algoritması, Karınca Koloni Algoritması, Yapay Arı Kolonisi Algoritması, Yarasa Algoritması, Ateş Böceği Algoritması, Kandil Böceği Algoritması, Optimizasyon Problemleri

Comparison of The Performance of Current Optimization Techniques in The Solution of Mathematical Problems

Abstract

The successful path of insects to problem solving (such as finding the shortest route to the source of food) is investigated by researchers and optimization techniques are being created by trying to imitate the behavior of insects in swarm. Hunting search algorithm, particle swarm optimizer, ant colony optimizer, artificial bee colony algorithm, bat algorithm, firefly algorithm and glowworm algorithm are some of these techniques. In this study, the seven techniques mentioned were studied separately. Six optimization problems are solved separately by these seven algorithms. Each problem has its own constraints and design variables. At least fifteen thousand iterations have been performed for each problem, and according to the type of problem, this number has increased up to twenty thousand. Optimization techniques are compared within each problem individually. These six problems are also compared between themselves. Performance comparisons of algorithms over convergence have also been made by plotting convergence rate graph of techniques for each problem.

Keywords: Hunting search algorithm, particle swarm optimizer, ant colony optimizer, artificial bee colony algorithm, bat algorithm, firefly algorithm and glowworm algorithm

1 Giriş

Optimizasyon, bir problemde belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçme işlemidir. En iyileme anlamına da gelmektedir. Genel bir yapısal optimizasyon problemi, belirlenmiş bir amaç fonksiyonunu en büyük ya da en küçük yapacak optimum değerlerin, problem kısıtlarını karşılayacak şekilde tasarım değişkenleri arasından seçilmesi şeklinde ifade edilebilir.

Optimizasyon sürecinde optimizasyon probleminin matematiksel modelini oluşturmak en önemli adımdır. Bu modeli doğru oluşturabilmek tasarım değişkenlerinin, sınırlayıcıların ve amaç fonksiyonunun doğru formüle edilmesine bağlıdır.

Tipik bir optimizasyon probleminin matematiksel modeli aşağıdaki gibi gösterilebilir.

Amaç fonksiyonu,

$$z = f\ (x)\ (1.1)$$

Kısıtlar,

$$h_j(x) = 0,\ j = 1,2, \dots ,0\ (1.2)$$

$$g_k(x) \leq 0,\ k = p+1, \dots , m\ (1.3)$$

$$x_i \in X,\ X = \left(x_1, x_2, \dots , x_q\right)\ (1.4)$$

h_j eşitlik kısıtlarını ve g_k eşitsizlik kısıtlarını ifade etmektedir [1]. X tasarım değişkenlerini, q değişkenlerin toplam sayısını ifade eder. Sınırlayıcı sayısı ise m ile ifade edilmiştir [2].

Stokastik arama tekniklerine dayanan araştırma ve uygulamaların artması ile [3] sezgisel hesaplama tekniklerinde kullanılan kısıtlama (sınırlama) işlemi ön plana çıkmıştır [4].

Optimizasyon problemleri tasarım değişkenlerinin tipine bağlı olarak, sürekli ve süreksiz (kesikli) olmak üzere iki gruba ayrılmaktadır.

Bir optimizasyon probleminde değişkenler belli bir aralıkta tüm değerleri alabiliyorsa, bu optimizasyon problemi sürekli olarak adlandırılır.

Kesikli optimizasyonda, amaç fonksiyonu ve kısıtlar, karar değişkenlerinin sürekli bir fonksiyonu değildirler yani karar değişkenleri sadece belirli değerleri alabilmektedir.

Optimizasyon problemleri planlama aşamasında sabit değerler olarak verilmeyen birçok parametreye dayanmaktadır. Bunlara malzeme parametreleri (elastisite modülü, akma dayanımı, emniyet gerilmesi, moment kapasitesi), dış yükler, sürtünme katsayısı, atalet momenti gibi örnekler verilebilir. Bu parametreler, stokastik belirsizliklerin birçok tipine bağlı olarak rastgele değişkenlerin farklı olasılıklarını gösteren fonksiyonlarla modellenmelidir. Bu belirsizliklerin üstesinden gelebilmek için sıradan deterministik optimizasyon yöntemleri yerine stokastik optimizasyon yöntemleri kullanılır. Bu algoritmaların diğer algoritmalara göre avantajı, kesikli değişkenlere sahip optimizasyon problemlerine de uygulanabiliyor olmasıdır [5].

2 Optimizasyon Yöntemleri

2.1 Av Arama Optimizasyon Yöntemi

Av arama algoritmasında, kurt, aslan vb. hayvanların avlanmasından esinlenilmiştir. Bu hayvanların hepsi bir grupta avlanabilmektedir. Avı kuşatırlar ve yakalayana dek çemberi daraltırlar. Grubun her üyesi pozisyonunu, avın pozisyonuna ve kendi pozisyonuna bağlı olarak düzenler. Eğer bir av çemberden kaçmayı başarırsa avcılar, avı tekrar kuşatmak için grubu yeniden organize eder. Algoritmanın adımları aşağıda verilmiştir:

1. adımda yöntem parametreleri belirlenir. Bunlar: HGS:avcı sayısı, MML:lidere doğru en fazla hareket, HCGR: avcı gurubu önem oranı (0 ile 1 arasında değişir). MML ve HCGR parametreleri avcı pozisyonunu belirleyebilmek için kullanılır (çözüm vektörü).
2. adımda avcı sayısına (HGS) bağlı olarak gruptaki avcıların ilk konumu belirlenir.
3. adımda avcıların pozisyonları lidere doğru hareket etmeleri ile güncellenir. (Yeni çözüm vektörleri $x' = \{x'_1, x'_{2,}, \dots \dots , x'_2\}$)

Yeni çözüm vektörü:

$$x'_i = x_i + rand * MML * (x_i^l - x_i)\ (2.1)$$

Rand, 0 ile 1 arasında rastgele bir sayıdır. Her avcı için, avcının lidere doğru hareketi başarılı olursa yeni pozisyonunda kalır, eski pozisyonunun yenisine göre daha iyi sonuç vermesi durumunda ise önceki pozisyonuna geri döner.

4. adımda avcılar tekrar organize olur ve daha etkin avlanabilmek için birbirlerine göre pozisyonlarını

düzenlerler.

$$x_i^j \leftarrow \begin{cases} x_i^j \in \{x_i^1, x_i^2, \dots, x_i^{HGS}\}\ HGCR\ olasılığı\ ile \\ x_i^j = x_i^j \pm rand\ (1 - HGCR)\ olasılığı\ ile \end{cases} \quad (2.2)$$

5. adımda avcılar yerel optimuma sıkışmamak ve global optimum noktasını bulabilmek için pozisyonlarını güncellerler.

$$x_i' = x_i^l \pm rand * \left(x_i^{max} - x_i^{min}\right) * \alpha * \exp(-\beta * EN) \quad (2.3)$$

x^l =liderin pozisyonudur. Bu algoritma devam ederken çözüm optimum noktaya yakınsar.

3., 4. ve 5. adımlar en fazla iterasyon sayısı kadar tekrar edilir ve program durdurulur [6].

2.2 Parçacık Küme Optimizasyon Yöntemi

Reynolds 1987 yılında kuş sürülerine benzer parçacıkları modellediği çalışması ile parçacık sürü optimizasyon yönteminin temelini atmıştır [7]. Yöntemde parçacıklardan oluşan yapay bir sürü dikkate alınır. Sürüdeki her bir avcının davranışında üç esas temel alınır. İlki her parçacığın komşularının çok yakın olması durumunda onlardan uzaklaşmaya çalışmasıdır. İkincisi, her bir parçacığın komşularının ortalama istikametine yönlenmesidir. Üçüncüsü her parçacığın komşularının ortalama konumuna doğru yönelmesidir [7]. Eberhart ve Kenneddy ise 1995 yılında bu modellemelerin optimizasyon problemlerinin çözümü üzerine yoğunlaşmaya başlamışlar, popülasyon temelli sezgisel bir optimizasyon tekniği olan parçacık sürü optimizas-yon yöntemini geliştirmişlerdir [8]. Bu üç kuralı şu şekilde değiştirmişlerdir: her parçacık sürüdeki liderin konumuna yaklaşır, hangi konumun lidere daha yakın olduğunu hatırlar ve bu bilgiyi diğer parçacıklarla paylaşır. Algoritmanın adımları aşa-ğıda verilmiştir:

1. adımda parçacık kümesi x_0^1 pozisyonu ve tasarım kümesinden rastgele seçilmiş v_0^1 ilk hızı belirlenir.

$$x_0^{\text{i}} = x_{\min} + \text{r}(x_{\max} - x_{\min}) \quad (2.4)$$

$$v_0^{\text{i}} = [(x_{\min} + \text{r}(x_{\max} - x_{\min}))/\Delta\text{t} \quad (2.5)$$

r, 0 ile 1 arasında rastgele seçilen bir sayıyı, $x_{\max}$ ile $x_{\min}$ ise sırasıyla tasarım değişkenlerinin üst ve alt sınırlarını ifade eder.

2. adımda ilgili parçacıkların konumu (x_k^{i}) kullanılarak amaç fonksiyon değerleri f(x_k^{i}) hesaplanır.

3. adımda geçerli (k.) iterasyondaki optimum parçacık pozisyonu (p_k^{i}) ve global optimum parçacık pozisyonu (p_k^g) güncellenir.

4. adımda $x_{k+1}^{\text{i}} = x_k^{\text{i}} + v_{k+1}^{\text{i}}\Delta t$ denkleminden her parçacığın konumu güncellenir. x_{k+1}^{i} , (k+1). iterasyonda i parçacığının konumu, v_{k+1}^{i} ilgili hızdır.

5. adımda her parçacığın hızı güncellenir.

6. adım olarak, 2'den 5. adıma kadar olan adımlar durdurma kriteri sağlanıncaya kadar tekrarlanır [6].

2.3 Karınca Koloni Optimizasyon Yöntemi

Karınca koloni optimizasyon tekniği, karınca kolonilerinin besin kaynakları ile yuvaları arasındaki en kısa yolu bulabilmelerinden esinlenilmiştir. Tamamıyla kör olan karıncalar yuvaları ile besin kaynakları arasındaki en kısa yolu başarı ile bulabilmektedirler. Bunu, tipik özellikleri olan feromon salgılamalarıyla başarırlar. Feromonlar uçucu maddelerdir. Besin kaynağına giden karıncanın salgıladığı feromonları, arkasından gelen diğer karıncalar antenlerinde bulunan hassas alıcılar vasıtasıyla algılamakta ve böylelikle karıncalar birbirinin izini sürebilmektedirler. Bir karınca için iki yol arasında bir seçim yapma durumu ortaya çıktığında feromon yoğunluğunun daha fazla olduğu yolu seçer ve bu şekilde kısa yol uzun yola göre feromon yoğunluğunun artması ile güçlenir, bir süre sonra karıncaların büyük çoğunluğu kısa yolu tercih eder. Karınca koloni optimizasyon algoritmasının adımları aşağıda verilmiştir:

1. adımda her biri optimizasyon sorununun olası bir çözümünü temsil eden karıncaların sayısı seçilir. Optimizasyon probleminde her bir karar değişkeni için, parçacık değişkenlerinin alabileceği mümkün olan değerlerden oluşan bir havuz tanımlanır. Her karınca her bir karar değişkenine rasgele atanır. Başlangıç feromon miktarı (τ_0), τ_0=1/Z_{min} denkleminden hesaplanır. Z_{min} amaç fonksiyonunun min. değeridir.

2. adımda kolonideki her karınca ilk tasarım değişkenini seçer. Ve ardından karıncalar karar değişken değerlerini seçer. Bu seçim aşağıda verilen olasılık hesabına bağlı bir karar süreci ile gerçekleştirilir.

$$P_{ij}(t) = \frac{(\tau_{ij}(t))^{\alpha} * (v_{ij})^{\beta}}{\sum_j^{ndiv} (\tau_{ij}(t))^{\alpha} * (v_{ij})^{\beta}} \quad (2.6)$$

$P_{ij}(t)$ t zamanında i karar değişkeni için havuzdan j^{th} değerinin seçilme olasılığıdır. α ve β parametreleri sırasıyla yerel iz değerlerinin etkisini ve görünürlüğünü ayarlamak için kullanılan parametrelerdir. Bu süreç bütün karıncalar ilk tasarım değişkenlerine değerler atayana kadar devam eder.

3. adımda her turun sonunda lokal bir güncelleme kuralı uygulanır. Karıncalar için havuzdan seçilen

değerlerin feromon konsantrasyonu araştırmayı geliştirmek için düşürülür. Bu, gerçek hayatta feromon salgısının buharlaşmasına karşılık gelir. Bunun için kullanılan matematiksel tabir $\tau_{ij}(t)$=ζ*$\tau_{ij}(t)$ dir. ζ değeri 0 ile 1 arasında değişen lokal güncelleme katsayısıdır. Bu katsayı 1'e yakın seçilirse hızlı yakınsama oluşur ve algoritma lokal optimumla son bulur. Diğer taraftan 0'a yakın seçilirse problemde yakınsama zorluğu oluşur.

4. adımda bir sonraki karar değişkeni için havuzdan bir değer atanır ve kolonideki tüm karıncalar her bir karar değişkeni için bir değer alana kadar bu atama işlemine devam edilir. Bu turun sonunda lokal güncelleme kuralı uygulanır. Bu işlemin sonunda her karınca her karar değişkeni için bir değere sahip olur ve hepsi optimizasyon problemi için bir aday çözüm olur. Herhangi bir aday çözümün kısıtları karşılamaması durumunda Z aday çözümü yerini Z_p=Z(1 + C)$^{\varepsilon}$ eşitliğinde gösterilen cezalandırılmış aday çözüm Z_p'ye bırakır. C, toplam sınırlayıcı sayısı, ε ise cezalandırma katsayısıdır.

5. adımda aşağıdaki eşitlik kullanılarak global güncelleme uygulanır:

$$\tau_{ij}(t+n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (2.7)$$

ρ, 0 ile 1 arasında bir katsayıdır. $1-\rho$, t ile t+1 zamanları arasında feromonun buharlaşma miktarını temsil eder. $\Delta\tau_{ij}$, feromon miktarındaki değişimdir.

6. adım olarak, 2'den 5. adıma kadar olan adımlar durdurma kriteri (genellikle en fazla iterasyon sayısı) sağlanıncaya kadar tekrarlanır [6].

2.4 Yapay Arı Kolonisi Optimizasyon Yöntemi

Yapay arı koloni algoritması bal arısı kolonisinin yem ararken gösterdiği davranışını taklit eder. Yapay bal arısı algoritmasında farklı görevleri yerine getiren üç tip arı vardır. İlk arı grubu besin kaynağını bulan, nektarının miktarını değerlendiren ve daha iyi kaynakların yerini hafızasında tutan, 'işçi arılardır'. Bu arılar kovana geri döndüklerinde, bu bilgiyi dans alanındaki diğer arılarla dans ederek paylaşırlar. Dans süresi, gıda kaynağındaki nektar miktarını temsil eder. İkinci grup, dansı izleyen 'gözlemci arılardır' ve gıda kaynağının ziyaret edilmeye değer olduğuna karar verirlerse, gıda kaynağına uçmaya karar verebilirler. Üçüncü grup, kovan çevresinde rastgele yeni besin kaynakları araştıran 'izci arılardır'. Besin kaynağı diğer arılar tarafından terk edilen işçi arılar, izci arı olurlar. Her besin kaynağı, optimizasyon problemi için olası bir çözüm olarak düşünülür ve besin kaynağındaki nektar miktarı, uygunluk değeriyle tanımlanan çözümün kalitesini temsil eder. Yapay arı koloni algoritması dört aşamadan oluşur. Bu aşamalar başlatma safhası, işçi arı safhası, gözlemci arı safhası ve izci arı safhasıdır.

1. adımda besin kaynağı başlangıç popülasyonunun tüm vektörleri, (x_p , p=1,…,np) aşağıdaki bağıntı yardımı ile oluşturulur. Np popülasyon boyutudur (yapay arıların toplam sayısı). Her besin kaynağı, n kadar değişken içeren (x_{pi} , i=1,…,n) bir çözüm vektörü ve dikkate alınan optimizasyon problemi için potansiyel bir çözümdür.

$$x_{pi} = x_{li} + rand(0,1)(x_{ui} - x_{li}) \quad (2.8)$$

x_{li} ve x_{ui} , x_i'nin alt ve üst sınırlarıdır.

2. adımda işçi arılar aşağıdaki bağıntı yardımı ile yeni besin kaynağı araştırırlar.

$$v_{pi} = x_{pi} + \emptyset_{pi}(x_{pi} - x_{ki}) \quad (2.9)$$

k≠i rastgele seçilen bir besin kaynağı, $\emptyset_{pi}$ [-1,1] aralığında rastgele bir sayıdır. Yeni besin kaynağı üretildikten sonra uygunluk (sonuç) hesaplanır. Eğer sonucu x_{pi}'den daha iyi ise yeni besin kaynağı eskisi ile değiştirilir. Besin kaynağının uygunluğu aşağıdaki bağıntıya bağlı olarak oluşturulur.

$$sonuç(x_p) = \begin{cases} \frac{1}{1+f(x_p)} \; eğer \; x_p \geq 0 \\ 1 + abs\left(f(x_p)\right) \; eğer \; x_p < 0 \end{cases} \quad (2.10)$$

3. adımda işçi arılar buldukları besin kaynaklarının yer bilgisini gözlemci arılar ile paylaşırlar. Gözlemci arılar besin kaynaklarını, aşağıdaki bağıntıda gösterilen, popülasyondaki her besin kaynağının sonuç değeri kullanılarak hesaplanan olasılık değeri P_p'ye göre seçerler.

$$P_p = \frac{sonuç(x_p)}{\sum_{p=1}^{np} sonuç(x_p)} \quad (2.11)$$

Bir besin kaynağı x_{pi} gözlemci arı tarafından rastgele seçildikten sonra, komşu besin kaynağının sonucu ve olasılık değeri 2.10 ve 2.11 denklemi kullanılarak hesaplanır.

4. adımda belirli denemeden sonra çözümleri geliştirilemeyen işçi arılar izci arı olurlar ve çözümleri terk edilir. Yeni sonuçlar bulmak için aramaya başlarlar.

6. adım olarak, 2'den 4. adıma kadar olan adımlar durdurma kriteri sağlanıncaya kadar tekrarlanır [6].

2.5 Yarasa Algoritması Optimizasyon Yöntemi

Yarasa algoritması, 2010 yılında Yang tarafından, yarasaların ses (yankı) ile yön bulma yeteneklerini

taklit eden bir optimizasyon yöntemidir [9]. Yarasalar bu yetenekleri sayesinde avlarını yakalayabilir, engellerden kaçabilir ve karanlıkta konaklayabilirler. Yarasalar yüksek bir ses yayar ve bu sesin çevredeki nesnelerden geri dönen yankısını dinlerler. Sesi yaydıkları zaman ile yankısının geldiği zaman arasındaki farkı, yankılardaki değişimi kullanırlar. Hedefin uzaklığını, avın tipini ve hatta avın hareket hızını hesaplarlar. Algoritmanın adımları aşağıda verilmiştir:

1. adımda her yarasanın m tasarım değişkenli ve bir amaç fonksiyonlu optimizasyon probleminin aday bir çözümünü (x_i) temsil ettiği x_i pozisyon ve v_i hızında yarasa popülasyonu başlatılır.
2. adımda t zamanında yeni sonuçlar ve yeni hızlar hesaplanır.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (2.12)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*)f_i \quad (2.13)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (2.14)$$

β, 0 ile 1 arasında değişerek alınabilen rastgele bir sayıdır. x^*, n yarasa arasından bütün sonuçların kıyaslanmasından sonra, geçerli global en iyi konumdur (sonuç).
3. adımda eğer rastgele üretilen sayı rand > r_i ise en iyi çözümler arasından bir çözüm belirlenir.
4. adımda seçilen en iyi çözüm etrafında bir lokal çözüm, rastgele bir değişim ile belirlenir.

$$x_{yeni} = x_{eski} + \varepsilon A^t \quad (2.15)$$

ε, -1 ile 1 arasında değişen rastgele bir sayı, A^t ise bu adımdaki bütün yarasaların ortalama ses yüksekliğidir.
5. adımda eğer rastgele üretilen sayı rand > A_i ve $f(x_i) < f(x^*)$ ise yeni sonuçlar kabul edilir, r_i arttırılır, A_i azaltılır.

$$A_i^{t+1} = \alpha A_i^t \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \quad (2.16)$$

6. adımda yarasalar derecelendirilir ve en iyi x^* bulunur.
7. adım olarak durdurma kriteri sağlanıncaya kadar 2'den 6'ya kadar olan adımlar tekrar edilir [9].

2.6 Ateş Böceği Algoritması Optimizasyon Yöntemi

Ateş böceği algoritması, ateş böceklerinin yanıp sönme özellikleri esas alınarak oluşturulmuştur.
Ateş böceği algoritması 3 kurala dayanır. Bunlar:
1. Bütün ateş böcekleri çift cinsiyetlidir. Böylece her biri bir diğerini etkileyebilir.
2. Çekicilik ateş böceği parlaklığı ile doğru orantılıdır ve mesafe arttıkça ikisi de azalır. Her hangi bir ateş böceği çiftinde daha az parlak olan daha çok parlak olana doğru hareket eder. Eğer bir ateş böceğinden daha parlak bir şey yoksa rastgele hareket edecektir.
3. Ateş böceği parlaklığı amaç fonksiyonunun etkinliği ile direkt alakalıdır.
Algoritmanın adımları aşağıda verilmiştir:

1. adımda n kadar ateş böceği için başlangıç popülasyonu rastgele oluşturulur (x_i), (i=1,2,…,n). Her biri optimizasyon problemi için aday bir çözümü temsil eder.
2. adımda her ateş böceği için ışık şiddeti ve ateş böcekleri arasındaki mesafe hesaplanır.
3. adımda her ateş böceği diğer daha parlak ateş böceğine doğru hareket eder. Eğer daha parlak başka ateş böceği yoksa rastgele hareket eder.
4. adımda yeni sonuçlar değerlendirilir ve ışık şiddeti güncellenir.
5. adımda ateş böcekleri sıralanır ve en iyi sonuç bulunur.
6. adım olarak durdurma kriteri sağlanıncaya kadar 2'den 5'e kadar olan adımlar tekrar edilir [10].

2.7 Kandil Böceği Algoritması Optimizasyon Yöntemi

Kandil böceği optimizasyon yöntemi, kandil böceklerinin davranışından esinlenilerek oluşturulan bir optimizasyon yöntemidir. Kandil böcekleri, lüsiferin emisyonlarını değiştirerek farklı yoğunluklarda ışıldayabilme özelliğine sahiptirler. Birbirleriyle ışıldayarak etkileşime girerler. Eşlerini ya da avlarını da bu şekilde etkilerler. Işıldamaları ile etkileyicilikleri doğru orantılıdır. Yapay bir kandil böceği kolonisi oluşturmak için ilk olarak m kadar kandil böceği (temsilci) seçilir. Her temsilci optimizasyon probleminin potansiyel bir çözümüdür. Sürüdeki her kandil böceği komşularını ve onlardan aldığı sinyalin şiddetine göre hareket yönünü belirlemek için arama alanını kullanır.

Algoritmanın adımları aşağıda verilmiştir:
1. adımda başlangıç parametreleri alınır. ($\rho = 0.4, \gamma = 0.6, \beta = 0.08, n_t = 5, s = 0.03, l_0 = 5$)
2. adımda m kandil böceği optimizasyon probleminin arama alanında rastgele dağıtılır.
3. adımda her kandil böceği i'nin lüsiferin oranı güncellenir.

$$l_i(t) = (1-\rho)l_i(t-1) + \gamma J(x_i(t)) \quad (2.17)$$

4. adımda yakın çevresinde kendi lüsiferin değerinden daha yüksek lüsiferin değeri olan kandil böceklerinin sayısı seçilir.

5. adımda bu çevredeki her kandil böceği için $p_{ij}(t)$ olasılığı hesaplanır.

$$P_{ij}(t) = \frac{l_j(t)-l_i(t)}{\sum_{k\in N_i(t)} l_k(t)-l_i(t)} \quad (2.18)$$

6. adımda bu çevredeki j kandil böceği $P_{ij}(t)$ olasılığı ile seçilir ve i kandil böceğine doğru, aşağıdaki bağıntı yardımı ile hareket ettirilir.

$$x_i(t+1) = x_i(t) + st * \left\{\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}\right\} \quad (2.19)$$

7. adımda komşuluk aralığı ile güncellenir.

8. adım olarak durdurma kriteri sağlanıncaya kadar 3'den 7'ye kadar olan adımlar tekrar edilir [11].

3 Tasarım Örnekleri

Önceki bölümde açıklanmış olan optimizasyon yöntemleri literatürde bulunan bazı fonksiyonların çözümü için kullanılmış ve elde edilen optimum sonuçların irdelenmesi suretiyle yöntemler arasında performans kıyaslaması yapılmıştır.

3.1

İlk problem aşağıda fonksiyonu verilen, iki tasarım değişkeni ve iki sınırlayıcıya sahip olan bir minimizasyon problemidir.

$$\min f(x) = -(x_2 - 1.275x_1^2 + 5x_1 - 6)^2 - 10(1-1/8\pi)\cos(\pi x_1) - 10 \quad (3.1)$$

Kısıtlar;

$$g_1(x) = -\pi x_1 - x_2 \le 0 \quad (3.2)$$

$$g_2(x) = -\pi^2 x_1^2 + 4x_2 \le 0 \quad (3.3)$$

Tasarım değişkenlerinin aralıkları;

$$-1.5 \le x_1 \le 3.5 \quad (3.4)$$

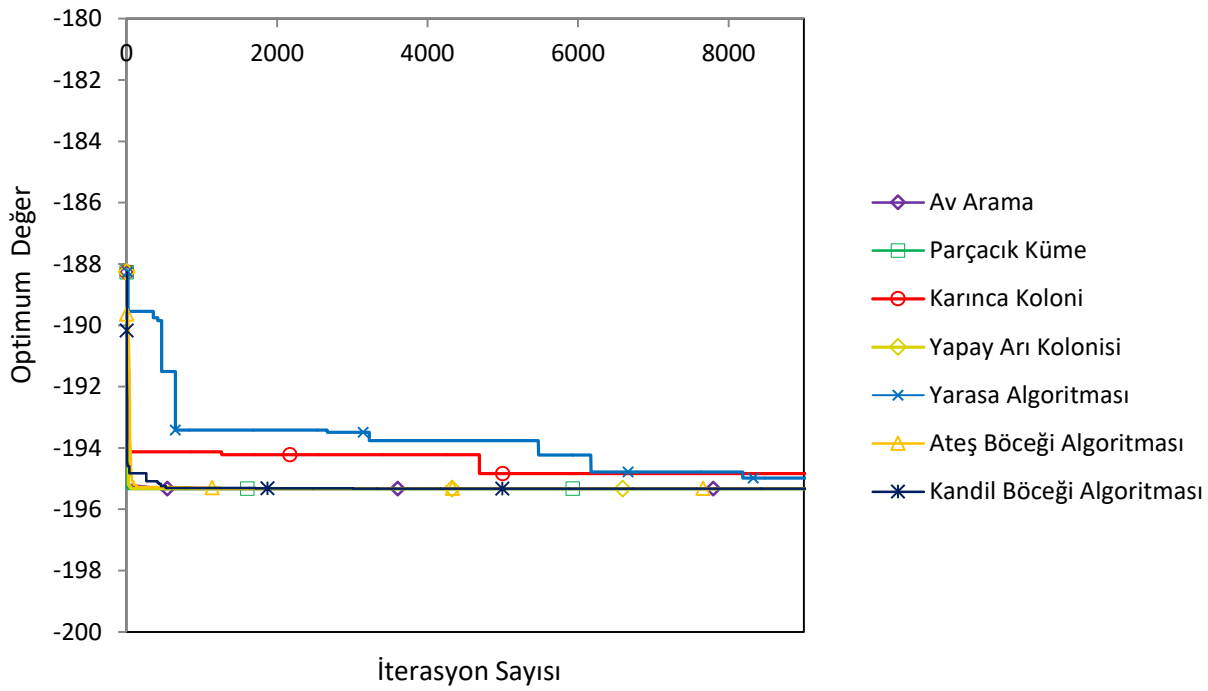
$$0 \le x_2 \le 15 \quad (3.5)$$

Bu karşılaştırma fonksiyonu 7 farklı optimizasyon yöntemine (HUS: Av Arama, PSO: Parçacık Küme, ACO: Karınca Koloni, ABC: Yapay Arı Kolonisi, BAT: Yarasa Algoritması, FFO: Ateş Böceği Algoritması, GLSO: Kandil Böceği Algoritması) tabi tutulmuş, her bir optimizasyon yöntemi için 40 avcı seçilmiş ve 15000 iterasyon yapılmıştır. Elde edilen optimum sonuçlar Tablo 1'de, amaç fonksiyonu değerinin iterasyon sayısına göre değişimi ise Şekil 1'de verilmiştir. Optimum sonucun av arama (Hus) ve kandil böceği (GLSO) Optimizasyon yöntemleri ile bulunduğu gözlenmiştir. Bu problemdeki en iyi yakınsama performansı kandil böceği algoritmasına aittir.

Tablo 1. Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Tasarım Değişkenleri	HUS	PSO	ACO	ABC	BAT	FFO	GLSO
x_1	2.46686	2.46686	2.87645	2.46686	2.46573	2.46687	2.46687
x_2	15.0000	15.0000	6.06089	15.0000	14.9845	14.9999	15.0000
$g_1(x)$	-22.7459	-22.746	-22.7499	-22.7459	-22.7269	-22.7458	-22.7459
$g_2(x)$	0.000004	-0.00001	-0.22667	-0.00001	-0.00681	-0.00061	-0.00006
f(x)	**-195.327**	**-195.3268**	**-194.838**	**-195.327**	**-194.980**	**-195.323**	**-195.327**



Şekil 1. Amaç Fonksiyonu Yakınsama Grafiği

3.2

Ele alınan ikinci problem aşağıda fonksiyonu verilen altı tasarım değişkeni ve altı kısıtlayıcıya sahip olan bir minimizasyon problemidir.

$$\min f(x) = -18\log(x_2 + 1) - 19.2\log(x_1 - x_2 + 1) + 5x_4 + 6x_5 + 8x_6 + 10x_1 - 7x_3 + 10 \quad (3.6)$$

Kısıtlar;

$$g_1(x) = -(0.8\log(x_2 + 1) + 0.96\log(x_1 - x_2 + 1) - 0.8x_3) \leq 0 \quad (3.7)$$

$$g_2(x) = -(\log(x_2 + 1) + 1.2\log(x_1 - x_2 + 1) - x_3 - 2x_6 + 2 \leq 0 \quad (3.8)$$

$$g_3(x) = x_2 - x_1 \leq 0 \quad (3.9)$$

$$g_4(x) = x_2 - 2x_4 \leq 0 \quad (3.10)$$

$$g_5(x) = -x_2 + x_1 - 2x_5 \leq 0 \quad (3.11)$$

$$g_6(x) = x_4 + x_5 - 1 \leq 0 \quad (3.12)$$

Tasarım değişkenlerinin aralıkları;

$$0 \leq x_1 \leq 2 \quad (3.13)$$

$$0 \leq x_2 \leq 2 \quad (3.14)$$

$$0 \leq x_3 \leq 1 \quad (3.15)$$

$$0 \leq x_4 \leq 1 \quad (3.16)$$

$$0 \leq x_5 \leq 1 \quad (3.17)$$

$$0 \leq x_6 \leq 1 \quad (3.18)$$

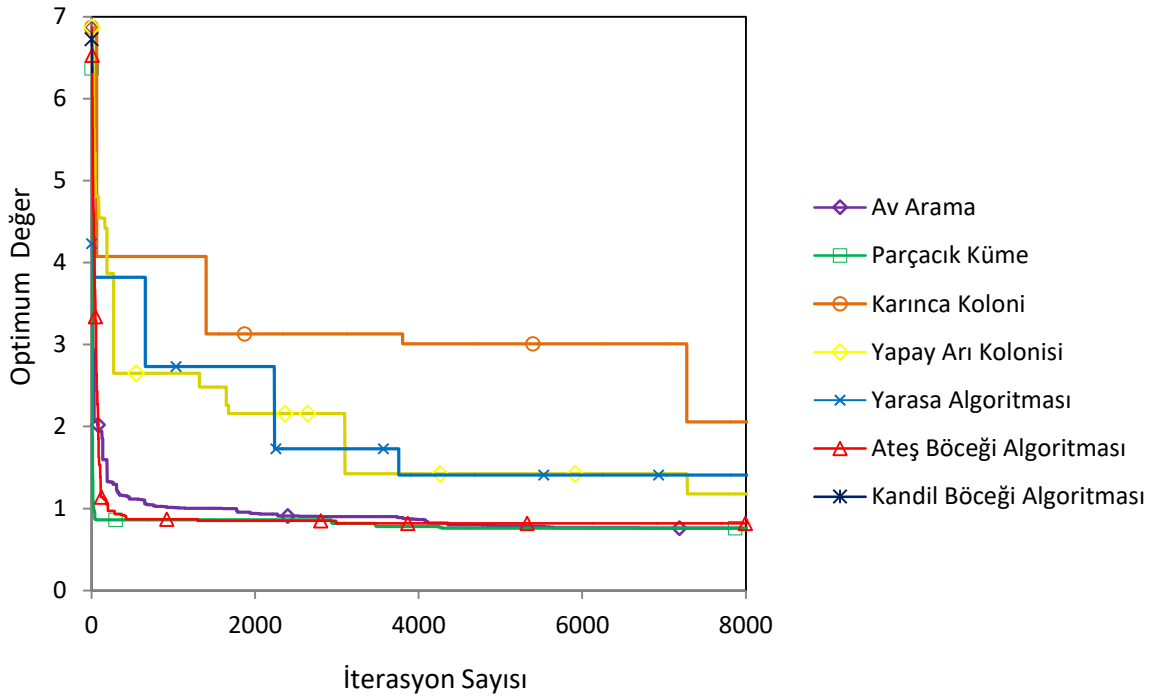
Fonksiyon 7 farklı optimizasyon yöntemine tabi tutulmuş, her bir optimizasyon yöntemi için 55 avcı seçilmiş ve 20000 iterasyon yapılmıştır. Elde edilen optimum sonuçlar Tablo 2'de, amaç fonksiyonu değerinin iterasyon sayısına göre değişimi ise Şekil 2'de verilmiştir. Optimum sonucun av arama optimizasyon yöntemi (Hus) ile bulunduğu gözlenmiştir. Bu problemdeki en iyi yakınsama performansı ise ateş böceği algoritmasına aittir.

Tablo 2. Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Tasarım Değişkenleri	HUS	PSO	ACO	ABC	BAT	FFO	GLSO
x1	1.14654	1.14543	0.26603	1.18521	1.11322	1.16665	1.23494
x2	0.54693	0.53854	1.38998	0.69524	0.39593	0.53563	0.49111
x3	1.00000	1.00000	0.24048	0.99908	0.97968	0.99956	1.00000

x_4	0.27346	0.26926	0.07142	0.37449	0.25517	0.26831	0.24564
x_5	0.29980	0.30344	0.50523	0.24706	0.37198	0.31911	0.37672
x_6	0.00000	0.00000	0.88481	0.01506	0.00176	0.00064	0.00000
$g_1(x)$	0.00001	0.000001	-0.0639	-0.0058	-0.0022	-0.0131	-0.0534
$g_2(x)$	-1.9999	-1.99999	-2.0581	-1.9771	-1.9992	-2.0151	-2.0668
$g_3(x)$	-0.5996	-0.60689	-0.6191	-0.4899	-0.71729	-0.6310	-0.7438
$g_4(x)$	0.00001	0.000001	-0.0839	-0.0537	-0.11441	-0.0010	-0.0002
$g_5(x)$	0.00001	0.000001	-0.0574	-0.0041	-0.02667	-0.0072	-0.0096
$g_6(x)$	-0.1269	-0.12383	-0.0286	-0.1313	-0.00087	-0.0934	-0.0009
f(x)	**0.75921**	**0.759738**	**1.86849**	**1.17692**	**1.409854**	**0.81716**	**0.9697**



Şekil 2. Amaç Fonksiyonu Yakınsama Grafiği

3.3

Bu problem, aşağıda fonksiyonu verilen iki tasarım değişkeni ve iki kısıtlayıcıya sahip olan bir maksimizasyon problemidir.

$$\max f(x) = [sin^3(2\pi x_1)\sin(2\pi x_2)]/[x_1^3(x_1 + x_2)] \quad (3.19)$$

Kısıtlar;

$$g_1(x) = x_1^2 - x_2 + 1 \leq 0 \quad (3.20)$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0 \quad (3.21)$$

Tasarım değişkenlerinin aralıkları;

$$0 \leq x_1 \leq 10 \quad (3.22)$$

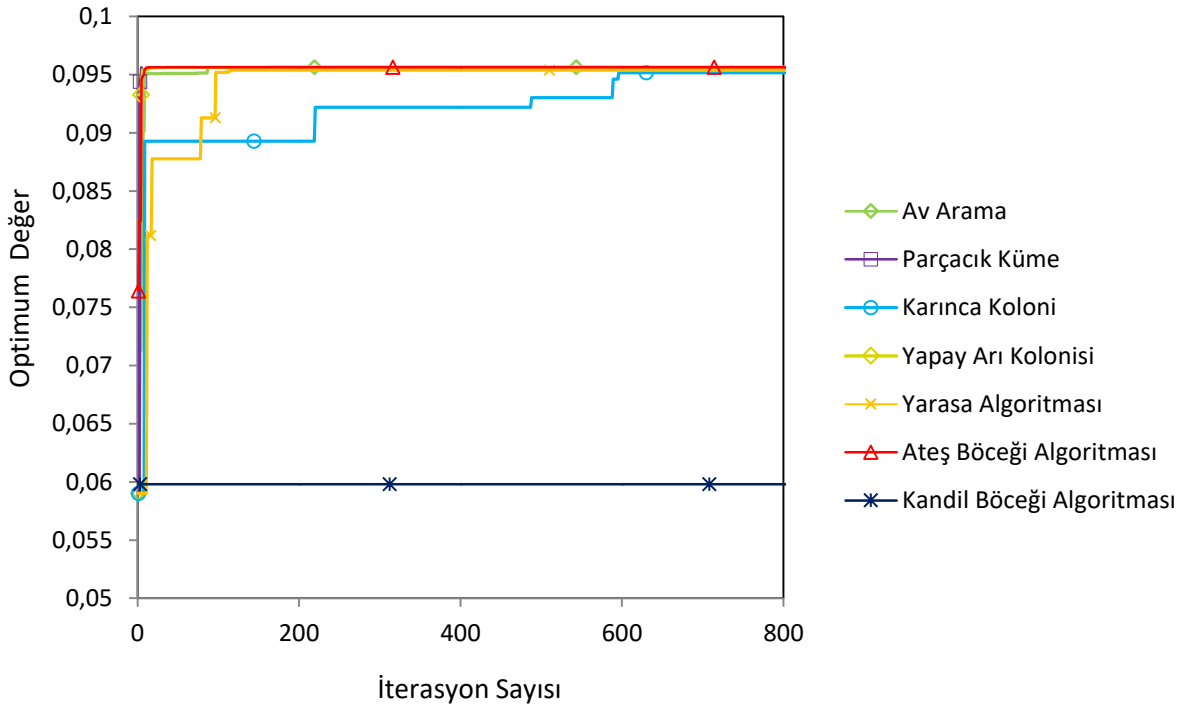
$$0 \leq x_2 \leq 10 \quad (3.23)$$

Maksimizasyon problemi 7 farklı optimizasyon yöntemine tabi tutulmuş, her bir optimizasyon yöntemi için 40 avcı seçilmiş ve 15000 iterasyon yapılmıştır. Elde edilen optimum sonuçlar Tablo 3'de, amaç fonksiyonu değerinin iterasyon sayısına göre değişimi ise Şekil 3'de verilmiştir. Bu problemin optimum sonucunu bulmada 5 optimizasyon yönteminin aynı sonuçları verdiği görülmüştür (HUS, PSO, ABC, BAT, FFO). Bu yöntemlerin yakınsama olarak da benzer performanslar gösterdiği sonuçlardan anlaşılmaktadır. Yakınsama ve optimum sonucu bulma anlamında kandil böceği algoritmasının bu problemde gösterdiği performansın iyi olmadığı görülmüştür.

Tablo 3. Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Tasarım Değiş-kenleri	HUS	PSO	ACO	ABC	BAT	FFO	GLSO
x_1	1.2286	1.2286	1.4701	1.2286	1.2287	1.2286	1.1919
x_2	4.2475	4.2475	4.3319	4.2475	4.2474	4.2474	4.3797
$g_1(x)$	-1.7380	-1.7380	-1.739	-1.7381	-1.7376	-1.7379	-1.9591
$g_2(x)$	-0.1673	-0.1673	-0.166	-0.1673	-0.1674	-0.0167	-0.0476
f(x)	**0.0956**	**0.0956**	**0.0955**	**0.0956**	**0.0956**	**0.0956**	**0.0598**



Şekil 3. Amaç Fonksiyonu Yakınsama Grafiği

3.4

Ele alınan dördüncü problem, iki tasarım değişkeni ve iki kısıtlayıcıya sahip olan ve aşağıda fonksiyonu verilen bir minimizasyon problemidir.

$$\min f(x) = 0.01x_1^2 + x_2^2 \quad (3.24)$$

Kısıtlar;

$$g_1(x) = -x_1x_2 + 25 \leq 0 \quad (3.25)$$

$$g_2(x) = -{x_1}^2 - {x_2}^2 + 25 \leq 0 \quad (3.26)$$

Tasarım değişkenlerinin aralıkları;

$$2 \leq x_1 \leq 50 \quad (3.27)$$

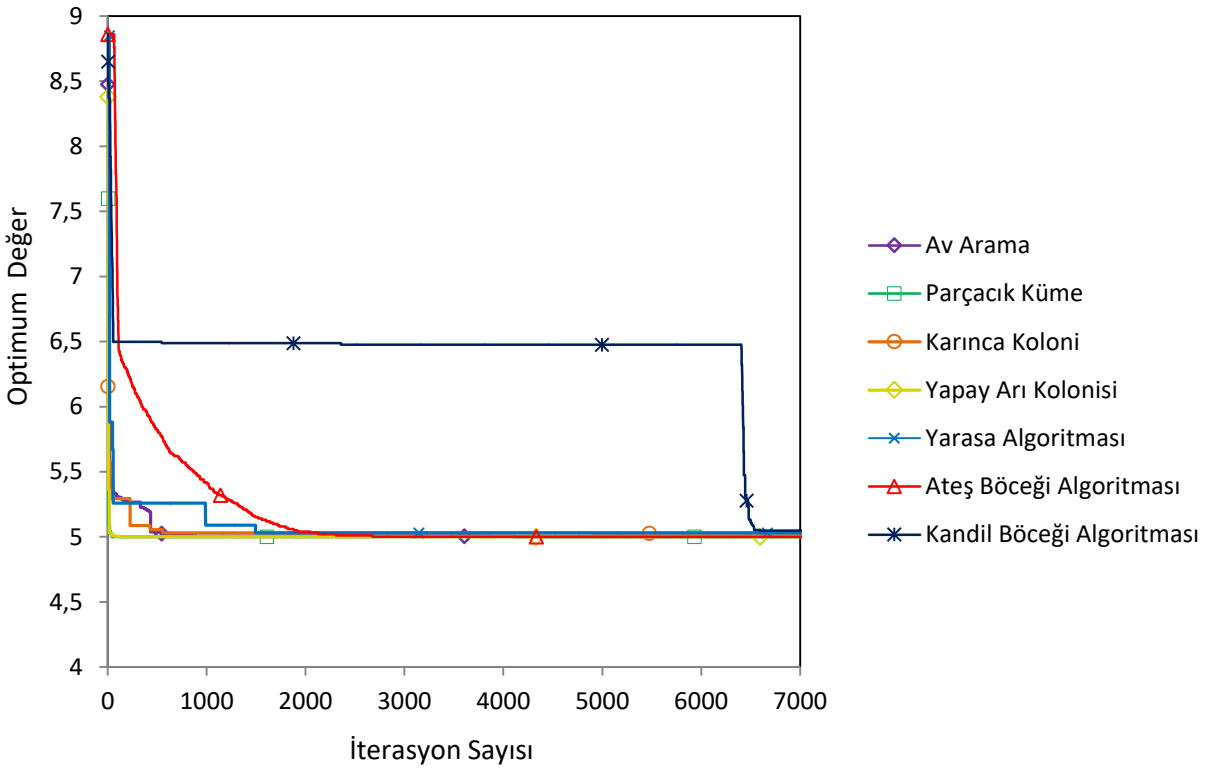
$$0 \leq x_2 \leq 50 \quad (3.28)$$

Fonksiyon 7 farklı optimizasyon yöntemine tabi tutulmuş, her bir optimizasyon yöntemi için 40 avcı seçilmiş ve 20000 iterasyon yapılmıştır. Elde edilen optimum sonuçlar Tablo 4'te, amaç fonksiyonu değerinin iterasyon sayısına göre değişimi ise Şekil 4'te verilmiştir. Elde edilen sonuçlar irdelendiğinde en iyi sonucu av arama yönteminin verdiği görülebilir. Bu problemdeki en iyi yakınsama performansı ateş böceği algoritmasına aittir.

Tablo 4. Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Tasarım Değiş-kenleri	HUS	PSO	ACO	ABC	BAT	FFO	GLSO
x_1	15.8131	15.81186	40.35514	15.8142	15.648	15.8159	16.93438
x_2	1.58097	1.581092	25.66774	1.58085	1.5988	1.58068	1.476845
$g_1(x)$	0.00001	-0.000001	-0.01813	0.000001	-0.0181	-0.00001	-0.00945
$g_2(x)$	-227.55	-227.5148	-234.568	-227.590	-222.41	-227.643	-263.954
f(x)	**4.99999**	**5.00000**	**5.005506**	**5.00000**	**5.00477**	**5.00000**	**5.048803**



Şekil 4. Amaç Fonksiyonu Yakınsama Grafiği

3.5

Performans kıyaslaması için kullanılan beşinci problem, altı tasarım değişkeni ve beş kısıtlayıcıya sahip olan aşağıda fonksiyonu verilen bir minimizasyon problemidir.

$$\min f(x) = -x_4 \quad (3.29)$$

Kısıtlar;

$$g_1(x) = 0.09755988x_1x_5 + x_1 - 1 \leq 0 \quad (3.30)$$

$$g_2(x) = 0.0965842812x_2x_6 + x_2 - x_1 \leq 0 \quad (3.31)$$

$$g_3(x) = 0.0391908x_3x_5 + x_3 + x_1 - 1 \leq 0 \quad (3.32)$$

$$g_4(x) = 0.03527172x_4x_6 + x_4 - x_1 + x_2 - x_3 \leq 0 \quad (3.33)$$

$$g_5(x) = \sqrt{x_5} + \sqrt{x_6} - 4 \leq 0 \quad (3.34)$$

Tasarım değişkenlerinin aralıkları;

$$0 \leq x_1 \leq 1 \quad (3.35)$$

$$0 \leq x_2 \leq 1 \quad (3.36)$$

$$0 \leq x_3 \leq 1 \quad (3.37)$$

$$0 \leq x_4 \leq 1 \quad (3.38)$$

$$0.00001 \leq x_5 \leq 16 \quad (3.39)$$

$$0.00001 \leq x_6 \leq 16 \quad (3.40)$$

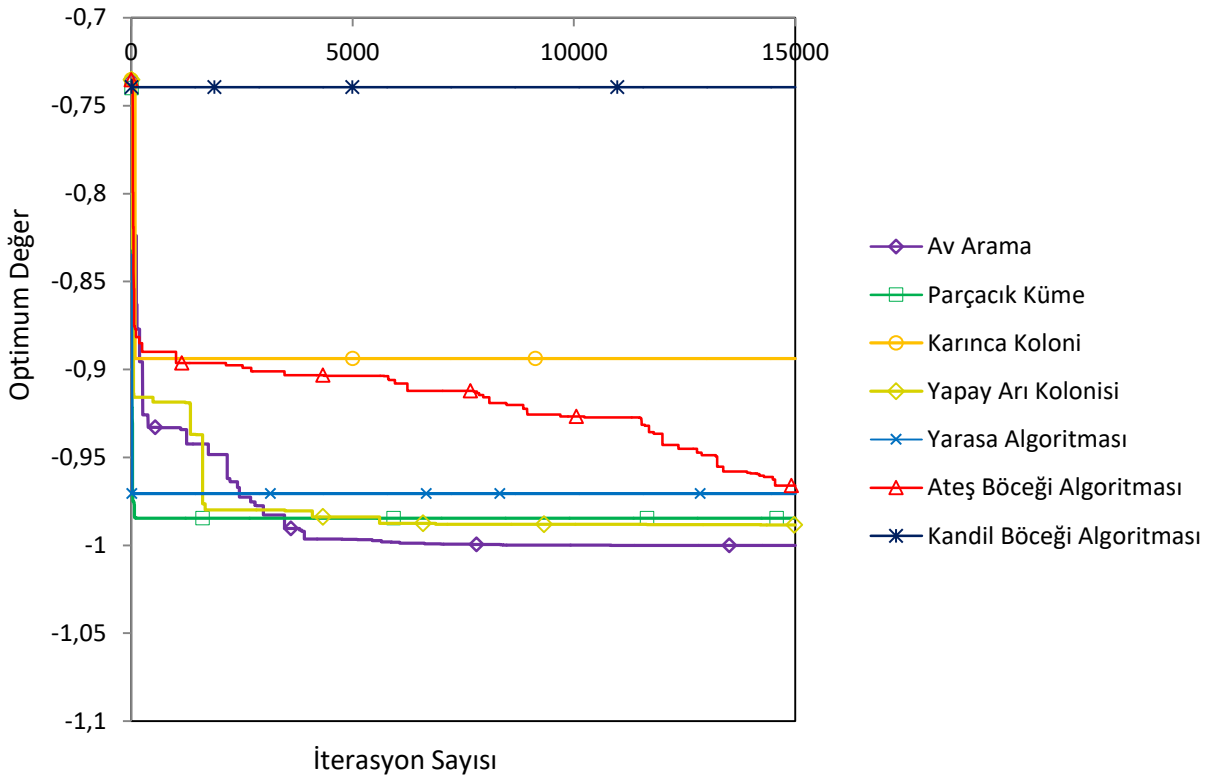
Fonksiyon 7 farklı optimizasyon yöntemine tabi tutulmuş, her bir optimizasyon yöntemi için 40 avcı seçilmiş ve 15000 iterasyon yapılmıştır. Elde edilen optimum sonuçlar Tablo 5'te, amaç fonksiyonu değerinin iterasyon sayısına göre değişimi ise Şekil 5'te verilmiştir. Optimum sonuçlardan da anlaşılacağı

üzere bu problem için en iyi performansı av arama algoritması göstermiştir. Yakınsama açısından da yine bu tekniğin diğerlerine olan üstünlüğü, amaç fonksiyonu değeri – iterasyon sayısı grafiğinden varılacak diğer bir sonuçtur.

Tablo 5. Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Tasarım Değiş-kenleri	HUS	PSO	ACO	ABC	BAT	FFO	GLSO
x_1	0.54441	0.3472	0.5644	0.1923	0.3963	0.8107	0.1917
x_2	0.00000	0.0000	0.1094	0.0013	0.0115	0.0019	0.1251
x_3	0.45559	0.6454	0.2990	0.7985	0.5673	0.1736	0.7926
x_4	1.00000	0.9846	0.1886	0.9883	0.9706	0.9659	0.7393
x_5	0.00009	0.2917	6.0937	0.0048	0.2680	2.2503	0.2102
x_6	0.00005	0.2311	1.9671	0.0085	0.4255	0.4793	1.7487
$g_1(x)$	-0.4556	-0.643	-0.929	-0.807	-0.593	-0.0112	-0.8043
$g_2(x)$	-0.5444	-0.347	-0.069	-0.191	-0.384	-0.8087	-0.0454
$g_3(x)$	0.00001	0.0000	-0.016	-0.009	-0.0303	-0.0003	-0.0091
$g_4(x)$	-0.0000	0.0000	-0.035	-0.001	.03304	-0.0001	-0.0742
$g_5(x)$	-3.9831	-2.979	-2.628	-3.838	-2.829	-1.8075	-2.2190
f(x)	**-1.0000**	**-0.985**	**-0.894**	**-0.988**	**-0.9707**	**-0.9659**	**-0.7393**



Şekil 5. Amaç Fonksiyonu Yakınsama Grafiği

3.6 Ele alınan altıncı optimizasyon problemi, 3.41. ba-

ğıntıda fonksiyonu verilen iki tasarım değişkeni ve iki kısıtlayıcıya sahip olan bir minimizasyon problemidir.

$$\min f(x) = x_1\sqrt{1+x_2^2} \quad (3.41)$$

Kısıtlar;

$$g_1(x) = 0.124\sqrt{1+x_2^2}\left(\frac{8}{x_1}+\frac{1}{x_1x_2}\right) - 1 \le 0 \quad (3.42)$$

$$g_2(x) = 0.124\sqrt{1+x_2^2}\left(\frac{8}{x_1}-\frac{1}{x_1x_2}\right) - 1 \le 0 \quad (3.43)$$

Tasarım değişkenlerinin aralıkları;

$$0.2 \le x_1 \le 4 \quad (3.44)$$

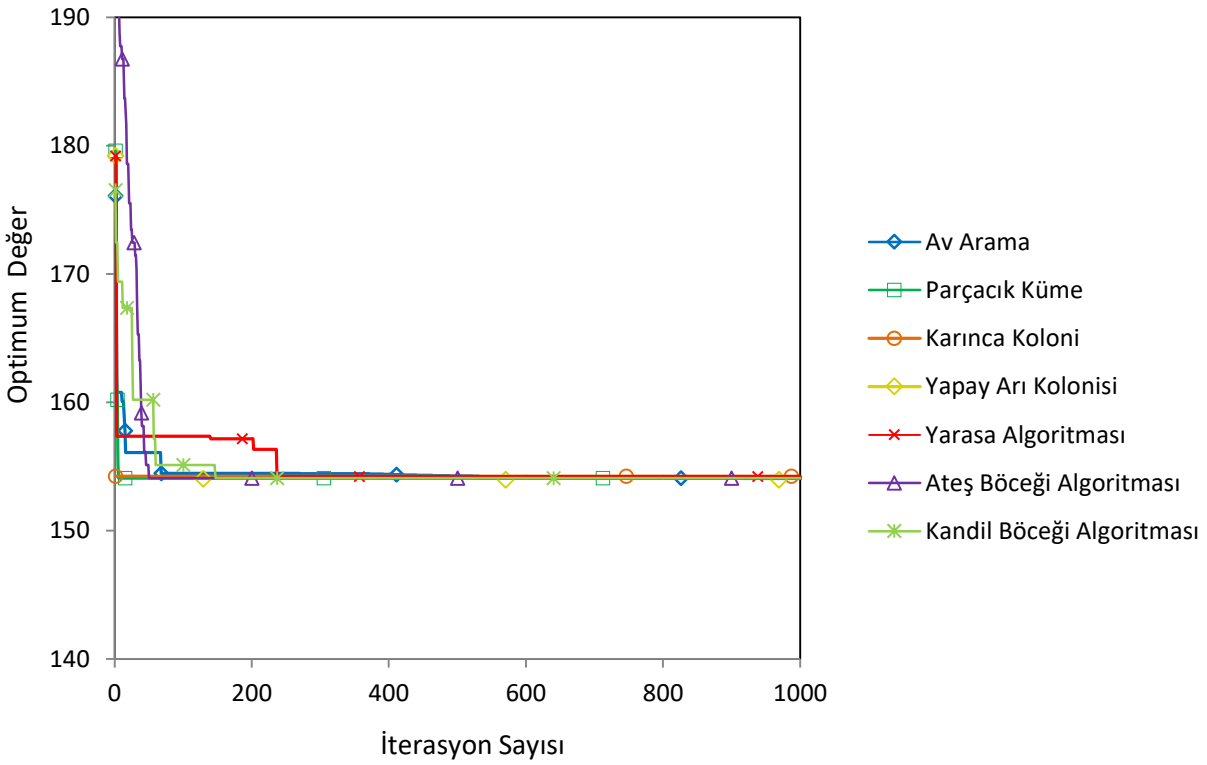
$$0.1 \le x_2 \le 1.6 \quad (3.45)$$

Fonksiyon 7 farklı optimizasyon yöntemine tabi tutulmuş, her bir optimizasyon yöntemi için 40 avcı seçilmiş ve 20000 iterasyon yapılmıştır. Elde edilen optimum sonuçlar Tablo 6'da, amaç fonksiyonu değerinin iterasyon sayısına göre değişimi ise Şekil 6'da verilmiştir. Bu problem için bütün teknikler benzer performanslar göstermekle birlikte en iyi sonucun av arama tarafından elde edildiği kaydedilmiştir. Yakınsama özelliği açısından da yarasa algoritması dışındaki yöntemlerin yakın performanslar gösterdiği izlenmiştir.

Tablo 6. Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar

Farklı Optimizasyon Yöntemlerinden Elde Edilen Optimum Sonuçlar**(42)**

Tasarım Değiş-kenleri	HUS	PSO	ACO	ABC	BAT	FFO	GLSO
x_1	1.41164	1.41160	3.45224	1.41142	1.41532	1.412130	1.41172
x_2	0.37702	0.37713	0.81947	0.37751	0.36943	0.376006	0.37687
$g_1(x)$	0.00001	0.000001	-0.49873	-0.0000	0.00002	0.000001	-0.00001
$g_2(x)$	-0.4979	-0.49787	-0.00010	-0.4975	-0.50562	-0.49899	-0.49813
f(x)	**1.50863**	**1.508651**	**1.508809**	**1.50865**	**1.508821**	**1.508655**	**1.508653**



Şekil 6. Amaç Fonksiyonu Yakınsama Grafiği

4 Sonuç ve Öneriler

Bu çalışmada av arama algoritması, parçacık küme algoritması, karınca koloni algoritması, yapay arı kolonisi algoritması, yarasa algoritması, ateş böceği algoritması ve kandil böceği algoritması olmak üzere toplam yedi adet optimizasyon tekniği incelenmiş olup bu teknikler matematiksel fonksiyonların optimum sonucunun bulunmasında kullanılmıştır. Bu amaçla altı adet matematiksel fonksiyon seçilmiştir. İlk fonksiyon için av arama ve kandil böceği optimizasyon yöntemlerinin, ikinci fonksiyon için ise av arama ve parçacık küme optimizasyon yöntemlerinin en iyi sonuç verdiği görülmüştür. Üçüncü fonksiyona bakıldığında altı optimizasyon tekniği ile aynı optimum sonucun bulunduğu, kandil böceği optimizasyon tekniği ile ise diğer-lerinden daha kötü sonuç bulunduğu görülmüştür. Dördüncü ve beşinci fonksiyonlarda ise av arama optimizasyon tekniği, en iyi sonucun bulunduğu teknik olmuştur. Teknikler yakınsama performansı açısından kıyaslanacak olursa, ilk problemde kandil böceği algoritması, ikinci ve dördüncü problemlerde ateş böceği algoritması, beşinci problemde av arama algoritması başarılı bulunmuştur. Üçüncü ve altıncı problemlerde ise problemlerde yarasa algoritması dışındaki algoritmalar optimum sonuca yakınsama performansı açısından başarılı bulunmuştur. Kullanılan sürü zekası tabanlı optimizasyon algoritmalarının optimum sonucu bulmada başarılı oldukları, problem tipine göre en iyi algoritmanın değiştiği görülmüştür. Fakat problemlerin çoğunluğu göz önüne alınacak olursa av arama optimizasyon yönteminin diğerlerine nazaran daha başarılı bulunduğu söylenebilir. Bununla birlikte, algoritma performansının, algoritma parametreleri için uygun değer seçimine bağlı oluşunun sezgisel tekniklerin genel bir özelliği olduğu dikkate alın-malıdır.

5 Referanslar

[1] Çarbaş, S.; Doğan, E.; Erdal, F.; Saka, M.P. Comparison of Metaheuristic Techniques in Finding The Solutions of Optimi-zation Problems.; 2 nd International Symposium on Computing in Science & Engineering, 2011.

[2] Doğan, E. Optimum Design Of Rigid And Semi-Rigid Steel Sway Frames Including Soil - Structure Interaction.; Ankara: ODTÜ Fen Bilimleri Enstitüsü. Doktora Tezi, 2010.

[3] Sun, W.; Y-X, Yuan. Optimization Theory and Methods; Nonlinear Programming, Springer-Verlag, 2006.

[4] Coello, C.A.C. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering, 2002; 1245–1287.

[5] Kurt, M. Stochastic Optimization Methods. Press: Springer, Munich, Germany, 2005; 314 pp.

[6] Saka, M.P.; Doğan, E.; Aydoğdu, İ. Review and Analysis of Swarm-Intelligence Based Algorithms.; Swarm Intelligence And Bio-Inspired Computation.; Elsevier; 450 pp.

[7] Reynolds, C.W. Flocks, herds, and schools: A distributed behavioral model. ACM Computer Graphics, 1987; 21, 25–34.

[8] Kennedy, J.; Eberhart, R. Particle swarm optimization, Institute of Electricaland Electronics Engineers Press 4. 1995; 1942-1948.

[9] Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization. Springer, 2010; 65-74.

[10] Yang, X. S. Firefly Algorithms for Multimodal Optimization. in: Stochastic Algorithms. Lecture Notes in Computer Science, 2009; 169-178.

[11] Krishnanand, K. N.; Ghose, D. Glowworm swarm Optimization: A New Method for Optimizing Multi-Modal Functions, International Journal of Computational intelligence studies, 1, 84–91.