DÜMF Mühendislik Dergisi
web: http://dergipark.gov.tr/dumf

Araştırma Makalesi / Research Article

Kısıtlı optimizasyon problemlerinin çözümü için atom arama optimizasyon algoritması

Erdal EKER*
Muş Alparslan Üniversitesi, Pazarlama ve Reklamcılık Bölümü, Muş
erdaleker4965@gmail.com ORCID: 0000-0002-5470-8384, Tel: (507) 696 63 26

Murat KAYRİ
Batman Üniversitesi, Bilgisayar Mühendisliği Bölümü, Batman
murat.kayri@batman.edu.tr ORCID: 0000-0002-5933-6444

Serdar EKİNCİ
Batman Üniversitesi, Bilgisayar Mühendisliği Bölümü, Batman
serdar.ekinci@batman.edu.tr ORCID: 0000-0002-7673-2553

Geliş: 15.03.2019, Revizyon:05.04.2019, Kabul Tarihi: 29.04.2019

Öz

Son yıllarda evrim, fizik, matematik ve sürü ilhamlı çok sayıdaki sezgisel-üstü optimizasyon teknikleri, bilim ve mühendislik alanlarına önerildi. Atom arama optimizasyonu (ASO), temel moleküler dinamiklerden esinlenen popülasyon tabanlı yeni bir optimizasyon algoritmasıdır. ASO, basitliği ve az sayıda kontrol parametresi sayesinde optimizasyon problemlerine kolaylıkla uygulanabilir. ASO, en çok bilinen sekiz test fonksiyonuna (Sphere, Rosenbrock, Step, Schwefel, Rastrigin, Ackley, Griewank ve Egg Crate) uygulandı. Ayrıca, her test fonksiyonu için ASO ile elde edilen istatistiksel sonuçlar (ortalama, standart sapma ve en iyi değer) literatürdeki diğer algoritmalarla elde edilen sonuçlarla karşılaştırıldı. Parçacık sürüsü optimizasyonu (PSO), yapay arı kolonisi (ABC) ve sinüs kosinüs algoritması (SCA) karşılaştırma için seçilen diğer metotlardır. Tüm test fonksiyonları için elde edilen istatistiksel sonuçlar ve yakınsama hızlarına bakıldığında, ASO algoritmasının kısıtlı optimizasyon problemlerini çözmedeki üstün performansı göze çarpmaktadır.

Anahtar Kelimeler: *Atom arama optimizasyonu; sezgisel-üstü; optimizasyon, test fonksiyonları.*

* Yazışmaların yapılacağı yazar

DOI: 10.24012/dumf.540402

Giriş

Optimizasyon, verilen bir sistemin çıkışını eniyilemek için sistem parametrelerinin mümkün olan tüm değerler arasından optimum olanları bulma işlemini ifade eder. Optimizasyon problemleri, tüm araştırma alanlarında bulunabilir, bu da optimizasyon tekniklerinin geliştirilmesini zorunlu kılarken bu alanda çalışan araştırmacılar için de ilgi odağı haline gelir. Geleneksel optimizasyon paradigmalarının, yerel optimumda takılı kalma ve arama alanının belirlenme ihtiyacı gibi sakıncaları nedeniyle, son yirmi yılda sezgisel üstü optimizasyon yaklaşımlarına ilginin arttığı gözlenmektedir (Mirjalili, 2016b; Boussaïd vd., 2013; Ekinci ve Hekimoglu, 2019).

Sezgisel üstü eniyileme algoritmaları akıllı hesaplamalarda giderek daha yaygın hale gelerek çok sayıda gerçek mühendislik problemlerine de yaygın olarak uygulanmaktadır (Ekinci ve Demiroren, 2016). Yaygın hale gelmeleri aşağıdaki yönlerinden kaynaklanmaktadır. İlk olarak, tüm bu optimizasyon tekniklerinin, gerçek hayatta bulunan ve her türlü fiziksel olay veya biyolojik davranıştan esinlenilen, makul olduğu kanıtlanmış bazı temel teorileri ve matematiksel modelleri vardır (Jaddi vd., 2017). Teoriler basit ve anlaşılması kolaydır. İkincisi, optimizasyon algoritmaları bir kara kutu gibi düşünülebilir. Bu, algoritmaların herhangi bir optimizasyon problemi için verilen bir giriş kümesi için kolaylıkla bir çıkış kümesi oluşturabildiği anlamına gelir. Araştırmacı, daha iyi çözümler elde etmek için algoritmaların yapılarını ve parametrelerini değiştirebildiğinden çok esnek ve çok yönlüdürler. Üçüncüsü, sezgisel üstü algoritmalar, çoğu mühendislik problemleri çok modlu fonksiyonlar olarak kabul edildiğinden, bu problemlerin çözümünde yerel minimumlardan etkin bir şekilde sakınmak gibi çok değerli bir özelliğe sahiptir. Ek olarak, makul bir süre içinde çözümlerin doğruluğunu artırmak için diğer algoritmaların iyi yanları kullanılarak farklı versiyonları geliştirilebilir. Dördüncüsü, sezgisel-üstü optimizasyon algoritmaları, tek amaçlı ve çok amaçlı problemler, düşük boyutlu ve yüksek boyutlu problemler, tek modlu ve çok modlu problemler ile ayrık ve sürekli problemler dahil ancak bunlarla sınırlı olmamak üzere farklı türde problemlerle başa çıkabilir (Zhao vd., 2019a). Literatürdeki bazı sezgisel üstü optimizasyon algoritmalarına örnekler Tablo 1'de gösterilmiştir. Sezgisel-üstü algoritmalar veri madenciliği, zamanlama ve çizelgeleme, örüntü tanıma, endüstri, mühendislik ve ekonomi gibi birçok alanda uygulanmışlardır (Jaddi vd., 2017; Ekinci ve Hekimoglu, 2017).

Tablo 1. Doğadan ilham alan optimizasyon algoritmaları hakkında kısa literatür taraması

Algoritma	**Esin Kaynağı**
Genetik algoritma (GA) (Holland, 1975)	Evrim
Benzetilmiş tavlama (SA) (Kirkpatrick vd., 1983)	Metalürjideki tavlama süreci
Parçacık sürü optimizasyonu (PSO) (Eberhart ve Kennedy, 1995)	Kuş sürülerinin sosyal davranış zekâsı
Yapay arı kolonisi (ABC) (Basturk ve Karaboga, 2006)	Bal arıları
Ateşböceği algoritması (FA) (Yang, 2009)	Ateşböceklerinin sosyal davranışları
Yerçekimsel arama algoritması (GSA) (Rashedi vd., 2009)	Yerçekimi kanunu ve kütle etkileşimleri
Yarasa algoritması (BA) (Yang, 2010)	Yarasaların yankı ile yer tayin davranışları
Çiçek tozlaşma algoritması (FPA) (Yang, 2012)	Çiçekli türlerin döllenme için tozlaşma süreci
Krill sürüsü (KH) (Gandomi ve Alavi, 2012)	Krill bireylerin doğadaki sürü davranışları
Yusufçuk algoritması (DA) (Mirjalili, 2016a)	Yusufçukların statik ve dinamik sürü davranışları
Karınca aslanı optimizasyonu (ALO) (Mirjalili, 2015)	Karınca aslanlarının doğadaki avlanma mekanizması
Virüs kolonisi arama (VCS) (Li vd., 2016)	Virüs enfeksiyonu ve yayılma stratejileri
Balina optimizasyon algoritması (WOA) (Mirjalili ve Lewis, 2016)	Kambur balinaların sosyal davranışları
Sinüs kosinüs algoritması (SCA) (Mirjalili, 2016b)	Matematiksel fonksiyonlara dayalı model
Çekirge optimizasyon algoritması (GOA) (Saremi vd., 2017)	Çekirgelerin sürü davranışları
Böbrek-ilhamlı algoritma (KA) (Jaddi vd., 2017)	İnsan vücudundaki böbreklerin çalışma süreci
Sincap arama algoritması (SSA) (Jain vd., 2019)	Sincapların dinamik besin arama davranışı
Atom arama optimizasyonu (ASO) (Zhao vd., 2019a)	Temel moleküler dinamikler

Tablo 1'de verilen PSO ve ABC, sezgisel-üstü algoritmalar içerisinde üzerinde en çok çalışılan tekniklerdir. Bununla birlikte SCA, yeni bir algoritma olmasına rağmen basit ve karmaşık optimizasyon problemlerinde en çok tercih edilen tekniktir. ASO ise en güncel ve etkili algoritmalardan biridir ancak mühendislik ve diğer bilim dallarındaki uygulamaları neredeyse yoktur. Bu çalışmada, PSO, ABC, SCA ve ASO algoritmalarının performansları bazı test fonksiyonları ele alınarak uygun şekilde karşılaştırılacaktır ve başarım oranları belirlenecektir.

PSO, ABC, SCA ve ASO Algoritmalarına Genel Bakış

Parçacık Sürüsü Optimizasyonu (PSO)

Eberhart ve Kennedy (1995) tarafından geliştirilen PSO, sürü davranışı gösteren kuş ve balık sürüleri dikkate alınarak oluşturulan stokastik bir optimizasyon metodudur. Bu metot birkaç yönden genetik algoritmaya (GA) benzemesine rağmen herhangi bir evrim operatörü yoktur. Popülasyon, sürü olarak adlandırılır ve parçacıklardan (bireylerden) oluşur. Parçacıklar optimize edilen fonksiyonun *D* boyutlu uzayında hareket ederler. Konum ve hız vektörleri sırasıyla $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ ve $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ olarak ifade edilir (Eberhart ve Kennedy, 1995).

Her bir parçacık kendi en iyi konumuna (*pbest*) ve tüm kümenin en iyi konumuna (*gbest*) bağlı olarak konumunu ve hızını günceller. Her bir adımda en iyi iki değerini bulduktan sonra aşağıdaki eşitliğe göre kendisini günceller (Eberhart ve Kennedy, 1995; Ekinci, 2016).

$$v_{id}^{t+1} = w * {v_{id}}^{t} + c_1 * r_1 * (pbest_{id}^{t} - x_{id}^{t}) + c_2 * r_2 * (gbest_{d}^{t} - x_{id}^{t}) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

$i = 1,2, \dots, NP$ olup *NP* sürünün büyüklüğünü, w her bir iterasyon için atalet ağırlığının doğrusal azalışını, c_1 ve c_2 sırasıyla bilişsel ve sosyal bileşenlerin göreceli etkisini belirleyen öğrenme faktörleridir. $r_1$ve r_2, [0,1] aralığındaki herhangi rasgele sayılardır. x_{id}^{t} , ${v_{id}}^{t}$, $pbest_{id}^{t}$ sırasıyla *i*. parçacığın *t*. iterasyonu için *d*. boyuttaki konum, hız ve kişisel en iyisidir. Aynı koşullarda tüm sürünün en iyi parçacığı ise $gbest_{d}^{t}$'dir. Bu çalışmada, c_1 ve c_2 öğrenme faktörleri 2.0 olarak alınmıştır.

Yapay Arı Kolonisi (ABC) Algoritması

Basturk ve Karaboga (2006) tarafından bal arılarının besin arama davranışlarından esinlenerek geliştirilen ABC algoritması, iki grup arıdan oluşur. Koloninin ilk yarısı yapay işçi arılardan ikinci yarısı ise gözcü arılardan oluşur. Besin kaynağı miktarınca işçi arı olur. Gözcü arılar arama bölgesinde işçi arıların besin kaynağı ile ilgili bilgiyi paylaşırlar. Birkaç döngü sonucunda besin kaynağı iyileştirilmezse kâşif arılar yeni besin kaynakları için rastgele araştırmalar yaparlar. Bir besin kaynağının konumu, optimizasyon probleminde olası bir çözümü temsil eder. Kaynaktaki besin miktarı ise ilgili çözümün uygunluğunu gösterir ve aşağıdaki eşitlik ile ifade edilir (Basturk ve Karaboga, 2006; Ekinci ve Demiroren, 2016).

$$fit_i = \begin{cases} \frac{1}{1+f_i} & eğer\ f_i \geq 0 \\ 1 + abs(f_i) & eğer\ f_i < 0 \end{cases} \quad (3)$$

Bir gözcü arı, işçi arının tüm olası çözümleri arasından en uygun değeri seçer ve bu da aşağıdaki gibi ifade edilir (Basturk ve Karaboga, 2006).

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (4)$$

Son eşitlikte *SN* işçi arı sayısına eşit olan besin kaynağı sayısını vermektedir. ABC hafızasındaki eski yiyecek kaynaklarından birini düzenlemek için aşağıdaki eşitlikten yararlanır (Basturk ve Karaboga, 2006; Ekinci, 2016).

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (5)$$

Eşitlik (5)'te $k \in \{1,2, \dots, SN\}$ ve $j \in \{1,2, \dots, D\}$, x_{kj} , x_{ij}'den farklı seçilen rasgele bir çözümü ifade ederken v_{ij} yeni çözümü, ϕ_{ij}, [-1,1] aralığında rasgele bir sayıyı ve *D* ise

optimizasyon parametresinin sayısını göstermektedir.

ABC algoritmasında bir besin kaynağı daha önceki iterasyonlardan daha fazla gelişme kaydetmiyorsa o besin kaynağının terk edilmesi gerekir. ABC algoritmasında önceki iterasyon sayısı önemli bir kontrol parametresidir. Kaşif arılar, aşağıdaki eşitlik vasıtasıyla işçi arıların terk ettiği besin kaynaklarının yerine yenilerini üretmek için yönlendirilirler (Basturk ve Karaboga, 2006).

$$x_{ij} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}) \quad (6)$$

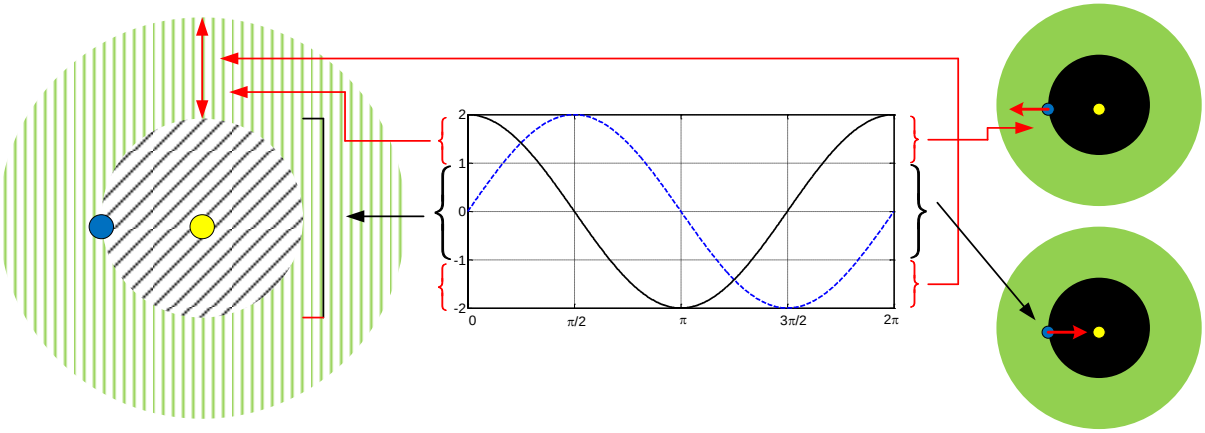
Son eşitlikte $x_{max,j}$ ve $x_{min,j}$ sırasıyla j. optimizasyon değişkenlerinin üst ve alt sınırlarıdır. $rand(0,1)$ ise [0,1] aralığındaki rasgele dağılmış bir sayıyı ifade etmektedir.

Sinüs Kosinüs Algoritması (SCA)

SCA, Mirjalili (2016b) tarafından geliştirilen trigonometrik fonksiyonlara dayalı sezgisel-üstü bir algoritmadır. SCA, başlangıçta çoklu rasgele çözümler üretir sonra en iyi çözüme doğru ya da onun dışına doğru salınım oluşturur. Ayrıca, araştırma uzayının keşfi ve sömürülmesi için birkaç rasgele ve uyarlanabilir değişkeni algoritmaya eklemler. Keşif ve sömürme stokastik temelli optimizasyon sürecinin en çok kullanılan iki aşamasıdır ve SCA içinde aşağıdaki denklem geliştirilmiştir (Mirjalili, 2016b; Hekimoglu, 2019a).

$$X_i^{t+1} = \begin{cases} X_i^t + r_1.\sin(r_2).|r_3 P_i^t - X_i^t|, & r_4 < 0{,}5 \\ X_i^t + r_1.\cos(r_2).|r_3 P_i^t - X_i^t|, & r_4 \geq 0{,}5 \end{cases} \quad (7)$$

Eşitlik (7)'de t. iterasyondaki i. boyutta X_i^t güncel çözüm pozisyonunu, P_i^t hedef noktasının boyutunu ve $|.|$ mutlak değeri ifade eder. [–2,2] değer aralığına sahip olan r_1, bir sonraki konum bölgesini tanımlar. [0,2π] değer aralığına sahip olan r_2 *ise* hareketin hedef noktasına veya onun uzağına ne kadar mesafede olduğunu hesaplar. r_3 hedef için rasgele bir ağırlığı temsil eder ve $r_3 > 1$ ise hedef noktasının mesafe belirlemede değerini stokastik olarak arttırırken, $r_3 < 1$ ise azaltır. r_4, [0,1] değer aralığına sahiptir ve sinüsten kosinüse ya da kosinüsten sinüse fonksiyonlar arası geçişi sağlar (Mirjalili, 2016b; Ekinci, 2019). Sinüs ve kosinüs fonksiyonlarının Eşitlik (7)'deki sonraki konuma etkileri Şekil 1'de gösterilmektedir.



Şekil 1. Sinüs ve kosinüs fonksiyonlarının sonraki konuma etkileri (Mirjalili, 2016b)

Atom Arama Optimizasyonu (ASO)

ASO, moleküler dinamikten esinlenen popülasyon tabanlı sezgisel-üstü bir algoritmadır (Zhao vd., 2019a). ASO'daki her bir atomun arama uzayındaki konumu, kendisinden daha ağır ya da daha hafif bir kütleye işaret eden bir çözüme sahiptir. Popülasyondaki tüm atomlar ağır olanların hafif olanları kendine çekmesi gibi birbirini ya çekecek ya da itecektir. Daha ağır atomların hız değişimi daha yavaş olduğundan yerel uzayda daha iyi çözümler üretmelerine neden olurken

hafif atomlar daha hızlı olduğundan yeni ümit verici alanları bulmak için tüm arama uzayında tarama yaparlar. (Zhao vd., 2019b)

ASO'da her bir atom en iyi atom ile bir kovalent bağa sahiptir. Yani her bir atom en iyi atoma doğru hareket etmeye zorlanır ve *i.* atomun kısıdı ise aşağıdaki eşitlikte ifade edilmiştir (Zhao vd., 2019b).

$$\theta_i(t) = \left[|x_i(t) - x_{best}(t)|^2 - b_{i,best}^2\right] \quad (8)$$

burada $x_{best}(t)$ en iyi atomun *t.* iterasyondaki konumu ve $b_{i,best}^2$ en iyi atom ile *i.* atom arasındaki sabit bağ uzunluğudur. Dolayısıyla kısıtlama kuvveti ise aşağıdaki eşitlikteki gibidir (Zhao vd., 2019b).

$$G_i^d(t) = \lambda(t)(x_{best}^d(t) - x_i^d(t)) \quad (9)$$

burada $\lambda(t) = \beta e^{-\frac{20t}{T}}$ Langragian çarpanıdır ve β ise çarpan ağırlığıdır. Bu çalışmada β 0.2 olarak alınmıştır. *i.* atomun *t.* iterasyondaki kütlesi aşağıdaki şekilde hesaplanır (Zhao vd., 2019b).

$$M_i(t) = e^{-\frac{Fit_i(t) - Fit_{best}(t)}{Fit_{worst}(t) - Fit_{best}(t)}} \quad (10)$$

$$m_i(t) = \frac{M_i(t)}{\sum_{j=1}^{N} M_j(t)} \quad (11)$$

burada $Fit_{best}(t)$ ve $Fit_{worst}(t)$ sırasıyla *t.* iterasyondaki minimum ve maksimum uygunluk değerine sahip atomlardır. $Fit_i(t)$ ise *i.* atomun *t.* iterasyondaki uygunluk değer fonksiyonudur.

Algoritmayı basitleştirmek için *i.* atomun (t+1). iterasyondaki konum ve hız vektörleri de aşağıdaki gibi ifade edilir (Zhao vd., 2019a).

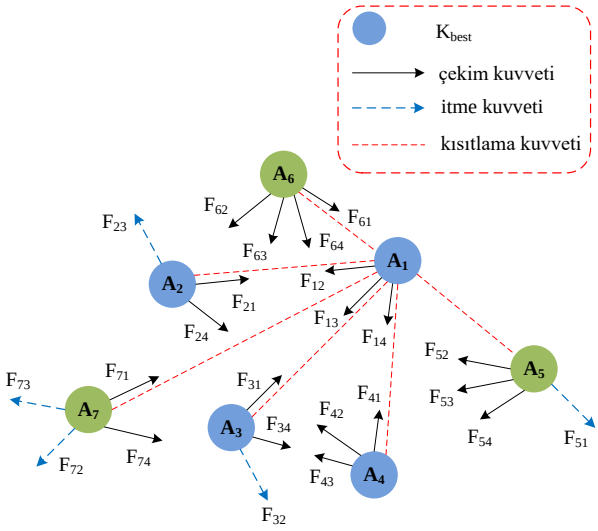
$$v_i^d(t+1) = rand_i^d v_i^d(t) + a_i^d(t) \quad (12)$$

$$x_i^d(t+1) = x_i^d + v_i^d(t+1) \quad (13)$$

ASO algoritmasında, keşiflerin ilk aşamasında keşfi geliştirmek için, her bir atomun *K* komşuları kadar daha iyi uygunluk değerlerine sahip birçok atomla etkileşime girmesi gerektiği gibi iterasyonların son aşamasında da sömürüyü arttırmak için, atomların *K* komşuları kadar daha iyi uygunluk değerlerine sahip birkaç atomla etkileşime girmeleri gerekir. Böylece, bir zaman fonksiyonu olarak *K*, iterasyonlar arttıkça yavaş yavaş azalır ve *K* aşağıdaki şekilde hesaplanır (Zhao vd., 2019b).

$$K(t) = N - (N-2) * \sqrt{\frac{t}{T}} \quad (14)$$

burada *N* sistemdeki toplam atom sayısı, *t* mevcut iterasyon sayısı ve *T* ise maksimum iterasyon sayısıdır. Bir atom popülasyonunun kuvvetleri, Şekil 2'de gösterilmiştir. Burada, en iyi uygunluk değerine sahip ilk 5 atom K_{best} olarak kabul edilir. ASO'nun daha detaylı açıklamaları için Zhao vd. (2019a) ve Hekimoglu (2019b)'ye bakılabilir.



Şekil 2. K=5 için K_{best} içeren bir atom sisteminin kuvvetleri (Zhao vd., 2019a)

Test Fonksiyonları

İlgili dört sezgisel-üstü algoritmanın performanslarını test etmek için üç adet tek modlu (Sphere, Rosenbrock ve Step) ve beş adet çok modlu (Schwefel, Rastrigin, Ackley, Griewank ve Egg Crate) olmak üzere sekiz farklı test fonksiyonu ele alınmıştır. Bu test fonksiyonları Tablo 2'de özetlenmiştir. Bu fonksiyonların detaylı bilgileri Rashedi vd. (2009)'da verilmiştir.

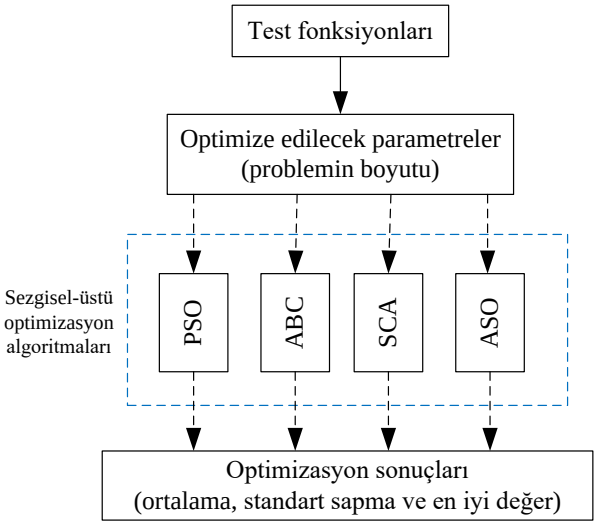
Tablo 2. Tek modlu ve çok modlu bazı test fonksiyonları

Fonksiyon	Türü	Fonksiyonu tanımı	Boyut (d)	Değer aralığı	Optimum değeri
Sphere	tek modlu	$f_1(x) = \sum_{i=1}^{d} x_i^2$	30	[−100, 100]	0
Rosenbrock	tek modlu	$f_2(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	20	[−30, 30]	0
Step	tek modlu	$f_3(x) = \sum_{i=1}^{d} (x_i + 0.5)^2$	30	[−100, 100]	0
Schwefel	çok modlu	$f_4(x) = -\sum_{i=1}^{d} (x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	4	[−500, 500]	−1675.9315
Rastrigin	çok modlu	$f_5(x) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	10	[−5.12, 5.12]	0
Ackley	çok modlu	$f_6(x) = -20exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} (cos2\pi x_i)\right) + 20 + e$	30	[−32, 32]	0
Griewank	çok modlu	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	10	[−600, 600]	0
Egg Crate	çok modlu	$f_8(x) = x_1^2 + x_2^2 + 25[sin^2(x_1) + sin^2(x_2)]$	2	[−5, 5]	0

Karşılaştırma Sonuçları

PSO için Yarpiz Team (2019), ABC için Karaboga (2019), SCA için Mirjalili (2019) ve ASO için Zhao (2019)'deki MATLAB tabanlı kodlardan değişiklikler yapılmak suretiyle yararlanılmıştır. Bu dört algoritma en çok bilinen sekiz farklı test fonksiyonlarına uygulandı ve her bir algoritma her bir fonksiyon için 30 kez çalıştırıldı. Uygun bir kıyaslama olması bakımından her bir algoritma için popülasyon büyüklüğü 50 ve maksimum iterasyon sayısı 500 olarak seçilmiştir. Şekil 3, optimizasyon algoritmalarının test fonksiyonlarına uygulanmasını göstermektedir. Sekiz test fonksiyonu için elde edilen istatistiksel sonuçlar Tablo 3'te sunulmuştur. En iyi sonuçlar kalın yazı tipi ile vurgulanmıştır.

Her bir test fonksiyonu optimize edilirken, PSO, ABC, SCA ve ASO algoritmalarının çalıştırılmasıyla geçen süreler yaklaşık olarak sırasıyla 2.419 s, 1.128 s, 0.237 s ve 2.896 s'dir.



Şekil 3. Optimizasyon aşaması

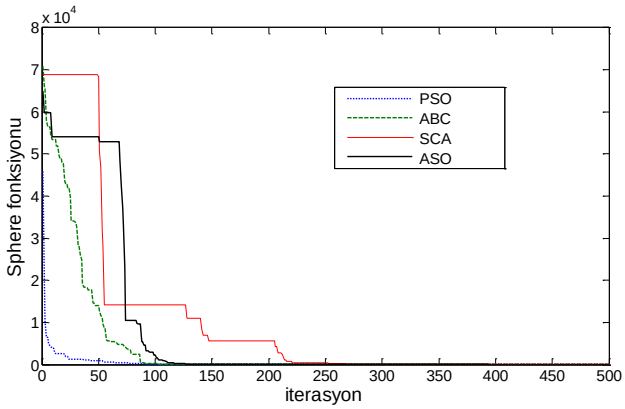
Tabloya göre, *Sphere*, *Step*, *Ackley*, *Griewank* ve *Egg Crate* test fonksiyonları için ortalama, standart sapma ve en iyi değer indeksleri açısından en iyi sonuçlar ASO algoritması ile elde edilmiştir. Benzer şekilde, *Rosenbrock* ve

Rastrigin test fonksiyonları için en iyi sonuçlar ABC algoritması ile bulunmuştur. *Schwefel* test fonksiyonu için ise ABC ve ASO algoritmaları birbirine yakın sonuçlar bulmuştur.

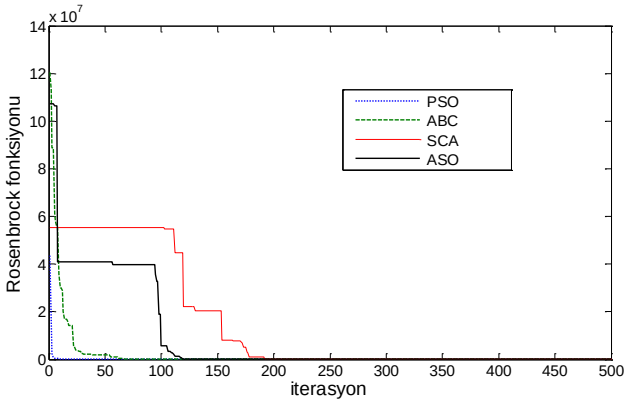
Tüm sekiz test fonksiyonu için PSO, ABC, SCA ve ASO algoritmalarının optimizasyon ilerlemeleri (fonksiyon değerlerinin iterasyon sayılarına göre değişimleri) Şekil 4–11'de gösterilmiştir. Bu şekiller, 30 kez çalıştırılma içerisinden seçilen rastgele bir çalıştırılmadan elde edilmiştir. Tablo 3'teki istatistiksel sonuçlara ve Şekil 4–11'deki yakınsama eğrilerine bakıldığında, ASO algoritmasının yeni ve etkili bir teknik olan SCA algoritmasına ve diğer iki algoritmaya (PSO ve ABC) nazaran daha üstün performansa sahip olduğu anlaşılmaktadır. Bununla birlikte, kaotik haritalar yaklaşımı veya karşıt tabanlı öğrenme stratejisi uygulanarak ASO algoritmasının verimliliği ve etkinliği daha da artırılabilir.

Tablo 3. Çeşitli test fonksiyonları için sonuçların karşılaştırılması

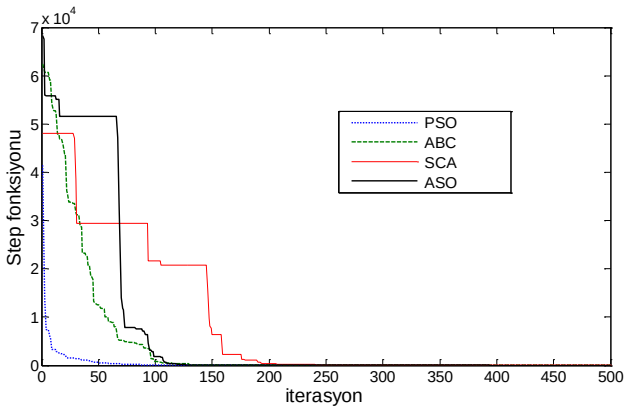
Fonksiyon Adı	**İndeks**	**Sezgisel-üstü optimizasyon teknikleri**			
		PSO	**ABC**	**SCA**	**ASO**
Sphere	Ortalama	4.16E-09	2.88E-05	0.259636	**4.55E-16**
	Standart sapma	8.68E-09	1.74E-05	0.283591	**1.75E-15**
	En iyi değer	8.30E-12	1.08E-05	0.004759	**8.00E-20**
Rosenbrock	Ortalama	25.65893	**7.458535**	17.49040	15.54956
	Standart sapma	21.80477	4.648221	**0.283592**	7.548646
	En iyi değer	6.323248	**0.578422**	17.60098	10.20560
Step	Ortalama	4.83E-08	0.003640	5.428303	**1.12E-18**
	Standart sapma	2.71E-07	0.016399	1.288786	**2.20E-18**
	En iyi değer	9.50E-13	6.69E-06	3.666500	**2.87E-20**
Schwefel	Ortalama	-1484.964	**-1675.9315**	-1440.078	-1615.626
	Standart sapma	88.42275	**0**	97.58904	68.76877
	En iyi değer	**-1675.9315**	**-1675.9315**	-1644.343	**-1675.9315**
Rastrigin	Ortalama	6.400986	**6.46E-15**	0.000724	5.107576
	Standart sapma	2.702742	**7.42E-15**	0.003799	2.857608
	En iyi değer	1.989976	**0**	**0**	0.995053
Ackley	Ortalama	0.000274	0.101194	7.171143	**5.58E-10**
	Standart sapma	0.001066	0.094629	9.253197	**3.25E-10**
	En iyi değer	2.88E-07	0.009319	0.018818	**5.34E-11**
Griewank	Ortalama	0.091754	0.008473	0.061780	**0**
	Standart sapma	0.042514	0.007385	0.097008	**0**
	En iyi değer	0.009901	1.86E-10	6.27E-15	**0**
Egg Crate	Ortalama	2.12E-212	3.08E-19	3.53E-77	**0**
	Standart sapma	6.38E-213	5.59E-19	5.86E-78	**0**
	En iyi değer	1.42E-215	4.54E-20	3.24E-81	**0**



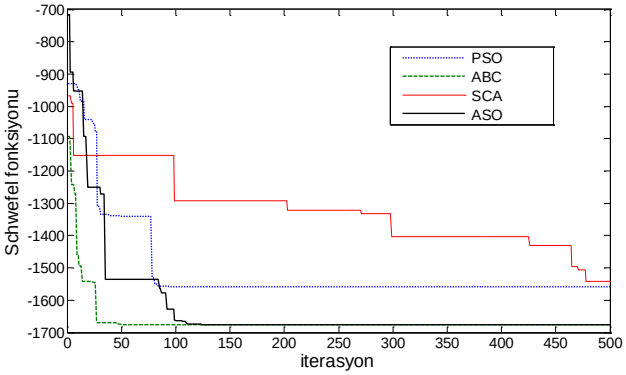
Şekil 4. Farklı algoritmalar için Sphere fonksiyonunun optimizasyon süreci



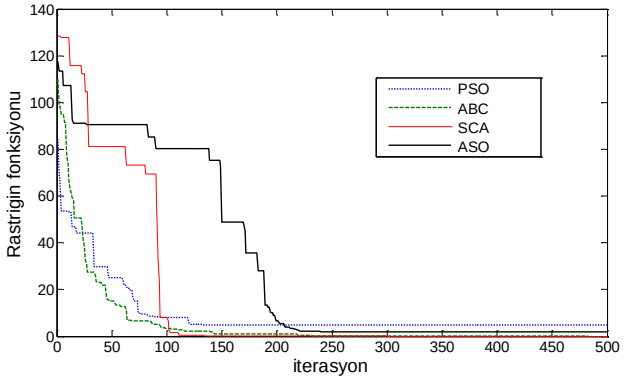
Şekil 5. Farklı algoritmalar için Rosenbrock fonksiyonunun optimizasyon süreci



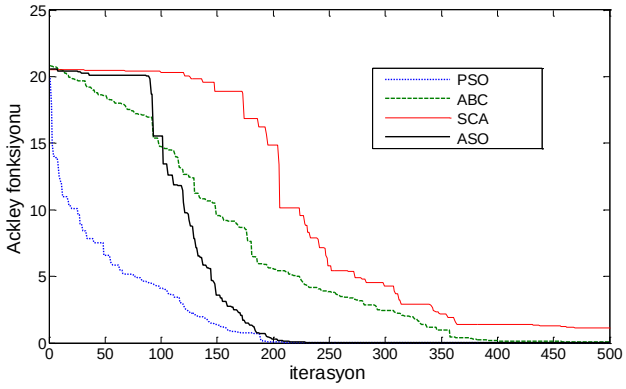
Şekil 6. Farklı algoritmalar için Step fonksiyonunun optimizasyon süreci



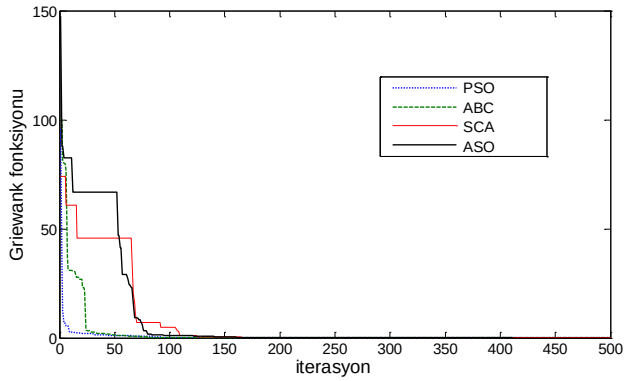
Şekil 7. Farklı algoritmalar için Schwefel fonksiyonunun optimizasyon süreci



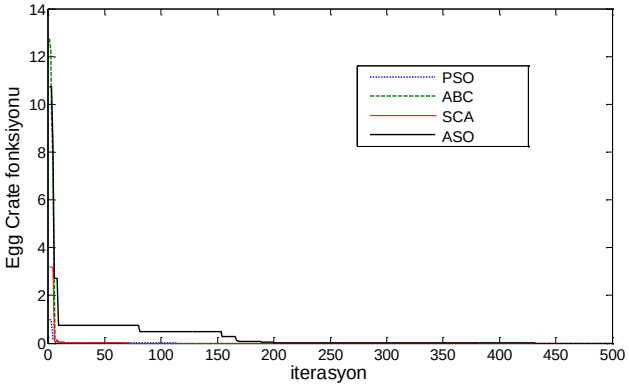
Şekil 8. Farklı algoritmalar için Rastrigin fonksiyonunun optimizasyon süreci



Şekil 9. Farklı algoritmalar için Ackley fonksiyonunun optimizasyon süreci



Şekil 10. Farklı algoritmalar için Griewank fonksiyonunun optimizasyon süreci



Şekil 11. Farklı algoritmalar için Egg Crate fonksiyonunun optimizasyon süreci

Sonuçlar

Esinlenme kaynakları farklı olan sezgisel-üstü eniyileştirme teknikleri, çok sayıdaki optimizasyon problemlerine başarıyla uygulanmıştır. Kişisel bir bilgisayar üzerinden uygun koşullar altında hangisinin en iyi performansa sahip olduğu kestirilebilir. Bu çalışmada en çok bilinen ve atıf alan PSO ve ABC ile güncel algoritmalar sınıfına giren SCA ve ASO algoritmalarının karşılaştırmalı analizi yapıldı. Karşılaştırma amacı için üçü tek modlu ve beşi çok modlu olan toplam sekiz adet kısıtlı kıyaslama fonksiyonu test için kullanıldı. Elde edilen istatistiksel değerlerde (ortalama, standart sapma ve en iyi değer) ve yakınsama eğrilerinde çoğunlukla ASO algoritmasının en iyi sonuçlar verdiği anlaşılmıştır. Gelecekteki çalışmalarda ise ASO algoritmasının performansını keşif ve sömürü açısından geliştirmek için kaotik haritalama önerilip yapay sinir ağlarının optimizasyonu yapılacaktır.

Kaynaklar

Basturk, B., Karaboga, D., (2006). An artificial bee colony (ABC) algorithm for numeric function optimization, IEEE Swarm Intelligence Symposium, Indianapolis.

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

Eberhart, R., Kennedy, J., (1995). A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, Nagoya.

Ekinci, S., (2016). Application and comparative performance analysis of PSO and ABC algorithms for optimal design of multi-machine power system stabilizers, *Gazi University Journal of Science*, *29*(2), 323-334.

Ekinci, S., Demiroren, A., (2016). Modeling, simulation, and optimal design of power system stabilizers using ABC algorithm, *Turkish Journal of Electrical Engineering & Computer Sciences*, *24*(3), 1532-1546.

Ekinci, S., Hekimoglu, B., (2017). Multi-machine power system stabilizer design via HPA algorithm, *Journal of the Faculty of Engineering and Architecture of Gazi University*, *32*(4), 1271-1285.

Ekinci S., (2019). Optimal design of power system stabilizer using sine cosine algorithm, *Journal of the Faculty of Engineering and Architecture of Gazi University*, https://dx.doi.org/10.17341/gazimmfd.460529.

Ekinci, S., Hekimoğlu, B., (2019). Improved Kidney-Inspired Algorithm Approach for Tuning of PID Controller in AVR System. *IEEE Access*, 7, 39935-39947.

Gandomi, A.H., Alavi, A.H., (2012). Krill herd: a new bio-inspired optimization algorithm, *Communications in Nonlinear Science and Numerical Simulation*, *17*(12), 4831-4845.

Hekimoğlu, B., (2019a). Sine-cosine algorithm-based optimization for automatic voltage regulator system. *Transactions of the Institute of Measurement and Control*, 41(6), 1761-1771.

Hekimoğlu, B., (2019b). Optimal Tuning of Fractional Order PID Controller for DC Motor Speed Control via Chaotic Atom Search Optimization Algorithm. *IEEE Access*, 7, 38100-38114.

Holland, J.H., (1975). *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor.

Jaddi, N.S., Alvankarian, J., Abdullah, S., (2017). Kidney-inspired algorithm for optimization problems, *Communications in Nonlinear Science and Numerical Simulation*, *42*, 358-369.

Jain, M., Singh, V., Rani, A., (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm, *Swarm and Evolutionary Computation*, *44*, 148-175.

Karaboga, D., Artificial Bee Colony (ABC) Algorithm, https://abc.erciyes.edu.tr, Erişim tarihi Şubat 16, 2019.

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

Li, M.D., Zhao, H., Weng, X.W., Han, T., (2016). A novel nature-inspired algorithm for optimization: Virus colony search, *Advances in Engineering Software*, *92*, 65-88.

Mirjalili, S., (2015). The ant lion optimizer, *Advances in Engineering Software*, *83*, 80-98.

Mirjalili, S., (2016a). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Computing and Applications*, *27*(4), 1053-1073.

Mirjalili, S., (2016b). SCA: a sine cosine algorithm for solving optimization problems, *Knowledge-Based Systems*, *96*, 120-133.

Mirjalili, S., Lewis, A., (2016). The whale optimization algorithm, *Advances in Engineering Software*, *95*, 51-67.

Mirjalili, S., Sine Cosine Algorithm (SCA), http://www.alimirjalili.com/SCA.html, Erişim tarihi Şubat 18, 2019.

Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., (2009). GSA: a gravitational search algorithm, *Information Sciences*, 179(13), 2232-2248.

Saremi, S., Mirjalili, S., Lewis, A., (2017). Grasshopper optimisation algorithm: theory and application, *Advances in Engineering Software*, *105*, 30-47.

Yang, X.S., (2009). Firefly algorithms for multimodal optimization, International Symposium on Stochastic Algorithms, Springer, Heidelberg.

Yang, X.S., (2010). A new metaheuristic bat-inspired algorithm, Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, Heidelberg.

Yang, X.S., (2012). Flower pollination algorithm for global optimization, International Conference on Unconventional Computing and Natural Computation, Springer, Heidelberg.

Yarpiz Team, Particle Swarm Optimization (PSO), https://www.mathworks.com/matlabcentral/fileexchange/52857-particle-swarm-optimization-pso , Erişim tarihi Şubat 17, 2019.

Zhao, W., Atom Search Optimization (ASO) Algorithm, https://www.mathworks.com/matlabcentral/fileexchange/67011-atom-search-optimization-aso-algorithm, Erişim tarihi Şubat 21, 2019.

Zhao, W., Wang, L., Zhang, Z., (2019a). Atom search optimization and its application to solve a hydrogeologic parameter estimation problem, *Knowledge-Based Systems*, *163*, 283-304.

Zhao, W., Wang, L., Zhang, Z., (2019b). A novel atom search optimization for dispersion coefficient estimation in groundwater, *Future Generation Computer Systems*, *91*, 601-610.

Atom search optimization algorithm for solving constrained optimization problems

Extended abstract

Optimization refers to the process of finding optimal parameters among all possible values of system parameters to optimize the output of a given system. Optimization problems can be found in all areas of research, which in turn necessitates the development of optimization techniques and becomes the focus of interest for researchers in this field. Due to the drawbacks of traditional optimization paradigms, such as the local optimum stagnation and the need to determine the search space, the interest in meta-heuristic optimization approaches over the last two decades has increased.

Atom search optimization (ASO) is a novel population-based optimization algorithm based on basic molecular dynamics. The position of each atom in the ASO in the search space has a solution that points to a heavier or lighter mass than itself. All the atoms in the population will attract or push each other like heavy ones to attract the light ones. Since heavier atoms are slower, they produce better solutions in the search space, while light atoms are faster, so they search across the entire search space to find new promising areas. ASO can be easily applied to optimization problems thanks to its simplicity and few control parameters.

In this study, ASO was applied to eight test functions (Sphere, Rosenbrock, Step, Schwefel, Rastrigin, Ackley, Griewank and Egg Crate). In addition, the statistical results (mean, standard deviation and best value) obtained with ASO for each test function was compared with the results obtained by other algorithms in the literature. Particle swarm optimization (PSO), artificial bee colony (ABC) and sinus cosine algorithm (SCA) are the methods chosen for comparison.

PSO is a stochastic optimization method based on herd of flocks and birds. Although this method is similar to genetic algorithm (GA) in several respects, there is no evolution operator. The population is called the swarm and consists of particles (individuals). The particles move in the D-dimensional space of the optimized function.

The artificial bee colony (ABC) algorithm, which is inspired by the honey bees' foraging behavior, consists of two groups of bees. The first half of the colony consists of artificial worker bees and the second half consists of observer bees. The amount of food source becomes the number of worker bees. Observer bees share information about the food source of worker bees in the search area. If the food source is not improved after several cycles, exploratory bees do random investigations for new food sources.

Sine cosine algorithm (SCA) is a meta-heuristic algorithm based on mathematical functions. The SCA generates multiple random solutions at first, and then generates the oscillation towards or out of the best solution. In addition, several random and adaptive variables are integrated into the algorithm for the exploration and exploitation of research space. Exploration and exploitation are the two most commonly used stages of stochastic-based optimization.

For the Sphere, Step, Ackley, Griewank and Egg Crate test functions, the best results of the performance criteria such as mean, standard deviation and best value were obtained by the ASO algorithm. Similarly, the best results for Rosenbrock and Rastrigin test functions were found by ABC algorithm. For the Schwefel test function, ABC and ASO algorithms found similar results. From the obtained statistical values and convergence curves, it was found that mostly the ASO algorithm gave the best results.

However, the efficiency and effectiveness of the ASO algorithm can be further enhanced by applying a chaotic maps approach or an opposition-based learning strategy. In future studies, chaotic mapping will be proposed to improve the performance of the ASO algorithm in terms of exploration and exploitation and optimization of the artificial neural networks will be done.

Keywords: *Atom search optimization, meta-heuristic, optimization, benchmark functions*