OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE SİNÜS KOSİNÜS ALGORİTMASI (SKA) YÖNTEMİNİN KULLANILMASI

Gökhan DEMİR

Yüksek Lisans Tezi
Yazılım Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Erkan TANYILDIZI
TEMMUZ-2017

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE SİNÜS KOSİNÜS ALGORİTMASI (SKA) YÖNTEMİNİN KULLANILMASI

YÜKSEK LİSANS TEZİ

Gökhan DEMİR

(151137103)

Tezin Enstitüye Verildiği Tarih : 25.07.2017
Tezin Savunulduğu Tarih : 25.07.2017

Tez Danışmanı: Doç. Dr. Erkan TANYILDIZI (F.Ü.)

Diğer Jüri Üyeleri: Prof. Dr. İbrahim TÜRKOĞLU (F.Ü.)

Doç. Dr. Cihan VAROL (S.H.S.U.)

TEMMUZ-2017

ÖNSÖZ

Yüksek lisans tez çalışmalarım boyunca bilgi ve tecrübe anlamında değerli katkılarıyla beni yönlendiren, ilgi ve hoşgörüsünü esirgemeyen ve her türlü olanağı sağlayan Doç. Dr. Erkan TANYILDIZI'na ve Doç. Dr. Bilal ALATAŞ'a teşekkür ederim.

Gökhan DEMİR

Elazığ, 2017

İÇİNDEKİLER

Sayfa No

ÖZET

Yüksek Lisans Tezi

OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE SİNÜS KOSİNÜS ALGORİTMASI (SKA) YÖNTEMİNİN KULLANILMASI

Gökhan DEMİR

Fırat Üniversitesi

Fen Bilimleri Enstitüsü

Yazılım Mühendisliği Anabilim Dalı

2017, Sayfa:66

Farklı kaynaklardan ilham alınarak pek çok metasezgisel algoritma geliştirilmesine rağmen matematikten ilham alan metasezgisel algoritma sayısı oldukça azdır. Bu tez çalışmasında yakın zamanda önerilen matematik tabanlı Sinüs Kosinüs Algoritması (SKA)'nın optimizasyon problemlerinin çözümünde kullanılması incelenmektedir. SKA'nın henüz yeni bir algoritma olması ve az sayıda parametre içermesinden dolayı algoritmada iyileştirmeler yapılarak daha iyi sonuçlar elde edilebileceği düşüncesiyle Adaptif Sinüs Kosinüs Algoritması önerilmektedir.

Ayrıca bu tez çalışması kapsamında optimizasyon problemlerinin çözümü için matematik tabanlı yeni bir metasezgisel yöntem olan Altın Sinüs Algoritması (ASA) önerilmektedir. ASA bu çalışma kapsamında ilk kez önerilen normalize ve hibrit yapıdaki kaotik harita yöntemi ile iyileştirilerek Kaotik Altın Sinüs Algoritmaları da sunulmaktadır.

Anahtar Kelimeler: Sinüs Kosinüs Algoritması, Adaptif Sinüs Kosinüs Algoritması, Altın Sinüs Algoritması, Optimizasyon, Metasezgisel Algoritmalar, Yapay Zeka

SUMMARY

M.S. Thesis

THE USE OF SINE COSINE ALGORITHM (SCA) METHOD IN THE SOLUTION OF OPTIMIZATION PROBLEMS

Gökhan DEMİR

Fırat University

Institute of Science and Technology

Department of Software Engineering

2017, Pages: 66

Although many metaheuristic algorithms are inspired by different sources, the number of metaheuristic algorithms inspired by mathematics is very small. In this thesis study, the use of the recently proposed math-based Sine Cosine Algorithm (SCA) to solve optimization problems is investigated. Adaptive Sine Cosine Algorithm is employed since it is a new algorithm which includes few parameters that can improve the results obtained by the original algorithm.

In addition, Golden Sine Algorithm (Gold-SA), a new math-based metaheurisitic method for solving optimization problems, is proposed in this thesis study. Gold-SA is presented with Chaotic Golden Sine Algorithms by using the normalized and hybrid structure chaotic map method which is proposed for the first time in this work. Gold-SA is also presented with the Chaotic Golden Sine Algorithms, which are improved by the chaotic mapping method of the normalized and hybrid structure.

Keywords: Sine Cosine Algorithm, Adaptive Sine Cosine Algorithm, Golden Sine Algorithm, Optimization, Metaheuristic Algorithms, Artificial Intelligence

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR

ASA : Altın Sinüs Algoritması

ASKA : Adaptif Sinüs Kosinüs Algoritması

BOA : Balina Optimizasyon Algoritması

ERA : Emperyalist Rekabetçi Algoritma

SKA : Sinüs Kosinüs Algoritması

PSO : Parçacık Sürü Optimizasyonu

KKO : Karınca Koloni Optimizasyon Algoritması

YAA : Yerçekimi Arama Algoritması

SEMBOLLER LİSTESİ

t : Mevcut iterasyon

T : Maksimum iterasyon sayısı

c : Sabit bir sayı

X : Çözüm

P : Pozisyon

D : Hedef çözüm

x_i^t : *i*. boyutun *t*. iterasyonundaki güncel çözümü

r_1, r_2, r_3 : Rastgele sayılar

r_4 : [0, 1] aralığında rastgele bir sayı

P_i : *i*. boyuttaki hedef noktanın pozisyonu

|| : Mutlak değer

w : Tel çapı

d : Ortalama bobin çapı

L : Uzunluk (veya bobin sayısı)

T_s : Kabuk kalınlığı

T_h : Basınçlı kabın baş kalınlığı

R : İç yarıçap

L : Kabın silindirik bölümünün başlık haricindeki uzunluğu

K : Kaotik harita

i : Kaotik haritanın indeksi

a : Altın kesit arama yöntemi başlangıç aralığının minimum değeri

b : Altın kesit arama yöntemi başlangıç aralığının maksimum değeri

p : [0, π] aralığında rastgele bir sayı

1. GİRİŞ

Optimizasyon, bir problemin global optimumum çözümünü belirli koşullar altında arama sürecidir. Optimizasyon algoritmaları temel olarak deterministik ve stokastik tabanlı algoritmalar olmak üzere iki sınıfa ayrılır. Deterministik yöntemler de kendi içinde hesaplamalı yöntemler ve gradyan tabanlı yöntemler olarak sınıflandırılır. Hesaplamalı yöntemler gradyan fonksiyonu hesaplamalarına ihtiyaç duymazlar ancak oldukça yavaş ve etkisiz yöntemlerdir. Gradyan tabanlı yöntemler amaç fonksiyonunun eğimini veya türevini kullanırlar. Ancak bu yöntemler amaç fonksiyonunun düz ve pürüzsüz olmadığı durumlarda global optimum noktaya yakınsamayı garanti etmezler. Deterministik yöntemler ile yüksek boyutlu, çok modlu veya türevlenemeyen problemleri çözmek oldukça zor hatta imkansızdır [1]. Stokastik tabanlı yöntemler yeterli sayıda rastgele çözüm ve optimizasyon adımları ile global optimumu bulmayı amaçlamaktadır.

Mühendislik, ekonomi ve işletme gibi pek çok bilim dalında, klasik yöntemlerle makul bir sürede veya kesin olarak çözülemeyen karmaşık optimizasyon problemleri ortaya çıkmıştır. Doğa, bu gibi karmaşık optimizasyon problemlerini çözmede kullanılabilecek yapay hesaplama yöntemleri tasarlamak için mekanizmalar, ilkeler ve kavramlar gibi kaynaklar sağlar [2]. Son yıllarda araştırmacılar, problemlerin çözümü için doğanın belirli olgu veya davranışlarını taklit eden doğadan esinlenmiş birçok metasezgisel algoritma geliştirmişlerdir.

Metasezgisel yöntemler gradyan bilgisine ihtiyaç duymazlar ve amaç fonksiyonu defalarca çağırarak global optimumu ararlar. Bu algoritmalar arama uzayını daraltırlar ve etkili bir şekilde çözüm bulmaya çalışırlar. Geliştirilen algoritmalar arasındaki temel fark, keşif (global arama yeteneği) ile sömürü arasındaki denge (optimal çözüm çevresinde yerel arama yeteneği) yaklaşımında yatmaktadır [1].

Metasezgiseller bir problemi çözmek için sezgiseller veya yöntemler kullanarak eksik veya sınırlı bilgilerle kabul edilebilir bir süre içinde yeterince iyi bir çözüm sağlayan algoritma sınıfıdır. Bu algoritmalar genellikle stokastik optimizasyon teknikleri kullanarak bir çözüm üretmek için yinelemeli yöntemler ve sezgiseller kullanır. Metasezgisel algoritmaların bir diğer avantajı ise, probleme bağımlı olmamasıdır. Hemen hemen tüm çeşit problemleri çözmek için kullanılabilen genel amaçlı yöntemlerdir. Buna karşılık sonuçların

doğruluğundan minimum hata değeri kadar taviz verilir [3]. Metasezgisel algoritmalar doğa, biyoloji, fizik, kimya, insan, spor, matematik, vb. gibi kavramlardan ilham alır.

Metasezgisel algoritmalar son yıllarda gerçek mühendislik problemlerinin optimum çözümünün bulunması için oldukça sık kullanılmaktadır. Genel amaçlı metasezgisel yöntemlerin yaygın olarak kullanılmasında basitlik, esneklik, türetilebilme ve lokal optimumdan kaçınma olarak 4 ana neden gösterilebilir [4]. Bu yöntemler ticaret, sanayi, mühendislik, vb. tüm sektörlerde düzenli olarak kullanılmaktadır [5].

Doğanın zengin kaynağı araştırmacılara pek çok açıdan ilham kaynağı olmuştur. Günümüzde yeni algoritmaların çoğu doğadan ilham alınarak geliştirilmektedir. Doğadan ilham alan algoritmalar biyolojik sistemin bazı başarılı özelliklerine dayanmaktadır. Dolayısıyla doğadan ilham alan algoritmaların en büyük kısmı biyolojiden ilham alınarak geliştirilmiştir [6]. Biyoloji tabanlı algoritmalar sürü davranışlarını doğrudan kullanmazlar. Genetik Algoritma [7] doğal seçilim sürecinden ilham alır. Diferansiyel Gelişim Algoritması [8] işleyiş ve operatörleri itibariyle genetik algoritmaya dayanır. Biyocoğrafya Tabanlı Optimizasyon [9] Algoritması, biyolojik canlıların coğrafik dağılımı olan biyocoğrafyaya dayanır. Beyin Fırtınası Optimizasyonu [10], insanların yaratıcı bir şekilde problem çözme sürecini taklit eder. Yunus Ekolokasyon Optimizasyon Algoritması [11], yunusların çeşitli çevrelerde navigasyon ve avlanma için kullandıkları biyolojik bir sonar olan ekolokasyon yeteneklerinden ilham alır. Eko-İlhamlı Evrimsel Algoritma [12], habitatlar, ekolojik ilişkiler ve ekolojik süksesyon gibi ekolojik kavramlara dayanır. Çiçek Tozlaşma Algoritması [13], çiçeklerin tozlaşma sürecini modeller. İstilacı Yabani Ot Optimizasyon [14] Algoritması, istilacı yabani otlardan ilham alır. Termit Koloni Optimizasyon Algoritması [15] termitlerin akıllı davranışlarından esinlenen popülasyon tabanlı bir optimizasyon tekniğidir. Atmosfer Bulutları Modeli Optimizasyon Algoritması [16] bulutların hareketini taklit eder.

Biyolojiden ilham alan algoritmalar arasında, sürü zekasından ilham alınarak özel bir algoritma sınıfı geliştirilmiştir. Bu nedenle biyolojiden ilham alan algoritmalardan bazıları sürü zekası tabanlıdır. Sürü zekası, bazı kuralları takip eden birden fazla ajanın ortaklaşa hareket etmesi sonucu ortaya çıkan davranışlar ile ilgilenir. Parçacık Sürü Optimizasyon Algoritması [17], kuş ve balık sürülerinin hareketlerinden ilham alır. Karınca Koloni Optimizasyon Algoritması [18] karıncaların yiyecek arama için en kısa yolu bulma davranışını taklit eder. Bakteriyel Yem Arama Optimizasyon Algoritması [19], Escherichia

Coli'nin sosyal beslenme davranışına dayanır. Yarasa Algoritması [20], yarasaların avını bulabilmesini ve en karanlıkta bile farklı böcek türlerini ayırt edebilmesini sağlayan ekolasyon yeteneğinden ilham alır. Yapay Arı Koloni Algoritması [21], bal arısı sürüsünün zeki davranışlarını taklit eder. Kurt Arama Algoritması [22], kurtların yiyecek arama ve düşmanlarından kaçarak hayatta kalma davranışlarını modeller. Guguk Kuşu Arama Algoritması [23], bazı guguk kuşu türlerinin kuluçka davranışlarıyla birlikte bazı meyve sineklerinin ve kuşların Levy uçuş davranışlarını taklit eder. Ateş Böceği Algoritması [24], ateş böceklerinin yanıp sönme davranışlarından ilham alır. Karınca Aslanı Optimizasyon Algoritması [25], doğadaki karınca aslanlarının avlanma mekanizmasını taklit eder. Gri Kurt Optimizasyon Algoritması [26], doğadaki gri kurtların liderlik hiyerarşisini ve avlanma mekanizmasını taklit eder. Balina Optimizasyon Algoritması [27] kambur balinaların kabarcık ağı avlanma stratejisine dayanır.

Metasezgisel algoritmaların hepsinin ilham kaynağı biyoloji değildir. Bazı metasezgisel algoritmalar fizik ve kimyadan ilham alınarak geliştirilmiştir. Bu algoritmalar elektrik yükleri, yerçekimi, nehir sistemleri vb. içeren belli fiziksel ve kimyasal kanunlar taklit edilerek geliştirilmiştir. Büyük Patlama-Büyük Çöküş Algoritması [28], Büyük Patlama ve Büyük Çöküş teorilerinden ilham alır. Kara Delik Algoritması [29], kara delik olgusunun gözlemlenebilir gerçekliklerinden ilham alır. Merkezi Kuvvet Optimizasyon Algoritması [30], yerçekimi kinematiği metaforuna dayanır. Yüklü Sistem Arama Algoritması [31], bazı fizik ve mekanik kurallarına dayanır. Elektromanyetizma Optimizasyon Algoritması [32], elektromanyetizma ilkelerine dayanır. Yerçekimi Arama Algoritması [33], yerçekimi ve kütle etkileşimi yasalarına dayanır. Armoni Arama Algoritması [34], müziğin amacının mükemmel armoniyi arıyor olmasına dayanır. Su Döngüsü Algoritması [35] su döngüsü sürecinden ve nehirlerin, akarsuların denize akışından ilham alır. Spiral Optimizasyon Algoritması [36] doğadaki spiral olguyu taklit eder.

Doğadaki diğer canlıların yanı sıra insan davranışlarından ilham alınarak geliştirilen metasezgisel algoritmalar da mevcuttur. Anarşik Toplum Optimizasyon [37] Algoritması, üyelerin durumlarını iyileştirmek için anarşik davrandıkları sosyal gruplamadan ilham alır. Emperyalist Rekabetçi Algoritma [38] insanların sosyo-politik evrim sürecinden ilham alır. Sosyal-Tabanlı Algoritma [39], Evrim Algoritması ve Emperyalist Rekabetçi Algoritma'yı birleştirerek yeni bir yaklaşım önermektedir.

Ayrıca spor kavramlarından ilham alınarak geliştirilen spor tabanlı metasezgisel algoritmalar da günümüzde oldukça popülerdir. Bu algoritmalarda insanların davranışları veya spor ligleri simüle edilmektedir. Lig Şampiyonası Algoritması [40], yapay bir ligde yapay takımların oynadığı bir şampiyona ortamını taklit etmeye çalışır. Futbol Lig Müsabakası Algoritması [41], futbol liglerindeki takımlar arasındaki müsabakalardan ve oyuncular arasındaki mücadelelerden ilham alır. Altın Top Algoritması [42] futbol kavramlarıa dayanır.

Tüm bu algoritmaların yanı sıra matematikten ilham alan algoritmalar da son derece önemlidir. Metasezgisel ve matematiksel programlama (MP) tekniklerinin birlikte kullanılmasıyla geliştirilen sezgisel algoritmalara Matheuristic adı verilmektedir. Ancak literatürde matematikten ilham alan algoritmaların sayısı ne yazık ki oldukça azdır. Baz Optimizasyon Algoritması [43] çözümleri optimum noktaya yönlendiren bir yer değiştirme parametresi ile birlikte temel aritmetik operatörlerin kombinasyonunu kullanır.

Bu tez çalışmasında tanıtılan Sinüs Kosinüs Algoritması (SKA) Seyedali Mirjalili tarafından sinüs ve kosinüs fonksiyonlarına dayalı bir matematiksel model kullanılarak optimizasyon problemlerinin çözümü için geliştirilen matematik tabanlı metasezgisel bir algoritmadır [44].

Her ne kadar literatüre kazandırılmış çok başarılı algoritmalar ve teknikler geliştirilmiş olsa da; bilimsel alanda sürekli iyileşme ve daima daha iyiyi arama felsefesi altında yeni tekniklerin tasarlanması, geliştirilmesi ve uygulanması önemli bir görevdir. [45]. Bu amaç doğrultusunda bu çalışmada kapsamında SKA üzerinde iyileştirmeler yapılarak Adaptif Sinüs Kosinüs Algoritması önerilmektedir. Ayrıca optimizasyon problemlerinin çözümünde kullanılmak üzere yeni bir genel amaçlı metasezgisel algoritma olan Altın Sinüs Algoritması geliştirilip bu algoritmanın da kaotik haritalarla iyileştirilmesi sağlanmıştır. Burada kullanılan kaotik haritalar normalize edilen ve hibrit yapıda olan yeni bir yaklaşım olarak sunulmaktadır.

2. SİNÜS KOSİNÜS ALGORİTMASI (SKA)

Sinüs Kosinüs Algoritması (SKA) Seyedali Mirjalili tarafından sinüs ve kosinüs fonksiyonlarına dayalı bir matematiksel model kullanılarak optimizasyon problemlerinin çözümü için geliştirilen popülasyon tabanlı metasezgisel bir algoritmadır [46]. Popülasyon tabanlı optimizasyon teknikleri genel olarak rastgele çözüm kümesi ile optimizasyon sürecini başlatır. Bu rastgele küme uygunluk fonksiyonu ile defalarca değerlendirilir ve optimizasyon tekniğinin temeli olan kural seti ile iyileştirilir. Stokastik tabanlı optimizasyon teknikleri, optimizasyon probleminin optimumunu stokastik olarak aradığından tek bir adımda çözümü bulmayı garanti etmez. Ancak yeterli sayıda rastgele çözüm ve optimizasyon adımları ile global optimumun bulunma olasılığı artar.

Popülasyon tabanlı stokastik algoritmalar arasındaki farklar her ne olursa olsun, optimizasyon süreci keşif ve sömürü olmak üzere ortak iki aşamadan oluşur. Optimizasyon algoritması arama uzayının umut verici alanlarını bulmak için yüksek rastgelelik oranı ile çözüm kümesindeki rastgele çözümleri hızlıca bir araya getirir. Bununla birlikte, sömürü aşamasında rastgele çözümlerde kademeli olarak değişiklikler yapılır ve rastgele varyasyonlar keşif aşamasındakinden çok daha azdır.

Bu çalışmada, her iki aşama için önerilen pozisyon güncelleme denklemleri Denklem 2.1 ve Denklem 2.2'deki gibidir:

$$X_i^{t+1} = X_i^t + r_1 \times \sin(r_2) \times |r_3 P_i^t - X_i^t| \tag{2.1}$$

$$X_i^{t+1} = X_i^t + r_1 \times \cos(r_2) \times |r_3 P_i^t - X_{ii}^t| \tag{2.2}$$

X_i^t: i. boyutun t. iterasyonundaki güncel çözümü

r_1, r_2, r_3 : Rastgele sayılar

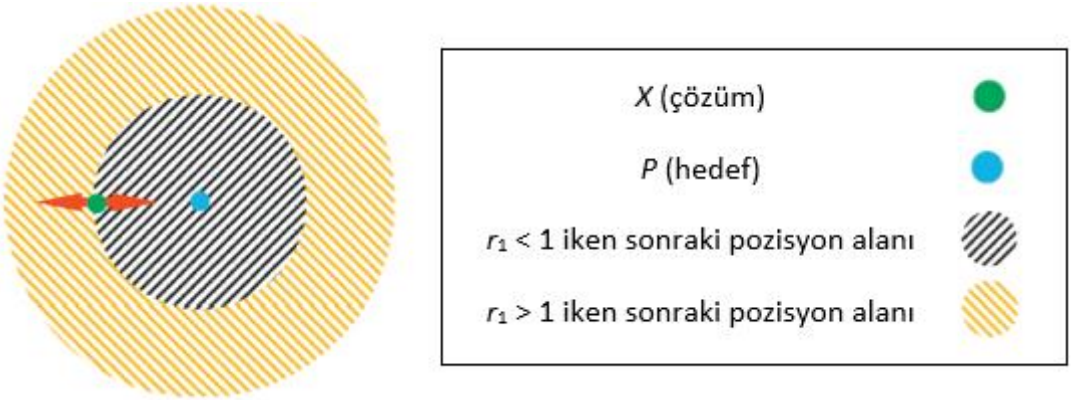
r_4 : [0, 1] aralığında rastgele bir sayıdır

P_i : i. boyuttaki hedef noktanın pozisyonu

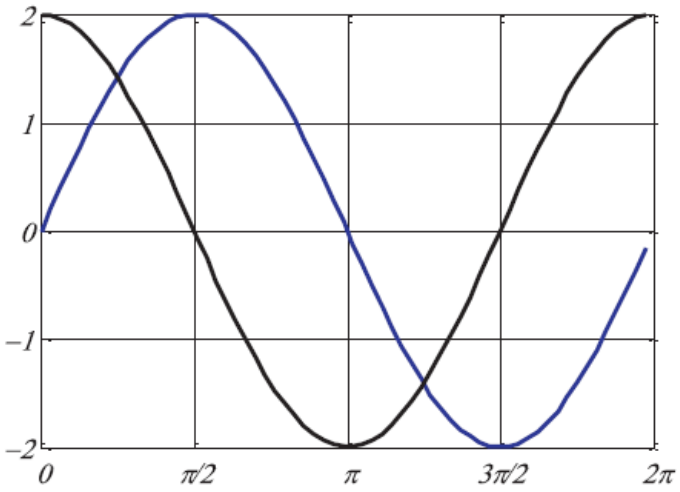
|| : Mutlak değer

Bu iki denklem Denklem 2.3'teki gibi birlikte kullanılır.

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times |r_3 P_i^t - X_i^t|, r_4 < 0.5 \\ X_i^t + r_1 \times \cos(r_2) \times |r_3 P_i^t - X_{ii}^t|, r_4 \geq 0.5 \end{cases} \tag{2.3}$$



Şekil 2.1. Denklem 2.1 ve Denklem 2.2'deki sinüs ve kosinüsün bir sonraki pozisyon üzerindeki etkileri [44]



Şekil 2.2. [-2, 2] aralığında sinüs ve kosinüs [44]

Yukarıdaki denklemlerde görüleceği üzere SKA'da; r_1, r_2, r_3, r_4 olmak üzere 4 ana parametre kullanılır.

r_1: Bir sonraki pozisyon bölgesini (veya hareket yönü) belirler.

r_2: Hedefe ulaşmak için, içe doğru ya da dışa doğru ne kadar hareket edileceğini belirler.

r_3: Stokastik ağırlığı rastgele belirler. $r_3 > 1$ olması stokastikliğin önemli olduğunu, $r_3 < 1$ olması ise stokastikliğin daha az etkili olduğunu belirtir.

r_4: Denklemdeki sinüs ve kosinüs bileşenleri arasındaki geçişi sağlar.

Bu yöntemde sinüs ve kosinüs kullanıldığından algoritma Sinüs Kosinüs Algoritması (SKA) olarak adlandırılmaktadır. Denklemlerdeki sinüs ve kosinüs fonksiyonlarının etkileri Şekil 2.1'de gösterilmektedir. Bu şekil, önerilen denklemlerin arama uzayındaki iki çözüm arasındaki bir alanın nasıl belirlendiğini gösterir. Şekil 2.1'de iki boyutlu bir model gösterilmesine rağmen bu denklemin daha yüksek boyutlar için genişletilebileceği unutulmamalıdır. Sinüs ve kosinüs fonksiyonlarının döngüsel deseni diğer bir çözüm etrafında yeniden konumlandırılabilir bir çözüm sağlar. Bu durum iki çözüm arasındaki tanımlı alanı sömürmeyi garanti eder. Arama alanını keşfetmek için, ilgili hedeflere karşılık gelen alanlar arasındaki boşluk dışında da çözümler aramak gerekir. Bu, Şekil 2'de gösterilen sinüs ve kosinüs fonksiyonlarının değişim aralığı ile elde edilebilir.

Sinüs ve kosinüs fonksiyonlarının [-2, 2] aralığındaki etkilerinin kavramsal modeli Şekil 3'te gösterilmiştir. Bu şekil, sinüs ve kosinüs fonksiyon aralığındaki değişimi ve başka bir çözüm ile kendisi arasındaki alanda pozisyonu içe doğru veya dışa doğru güncellemek için nasıl bir çözüm bulunması gerektiğini göstermektedir. Rastgele konum, içeride veya dışarıda olmak üzere Denklem 2.3'teki gibi [0, 2π] aralığında rastgele bir r_2 sayısı belirlenmesiyle elde edilir. Bu nedenle bu mekanizma arama uzayında keşif ve sömürüyü garanti eder.

Algoritma, arama uzayının umut verici alanlarını bulmak ve global optimuma yakınsamak için keşif ve sömürüyü dengelemelidir. Keşif ve sömürüyü dengelemek amacıyla, Denklem 2.4 kullanılarak Denklem 2.1, 2.2 ve 2.3'teki sinüs ve kosinüs aralığı adaptif olarak değiştirilir:

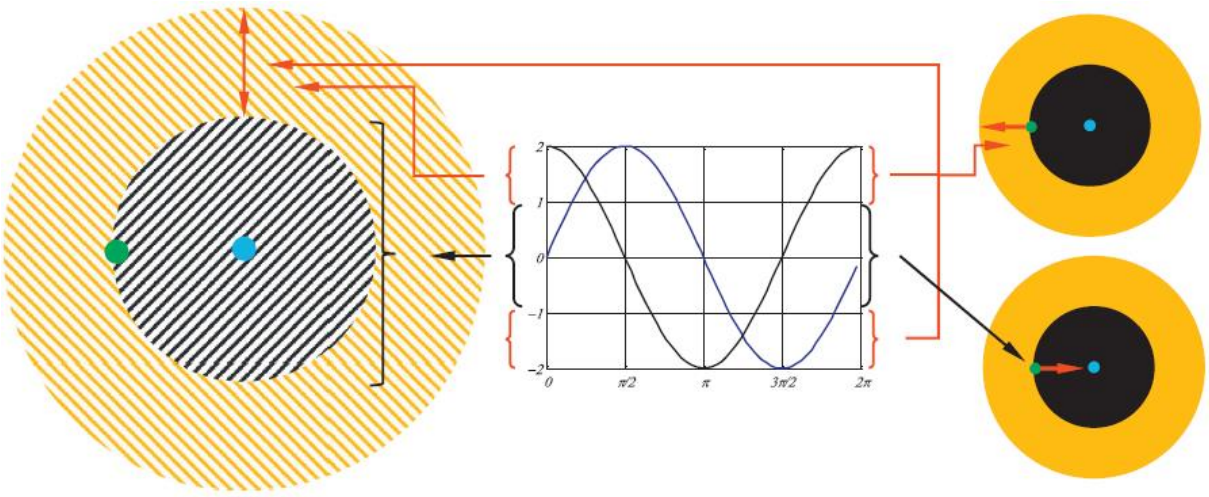
$$r_1 = c - t\frac{c}{T} \tag{2.4}$$

t, mevcut iterasyondur. *T* maksimum iterasyon sayısıdır ve *c* bir sabittir.

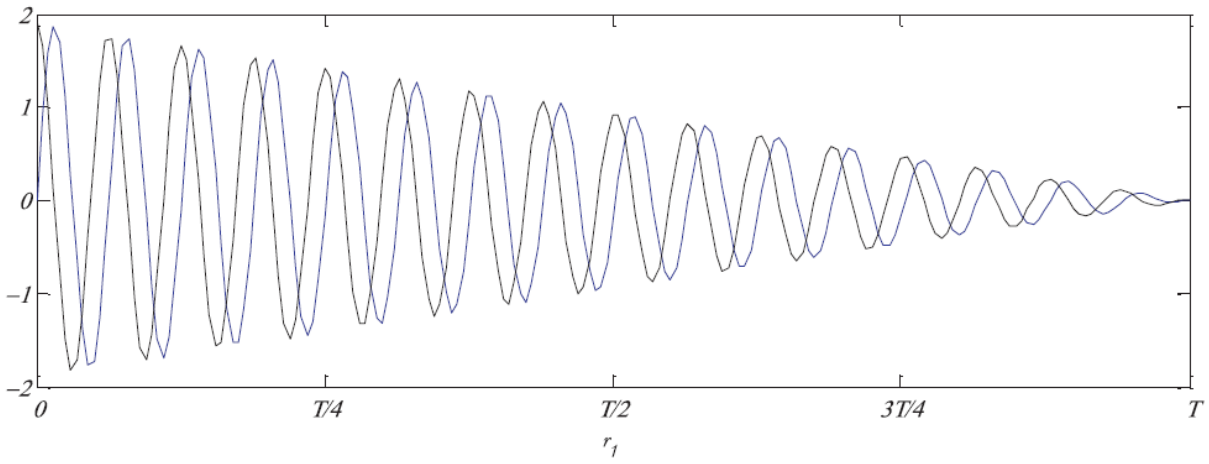
Şekil 2.4, bu denklemin iterasyonlar boyunca sinüs ve kosinüs fonksiyon aralığının azalışını gösterir. Şekil 2.3 ve Şekil 2.4'ten tahmin edilebileceği gibi, SKA'nın sinüs ve kosinüs fonksiyonları, arama uzayını (1, 2] ve [-2, 1) aralığında araştırır. Ancak, algoritma [-1, 1] arama alanını sömürür.

SKA'nın sözde kodu Şekil 2.5'te gösterilmektedir. SKA optimizasyon sürecinde, rastgele çözüm kümesi ile başlamaktadır. Algoritma, elde edilen en iyi çözümü kaydeder, onu hedef noktası olarak atar ve bu çözüme göre diğer çözümleri günceller. Bu arada, sinüs

ve kosinüs fonksiyon aralığı sömürüyü garanti etmek için iterasyon sayısı arttıkça güncellenir. Algoritma varsayılan olarak iterasyon sayacının maksimum iterasyon sayısını aşması ile optimizasyon sürecini sonlandırır. Bununla birlikte sonlandırma koşulu olarak maksimum fonksiyon değerlendirme sayısı veya elde edilen global optimumun doğruluğu dikkate alınabilir.



Şekil 2.3. [-2, 2] aralığında sinüs ve kosinüs; bir çözümün hedefin etrafında veya ötesinde dolaşmasına verir [44]



Şekil 2.4. Sinüs ve kosinüs aralığı için azalan model (c = 3) [44]

Başlat: Arama ajanları kümesini (çözümler) (X) başlat.

Do

Değerlendir: Her arama ajanını uygunluk fonksiyonu ile değerlendir.

Güncelle: Şimdiye kadar elde edilen en iyi çözümü güncelle ($P = X$').

Güncelle: r_1, r_2, r_3 ve r_4 güncelle.

Güncelle: Denklem 2.3'ü kullanarak arama ajanlarının pozisyonlarını güncelle.

While ($t <$ maksimum iterasyon sayısı)

Return: Şimdiye kadar global optimum olarak elde edilen en iyi çözümü döndür.

Şekil 2.5. SKA'nın sözde kodu

SKA'nın belirtilen operatörler ile optimizasyon problemlerinin global optimumunu belirlemesi teorik olarak aşağıdaki nedenlerden dolayı mümkündür:

1. SKA belirli bir problem için rastgele çözümler dizisi oluşturur ve geliştirir, bu nedenle bireysel tabanlı algoritmalara kıyasla yüksek keşif ve yerel optimumdan kaçınmayı sağlar.
2. Sinüs ve kosinüs fonksiyonları 1'den büyük veya -1'den düşük bir değer döndürdüğünde arama uzayının farklı bölgeleri keşfedilir.
3. Arama uzayının umut verici alanları sinüs ve kosinüs -1 ile 1 aralığında değer döndürdüğünde sömürülür.
4. SKA, sinüs ve kosinüs fonksiyon aralığının adaptif kullanımı ile keşiften sömürüye kolayca geçer.
5. Global optimumun en iyi yakınsaması hedef nokta olarak bir değişkende saklanır ve optimizasyon sırasında asla kaybolmaz.
6. Çözümler, pozisyonlarını elde edilen en iyi çözümler etrafında güncellediğinden optimizasyon boyunca arama uzayının en iyi bölgelerine doğru bir eğilim vardır.
7. Önerilen algoritma, optimizasyon problemini kara kutu olarak gördüğünden farklı alanlardaki problemlere kolayca uygulanabilir.

2.1. Deney ve Sonuçlar

SKA'nın performansını değerlendirmek için literatürde yaygın olarak kullanılan 23 kalite testi fonksiyonu üzerinde testler yapılmıştır. Bu fonksiyonlar tek modlu fonksiyonlar, çok modlu fonksiyonlar ve sabit boyutlu çok modlu fonksiyonlar olmak üzere üç farklı kategoride incelenebilir. Tablo 2.1'de gösterilen F1 - F7 arasındaki fonksiyonlar tek modlu fonksiyonlardır ve bir tek global optimuma sahiptirler. Bu fonksiyonlar arama algoritmalarının yakınsama oranını test etmek için tasarlanmıştır. Birden fazla lokal minimuma sahip olan ve bundan dolayı optimize edilmesi oldukça zor olan F8 – F13 arasındaki çok modlu fonksiyonlar Tablo 2.2'de gösterilmiştir. Çok modlu fonksiyonlarda problem boyutu sayısı arttıkça yerel optimum sayısı da artmaktadır. Bu nedenle bu tür test problemleri optimizasyon algoritmalarının arama kapasitelerini değerlendirmede oldukça önemlidir. Tablo 2.3'te gösterilen F14 – F23 arasındaki sabit boyutlu çok modlu fonksiyonların çok modlu fonksiyonlardan tek farkı boyutlarının düşük sayıda olmasından dolayı az sayıda yerel minimum içermeleridir. SKA, ilk kez mühendislik tasarım problemlerinden biri olan germe / sıkıştırma yay tasarım probleminin çözümünde kullanılarak algoritmanın kısıtlı problemler üzerindeki etkinliği de test edilmiştir.

SKA iyi bilinen sürü tabanlı algoritmalardan Parçacık Sürü Optimizasyonu (PSO), Karınca Koloni Optimizasyon (KKO) Algoritması ve güncel algoritmalar Balina Optimizasyon Algoritması (BOA), Yerçekimi Arama Algoritması (YAA) olmak üzere 4 metasezgisel algoritma ile karşılaştırılmıştır. SKA matematikten ilham alması yönüyle karşılaştırılan algoritmalardan farklıdır. Algoritmaların popülasyon büyüklüğü 30 ve iterasyon sayıları 1000 olarak kabul edilmiştir. Boyutları fonksiyon tanımlarında V_{no} olarak belirtilen her kalite testi fonksiyonu için algoritmalar 30 kez çalıştırılmıştır. Elde edilen ortalama, standart sapma, en iyi sonuç, en kötü sonuç ve ortalama çalışma zamanı istatistiksel sonuçları Tablo 2.4'te gösterilmektedir.

2.1.1. Kalite Testi Fonksiyonları Sonuçları

Kalite testi fonksiyonları sonuçları incelendiğinde SKA'nın optimum sonuca karşılaştırılan diğer tüm algoritmalardan (PSO, BOA, KKO, YAA) çok daha kısa sürede yakınsadığı açıkça görülmektedir. SKA F14, F16, F17 ve F18 kalite testi fonksiyonlarında optimum sonucu elde etmiştir.

Karşılaştırılan algoritmalarda kullanılan parametreler şu şekildedir:

1. PSO: *Atalet ağırlığı* = 1, *Atalet ağırlığı sönüm oranı* = 0.99, *Kişisel öğrenme katsayısı* = 1.5, *Küresel öğrenme katsayısı* = 2.0
2. KKO: *Örnek boyutu* = 40, *Yoğunlaşma faktörü* = 0.5, *Sapma - uzaklık oranı*=1
3. BOA: *b* = 1
4. YAA: *R_norm* = 2, *R_gücü* = 1, *Elitist kontrolü* = 1
5. SKA: *c* = 2

Tablo 2.1. Tek modlu kalite testi fonksiyonlarının tanımı

Fonksiyon	V_{no}	*Aralık*	F_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100, 100]	0
$F_2(x) = \sum_{i=1}^{n} x_i^2 \ \|x_i\| + \prod_{i=1}^{n} \|x_i\|$	30	[-10, 10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[-100, 100]	0
$F_4(x) = max\{\|x_i\|, 1 \leq i \leq n\}$	30	[-100, 100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30, 30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[-100, 100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$	30	[-1.28, 1.28]	0

Tablo 2.2. Çok modlu kalite testi fonksiyonlarının tanımı

Fonksiyon	V_{no}	*Aralık*	F_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500, 500]	-418.9829 x 30
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12, 5.12]	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[-32, 32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{i}\right) + 1$	30	[-600, 600]	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2 + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\}$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50, 50]	0
$F_{13}(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[-50, 50]	0

Tablo 2.3. Sabit boyutlu çok modlu kalite testi fonksiyonlarının tanımları

Fonksiyon	V_{no}	*Aralık*	F_{min}
$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$	2	[-65, 65]	1
$F_{15}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}]^2$	4	[-5, 5]	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5, 5]	-1.0316
$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	2	[-5, 5]	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2, 2]	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	[1, 3]	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	[0, 1]	-3.32
$F_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.4028
$F_{23}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.5363

Tablo 2.4. Kalite testi fonksiyonları sonuçları (*ort*: ortalama çözüm, *sd*: standart sapma, *en iyi*: en iyi çözüm, *en kötü*: en kötü çözüm, *süre*: saniye cinsinden ortalama çalışma süresi, *F*: fonksiyon *İ*: istatistikler)

F	*İ*	SKA	PSO	BOA	KKO	YAA
	ort	0.0310	6.5085e-16	1.4109e-154	1.2766	1.0976e-16
	sd	0.0866	1.1800e-15	7.0872e-154	0.8429	3.5117e-17
F1	*en iyi*	2.4792e-06	1.0232e-17	3.3634e-166	0.4405	6.6897e-17
	en kötü	0.3945	5.4858e-15	3.8875e-153	4.4251	2.1405e-16
	süre	1.2570	8.3309	4.2849	43.7911	12.0372
	ort	2.1026e-05	2.1071e-05	5.0254e-102	3.6260e+03	5.2733e-08
	sd	4.3726e-05	1.0554e-04	1.9523e-101	1.9618e+04	1.6126e-08
F2	*en iyi*	2.9768e-09	6.9921e-10	1.4491e-114	1.1190	2.7875e-08
	en kötü	2.1087e-04	5.7914e-04	9.2622e-101	1.0750e+05	1.0028e-07
	süre	1.3360	8.5741	4.5383	46.1645	10.7295
	ort	4.0282e+03	7.6568	1.9940e+04	1.0596e+05	432.9601
	sd	3.2698e+03	5.5415	1.1221e+04	2.9680e+04	155.3030
F3	*en iyi*	109.8140	1.0370	3.6202e+03	4.7676e+04	217.7493
	en kötü	1.4489e+04	22.0031	4.3344e+04	1.6262e+05	780.2057
	süre	6.2230	17.1235	9.6729	51.2220	14.0386
	ort	19.9222	0.6941	40.8308	78.1506	1.5515
	sd	11.2356	0.3446	32.4037	9.6495	1.5699
F4	*en iyi*	1.5395	0.2272	0.0519	42.8106	1.0044e-08
	en kötü	42.7265	1.3777	92.0102	92.1774	5.3571
	süre	1.6804	12.2517	5.2079	39.2201	7.3923
	ort	1.0127e+03	41.3337	27.2594	5.0124e+04	36.3915
	sd	2.8445e+03	32.4499	0.6228	4.5712e+04	53.9666
F5	*en iyi*	28.3801	2.3651	26.5891	6.4021e+03	24.5371
	en kötü	1.2227e+04	110.3879	28.7495	1.9159e+05	322.0843
	süre	1.7429	11.0633	4.7569	38.6504	9.5483
	ort	4.5788	2.7841e-15	0.1062	1.0898	0.2000
	sd	0.5852	8.7185e-15	0.1165	0.5781	0.7611
F6	*en iyi*	3.8463	1.7238e-17	0.0094	0.4107	0
	en kötü	6.7212	4.7394e-14	0.4399	3.1241	4.0000
	süre	1.8441	9.5745	4.7322	36.7716	10.1557
	ort	0.0421	0.0148	0.0024	0.1854	0.0647
	sd	0.0528	0.0059	0.0023	0.0757	0.0254
F7	*en iyi*	0.0047	0.0063	5.6430e-05	0.0546	0.0094
	en kötü	0.2775	0.0282	0.0090	0.4039	0.1131
	süre	2.3571	8.5231	6.2950	42.1256	10.7440
	ort	-3.9367e+03	-6.3686e+03	-1.1224e+04	-4.4590e+149	-2.4485e+03
	sd	258.2417	867.5216	1.6018e+03	2.4407e+150	425.4308
F8	*en iyi*	-4.5524e+03	-8.0276e+03	-1.2569e+04	-1.3368e+151	-3.5570e+03
	en kötü	-3.5344e+03	-4.3764e+03	-7.8490e+03	-3.0059e+98	-1.7879e+03
	süre	2.0996	8.6910	5.0582	40.6682	9.4981
	ort	23.4859	44.7399	0	252.9477	27.0629
	sd	32.5145	13.7505	0	18.7779	6.2785
F9	*en iyi*	6.9014e-06	23.8790	0	193.7615	17.9093
	en kötü	168.7336	81.5864	0	276.6462	41.7882
	süre	1.9521	9.8125	4.8869	39.8226	13.6292
	ort	12.6852	1.0915	4.0856e-15	0.6876	7.8281e-09
	sd	9.5190	0.7594	2.3511e-15	0.3804	1.6719e-09
F10	*en iyi*	2.5556e-04	2.7719e-09	8.8818e-16	0.1548	5.7326e-09
	en kötü	20.3227	2.3162	7.9936e-15	1.7909	1.3408e-08
	süre	2.1039	10.4712	5.0623	42.3656	15.1583
	ort	0.3685	0.0240	0.0050	0.9188	8.2013
	sd	0.3339	0.0246	0.0193	0.0798	3.2014
F11	*en iyi*	7.5275e-04	0	0	0.6492	2.6444
	en kötü	0.9431	0.0860	0.0875	1.0283	14.4065
	süre	2.2472	11.9485	5.6325	41.3976	14.9480
F12	*ort*	3.1533	0.2319	0.0110	3.2754e+04	0.1608

	sd	6.2240	0.5471	0.0177	7.9615e+04	0.2849
	en iyi	0.3258	1.0560e-18	0.0012	18.9615	3.5333e-19
	en kötü	33.4485	2.7038	0.0956	3.2040e+05	1.4847
	süre	3.9721	16.8464	4.5318	42.5764	15.5377
	ort	18.5548	0.1020	0.1962	8.1417e+04	0.0033
	sd	65.8292	0.4602	0.1581	1.1285e+05	0.0104
F13	*en iyi*	2.2029	5.5146e-18	0.0398	961.1513	4.1030e-18
	en kötü	365.0753	2.5085	0.7707	3.9818e+05	0.0548
	süre	3.9894	11.5014	7.6812	43.5993	13.4597
	ort	1.5275	4.4098	2.7961	1.3235	3.4221
	sd	0.8922	2.9700	3.2852	1.7829	2.6992
F14	*en iyi*	0.9980	0.9980	0.9980	0.9980	0.9980
	en kötü	2.9821	11.7187	10.7632	10.7632	13.8192
	süre	10.3350	20.2909	10.0869	19.4695	17.3979
	ort	9.7180e-04	3.4190e-04	5.6780e-04	0.0011	0.0023
	sd	3.8543e-04	1.6780e-04	2.7009e-04	3.1570e-04	8.7184e-04
F15	*en iyi*	3.4077e-04	3.0749e-04	3.0836e-04	8.8731e-04	6.2116e-04
	en kötü	0.0015	0.0012	0.0015	0.0019	0.0052
	süre	1.9942	9.4237	2.0983	8.7483	9.6892
	ort	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	sd	2.5344e-05	6.7752e-16	1.6070e-10	6.7752e-16	4.8787e-16
F16	*en iyi*	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	en kötü	-1.0315	-1.0316	-1.0316	-1.0316	-1.0316
	süre	1.3675	9.8322	1.4481	5.5078	7.5780
	ort	0.3985	0.3979	0.3979	0.3979	0.3979
	sd	6.0681e-04	0	3.9088e-06	0	0
F17	*en iyi*	0.3979	0.3979	0.3979	0.3979	0.3979
	en kötü	0.4003	0.3979	0.3979	0.3979	0.3979
	süre	1.2613	7.8293	1.3837	5.69707	7.2001
	ort	3.0000	3.0000	3.0000	3.0000	3.0000
	sd	3.2611e-05	1.7954e-15	9.2119e-05	1.3194e-15	3.1250e-15
F18	*en iyi*	3.0000	3.0000	3.0000	3.0000	3.0000
	en kötü	3.0001	3.0000	3.0005	3.0000	3.0000
	süre	1.3814	9.1013	1.4617	5.8880	7.2429
	ort	-3.8551	-3.8370	-3.8609	-3.8628	-3.8628
	sd	0.0024	0.1411	0.0022	2.7101e-15	2.2913e-15
F19	*en iyi*	-3.8618	-3.8628	-3.8628	-3.8628	-3.8628
	en kötü	-3.8519	-3.0898	-3.8549	-3.8628	-3.8628
	Süre	2.3266	10.1333	2.4045	7.7821	9.2076
	ort	-2.8866	-3.2863	-3.2429	-3.2665	-3.3220
	sd	0.4145	0.0554	0.1113	0.0603	1.4889e-15
F20	*en iyi*	-3.2333	-3.3220	-3.3219	-3.3220	-3.3220
	en kötü	-1.2291	-3.2031	-2.9868	-3.2031	-3.3220
	süre	1.8505	9.9590	3.1532	11.7540	8.5954
	ort	-2.2850	-5.4786	-9.0471	-5.6870	-6.4134
	sd	1.8892	3.4558	2.2807	3.7094	3.6846
F21	*en iyi*	-5.9900	-10.1532	-10.1530	-10.1532	-10.1532
	en kötü	-0.4965	-2.6305	-2.6303	-2.6829	-2.6305
	süre	2.6262	11.3522	2.7609	11.0864	8.8242
	ort	-3.5487	-6.9147	-7.7342	-6.8594	-10.4029
	sd	1.6587	3.6103	2.9214	2.5485	1.4378e-15
F22	*en iyi*	-5.2185	-10.4029	-10.4027	-10.4029	-10.4029
	en kötü	-0.9071	-2.7519	-3.7235	-5.0877	-10.4029
	süre	3.1160	11.5451	3.1399	14.0219	9.9985
	ort	-4.0940	-5.9754	-8.5166	-7.2484	-10.2897
	sd	1.7448	3.6315	2.9532	3.0040	1.3511
F23	*en iyi*	-7.9749	-10.5363	-10.5363	-10.5363	-10.5363
	en kötü	-0.9460	-2.4217	-2.4217	-2.4217	-3.1359
	süre	3.5207	12.0961	3.8171	13.2174	11.2024

2.1.2. Parametrik Olmayan İstatistiksel Analiz Sonuçları

Metasezgisel algoritmaların stokastikliği nedeniyle algoritmalar karşılaştırılırken daha güvenilir sonuçlar elde etmek için istatistiksel testlerden faydalanılmaktadır. İstatiksel analizler için geliştirilmiş yöntemler parametrik ve parametrik olmayan testler olmak üzere iki sınıfa ayrılır. Parametrik testler yaygın olarak sayısal zeka deneylerinin analizinde kullanılmaktadır.

Ancak parametrik testler, sayısal zekaya dayalı stokastik algoritmaların performansının analizinde ihlal edilen koşullara dayalıdır. Bu koşullar bağımsızlık, normallik ve eş varyanslık olarak bilinir. Bu problemi aşmak için parametrik olmayan testler kullanılmaktadır [47]. Genellikle sayısal optimizasyon problemlerinin çözümü için önerilen algoritmaların başarımını kıyaslamak için parametrik olmayan testlerden Wilcoxon işaretli sıralar testi ve Wilcoxon sıra toplamı testi kullanılmaktadır.

SKA ve diğer algoritmalar arasında yapılan Wilcoxon sıra toplamı testine ait sonuçlar Tablo 2.5'te gösterilmektedir.

Tablo 2.5. Wilcoxon sıra toplamı testi karşılaştırma sonuçları

F	SKA / PSO	SKA / BOA	SKA / KKO	SKA / YAA
	p - değeri	*p* - değeri	*p* - değeri	*p* - değeri
F1	3.0199e-11	3.0199e-11	3.0199e-11	3.0199e-11
F2	9.7917e-05	3.0199e-11	3.0199e-11	8.4848e-09
F3	3.0199e-11	5.5329e-08	3.0199e-11	1.4110e-09
F4	3.0199e-11	0.0933	3.0199e-11	1.4643e-10
F5	1.3250e-04	4.0772e-11	4.9752e-11	5.0723e-10
F6	3.0199e-11	3.0199e-11	3.0199e-11	4.3909e-12
F7	0.0058	1.7769e-10	5.5727e-10	2.6806e-04
F8	3.3384e-11	3.0199e-11	3.0199e-11	3.3384e-11
F9	7.7364e-06	1.2118e-12	3.0199e-11	0.0392
F10	6.8971e-04	1.6711e-11	0.0184	3.0199e-11
F11	2.2780e-05	2.0973e-11	1.2870e-09	3.0199e-11
F12	3.3520e-08	3.0199e-11	4.9752e-11	2.3701e-10
F13	4.9752e-11	3.0199e-11	3.0199e-11	3.0199e-11
F14	4.0001e-05	0.0484	4.5618e-11	3.0036e-04
F15	2.1532e-10	1.8682e-05	0.0594	6.5183e-09
F16	1.2118e-12	3.0199e-11	1.2118e-12	4.0806e-12
F17	1.2118e-12	1.6132e-10	1.2118e-12	1.2118e-12
F18	1.0975e-11	0.1120	2.3657e-12	2.8936e-11
F19	4.5618e-11	2.4386e-09	1.2118e-12	2.3638e-12
F20	2.9744e-11	1.4294e-08	6.0455e-11	8.8675e-12
F21	2.2821e-05	1.3289e-10	5.4561e-05	8.1951e-06
F22	0.0031	9.0632e-08	1.0398e-09	1.2455e-11
F23	0.1892	1.7294e-07	1.1947e-06	2.0585e-10

2.1.3. Kısıtlı Mühendislik Tasarım Problemlerinin Çözümünde SKA Kullanılması

Gerçek sistemler genellikle kısıtlı problemlerden oluşur. Tasarım sürecinde çözümlerin kullanılabilirliğini sağlayan kısıtlar; Denklem 2.5'te görüldüğü üzere eşitlik ve eşitsizlik kısıtları olmak üzere iki çeşittir.

Genel olarak kısıtlı optimizasyon problemleri Denklem 2.5'teki gibi tanımlanır:

$$min\ \ f(x)$$

kısıtlar: $g_k(x) \leq 0, k = 1, 2, 3, \dots, q$

$$h_j(x) = 0, j = 1, 2, 3, \dots, m \tag{2.5}$$

$$alt_i \leq x_i \leq ust_i, i = 1, 2, 3, \dots, D$$

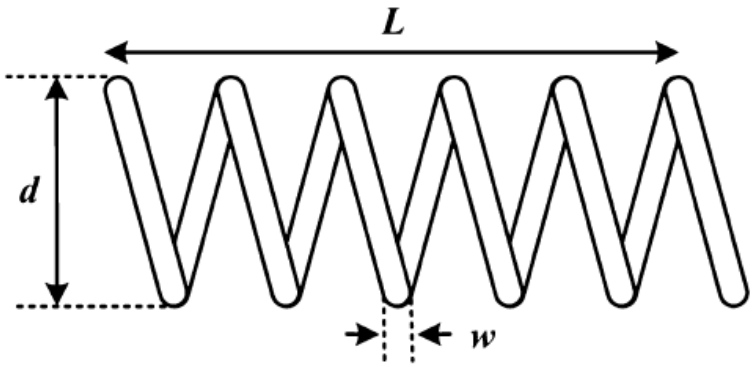
Burada $f(x)$ amaç fonksiyondur. $x = (x_1, x_2, \dots, x_D)$ olmak üzere D boyutlu bir vektördür. $g_k(x)$ eşitsizlik kısıtıdır ve $h_j(x)$ eşitlik kısıtıdır. alt_i ; x_i'nin alt sınırıdır ve ust_i ; x_i'nin üst sınırıdır.

Kısıtlı problemlerin optimize edilmesi sırasında dikkat edilmesi gereken en önemli husus kısıtların ihlal edilmemesidir. Bunu sağlamak için kısıtları kontrol eden bir metot algoritma ile entegre bir şekilde çalışmalıdır. Kısıtların kullanımı sağlayan; ceza metodu, özel operatörler, onarıcı algoritmalar, amaç fonksiyonunun ve kısıtların ayrılması, hibrit metotlar olmak üzere bazı metotlar bulunmaktadır [48]. SKA ile basınçlı kap probleminin çözümünde literatürde en çok kullanılan ve uygulanması kolay bir yöntem olan ceza metodu kullanılmıştır.

Şekil 2.6'da gösterilen germe / sıkıştırma yay tasarım probleminin çözümündeki temel amaç değişkenlerin optimum değerlerini elde ederek minimum ağırlıklı bir yay tasarlamaktır. Problem dört lineer olmayan eşitsizlik kısıtı ve tel çapı w (x_1), ortalama bobin çapı d (x_2), uzunluk (veya bobin sayısı) L (x_3) olmak üzere üç sürekli değişkenden oluşur. Bu kısıtlar ve problem Denklem 2.6'da gösterilmektedir.

SKA ve karşılaştırılan diğer algoritmalar (PSO, BOA, KKA, YAA) germe / sıkıştırma yay tasarım problemi üzerinde test edilirken popülasyon sayısı 200, iterasyon sayısı 10000 olarak belirlenmiştir ve algoritmalar 30 kez çalıştırılmıştır. Elde edilen sonuçlar Tablo 2.6'da gösterilmektedir. SKA problemi **71.69195** saniyede çözerek karşılaştırılan algoritmalar arasında sonucu en düşük sürede bulan algoritma olmuştur. Ayrıca optimum

maliyeti **0.012676** bularak PSO'dan sonra optimum maliyeti en düşük bulan ikinci algoritma konumundadır.



Şekil 2.6. Germe / sıkıştırma yay tasarım problemi

$$\min f(x_1, x_2, x_3) = (x_3 + 2)x_1^2 x_2$$

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

$$g_2(X) = \frac{x_2(4x_2 - x_1)}{12566x_1^3(x_2 - x_1)} + \frac{1}{5108x_1^2} - 1 \leq 0$$

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

$$g_4(X) = \frac{2(x_1 + x_2)}{3} - 1 \leq 0 \qquad (2.6)$$

$Değişken\ aralığı$: $0.05 \leq x_1 \leq 2,$

$0.25 \leq x_2 \leq 1.3$

$2.0 \leq x_3 \leq 15.0$

Tablo 2.6. Germe / sıkıştırma yay tasarım problemi karşılaştırma sonuçları

Algoritma	Optimum değişkenler			*Optimum Maliyet*	*Süre*
	w	*d*	*L*		
SKA	0.051108490847	0.342889153904	12.15302541048	0.012676201923	**71.69195**
PSO	0.051580471304	0.354110970914	11.44344489883	**0.012665448278**	517.93111
BOA	0.052614959700	0.379403901376	10.07314988814	0.012680631117	89.987781
KKO	0.057606226982	0.516451973173	5.739265672904	0.013263818150	511.46341
YAA	0.053462849967	0.395370059884	9.564393481895	0.013068653733	2463.76132

2.2. Sonuç

Bu bölümde yakın zamanda önerilen matematik tabanlı Sinüs Kosinüs Algoritması (SKA)'nın optimizasyon problemlerinin çözümünde kullanılması incelenmiştir. SKA 23 kalite testi fonksiyonu üzerinde ve mühendislik tasarım problemlerinden biri olan germe / sıkıştırma yay tasarım probleminin çözümünde diğer metasezgisel algoritmalar (PSO, BOA, KKA, YAA) ile karşılaştırılarak test edilmiştir. Ayrıca stokastik yöntemlerin performansının daha güvenilir bir şekilde belirlenmesini sağlayan istatiksel testlerden Wilcoxon sıra toplamı testi kullanılarak algoritmalardan elde edilen sonuçlar istatiksel olarak karşılaştırılmıştır. SKA'nın henüz yeni bir algoritması ve az sayıda parametre içermesinden dolayı algoritmada iyileştirmeler yapılarak daha iyi sonuçlar elde edilmesi amaçlanmaktadır.

3. ADAPTİF SİNÜS KOSİNÜS ALGORİTMASI

Sinüs Kosinüs Algoritması'nda her arama ajanının her boyutu için r_2, r_3 ve r_4 parametrelerinin rastgele olarak yeniden belirlenmesi hedefe ulaşmada sapmalara neden olmaktadır. Ayrıca her boyut bu parametrelerin yeniden belirlenmesi problemlerin çözümünde geçen süreyi artırmaktadır. Bu nedenle önerilen yeni yöntemde r_2, r_3 ve r_4 parametrelerinin her arama ajanı için belirlenip tüm boyutları için sabit tutularak hem hedeften sapmaların azaltılması hem de performansta iyileştirmelerin sağlanması amaçlanmaktadır.

Şekil 3.1'de sözde kodu verilen SKA üzerinde iyileştirmeler yapılarak geliştirilen Adaptif Sinüs Kosinüs Algoritması (ASKA)'nın sözde kodu Şekil 3.2'de gösterilmektedir.

1. *Başlangıç popülasyonunu her boyut için arama ajanı sayısı kadar düzgün dağılıma bağlı olarak rastgele oluştur*
2. *Arama ajanlarının uygunluğunu hesapla*
3. *En iyi arama ajanını bul ve hedef değer olarak ata*
4. ***while*** *maksimum iterasyon sayısı*
5. $t = 2, c = 2$
6. $r_1 = c - t*((c) / Maksimum\ iterasyon)$
7. ***for*** *arama ajanı sayısı*
8. ***for*** *boyut sayısı*
9. $r_2 \leftarrow 2\pi * rand$
10. $r_3 \leftarrow 2 * rand$
11. $r_4 \leftarrow rand$
12. ***if*** $r_4 < 0$
13. $X(i, j) \leftarrow X(i, j) + (r_1 * \sin(r_2) * |r_3 * P(j) - X(i,j)|$
14. **else**
15. $X(i, j) \leftarrow X(i, j) + (r_1 * \sin(r_2) * |r_3 * P(j) - X(i,j)|$
16. ***end if***
17. ***end for***
18. ***end for***
19. *En iyi çözümü (arama ajanı) bul ve P(j)'ye hedef değer olarak ata*
20. ***end while***
21. ***return*** *en iyi çözüm kümesi ve elde edien global optimum sonuç*

Şekil 3.1. Sinüs Kosinüs Algoritması sözde kodu

1. *Başlangıç popülasyonunu her boyut için arama ajanı sayısı kadar düzgün dağılıma bağlı olarak rastgele oluştur*
2. *Arama ajanlarının uygunluğunu hesapla*
3. *En iyi arama ajanını bul ve hedef değer olarak ata*
4. ***while*** *maksimum iterasyon sayısı*
5. $t = 2, c = 2$
6. $r_1 = c - t * ((c) / Maksimum\ iterasyon)$
7. ***for*** *arama ajanı sayısı*
8. $r_2 \leftarrow 2\pi * rand$
9. $r_3 \leftarrow 2 * rand$
10. $r_4 \leftarrow rand$
11. ***for*** *boyut sayısı*
12. ***if*** $r_4 < 0$
13. $X(i, j) \leftarrow X(i, j) + (r_1 * \sin(r_2) * |r_3 * P(j) - X(i,j)|$
14. **else**
15. $X(i, j) \leftarrow X(i, j) + (r_1 * \sin(r_2) * |r_3 * P(j) - X(i,j)|$
16. ***end if***
17. ***end for***
18. ***end for***
19. *En iyi çözümü (arama ajanı) bul ve P(j)'ye hedef değer olarak ata*
20. ***end while***
21. ***return*** *en iyi çözüm kümesi ve elde edien global optimum sonuç*

Şekil 3.2. Adaptif Sinüs Kosinüs Algoritması sözde kodu

3.1. Deney ve Sonuçlar

ASKA iyi bilinen sürü tabanlı algoritmalardan Parçacık Sürü Optimizasyonu (PSO), Karınca Koloni Optimizasyon (KKO) Algoritması ve güncel algoritmalar Balina Optimizasyon Algoritması (BOA), Yerçekimi Arama Algoritması (YAA), Sinüs Kosinüs Algoritması (SKA) olmak üzere 5 metasezgisel algoritma ile karşılaştırılmışır. Algoritmaların popülasyon büyüklüğü 30 ve iterasyon sayıları 1000 olarak kabul edilmiştir. Boyutları fonksiyon tanımlarında V_{no} olarak belirtilen her kalite testi fonksiyonu için algoritmalar 30 kez çalıştırılmıştır. Elde edilen ortalama, standart sapma, en iyi sonuç, en kötü sonuç ve ortalama çalışma zamanı istatistiksel sonuçları Tablo 3.1'de gösterilmektedir. Karşılaştırılan algoritmalarda kullanılan parametreler şu şekildedir:

1. PSO: *Atalet ağırlığı* = 1, *Atalet ağırlığı sönüm oranı* = 0.99, *Kişisel öğrenme katsayısı* = 1.5, *Küresel öğrenme katsayısı* = 2.0
2. KKO: *Örnek boyutu* = 40, *Yoğunlaşma faktörü* = 0.5, *Sapma - uzaklık oranı*=1
3. BOA: b = 1
4. YAA: *R_norm* = 2, *R_gücü* = 1, *Elitist kontrolü* = 1
5. SKA: c = 2

3.1.1. Kalite Testi Fonksiyonları Sonuçları

Tablo 3.1. Kalite testi fonksiyonları sonuçları (*ort*: ortalama çözüm, *sd*: standart sapma, *en iyi*: en iyi çözüm, *en kötü*: en kötü çözüm, *süre*: saniye cinsinden ortalama çalışma süresi, *F*: fonksiyon, *İ*:istatistik)

F	*İ*	ASKA	PSO	SKA	BOA	KKO	YAA
	ort	4.1321e-137	6.5085e-16	0.0310	1.4109e-154	1.2766	1.0976e-16
	sd	2.2632e-136	1.1800e-15	0.0866	7.0872e-154	0.8429	3.5117e-17
F1	*en iyi*	2.5945e-179	1.0232e-17	2.4792e-06	3.3634e-166	0.4405	6.6897e-17
	en kötü	1.2396e-135	5.4858e-15	0.3945	3.8875e-153	4.4251	2.1405e-16
	süre	1.01338	8.3309	1.2570	4.2849	43.7911	12.0372
	ort	1.3305e-70	2.1071e-05	2.1026e-05	5.0254e-102	3.6260e+03	5.2733e-08
	sd	4.3038e-70	1.0554e-04	4.3726e-05	1.9523e-101	1.9618e+04	1.6126e-08
F2	*en iyi*	8.3720e-84	6.9921e-10	2.9768e-09	1.4491e-114	1.1190	2.7875e-08
	en kötü	2.1926e-69	5.7914e-04	2.1087e-04	9.2622e-101	1.0750e+05	1.0028e-07
	süre	1.12487	8.5741	1.3360	4.5383	46.1645	10.7295
	ort	2.3615e-71	7.6568	4.0282e+03	1.9940e+04	1.0596e+05	432.9601
	sd	1.2678e-70	5.5415	3.2698e+03	1.1221e+04	2.9680e+04	155.3030
F3	*en iyi*	9.8711e-120	1.0370	109.8140	3.6202e+03	4.7676e+04	217.7493
	en kötü	6.9476e-70	22.0031	1.4489e+04	4.3344e+04	1.6262e+05	780.2057
	süre	5.92186	17.1235	6.2230	9.6729	51.2220	14.0386
	ort	7.9899e-69	0.6941	19.9222	40.8308	78.1506	1.5515
	sd	4.3646e-68	0.3446	11.2356	32.4037	9.6495	1.5699
F4	*en iyi*	4.6344e-89	0.2272	1.5395	0.0519	42.8106	1.0044e-08
	en kötü	2.3908e-67	1.3777	42.7265	92.0102	92.1774	5.3571
	süre	1.42658	12.2517	1.6804	5.2079	39.2201	7.3923
	ort	0.0337	41.3337	1.0127e+03	27.2594	5.0124e+04	36.3915
	sd	0.0428	32.4499	2.8445e+03	0.6228	4.5712e+04	53.9666
F5	*en iyi*	1.0003e-04	2.3651	28.3801	26.5891	6.4021e+03	24.5371
	en kötü	0.1531	110.3879	1.2227e+04	28.7495	1.9159e+05	322.0843
	süre	1.55189	11.0633	1.7429	4.7569	38.6504	9.5483
	ort	0.0020	2.7841e-15	4.5788	0.1062	1.0898	0.2000
	sd	0.0029	8.7185e-15	0.5852	0.1165	0.5781	0.7611
F6	*en iyi*	8.5915e-06	1.7238e-17	3.8463	0.0094	0.4107	0
	en kötü	0.0120	4.7394e-14	6.7212	0.4399	3.1241	4.0000
	süre	1.52137	9.5745	1.8441	4.7322	36.7716	10.1557
	ort	7.1326e-05	0.0148	0.0421	0.0024	0.1854	0.0647
	sd	7.7696e-05	0.0059	0.0528	0.0023	0.0757	0.0254
F7	*en iyi*	2.2908e-07	0.0063	0.0047	5.6430e-05	0.0546	0.0094
	en kötü	3.9592e-04	0.0282	0.2775	0.0090	0.4039	0.1131
	süre	2.03204	8.5231	2.3571	6.2950	42.1256	10.7440
	ort	-1.2569e+04	-6.3686e+03	-3.9367e+03	-1.1224e+04	-4.4590e+149	-2.4485e+03
	sd	0.0103	867.5216	258.2417	1.6018e+03	2.4407e+150	425.4308
F8	*en iyi*	-1.2569e+04	-8.0276e+03	-4.5524e+03	-1.2569e+04	-1.3368e+151	-3.5570e+03
	en kötü	-1.2569e+04	-4.3764e+03	-3.5344e+03	-7.8490e+03	-3.0059e+98	-1.7879e+03
	süre	1.74441	8.6910	2.0996	5.0582	40.6682	9.4981
	ort	0	44.7399	23.4859	0	252.9477	27.0629
	sd	0	13.7505	32.5145	0	18.7779	6.2785
F9	*en iyi*	0	23.8790	6.9014e-06	0	193.7615	17.9093
	en kötü	0	81.5864	168.7336	0	276.6462	41.7882
	süre	1.66793	9.8125	1.9521	4.8869	39.8226	13.6292
	ort	8.8818e-16	1.0915	12.6852	4.0856e-15	0.6876	7.8281e-09
	sd	0	0.7594	9.5190	2.3511e-15	0.3804	1.6719e-09
F10	*en iyi*	8.8818e-16	2.7719e-09	2.5556e-04	8.8818e-16	0.1548	5.7326e-09
	en kötü	8.8818e-16	2.3162	20.3227	7.9936e-15	1.7909	1.3408e-08
	süre	1.74515	10.4712	2.1039	5.0623	42.3656	15.1583
	ort	0	0.0240	0.3685	0.0050	0.9188	8.2013
	sd	0	0.0246	0.3339	0.0193	0.0798	3.2014
F11	*en iyi*	0	0	7.5275e-04	0	0.6492	2.6444
	en kötü	0	0.0860	0.9431	0.0875	1.0283	14.4065
	süre	1.90766	11.9485	2.2472	5.6325	41.3976	14.9480

	ort	5.5727e-05	0.2319	3.1533	0.0110	3.2754e+04	0.1608
	sd	1.0953e-04	0.5471	6.2240	0.0177	7.9615e+04	0.2849
F12	*en iyi*	4.6736e-08	1.0560e-18	0.3258	0.0012	18.9615	3.5333e-19
	en kötü	5.2938e-04	2.7038	33.4485	0.0956	3.2040e+05	1.4847
	süre	3.67295	16.8464	3.9721	4.5318	42.5764	15.5377
	ort	6.4582e-04	0.1020	18.5548	0.1962	8.1417e+04	0.0033
	sd	7.6868e-04	0.4602	65.8292	0.1581	1.1285e+05	0.0104
F13	*en iyi*	6.5958e-06	5.5146e-18	2.2029	0.0398	961.1513	4.1030e-18
	en kötü	0.0032	2.5085	365.0753	0.7707	3.9818e+05	0.0548
	süre	3.65425	11.5014	3.9894	7.6812	43.5993	13.4597
	ort	1.2626	4.4098	1.5275	2.7961	1.3235	3.4221
	sd	0.6860	2.9700	0.8922	3.2852	1.7829	2.6992
F14	*en iyi*	0.9980	0.9980	0.9980	0.9980	0.9980	0.9980
	en kötü	2.9821	11.7187	2.9821	10.7632	10.7632	13.8192
	süre	8.50384	20.2909	10.3350	10.0869	19.4695	17.3979
	ort	3.8053e-04	3.4190e-04	9.7180e-04	5.6780e-04	0.0011	0.0023
	sd	7.1784e-05	1.6780e-04	3.8543e-04	2.7009e-04	3.1570e-04	8.7184e-04
F15	*en iyi*	3.1178e-04	3.0749e-04	3.4077e-04	3.0836e-04	8.8731e-04	6.2116e-04
	en kötü	5.9624e-04	0.0012	0.0015	0.0015	0.0019	0.0052
	süre	1.63308	9.4237	1.9942	2.0983	8.7483	9.6892
	ort	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	sd	2.1956e-05	6.7752e-16	2.5344e-05	1.6070e-10	6.7752e-16	4.8787e-16
F16	*en iyi*	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	en kötü	-1.0315	-1.0316	-1.0315	-1.0316	-1.0316	-1.0316
	süre	1.19193	9.8322	1.3675	1.4481	5.5078	7.5780
	ort	0.3981	0.3979	0.3985	0.3979	0.3979	0.3979
	sd	4.8979e-04	0	6.0681e-04	3.9088e-06	0	0
F17	*en iyi*	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
	en kötü	0.3998	0.3979	0.4003	0.3979	0.3979	0.3979
	süre	0.7232	7.8293	1.2613	1.3837	5.69707	7.2001
	ort	3.0002	3.0000	3.0000	3.0000	3.0000	3.0000
	sd	2.3677e-04	1.7954e-15	3.2611e-05	9.2119e-05	1.3194e-15	3.1250e-15
F18	*en iyi*	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	en kötü	3.0008	3.0000	3.0001	3.0005	3.0000	3.0000
	süre	0.76903	9.1013	1.3814	1.4617	5.8880	7.2429
	ort	-3.8146	-3.8370	-3.8551	-3.8609	-3.8628	-3.8628
	sd	0.0800	0.1411	0.0024	0.0022	2.7101e-15	2.2913e-15
F19	*en iyi*	-3.8627	-3.8628	-3.8618	-3.8628	-3.8628	-3.8628
	en kötü	-3.6144	-3.0898	-3.8519	-3.8549	-3.8628	-3.8628
	Süre	2.05760	10.1333	2.3266	2.4045	7.7821	9.2076
	ort	-2.9617	-3.2863	-2.8866	-3.2429	-3.2665	-3.3220
	sd	0.2679	0.0554	0.4145	0.1113	0.0603	1.4889e-15
F20	*en iyi*	-3.2900	-3.3220	-3.2333	-3.3219	-3.3220	-3.3220
	en kötü	-2.0038	-3.2031	-1.2291	-2.9868	-3.2031	-3.3220
	süre	2.09464	9.9590	1.8505	3.1532	11.7540	8.5954
	ort	-10.1516	-5.4786	-2.2850	-9.0471	-5.6870	-6.4134
	sd	0.0033	3.4558	1.8892	2.2807	3.7094	3.6846
F21	*en iyi*	-10.1532	-10.1532	-5.9900	-10.1530	-10.1532	-10.1532
	en kötü	-10.1375	-2.6305	-0.4965	-2.6303	-2.6829	-2.6305
	süre	2.30390	11.3522	2.6262	2.7609	11.0864	8.8242
	ort	-10.4017	-6.9147	-3.5487	-7.7342	-6.8594	-10.4029
	sd	0.0025	3.6103	1.6587	2.9214	2.5485	1.4378e-15
F22	*en iyi*	-10.4029	-10.4029	-5.2185	-10.4027	-10.4029	-10.4029
	en kötü	-10.3923	-2.7519	-0.9071	-3.7235	-5.0877	-10.4029
	süre	3.05327	11.5451	3.1160	3.1399	14.0219	9.9985
	ort	-10.5352	-5.9754	-4.0940	-8.5166	-7.2484	-10.2897
	sd	0.0029	3.6315	1.7448	2.9532	3.0040	1.3511
F23	*en iyi*	-10.5363	-10.5363	-7.9749	-10.5363	-10.5363	-10.5363
	en kötü	-10.5221	-2.4217	-0.9460	-2.4217	-2.4217	-3.1359
	süre	3.14048	12.0961	3.5207	3.8171	13.2174	11.2024

3.1.2. Parametrik Olmayan İstatistiksel Analiz Sonuçları

İstatiksel anlamlılık değeri $\alpha = 0.05$ olmak üzere ASKA ile diğer metasezgisel algoritmalar arasında Wilcoxon işaretli sıralar testi yapılarak istatiksel sonuçlar Tablo 3.2'de gösterilmiştir. Burada $p < 0.05$ olması durumu karşılaştırılan algoritmalardan elde edilen sonuçlar arasında istatiksel açıdan anlamlı bir farkın olduğunu göstermektedir. Ayrıca R^+ ASKA'nın karşılaştırılan algoritmaya göre daha üstün sonuçlar elde ettiği rankların toplamını gösterirken, R^- ise karşılaştırılan algoritmanın ASKA'ya göre daha üstün sonuçlar elde ettiği rankların toplamını gösterir. *K* kazanma durumunu ifade eder.

Tablo 3.2. Wilcoxon işaretli sıralar testi karşılaştırma sonuçları

F	SKA / ASKA				PSO / ASKA				BOA / ASKA				KKO / ASKA				YAA / ASKA			
	p - value	R^+	R^-	K	*p*-değeri	R^+	R^-	K	*p* - value	R^+	R^-	K	*p* - value	R^+	R^-	K	*p* - value	R^+	R^-	K
F1	1.7344e-06	465	0	+	1.7344e-06	465	0	+	2.2248e-04	53	412	-	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F2	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	0	465	-	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F3	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F4	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F5	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F6	1.7344e-06	465	0	+	1.7344e-06	0	465	-	1.7344e-06	465	0	+	1.7344e-06	465	0	+	0.0028	87	378	-
F7	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.9209e-06	464	1	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F8	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.7344e-06	0	465	-	1.7344e-06	465	0	+
F9	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1	0	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F10	1.7344e-06	465	0	+	1.7344e-06	465	0	+	1.3388e-05	253	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F11	1.7344e-06	465	0	+	2.5631e-06	435	0	+	0.5000	3	0	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F12	1.7344e-06	465	0	+	0.6435	255	210	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	0.0166	349	116	+
F13	1.7344e-06	465	0	+	0.3820	275	190	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+	0.0571	140	325	-
F14	0.0020	383	82	+	1.1265e-05	446	19	+	0.4405	270	195	+	3.1123e-05	30	435	-	1.6394e-05	442	23	+
F15	1.7344e-06	465	0	+	1.6046e-04	49	416	-	4.8969e-04	402	63	+	1.7344e-06	465	0	+	1.7344e-06	465	0	+
F16	0.7189	250	215	+	1.7344e-06	0	465	-	1.7344e-06	0	465	-	1.7344e-06	0	465	-	1.7344e-06	0	465	-
F17	5.7064e-04	400	65	+	1.7344e-06	0	465	-	6.3391e-06	13	452	-	1.7344e-06	0	465	-	1.7344e-06	0	465	-
F18	0.0064	100	365	-	1.7344e-06	0	465	-	0.0024	85	308	-	1.7344e-06	0	465	-	1.7344e-06	0	465	-
F19	0.4165	193	272	-	3.1123e-05	30	435	-	6.1564e-04	66	399	-	1.7344e-06	0	465	-	1.7344e-06	0	465	-
F20	0.6733	253	212	+	1.7344e-06	0	465	-	1.0246e-05	18	447	-	2.6033e-06	4	461	-	1.7344e-06	0	465	-
F21	1.7344e-06	465	0	+	2.6134e-04	410	55	+	0.0024	380	85	+	0.0015	387	78	+	0.0087	360	105	+
F22	1.7344e-06	465	0	+	0.0207	345	120	+	2.3704e-05	438	27	+	2.6134e-04	410	55	+	1.7344e-06	0	465	-
F23	1.7344e-06	465	0	+	6.1564e-04	399	66	+	1.6394e-05	442	23	+	0.0036	374	91	+	3.1123e-05	30	435	-

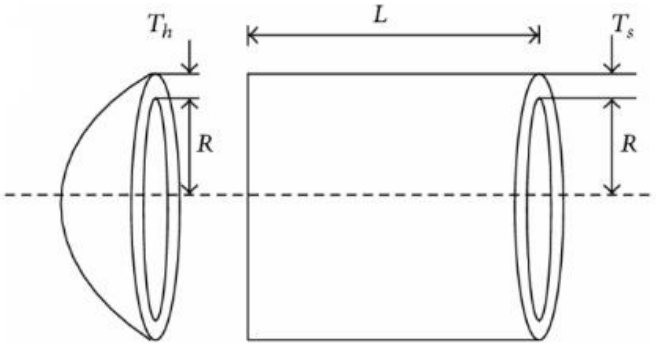
Tablo 3.2'deki Wilcoxon işaretli sıralar testi sonuçlarına göre ASKA 23 kalite testi fonksiyonu üzerinden; SKA'ya karşı 21/2, PSO'ya karşı 16/7, BOA'ya karşı 16/7, KKO'ya karşı 16/7, YAA'ya karşı14/9 üstün gelmektedir.

3.1.3. Kısıtlı Mühendislik Tasarım Problemlerinin Çözümünde ASKA Kullanılması

Önerilen yeni adaptif algoritma, mühendislik tasarım problemlerinden biri olan basınçlı kap probleminin çözümünde kullanılarak algoritmanın kısıtlı problemler üzerindeki etkinliği test edilmiştir. ASKA ve karşılaştırılan diğer algoritmalar (PSO, SKA, BOA, KKO, YAA) basınçlı kap problemi üzerinde test edilirken popülasyon sayısı 200, iterasyon sayısı 10000 olarak belirlenmiştir ve algoritmalar 30 kez çalıştırılmıştır. Elde edilen sonuçlar Tablo 3.3'te, Wilcoxon sıra toplam testi istatiksel sonuçları Tablo 3.4'te gösterilmektedir.

Şekil 3.3'te gösterildiği gibi silindirik bir kabın her iki ucu yarı küresel başlıklarla kapalıdır. Burada amaç; malzeme, şekillendirme ve kaynak maliyeti olmak üzere toplam maliyeti en aza indirmektir. Bu problemde dört tasarım değişkeni vardır: T_s (kabuk kalınlığı, x_1), T_h (başın kalınlığı, x_2), R (iç yarıçap, x_3) ve L (kabın silindirik bölümünün başlık haricindeki uzunluğu, x_4) [49].

Bu problem 4 kısıttan oluşur. Bu kısıtlar ve problem Denklem 3.1'de gösterilmektedir



Şekil 3.3. Basınçlı kap problemi

$$min\ f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$s.t.\quad g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0 \tag{3.1}$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1.296000 \leq 0$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0$$

Değişken aralığı $0 \leq x_1 \leq 99,$

$$0 \leq x_2 \leq 99,$$

$$10 \leq x_3 \leq 200,$$

$$10 \leq x_4 \leq 200$$

Tablo 3.3. Basınçlı kap problemi karşılaştırma sonuçları

Algoritma	Optimum değişkenler				*Optimum Maliyet*	*Süre*
	T_s	T_h	R	L		
ASKA	0.778566726025	0.387863076807	40.322074399104	200	**5897.91316**	**59.53190**
PSO	0.828797029813	0.409674804493	42.942851305476	166.44667183143	5977.62589	517.86163
SKA	0.780408762923	0.386271554329	40.406252027563	200	5920.53556	59.77302
BOA	0.778189825887	0.390165645138	40.319639981412	199.99970408291	5901.42948	87.78191
KKO	0.921103032395	0.455301706112	47.725545735306	117.48048122090	6177.27956	498.07486
YAA	0.896775543832	0.443276615916	46.465054100846	129.12087575297	6120.54193	2907.64910

Tablo 3.4. Basınçlı kap problemi Wilcoxon sıra toplam testi karşılaştırma sonuçları

	PSO / ASKA	SKA / ASKA	BOA / ASKA	KKO / ASKA	YAA / ASKA
p - değeri	0.0056	8.5641e-04	0.3555	0.0020	0.0083

ASKA basınçlı kap problemini **59.53190** saniyede çözerek problemin minimum maliyetini **5897.91316** olarak bulması yapılan iyileştirmenin gücünü kanıtlamaktadır. Çalışma zamanı ve minimum maliyeti bulma açısından ASKA basınçlı kap probleminin çözümünde karşılaştırılan algoritmalar arasında en iyi sonucu vermektedir. Tablo 3.4'te verilen Wilcoxon sıra toplamı testi karşılaştırma sonuçlarına göre de ASKA'nın rakiplerine karşı üstün geldiği açıkça görülmektedir.

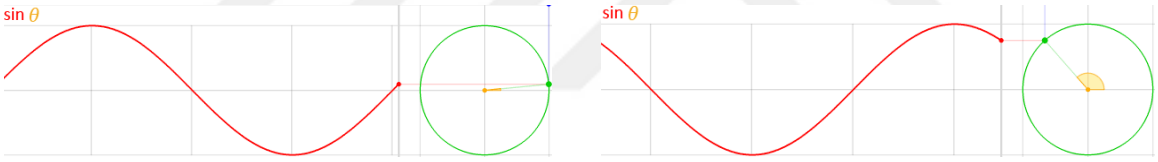
3.2. Sonuç

SKA üzerinde iyileştirmeler yapılarak geliştirilen Adaptif Sinüs Kosinüs Algoritması (ASKA) 23 kalite testi fonksiyonu ve mühendislik tasarım problemlerinden biri olan basınçlı kap probleminin çözümünde test edilmiştir. Test sonuçlarından önerilen yeni algoritmanın SKA'ya ve karşılaştırılan diğer algoritmalara göre daha iyi sonuçlar verdiği görülmektedir. Wilcoxon sıra toplamı testi sonuçları da ASKA'nın diğer algoritmalara karşı üstünlüğünü kanıtlamaktadır.

4. ALTIN SİNÜS ALGORİTMASI: MATEMATİKTEN İLHAM ALAN YENİ BİR METASEZGİSEL ALGORİTMA

Sezgisel yöntemler deterministik yöntemlerden farklı olarak stokastik tabanlı operatörler kullanırlar. Bu operatörlerin oluşturulmasında farklı kaynaklardan (biyoloji, fizik, müzik, spor, matematik vb.) ilham alınmaktadır. Yeni önerilen matematik tabanlı algoritmanın ilham kaynağı sinüs fonksiyonudur [50].

Trigonometrik bir fonksiyon olan sinüs, sin kısaltmasıyla ifade edilir. Merkezi orijin olan 1 birim yarıçaplı çember üzerindeki bir noktanın y eksenine göre koordinatıdır. Orijinden noktaya çizilen bir doğrunun y ekseniyle yaptığı açı kullanılarak ya da aynı açıya sahip bir dik üçgende, bu açının karşısındaki kenarın hipotenüse bölümüyle hesaplanır. Fonksiyonun tanım aralığı [-1, 1]'dir. Sinüs fonksiyonu değerlerini düzenli aralıklarla tekrar eden bir periyodik bir fonksiyondur. Sinüs fonksiyonun periyodu 2π 'dir ve Şekil 4.1'de gösterildiği gibi tüm sinüs değerleri ile birim çemberin tamamı taranabilir.



(a) (b)

Şekil 4.1. Sinüs periyodunun birim çemberde değişimi

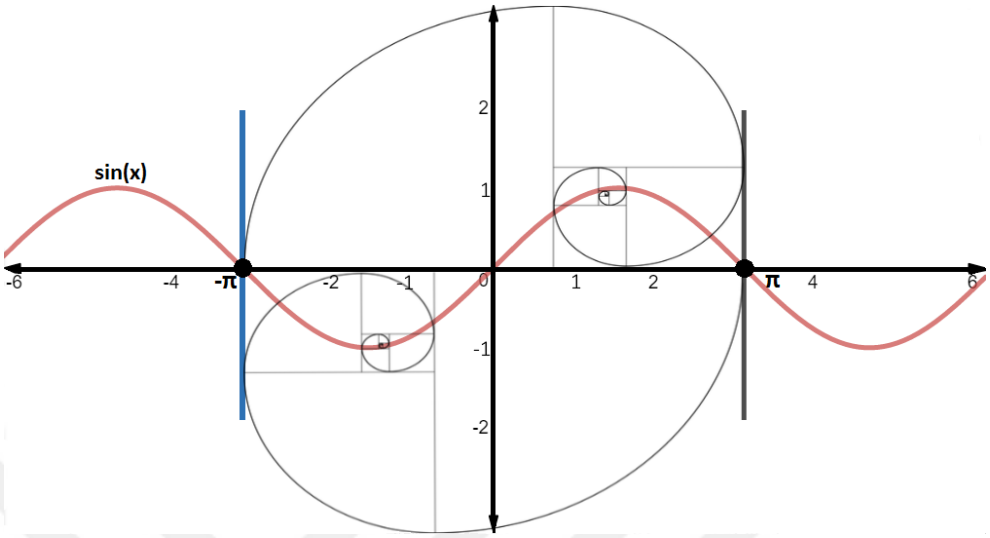
Sinüs fonksiyonun tüm değerlerinin birim çemberini taraması optimizasyon problemlerindeki arama uzayının taranmasına benzemektedir. Bu benzerlik ASA'nın geliştirilmesine ilham kaynağı olmuştur.

Şekil 4.2'de çalışma mekanizması gösterilen algoritmada kullanılan operatör Denklem 4.1'de gösterilmektedir.

$$V(i,j) = V(i,j) * |\sin(r_1)| - p * \sin(r_1) * |x_1 * D(j) - x_2 * V(i,j)| \quad (4.1)$$

Burada $V(i, j)$; i. çözümdeki j. boyutun mevcut değerini göstermektedir. D belirlenen hedef değerdir. $r_1 = [0, 2\pi]$ aralığında rastgele bir sayıdır. $p = [0, \pi]$ aralığında

rastgele bir sayıdır. x_1 ve x_2 ise altın kesit yöntemi ile elde edilen katsayılardır. Bu katsayılar arama uzayını daraltarak mevcut değeri hedef değere yaklaştırmayı sağlamaktadır.



Şekil 4.2. Altın Sinüs Algoritması (ASA)

Diğer popülasyon tabanlı algoritmalarda olduğu gibi ASA da rastgele oluşturulan bir popülasyon ile başlar. Başlangıç popülasyonunun iyi seçilmesi popülasyon tabanlı algoritmalar için oldukça önemlidir. ASA başlangıç popülasyonunu Denklem 4.2'de gösterildiği gibi her bir boyut için düzgün dağılımla rastgele oluşturarak arama uzayının daha iyi taranmasını hedeflemektedir.

$$V = rand(ajan\_no,\ boyut) * (\text{üst sınır} - alt\ sınır) + alt\ sınır \tag{4.2}$$

Sezgisel yöntemlerin temel hedefi arama uzayının en iyi sonuç vereceği düşünülen alanlarını keşfetmek ve bu alanların da mümkün olduğunca tamamının taranmasını sağlamaktır. Arama uzayının geniş olması problemlerin çözümü için büyük bir sorun oluşturmaktadır. Arama uzayının daraltılması işleminin nasıl yapıldığı problemlerin çözümünde sonuçları önemli ölçüde etkilemektedir. ASA bu işlemi en iyi şekilde yapmak için altın kesit yöntemini kullanmaktadır.

Altın kesit arama, tek bir tek modlu fonksiyonun maksimum veya minimum değerini bulmak için kullanılabilecek bir optimizasyon tekniğidir. Adını altın orandan almıştır. Altın oran, matematik ve sanatta, bir bütünün parçaları arasında gözlemlenen, uyum açısından en yetkin boyutları verdiği sanılan geometrik ve sayısal bir oran bağıntısıdır [51].

m ve *n* sayıları arasında altın oran var ise;

$$\frac{m+n}{m} = \frac{m}{n} = \tau \tag{4.3}$$

Denklem 4.3'ten Denklem 4.4 veya Denklem 4.5 yazılabilir.

$$1 + \frac{n}{m} = \tau \tag{4.4}$$

$$1 + \frac{1}{\tau} = \tau \tag{4.5}$$

İkinci dereceden denklem çözümünden Denklem 4.6 elde edilir.

$$\tau = \frac{1-\sqrt{5}}{2} = 0.618033 \ldots (\text{pozitif kök}) \tag{4.6}$$

τ estetikte önemli bir yere sahip olan (örn; Mısır Piramitleri) altın oran sayısıdır. Altın kesit arama yöntemi gradyan bilgisine ihtiyaç duymamaktadır. Bu yöntemin diğerlerine göre iki önemli avantajı vardır:

a) Her adımda sadece bir yeni fonksiyon değerlendirmesi gereklidir.

b) Her adımda sabit bir azalma faktörü vardır.

Altın kesit arama yönteminin [*a, b*] değer aralığındaki sözde kodu Şekil 4.3'te verilmiştir.

Adım 1: *a, b, tolerans*, *τ* değerlerini al

Adım 2: Hesapla

$$x_1 = a * (1 - \tau) + b * \tau$$

$$x_2 = a * \tau + b * (1 - \tau)$$

Adım 3: ***if*** $f(x_1) > f(x_2)$

then $a = x_1, x_1 = x_2$

$$x_2 = a * \tau + b * (1 - \tau)$$

else $a = x_2, x_2 = x_1,$

$$x_1 = a * (1 - \tau) + b * \tau$$

Adım 4: $|f(x_1) - f(x_2)| < tolerans$ oluncaya dek Adım 3'ü tekrarla

Adım5: Yakınsadı. $x^* = \alpha_1, f(x^*) = f(\alpha_1)$ yazdır

Şekil 4.3. Altın kesit yöntemi

ASA'da varsayılan olarak başlangıçta $a = -\pi$ ve $b = \pi$ olarak kabul edilir ve ilk iterasyonda bu iki katsayı mevcut ve hedef değere uygulanır. Ardından hedef değerin değişmesine göre x_1 ve x_2 katsayıları güncellenir. Altın kesit yönteminin ASA'da kullanımı Şekil 4.4'te gösterilmektedir.

Adım 1: ***if*** *mevcut değer < hedef değer*

then $b = x_2$, $x_2 = x_1$

$$x_1 = a * \tau + b * (1 - \tau)$$

else $a = x_1$, $x_1 = x_2$

$$x_2 = a * (1 - \tau) + b * \tau$$

Adım 2: ***if*** $x_1 == x_2$

then *a = random1, b = random2*

$$x_1 = a * \tau + b * (1 - \tau)$$

$$x_2 = a * (1 - \tau) + b * \tau$$

Şekil 4.4. Altın kesit yönteminin ASA'da kullanılması

x_1 ve x_2 değerlerinin eşitliği istenmeyen bir durum olduğundan bu durumun önüne geçmek için Şekil 4.5'te Adım 2'de gösterildiği gibi eşitlik kontrolü yapılır. Eğer iki değer birbirine eşit ise *a*; [0, π] aralığında rastgele bir *random1* ve *b*; [0, -π] aralığında rastgele bir *random2* sayısı seçilerek x_1 ve x_2 değerleri yeniden hesaplanır.

ASA'nın sözde kodu Şekil 4.5'te gösterilmektedir.

1. *Başlangıç popülasyonunu Denklem 2 ile her boyut için arama ajanı sayısı kadar düzgün dağılıma bağlı olarak rastgele oluştur*
2. *Arama ajanlarının uygunluğunu hesapla*
3. *En iyi arama ajanını bul ve hedef değer olarak ata*
4. ***while*** *maksimum iterasyon sayısı*
5. ***for*** *arama ajanı sayısı*
6. $r \leftarrow rand\ (0,1)$
7. $r_1 \leftarrow 2\pi * r$
8. $r_2 \leftarrow \pi * r$
9. ***for*** *boyut sayısı*
10. $V(i, j) \leftarrow V(i, j) * |sin(r_1)| - p * sin(r_1) * |x_1 * D(j) - x_2 * V(i, j)|$
11. ***end for***
12. ***end for***
13. *En iyi çözümü (arama ajanı) bul ve D(j)'ye hedef değer olarak ata*
14. ***if*** $V(i, j) < D(j)$
15. ***then*** $b \leftarrow x_2, x_2 \leftarrow x_1$
16. $x_1 \leftarrow a * \tau + b * (1 - \tau)$
17. ***else*** $a \leftarrow x_1, x_1 \leftarrow x_2$
18. $x_2 \leftarrow a * (1 - \tau) + b * \tau$
19. ***if*** $x_1 == x_2$
20. ***then*** $a \leftarrow rand(0, \pi), b \leftarrow rand(0, -\pi)$
21. $x_1 \leftarrow a * \tau + b * (1 - \tau)$
22. $x_2 \leftarrow a * (1 - \tau) + b * \tau$
23. ***end while***
24. ***return*** *en iyi çözüm kümesi ve elde edien global optimum sonuç*

Şekil 4.5. Altın Sinüs Algoritması'nın sözde kodu

4.1. Deney ve Sonuçlar

ASA'nın performansını değerlendirmek için literatürde yaygın olarak kullanılan 23 kalite testi fonksiyonu üzerinde testler yapılmıştır. Bu fonksiyonlar tek modlu fonksiyonlar, çok modlu fonksiyonlar ve sabit boyutlu çok modlu fonksiyonlar olmak üzere üç farklı

kategoride incelenebilir. ASA, mühendislik tasarım problemlerinden olan basınçlı kap ve germe / sıkıştırma yay problemlerinin çözümünde kullanılarak algoritmanın kısıtlı problemler üzerindeki etkinliği de test edilmiştir.

ASA iyi bilinen sürü tabanlı algoritmalardan Parçacık Sürü Optimizasyonu (PSO), Karınca Koloni Optimizasyon (KKO) Algoritması ve güncel algoritmalar Balina Optimizasyon Algoritması (BOA), Yerçekimi Arama Algoritması (YAA), Sinüs Kosinüs Algoritması (SKA) olmak üzere 5 metasezgisel algoritma ile karşılaştırılmışır. Algoritmaların popülasyon büyüklüğü 30 ve iterasyon sayıları 1000 olarak kabul edilmiştir. Boyutları fonksiyon tanımlarında V_{no} olarak belirtilen her kalite testi fonksiyonu için algoritmalar 30 kez çalıştırılmıştır. Elde edilen ortalama, standart sapma, en iyi sonuç, en kötü sonuç ve ortalama çalışma zamanı istatistiksel sonuçları Tablo 4.1'de gösterilmektedir. Karşılaştırılan algoritmalarda kullanılan parametreler şu şekildedir:

1. PSO: *Atalet ağırlığı* = 1, *Atalet ağırlığı sönüm oranı* = 0.99, *Kişisel öğrenme katsayısı* = 1.5, *Küresel öğrenme katsayısı* = 2.0
2. KKO: *Örnek boyutu* = 40, *Yoğunlaşma faktörü* = 0.5, *Sapma - uzaklık oranı*=1
3. BOA: b = 1
4. YAA: *R_norm* = 2, *R_gücü* = 1, *Elitist kontrolü* = 1
5. SKA: c = 2

4.1.1. Kalite Testi Fonksiyonları Sonuçları

Tablo 4.1. Kalite testi fonksiyonları sonuçları (*ort*: ortalama çözüm, *sd*: standart sapma, *en iyi*: en iyi çözüm, *en kötü*: en kötü çözüm, *süre*: saniye cinsinden ortalama çalışma süresi)

F	I	ASA	PSO	SKA	BOA	KKO	YAA
	ort	0	6.5085e-16	0.0310	1.4109e-154	1.2766	1.0976e-16
	sd	0	1.1800e-15	0.0866	7.0872e-154	0.8429	3.5117e-17
F1	*en iyi*	0	1.0232e-17	2.4792e-06	3.3634e-166	0.4405	6.6897e-17
	en kötü	0	5.4858e-15	0.3945	3.8875e-153	4.4251	2.1405e-16
	süre	1.1532	8.3309	1.2570	4.2849	43.7911	12.0372
	ort	5.4526e-214	2.1071e-05	2.1026e-05	5.0254e-102	3.6260e+03	5.2733e-08
	sd	0	1.0554e-04	4.3726e-05	1.9523e-101	1.9618e+04	1.6126e-08
F2	*en iyi*	0	6.9921e-10	2.9768e-09	1.4491e-114	1.1190	2.7875e-08
	en kötü	1.6358e-212	5.7914e-04	2.1087e-04	9.2622e-101	1.0750e+05	1.0028e-07
	süre	1.2681	8.5741	1.3360	4.5383	46.1645	10.7295
	ort	0	7.6568	4.0282e+03	1.9940e+04	1.0596e+05	432.9601
	sd	0	5.5415	3.2698e+03	1.1221e+04	2.9680e+04	155.3030
F3	*en iyi*	0	1.0370	109.8140	3.6202e+03	4.7676e+04	217.7493
	en kötü	0	22.0031	1.4489e+04	4.3344e+04	1.6262e+05	780.2057
	süre	5.9975	17.1235	6.2230	9.6729	51.2220	14.0386
	ort	1.0387e-271	0.6941	19.9222	40.8308	78.1506	1.5515
	sd	0	0.3446	11.2356	32.4037	9.6495	1.5699
F4	*en iyi*	0	0.2272	1.5395	0.0519	42.8106	1.0044e-08
	en kötü	2.7315e-270	1.3777	42.7265	92.0102	92.1774	5.3571
	süre	1.5333	12.2517	1.6804	5.2079	39.2201	7.3923
	ort	0.0011	41.3337	1.0127e+03	27.2594	5.0124e+04	36.3915
	sd	0.0015	32.4499	2.8445e+03	0.6228	4.5712e+04	53.9666
F5	*en iyi*	6.1189e-08	2.3651	28.3801	26.5891	6.4021e+03	24.5371
	en kötü	0.0050	110.3879	1.2227e+04	28.7495	1.9159e+05	322.0843
	süre	1.6794	11.0633	1.7429	4.7569	38.6504	9.5483
	ort	6.0554e-05	2.7841e-15	4.5788	0.1062	1.0898	0.2000
	sd	1.2696e-04	8.7185e-15	0.5852	0.1165	0.5781	0.7611
F6	*en iyi*	1.5292e-08	1.7238e-17	3.8463	0.0094	0.4107	0
	en kötü	6.6993e-04	4.7394e-14	6.7212	0.4399	3.1241	4.0000
	süre	1.6973	9.5745	1.8441	4.7322	36.7716	10.1557
	ort	7.7789e-05	0.0148	0.0421	0.0024	0.1854	0.0647
	sd	7.4299e-05	0.0059	0.0528	0.0023	0.0757	0.0254
F7	*en iyi*	5.7196e-06	0.0063	0.0047	5.6430e-05	0.0546	0.0094
	en kötü	3.3952e-04	0.0282	0.2775	0.0090	0.4039	0.1131
	süre	1.6257	8.5231	2.3571	6.2950	42.1256	10.7440
	ort	-1.2569e+04	-6.3686e+03	-3.9367e+03	-1.1224e+04	-4.4590e+149	-2.4485e+03
	sd	0.0632	867.5216	258.2417	1.6018e+03	2.4407e+150	425.4308
F8	*en iyi*	-1.2569e+04	-8.0276e+03	-4.5524e+03	-1.2569e+04	-1.3368e+151	-3.5570e+03
	en kötü	-1.2569e+04	-4.3764e+03	-3.5344e+03	-7.8490e+03	-3.0059e+98	-1.7879e+03
	süre	1.3224	8.6910	2.0996	5.0582	40.6682	9.4981
	ort	0	44.7399	23.4859	0	252.9477	27.0629
	sd	0	13.7505	32.5145	0	18.7779	6.2785
F9	*en iyi*	0	23.8790	6.9014e-06	0	193.7615	17.9093
	en kötü	0	81.5864	168.7336	0	276.6462	41.7882
	süre	1.4622	9.8125	1.9521	4.8869	39.8226	13.6292
	ort	8.8818e-16	1.0915	12.6852	4.0856e-15	0.6876	7.8281e-09
	sd	0	0.7594	9.5190	2.3511e-15	0.3804	1.6719e-09
F10	*en iyi*	8.8818e-16	2.7719e-09	2.5556e-04	8.8818e-16	0.1548	5.7326e-09
	en kötü	8.8818e-16	2.3162	20.3227	7.9936e-15	1.7909	1.3408e-08
	süre	1.8509	10.4712	2.1039	5.0623	42.3656	15.1583
	ort	0	0.0240	0.3685	0.0050	0.9188	8.2013
	sd	0	0.0246	0.3339	0.0193	0.0798	3.2014
F11	*en iyi*	0	0	7.5275e-04	0	0.6492	2.6444
	en kötü	0	0.0860	0.9431	0.0875	1.0283	14.4065
	süre	2.005	11.9485	2.2472	5.6325	41.3976	14.9480
F12	*ort*	2.6557e-06	0.2319	3.1533	0.0110	3.2754e+04	0.1608

	sd	7.0487e-06	0.5471	6.2240	0.0177	7.9615e+04	0.2849
	en iyi	1.0482e-10	1.0560e-18	0.3258	0.0012	18.9615	3.5333e-19
	en kötü	3.4562e-05	2.7038	33.4485	0.0956	3.2040e+05	1.4847
	süre	3.8794	16.8464	3.9721	4.5318	42.5764	15.5377
	ort	5.4864e-06	0.1020	18.5548	0.1962	8.1417e+04	0.0033
	sd	8.9948e-06	0.4602	65.8292	0.1581	1.1285e+05	0.0104
F13	*en iyi*	1.4681e-08	5.5146e-18	2.2029	0.0398	961.1513	4.1030e-18
	en kötü	4.0901e-05	2.5085	365.0753	0.7707	3.9818e+05	0.0548
	süre	3.8626	11.5014	3.9894	7.6812	43.5993	13.4597
	ort	1.0311	4.4098	1.5275	2.7961	1.3235	3.4221
	sd	0.1815	2.9700	0.8922	3.2852	1.7829	2.6992
F14	*en iyi*	0.9980	0.9980	0.9980	0.9980	0.9980	0.9980
	en kötü	1.9920	11.7187	2.9821	10.7632	10.7632	13.8192
	süre	9.5544	20.2909	10.3350	10.0869	19.4695	17.3979
	ort	3.6152e-04	3.4190e-04	9.7180e-04	5.6780e-04	0.0011	0.0023
	sd	4.5117e-05	1.6780e-04	3.8543e-04	2.7009e-04	3.1570e-04	8.7184e-04
F15	*en iyi*	3.0868e-04	3.0749e-04	3.4077e-04	3.0836e-04	8.8731e-04	6.2116e-04
	en kötü	5.1038e-04	0.0012	0.0015	0.0015	0.0019	0.0052
	süre	1.7220	9.4237	1.9942	2.0983	8.7483	9.6892
	ort	-1.0303	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	sd	0.0025	6.7752e-16	2.5344e-05	1.6070e-10	6.7752e-16	4.8787e-16
F16	*en iyi*	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	en kötü	-1.0184	-1.0316	-1.0315	-1.0316	-1.0316	-1.0316
	süre	0.8383	9.8322	1.3675	1.4481	5.5078	7.5780
	ort	0.3980	0.3979	0.3985	0.3979	0.3979	0.3979
	sd	1.9567e-04	0	6.0681e-04	3.9088e-06	0	0
F17	*en iyi*	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
	en kötü	0.3988	0.3979	0.4003	0.3979	0.3979	0.3979
	süre	1.1924	7.8293	1.2613	1.3837	5.69707	7.2001
	ort	3.0005	3.0000	3.0000	3.0000	3.0000	3.0000
	sd	5.0420e-04	1.7954e-15	3.2611e-05	9.2119e-05	1.3194e-15	3.1250e-15
F18	*en iyi*	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	en kötü	3.0018	3.0000	3.0001	3.0005	3.0000	3.0000
	süre	0.8128	9.1013	1.3814	1.4617	5.8880	7.2429
	ort	-3.8148	-3.8370	-3.8551	-3.8609	-3.8628	-3.8628
	sd	0.0674	0.1411	0.0024	0.0022	2.7101e-15	2.2913e-15
F19	*en iyi*	-3.8628	-3.8628	-3.8618	-3.8628	-3.8628	-3.8628
	en kötü	-3.6041	-3.0898	-3.8519	-3.8549	-3.8628	-3.8628
	Süre	1.8077	10.1333	2.3266	2.4045	7.7821	9.2076
	ort	-3.0407	-3.2863	-2.8866	-3.2429	-3.2665	-3.3220
	sd	0.2347	0.0554	0.4145	0.1113	0.0603	1.4889e-15
F20	*en iyi*	-3.3011	-3.3220	-3.2333	-3.3219	-3.3220	-3.3220
	en kötü	-2.0149	-3.2031	-1.2291	-2.9868	-3.2031	-3.3220
	süre	1.6953	9.9590	1.8505	3.1532	11.7540	8.5954
	ort	-10.1528	-5.4786	-2.2850	-9.0471	-5.6870	-6.4134
	sd	5.3227e-04	3.4558	1.8892	2.2807	3.7094	3.6846
F21	*en iyi*	-10.1532	-10.1532	-5.9900	-10.1530	-10.1532	-10.1532
	en kötü	-10.1514	-2.6305	-0.4965	-2.6303	-2.6829	-2.6305
	süre	2.3810	11.3522	2.6262	2.7609	11.0864	8.8242
	ort	-10.4016	-6.9147	-3.5487	-7.7342	-6.8594	-10.4029
	sd	0.0028	3.6103	1.6587	2.9214	2.5485	1.4378e-15
F22	*en iyi*	-10.4028	-10.4029	-5.2185	-10.4027	-10.4029	-10.4029
	en kötü	-10.3902	-2.7519	-0.9071	-3.7235	-5.0877	-10.4029
	süre	2.7066	11.5451	3.1160	3.1399	14.0219	9.9985
	ort	-10.5360	-5.9754	-4.0940	-8.5166	-7.2484	-10.2897
	sd	5.9386e-04	3.6315	1.7448	2.9532	3.0040	1.3511
F23	*en iyi*	-10.5363	-10.5363	-7.9749	-10.5363	-10.5363	-10.5363
	en kötü	-10.5336	-2.4217	-0.9460	-2.4217	-2.4217	-3.1359
	süre	3.2553	12.0961	3.5207	3.8171	13.2174	11.2024

Tablo 4.1'deki kalite testi fonksiyonları sonuçları incelendiğinde ASA'nın optimum sonuca karşılaştırılan diğer tüm algoritmalardan (PSO, SKA, BOA, KKO, YAA) çok daha hızlı yakınsadığı açıkça görülmektedir. Ayrıca F1 - F7 tek modlu fonksiyonlarda ve F8 - F13 çok modlu fonksiyonlarda ASA'nın karşılaştırılan algoritmalardan oldukça üstün olduğu dikkat çekmektedir. Önerilen algoritma tek modlu fonksiyonlardan F1, F2, F3 ve F4 fonksiyonlarında; çok modlu fonksiyonlardan F9 ve F11 fonksiyonlarında; sabit boyutlu çok modlu fonksiyonlardan F16, F17, F18, F19, F21, F22 ve F23 fonksiyonlarında optimum sonuca yakınsamaktadır.

4.1.2. Parametrik olmayan istatistiksel analiz sonuçları

İstatiksel anlamlılık değeri $\alpha = 0.05$ olmak üzere ASA ile diğer metasezgisel algoritmalar arasında Wilcoxon işaretli sıralar testi yapılarak istatiksel sonuçlar Tablo 4.2'de gösterilmiştir. Burada $p < 0.05$ olması durumu karşılaştırılan algoritmalardan elde edilen sonuçlar arasında istatiksel açıdan anlamlı bir farkın olduğunu göstermektedir. Ayrıca R^+ ASA'nın karşılaştırılan algoritmaya göre daha üstün sonuçlar elde ettiği rankların toplamını gösterirken, R^- ise karşılaştırılan algoritmanın ASA'ya göre daha üstün sonuçlar elde ettiği rankların toplamını gösterir.

Tablo 4.2. Wilcoxon işaretli sıralar testi karşılaştırma sonuçları

F	PSO / ASA				SKA / ASA				BOA / ASA				KKO / ASA				YAA / ASA			
	p - value	R^+	R^-	K	*p* - value	R^+	R^-	K	*p* - value	R^+	R^-	K	*p* - value	R^+	R^-	K	*p* - value	R^+	R^-	K
F1	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F2	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F3	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F4	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F5	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F6	1.73e-06	0	465	-	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	0.0028	87	378	-
F7	1.73e-06	465	0	+	1.73e-06	465	0	+	1.92e-06	464	1	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F8	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	0	465	-	1.73e-06	465	0	+
F9	1.73e-06	465	0	+	1.73e-06	465	0	+	1	0	0	=	1.73e-06	465	0	+	1.73e-06	465	0	+
F10	1.72e-06	465	0	+	1.73e-06	465	0	+	1.33e-05	253	212	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F11	2.56e-06	435	30	+	1.73e-06	465	0	+	0.5000	3	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F12	0.6435	255	210	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	0.0093	359	106	+
F13	0.3820	275	190	+	1.73e-06	465	0	+	1.73e-06	465	0	+	1.73e-06	465	0	+	0.0571	140	325	-
F14	3.18e-06	459	6	+	1.49e-05	443	22	+	0.0687	321	144	+	3.11e-05	30	435	-	5.75e-06	453	12	+
F15	3.06e-04	57	408	-	3.18e-06	459	6	+	4.07e-05	432	35	+	1.73e-06	465	0	+	1.73e-06	465	0	+
F16	1.73e-06	0	465	-	2.60e-06	4	461	-	1.73e-06	0	465	-	1.73e-06	0	465	-	1.73e-06	0	465	-
F17	1.73e-06	0	465	-	6.15e-04	399	66	+	1.73e-06	0	465	-	1.73e-06	0	465	-	1.73e-06	0	465	-
F18	1.73e-06	0	465	-	1.92e-06	8	457	-	1.73e-06	20	445	-	1.73e-06	0	465	-	1.73e-06	0	465	-
F19	3.11e-05	30	435	-	0.0093	106	359	-	8.18e-05	41	424	-	1.73e-06	0	465	-	1.73e-06	0	465	-
F20	3.51e-06	7	458	-	0.0786	318	147	+	1.02e-05	18	447	-	2.87e-06	5	460	-	1.73e-06	0	465	-
F21	2.61e-04	410	55	+	1.73e-06	465	0	+	5.75e-06	453	12	+	0.0015	387	78	+	0.0087	360	105	+
F22	0.0207	345	120	+	1.73e-06	465	0	+	2.16e-05	439	26	+	0.6733	410	55	+	1.73e-06	0	465	-
F23	6.15e-04	399	66	+	1.73e-06	465	0	+	1.97e-05	440	25	+	0.0148	374	91	+	3.11e-05	30	435	-

Tablo 4.2'de gösterilen Wilcoxon işaretli sıralar testi karşılaştırma sonuçlarına göre 23 fonksiyon üzerinden ASA; PSO'ya karşı 16/7, SKA'ya karşı 20/3, BOA'ya karşı 17/5, KKO'ya karşı 17/6 ve YAA'ya karşı 14/9 daha başarılı olduğu görülmektedir. F15 - F20 sabit boyutlu çok modlu kalite testi fonksiyonlarında karşılaştırılan algoritmalar ASA'dan daha başarılı olmuştur.

4.1.3. Duyarlılık analizi

Kontrol parametreleri üzerinde yapılan değişikliklere algoritmanın en az hassasiyetle cevap vermesi algoritmanın sağlamlığını gösteren önemli bir kriterdir. Önerilen yeni algoritmanın sağlamlığını test etmek amacıyla kontrol parametrelerinden olan başlangıç popülasyonu büyüklüğü üzerinde değişiklik yapılarak duyarlılık analizi yapılmıştır.

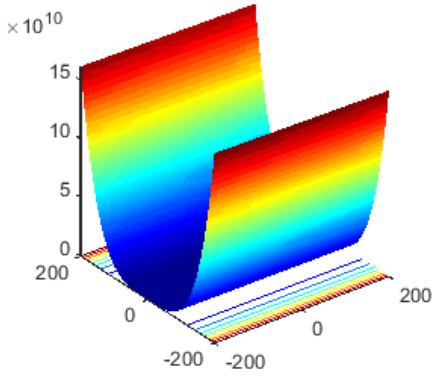
Tablo 4.3'te başlangıç popülasyon büyüklüğünün 10, 30 ve 50 olması durumlarında F5, F9, F13 fonksiyonları üzerinde elde edilen sonuçlar yer almaktadır. Sonuçlar

algoritmanın her bir fonksiyon için 1000 iterasyon ve 30 kez çalıştırılması sonucu elde edilmiştir. F5 fonksiyonu yakınsama grafikleri Şekil 4.6'da, F9 fonksiyonu yakınsama grafikleri Şekil 4.7'de ve F13 fonksiyonu yakınsama grafikleri Şekil 4.8'de gösterilmektedir.

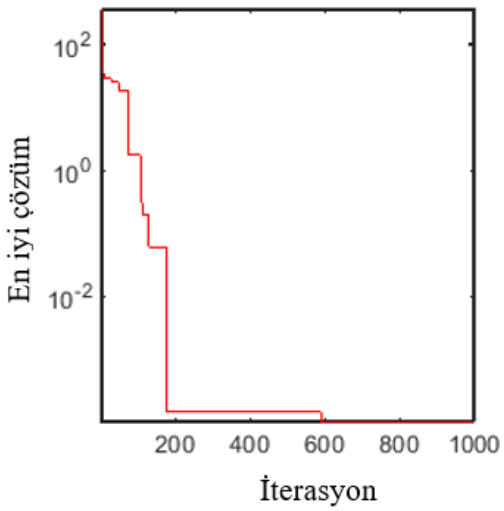
Tablo 4.3. Popülasyon büyüklüğünün değişimine göre elde edilen sonuçlar

F	istatistik	*boyut* = 10	*boyut* = 30	*boyut* = 50
F5	*ort*	0.0101	0.0011	6.7071e-04
	sd	0.0262	0.0021	0.0016
	en iyi	1.8910e-06	1.6302e-07	4.9791e-08
	en kötü	0.1344	0.0089	0.0078
	süre	0.6857	1.5172	2.6075
F9	*ort*	0	0	0
	sd	0	0	0
	en iyi	0	0	0
	en kötü	0	0	0
	süre	0.5947	1.3725	2.2918
F13	*ort*	7.3204e-05	1.4030e-05	4.3964e-06
	sd	1.6166e-04	2.3757e-05	8.7681e-06
	en iyi	3.8238e-08	1.5544e-08	1.8025e-09
	en kötü	7.6928e-04	1.1967e-04	4.4376e-05
	süre	0.5362	3.2876	6.1802

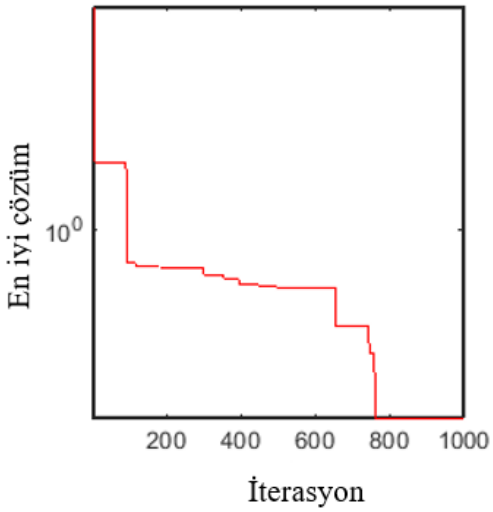
Tablo 4.3'teki istatiksel sonuçlar popülasyon büyüklüğündeki artışın F5, F9 ve F13 fonksiyonlarının ortalama çalışma sürelerinde de artışa neden olduğunu göstermektedir. F5 ve F9 fonksiyonlarında popülasyon boyutunun artırılmasının optimum sonuca yakınsamada az da olsa etkili olduğu, ancak F9 fonksiyonun çözümünü etkilemediği görülmektedir.



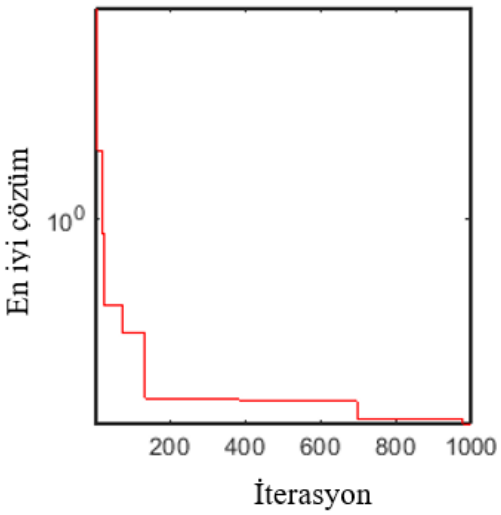
(a) RosenbrockFonksiyonu (F5)



(b) Popülasyon sayısı = 10

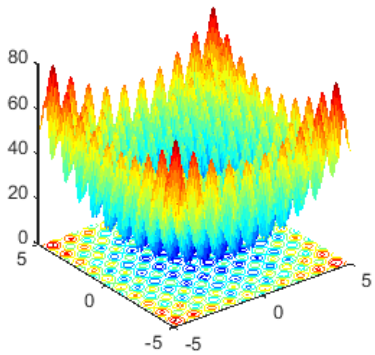


(c) Popülasyon sayısı = 30

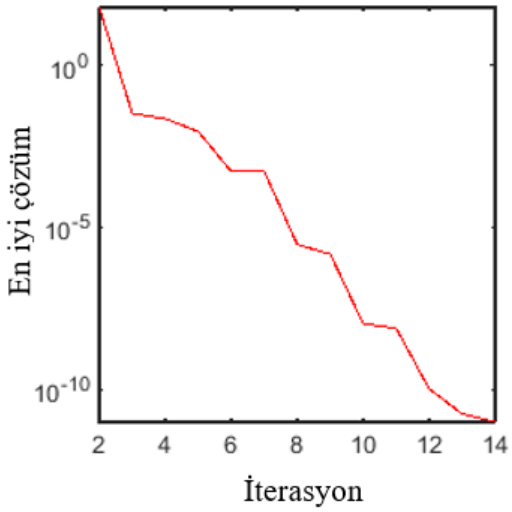


(d) Popülasyon Sayısı = 50

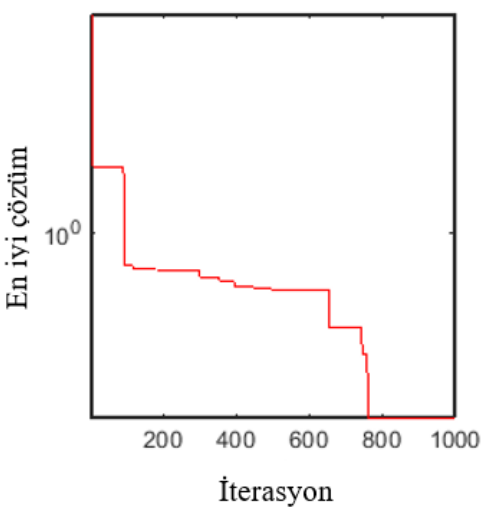
Şekil 4.6. Rosenbrock Fonksiyonu (F5) yakınsama grafikleri



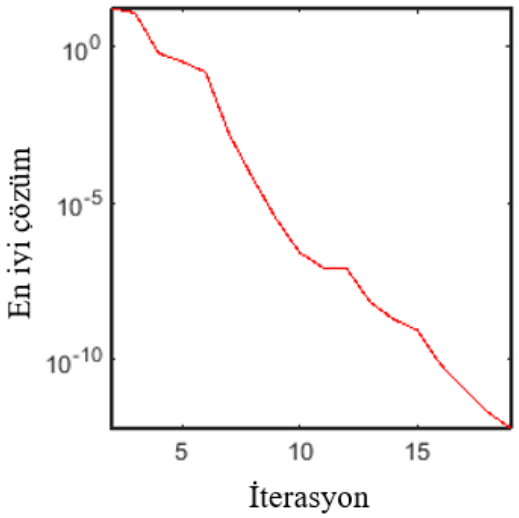
(a) Rastrigin Fonksiyonu (F9)



(b) Popülasyon sayısı = 10

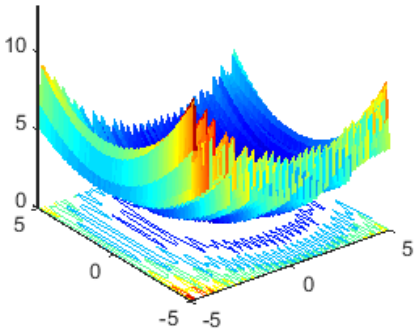


(c) Popülasyon sayısı = 30

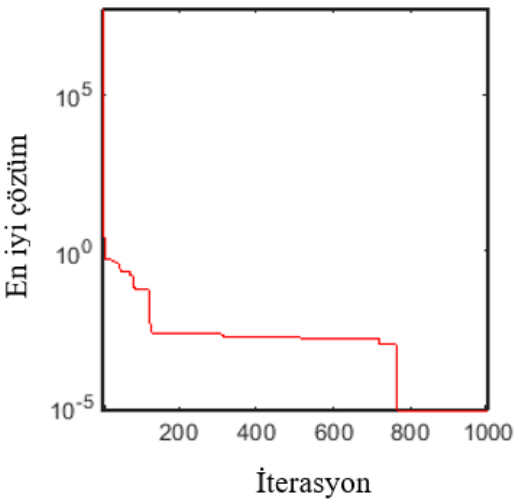


(d) Popülasyon sayısı = 30

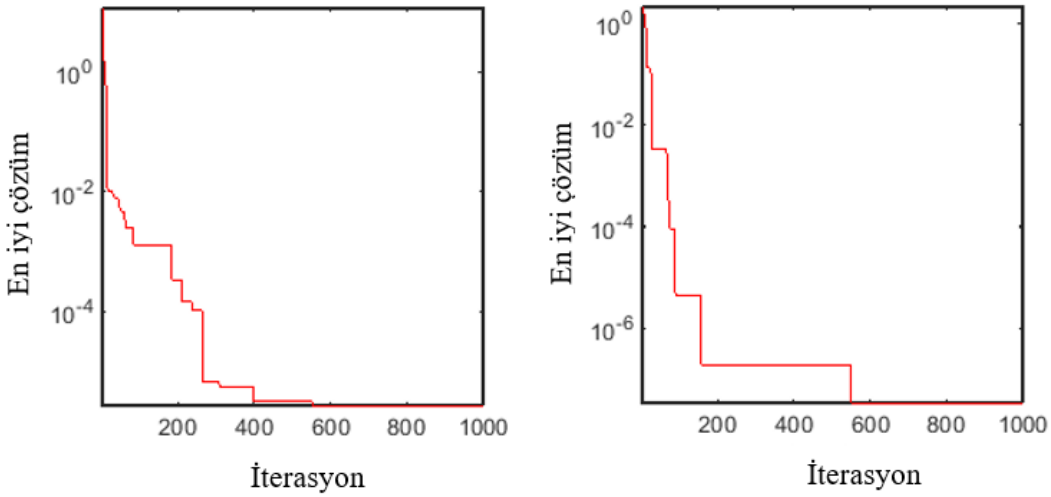
Şekil 4.7. Rastrigin Fonksiyonu (F9) yakınsama grafikleri



(a) Fonksiyon F13



(b) Popülasyon sayısı = 10



(c) Popülasyon sayısı = 30 (d) Popülasyon sayısı = 50

Şekil 4.8. F13 Fonksiyonu yakınsama grafikleri

4.1.4. Kısıtlı Mühendislik Tasarım Problemlerinin Çözümünde ASA Kullanılması

Önerilen yeni algoritma, mühendislik tasarım problemlerinden basınçlı kap problemi ve germe / sıkıştırma yay probleminin çözümünde kullanılarak algoritmanın kısıtlı problemler üzerindeki etkinliği test edilmiştir. ASA ve karşılaştırılan diğer algoritmalar (PSO, SKA, BOA, KKO, YAA) basınçlı kap problemi ve germe / sıkıştırma yay probleminin üzerinde test edilirken popülasyon sayısı 200, iterasyon sayısı 10000 olarak belirlenmiştir ve algoritmalar 30 kez çalıştırılmıştır. Basınçlı kap probleminin çözümünde elde edilen sonuçlar Tablo 4.4'te, germe / sıkıştırma yay problemlerinin çözümünde elde edilen sonuçlar Tablo 4.6'da gösterilmektedir. Basınçlı kap probleminin çözümüne ait Wilcoxon işaretli sıralar testi istatiksel sonuçları Tablo 4.5'te, germe / sıkıştırma yay probleminin çözümüne ait Wilcoxon işaretli sıralar testi istatiksel sonuçları Tablo 4.7'de gösterilmektedir.

Tablo 4.4. Basınçlı kap problemi karşılaştırma sonuçları

Algoritma	Optimum değişkenler				*Optimum Maliyet*	*Süre*
	T_s	T_h	R	L		
ASA	0.779350523979	0.385169061742	40.323163712897	200	**5895.98803**	**58.97245**
PSO	0.828797029813	0.409674804493	42.942851305476	166.44667183143	5977.62589	517.86163
SKA	0.780408762923	0.386271554329	40.406252027563	200	5920.53556	59.77302
BOA	0.778189825887	0.390165645138	40.319639981412	199.99970408291	5901.42948	87.78191
KKO	0.921103032395	0.455301706112	47.725545735306	117.48048122090	6177.27956	498.07486
YAA	0.896775543832	0.443276615916	46.465054100846	129.12087575297	6120.54193	2907.64910

Tablo 4.5. Basınçlı kap problemi Wilcoxon işaretli sıralar testi karşılaştırma sonuçları

	p-value	R^+	R^-	K
PSO / ASA	0.1846	297	168	+
SKA / ASA	0.6288	256	209	+
BOA / ASA	3.8811e-04	405	60	+
KKO / ASA	0.0012	390	75	+
YAA / ASA	0.1650	300	165	+

Tablo 4.6. Germe / Sıkıştırma yay problemi karşılaştırma sonuçları

Algoritma	Optimum değişkenler			*Optimum Maliyet*	*Süre*
	w	*d*	*L*		
ASA	0.051797597638	0.359216890016	11.15048159186	0.0126741	**25.49906**
PSO	0.051580471304	0.354110970914	11.44344489883	**0.0126654**	517.93111
SKA	0.051108490847	0.342889153904	12.15302541048	0.0126762	71.69195
BOA	0.052614959700	0.379403901376	10.07314988814	0.0126806	89.987781
KKO	0.057606226982	0.516451973173	5.739265672904	0.0132638	511.46341
YAA	0.053462849967	0.395370059884	9.564393481895	0.0130686	2463.76132

Tablo 4.7. Wilcoxon işaretli sıralar testi karşılaştırma sonuçları

	p - değeri	R^+	R^-	K
PSO / ASA	0.1254	307	158	+
SKA / ASA	0.0030	377	88	+
BOA / ASA	4.7292e-06	455	10	+
KKO / ASA	1.7344e-06	465	0	+
YAA / ASA	1.7344e-06	465	0	+

ASA'nın basınçlı kap problemini **58.97245** saniye ve germe / sıkıştırma yay problemini **25.49906** saniye gibi oldukça kısa sürelerde çözerek basınçlı kap probleminin minimum maliyetini **5895.98803**, germe / sıkıştırma yay probleminin minimum maliyetini **0.01267** olarak bulması oldukça etkileyicidir. Çalışma zamanı ve minimum maliyeti bulma açısından ASA basınçlı kap probleminin çözümünde karşılaştırılan algoritmalar arasında en iyi sonucu vermektedir. Germe / sıkıştırma yay probleminin çözümünde ise karşılaştırılan algoritmalar arasında en kısa sürede ve PSO'dan sonra en iyi sonucu veren algoritma konumundadır. Tablo 4.5'te ve Tablo 4.7'de verilen Wilcoxon işaretli sıralar testi karşılaştırma sonuçlarına göre de ASA'nın rakiplerine karşı üstün geldiği açıkça görülmektedir.

4.2. Sonuç

Altın Sinüs Algoritması (ASA) matematik tabanlı genel amaçlı yeni bir arama algoritması olarak önerilmektedir. Önerilen algoritma trigonometrik bir fonksiyon olan Sinüs'ten ilham almaktadır. ASA'nın çözüm uzayını altın kesit ile daraltması lokal optimum çözümlerden kaçınarak global optimuma kısa sürede yakınsamasını sağlamaktadır. Geliştirilen yeni algoritma 23 kalite testi fonksiyonunun, mühendislik tasarım problemlerinden olan basınçlı kap problemi ve germe / sıkıştırma yay probleminin çözümünde diğer metasezgisel algoritmalar (PSO, SKA, BOA, KKO, YAA) ile karşılaştırılarak test edilmiştir. Ayrıca stokastik yöntemlerin performansının daha güvenilir bir şekilde belirlenmesini sağlayan istatiksel testlerden Wilcoxon işaretli sıralar testi kullanılarak algoritmalardan elde edilen sonuçlar istatiksel olarak karşılaştırılmıştır. ASA'nın yeni bir algoritma olması, algoritmaya bağımlı az sayıda parametre ve operatör içermesi gelecek çalışmalarda algoritmanın geliştirilmesine ve diğer yöntemlerle hibrit olarak kullanılmasına da olanak sağlamaktadır.

5. KAOTİK HARİTALI ALTIN SİNÜS ALGORİTMALARI

Kaos teorisi, kaotik dinamik sistemlerin çalışmasını ifade eder. Kaotik sistemler, başlangıç koşullarına duyarlı doğrusal olmayan dinamik sistemlerdir. Başlangıç koşullarındaki küçük değişiklikler, sistemin sonucunda yüksek varyasyonlara neden olur. Kaotik sistemler rastgele davranışlar sergilese bile, bir sistemin kaotik davranışlar sergilemesi için mutlaka rastgele olması gerekmez. Başka bir deyişle, deterministik sistemler de kaotik davranışlar gösterebilir. Son zamanlarda, bu özellikler popülasyon tabanlı metasezgisel optimizasyon algoritmalarının performansını arttırmak için kullanılmaktadır [52].

2009'da Alatas ve ark. 12 kaotik haritayı PSO'ya uygulayarak PSO'nun performansının kaos tarafından geliştirilebileceğini kanıtladılar [53]. Alatas ayrıca, kaosun Yapay Arı Koloni Algoritması'nın yanı sıra Armoni Arama Algoritması'nın da performansını geliştirebildiğini gösterdi [53-54]. 2012'de, hızlandırılmış PSO'nun kaosla güçlendirilmiş bir versiyonu da Gandomi ve ark. tarafından önerildi. Kaos ile geliştirilmiş diğer metasezgisel algoritmalardan bazıları Kaotik Genetik Algoritmalar [55], Kaotik Diferansiyel Gelişim [56], Kaotik Benzetilmiş Tavlama [57] ve Kaotik Ateşböceği Algoritması [58], Kaotik Kril Sürü Algoritması [59] ve Kaotik Biyocoğrafya Tabanlı Optimizasyon [60] dur.

2010 ve 2012'de kaotik haritalar Emperyalist Rekabetçi Algoritma'ya (ERA) entegre edilerek kaotik ERA'nın orijinal ERA ve diğer algoritmaların performansını önemli ölçüde artırdığı belirlenmiştir [61]. Kaotik Yarasa Algoritması ve Kaotik Guguk Kuşu Arama Algoritması, 2014, 2015 ve 2016'da önerilen yeni araştırma çalışmalarıdır [62-63]. Bu çalışmaların sonuçları, kaosun metasezgisel algoritmalara başarıyla uygulanabilirliğini teyit etmektedir.

5.1. ASA'da Kullanılan Kaotik Haritalar

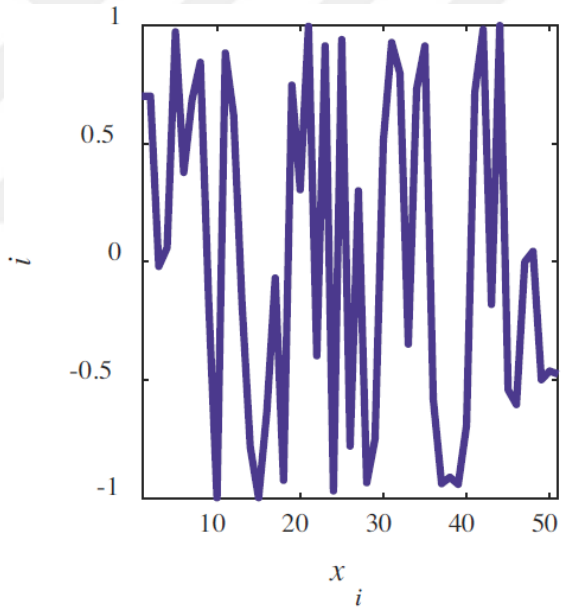
Bu bölümde, kullanılan kaotik haritalar sunulmaktadır. Bölüm 5.2'de ASA'ya entegre edilme yöntemi önerilmektedir. Seçilen kaotik haritalar aşağıda listelenmiştir:

5.1.1. Chebyshev Kaotik Harita

Denklem 5.1'de gösterilen Chebyshev kaotik haritanın grafiği Şekil 5.1'de gösterilmektedir.

$$x_{i+1} = \cos(i\cos^{-1}(x_i)) \tag{5.1}$$

Bu harita, (-1,1) aralığında bulunur.



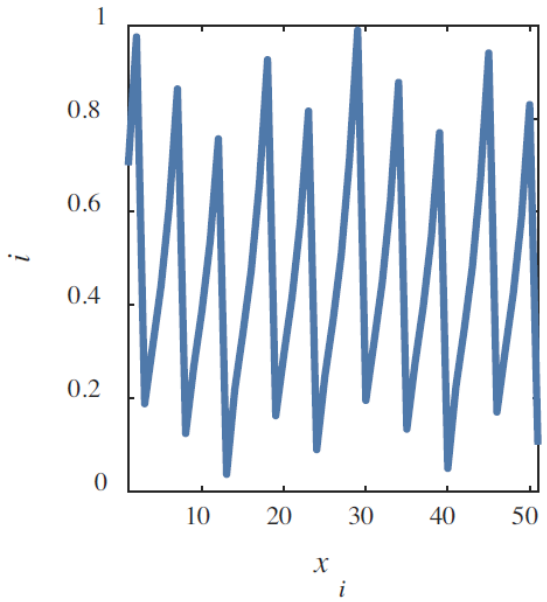
Şekil 5.1. Chebyshev kaotik harita

5.1.2. Çember Kaotik Harita

Denklem 5.2'de gösterilen çember kaotik haritanın grafiği Şekil 5.2'de gösterilmektedir.

$$x_{i+1} = mod\left(x_i + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_i), 1\right), a = 0.5, b = 0.2 \tag{5.2}$$

Bu harita, (0,1) aralığında bulunur.



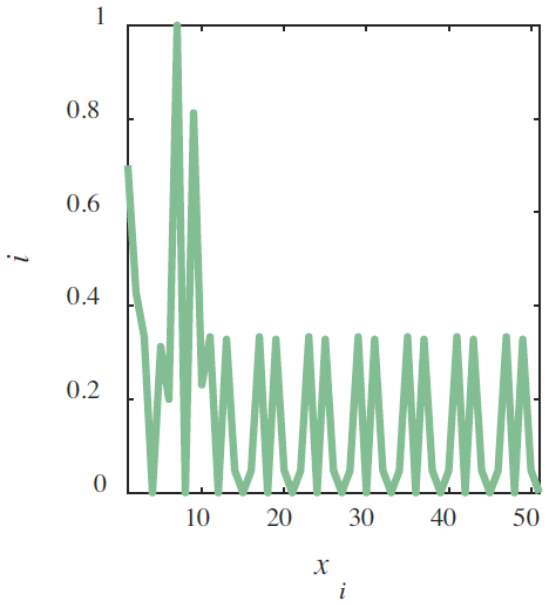
Şekil 5.2. Çember kaotik harita

5.1.3. Gauss/Mouse Kaotik Harita

Denklem 5.3'te gösterilen Gauss/mouse kaotik haritanın grafiği Şekil 5.3'te gösterilmektedir.

$$x_{i+1} = \begin{cases} 1 & x_i = 0, \\ \frac{1}{mod(x_i,1)} & aksi\ takdirde \end{cases} \tag{5.3}$$

Bu harita, (0,1) aralığında bulunur.

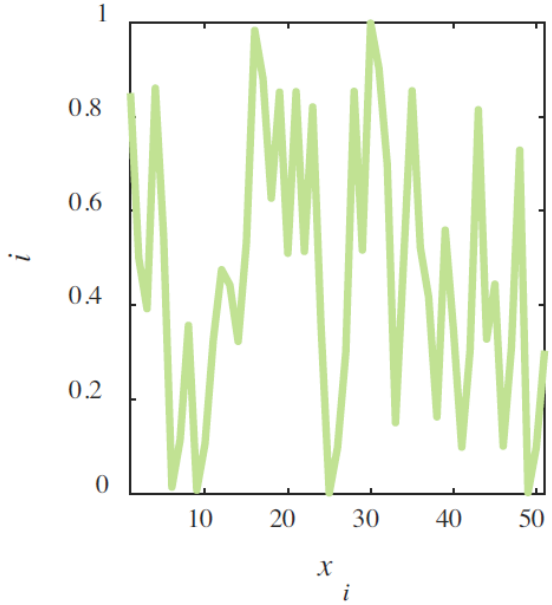


Şekil 5.3. Gauss/mouse kaotik harita

5.1.4. İteratif Kaotik Harita

Denklem 5.4'te gösterilen iteratif kaotik haritanın grafiği Şekil 5.4'te gösterilmektedir.

$$x_{i+1} = \sin\left(\frac{a\pi}{x_i}\right) a = 0.7 \tag{5.4}$$



Şekil 5.4. İteratif kaotik harita

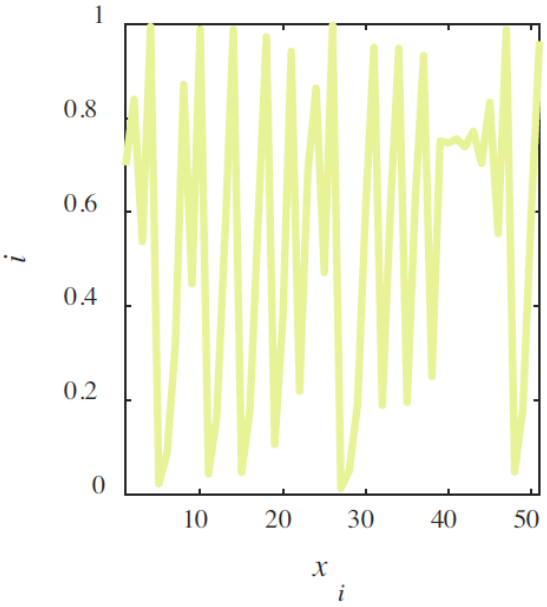
Bu harita, (-1,1) aralığında bulunur.

5.1.5. Lojistik Kaotik Harita

Denklem 5.5'te gösterilen lojistik kaotik haritanın grafiği Şekil 5.5'te gösterilmektedir.

$$x_{i+1} = ax_i(1 - x_i)a = 4, \tag{5.5}$$

Bu harita, (0,1) aralığında bulunur.



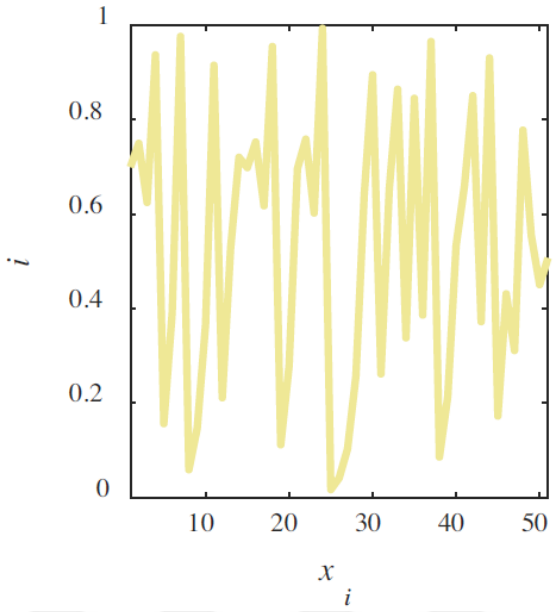
Şekil 5.5. Lojistik kaotik harita

5.1.6. Parçalı Kaotik Harita

Denklem 5.6'da gösterilen parçalı kaotik haritanın grafiği Şekil 5.6'da gösterilmektedir.

$$x_{i+1} = \begin{cases} \frac{x_i}{P} & 0 \le x_i < P \\ \frac{x_i - P}{0.5 - P} & P \le x_i < 0.5, \\ \frac{1 - P - x_i}{0.5 - P} & 0.5 \le x_i < 1 - P, 1 - P \le x_i < 1 \\ \frac{1 - x_i}{P} & \end{cases} \tag{5.6}$$

Bu harita, (0,1) aralığında bulunur.



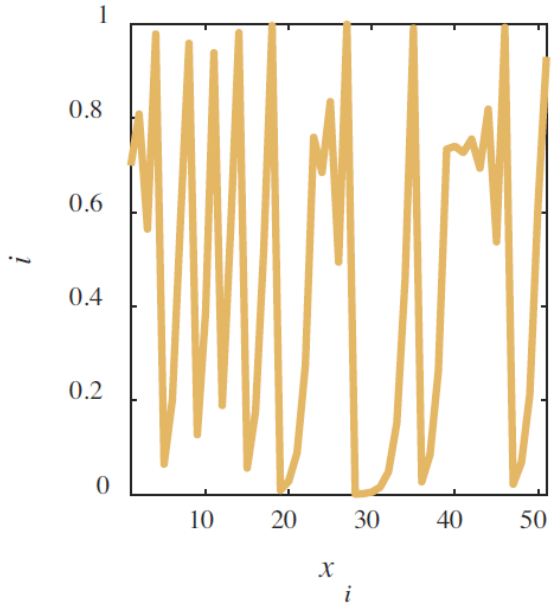
Şekil 5.6. Parçalı kaotik harita

5.1.7. Sinüs Kaotik Harita

Denklem 5.7'de gösterilen Sinüs kaotik haritanın grafiği Şekil 5.7'de gösterilmektedir.

$$x_{i+1} = \frac{a}{4}\sin(\pi x_i)\ \ a = 4, \tag{5.7}$$

Bu harita, (0,1) aralığında bulunur.



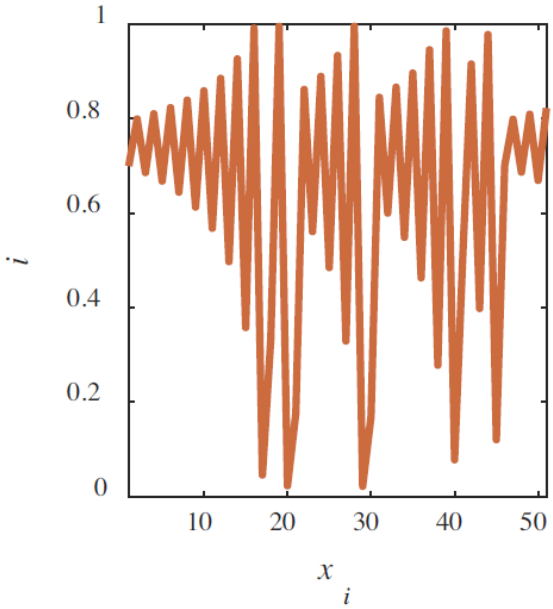
Şekil 5.7. Sinüs kaotik harita

5.1.8. Singer Kaotik Harita

Denklem 5.8'de gösterilen Singer kaotik haritanın grafiği Şekil 5.8'de gösterilmektedir.

$$x_{i+1} = \mu(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4), \mu = 2.3 \quad (5.8)$$

Bu harita, (0,1) aralığında bulunur.



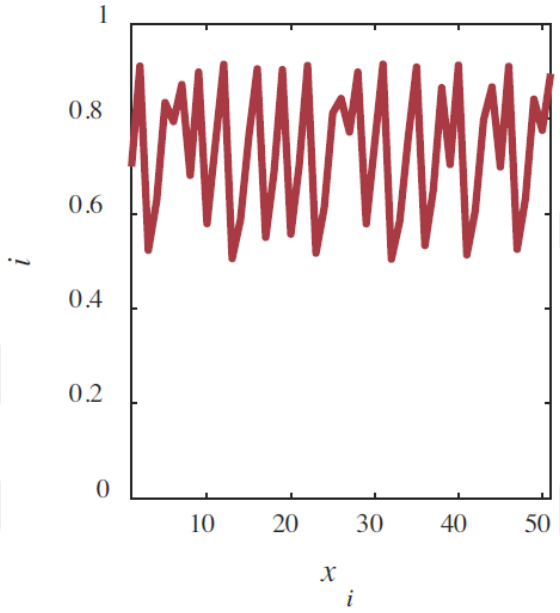
Şekil 5.8. Singer kaotik harita

5.1.9. Sinüzoidal Kaotik Harita

Denklem 5.9'da gösterilen sinüzoidal kaotik haritanın grafiği Şekil 5.9'da gösterilmektedir.

$$x_{i+1} = ax_i^2 \sin(\pi x_i), a = 2.3 \tag{5.9}$$

Bu harita, (0,1) aralığında bulunur.



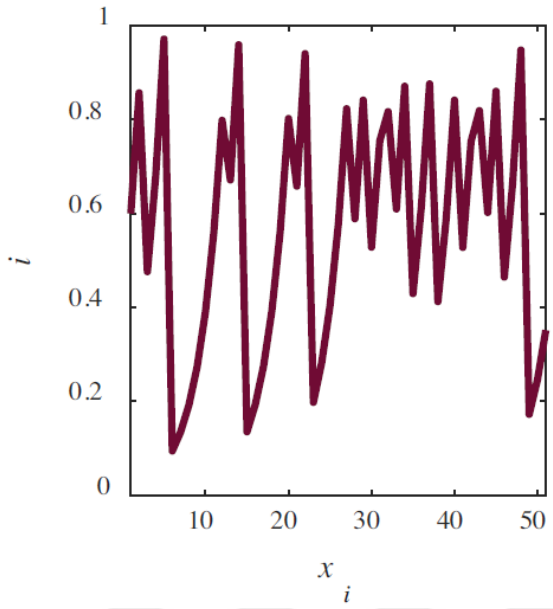
Şekil 5.9. Sinüzoidal kaotik harita

5.1.10. Tent Kaotik Harita

Denklem 5.10'da gösterilen Tent kaotik haritanın grafiği Şekil 5.10'da gösterilmektedir.

$$x_{i+1} = \begin{cases} \frac{x_i}{0.7} x_i < 0.7 \\ \frac{10}{3}(1 - x_i)x_i \geq 0.7 \end{cases} \tag{5.10}$$

Bu harita, (0,1) aralığında bulunur.



Şekil 5.10. Tent kaotik harita

Bütün kaotik haritalar 0.7'den (x_0 = 0.7) başlamaktadır.

5.2. Kaotik Haritaları ASA'ya Uygulama Yöntemi

Altın Sinüs Algoritması'nda p parametresi her arama ajanı için $[0, \pi]$ aralığında rastgele olarak belirlenmektedir. Rastgele üretilen sayılar bazen arama uzayının tamamının sömürülmesini engellemektedir. Kaotik haritalar çoğu zaman rastgele sayılar yerine kullanılarak optimizasyon problemlerinin çözümünde algoritmaların performansını artırmada başarılı sonuçlar vermektedir. Seyedali Mirjalili ve Amir H. Gandomi "Yerçekimi Arama Algoritması için Kaotik Yerçekimi Sabitleri" [52] adlı makalede kaotik haritaları Denklem 5.11 ve Denklem 5.12 ile normalize ederek başarılı sonuçlar elde etmişlerdir. Bu çalışmada normalize edilmiş kaotik haritalara entegre olarak Sinüs Kosinüs Algoritması'nda adaptif olarak azalan Denklem 2.4'te gösterilen r_1 deseni ilk kez kullanılarak arama uzayının mümkün olduğunca tamamının taranması hedeflenmektedir.

t, mevcut iterasyondur. T maksimum iterasyon sayısıdır ve a bir sabittir.

$$V(t) = Max - \frac{t}{T}(Max - Min) \tag{5.11}$$

$Normalize\ (K_i(t);\ [a,b]'dan\ [0,V(t)]'ye)$:

$$K_i^{norm}(t) = \frac{(K_i(t)-a)\times(V(t)-0)}{(b-a)} + 0 = \frac{(K_i(t)-a)\times(V(t))}{(b-a)} \quad (5.12)$$

K: Kaotik harita,

i: Kaotik haritanın indeksi,

t: Mevcut iterasyon,

T: Maksimum iterasyon sayısı,

[*Max*, *Min*]: Uyarlanabilir aralığı temsil eder,

[a, b]: Kaotik haritaların aralığını gösterir.

$V(t)$ her iterasyonda azaltılırken her iterasyonda [a,b], [0,$V(t)$] ile eşleşir. Bu, adaptif normalizasyon sürecini ifade eder.

Sonuç olarak *p* katsayısının değeri Denklem 5.13 kullanılarak güncelleştirilir.

$$p(t) = K_i^{norm}(t) + c - t \times ((c)/T) \quad (5.13)$$

Kaotik haritalar teorik olarak şu nedenlerden dolayı faydalıdır:

- Kaotik haritalar, herhangi bir özel artan veya azalan desen izlememektedir, bu nedenle yinelemeler boyunca *p* için farklı değerler sağlamaktadırlar.
- Kaotik haritalar, *p*'nin değerini aniden değiştirerek yerel minimumdan kurtulmaya yardımcı olur.
- Kaotik haritalar, daha iyi yakınsama hızı sağlamaktadır.
- Uyarlamalı normalleştirme yaklaşımı, kaos tabanlı ASA algoritmalarının arama evresinden sömürü evresine yavaş yavaş geçmesine yardımcı olur.
- *p*'ye kaotik haritalar eklemek, hem adaptif *p*'nin hem de kaotik haritaların rastgele davranışına aynı anda katkıda bulunur.

Bu nedenle on kaotik harita ASA'nın performansını artırmak için kullanılmıştır.

5.3. Deney ve Sonuçlar

Geliştirilen yeni yöntem F5, F12, F13, F19 ve F20 kalite testi fonksiyonlarına uygulanarak sonuçlar Tablo 5.1'de gösterilmektedir. F5 fonksiyonuna ait yakınsama grafiği Şekil 5.11'de, F12 fonksiyonuna ait yakınsama grafiği Şekil 5.12'te, F13 fonksiyonuna ait

yakınsama grafiği Şekil 5.13'te, F19 fonksiyonuna ait yakınsama grafiği Şekil 5.14'te, F20 fonksiyonuna ait yakınsama grafiği Şekil 5.15'da gösterilmektedir.

Tablo 5.1. Kalite testi fonksiyonları sonuçları (*ort*: ortalama çözüm, *sd*: standart sapma, *en iyi*: en iyi çözüm, *en kötü*: en kötü çözüm, *F*:fonksiyon, *İ*: istatistikler)

F	*İ*	F5	F12	F13	F19	F20
ASA	*ort*	7.6412e-04	1.5886e-06	1.9802e-05	-3.7841	-2.9592
	sd	0.0011	3.0044e-06	6.0729e-05	0.0865	0.3052
	en iyi	1.1344e-06	1.0691e-08	7.5857e-08	-3.8627	-3.3036
	en kötü	0.0042	1.3794e-05	3.3367e-04	-3.5980	-1.4908
K-ASA1	*ort*	5.9770e-04	1.8438e-06	1.8587e-05	-3.7973	-2.9281
	sd	0.0017	4.6076e-06	4.9544e-05	0.0678	0.3884
	en iyi	1.3246e-08	**4.5515e-11**	2.4121e-09	-3.8624	-3.3114
	en kötü	0.0090	2.0640e-05	2.1019e-04	-3.6667	-1.6320
K-ASA2	*ort*	7.1666e-04	7.3447e-07	1.1229e-05	-3.8036	-2.9376
	sd	0.0025	1.2626e-06	2.5533e-05	0.0766	0.2701
	en iyi	6.9253e-08	1.9028e-09	2.6391e-09	-3.8626	-3.2262
	en kötü	0.0139	4.3577e-06	1.2592e-04	-3.6089	-1.7379
K-ASA3	*ort*	0.0021	3.5520e-06	2.3832e-05	-3.7973	-2.8948
	sd	0.0074	1.4880e-05	4.5442e-05	0.0745	0.4178
	en iyi	5.6950e-08	3.8111e-09	3.5240e-09	-3.8622	-3.2817
	en kötü	0.0403	8.1773e-05	1.9561e-04	-3.6044	-1.1704
K-ASA4	*ort*	8.4300e-04	8.0683e-07	5.7996e-05	-3.7849	-2.9479
	sd	0.0028	1.3490e-06	1.7730e-04	0.0818	0.3961
	en iyi	4.6070e-08	1.7830e-09	1.0332e-08	-3.8627	-3.2552
	en kötü	0.0145	5.0969e-06	9.5103e-04	-3.6018	-1.4650
K-ASA5	*ort*	0.0023	1.6635e-06	2.7116e-05	-3.7810	-2.9439
	sd	0.0072	4.6690e-06	6.1372e-05	0.0761	0.2922
	en iyi	5.8983e-08	4.6439e-10	**4.7801e-10**	-3.8625	-3.2723
	en kötü	0.0306	2.4306e-05	2.7017e-04	-3.6007	-1.5972
K-ASA6	*ort*	8.9644e-04	1.7741e-06	1.1126e-05	-3.8007	-2.9447
	sd	0.0014	5.0282e-06	1.8043e-05	0.0714	0.3346
	en iyi	2.9675e-08	2.5383e-10	3.1385e-09	**-3.8628**	**-3.3052**
	en kötü	0.0052	2.6333e-05	6.6479e-05	-3.5950	-1.3301
K-ASA7	*ort*	5.2884e-04	1.2585e-06	3.3608e-05	-3.7955	-2.9658
	sd	0.0010	3.6219e-06	6.8010e-05	0.0649	0.1243
	en iyi	8.4680e-08	4.3978e-10	5.7179e-09	-3.8627	-3.1448
	en kötü	0.0052	1.5491e-05	3.1941e-04	-3.6213	-2.6929
K-ASA8	*ort*	1.0084e-04	7.2342e-07	2.0493e-05	-3.7863	-2.9263
	sd	1.7375e-04	1.1624e-06	3.6663e-05	0.0795	0.3267
	en iyi	4.3724e-08	4.7235e-10	4.6726e-09	-3.8614	-3.2193
	en kötü	8.3789e-04	3.8443e-06	1.2470e-04	-3.6099	-1.3824
K-ASA9	*ort*	2.7427e-04	8.4349e-07	1.7158e-05	-3.8101	-2.9757
	sd	9.3238e-04	2.6673e-06	4.4625e-05	0.0575	0.2004
	en iyi	2.8276e-07	4.3278e-10	8.5489e-09	-3.8622	-3.2979
	en kötü	0.0051	1.4665e-05	2.0350e-04	-3.6634	-2.3267
K-ASA10	*ort*	0.0015	1.7372e-06	1.7069e-05	-3.7870	-2.8510
	sd	0.0042	3.6962e-06	3.7362e-05	0.0823	0.4274
	en iyi	**2.2575e-09**	3.0208e-09	4.6515e-08	-3.8623	-3.1764
	en kötü	0.0199	1.3951e-05	1.3582e-04	-3.6016	-1.1698

K-ASA1: Chebyshev kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

K-ASA2: Çember kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

K-ASA3: Gauss/mouse kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

K-ASA4: İteratif kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

K-ASA5: Lojistik kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

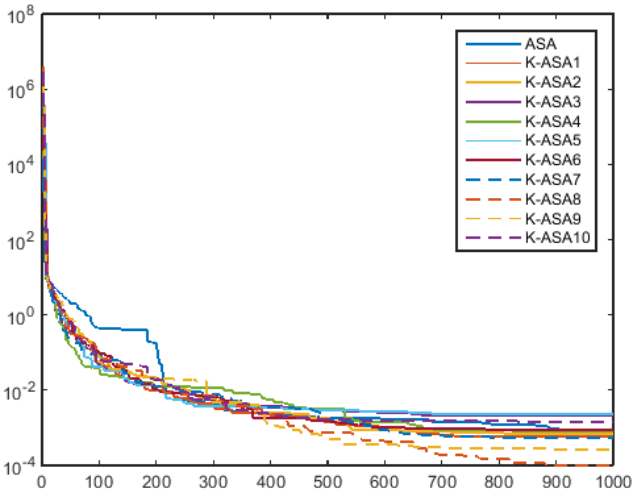
K-ASA6: Parçalı kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

K-ASA7: Sinüs kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

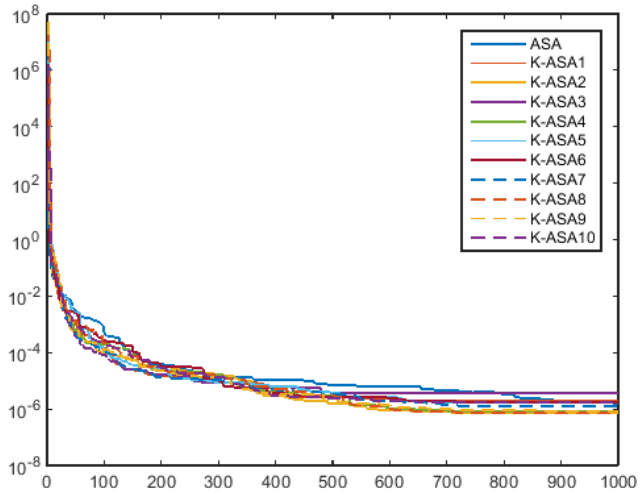
K-ASA8: Singer kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

K-ASA9: Sinüzoidal kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.

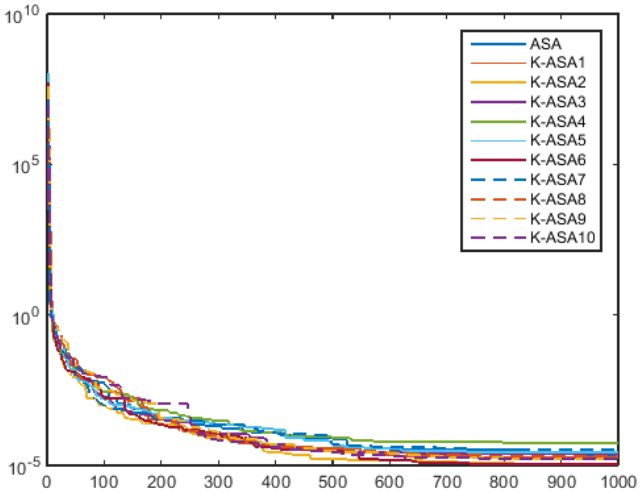
K-ASA10: Tent kaotik harita kullanılarak geliştirilen Kaotik Altın Sinüs Algoritması.



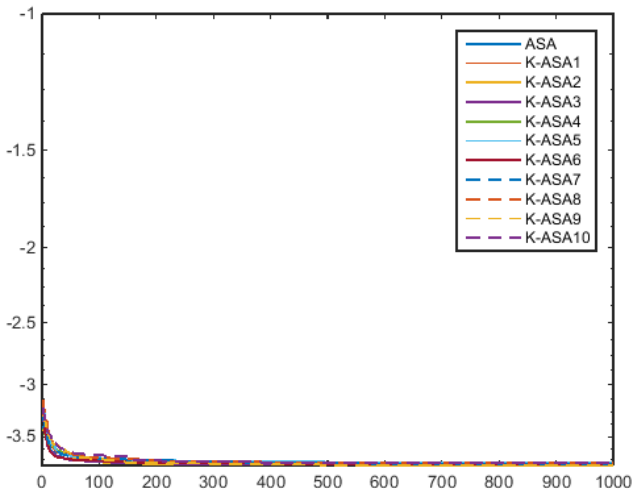
Şekil 5.11. F5 fonksiyonu yakınsama eğrileri



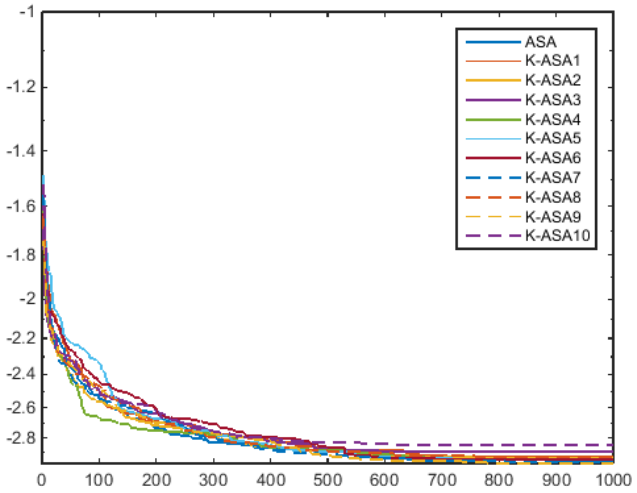
Şekil 5.12. F12 fonksiyonu yakınsama eğrileri



Şekil 5.13. F13 fonksiyonu yakınsama eğrileri



Şekil 5.14. F19 fonksiyonu yakınsama eğrileri



Şekil 5.15. F20 fonksiyonu yakınsama eğrileri

Tablo 5.2. Wilcoxon işaretli sıralar testi karşılaştırma sonuçları

F	İ	F5	F12	F13	F19	F20
ASA / K-ASA1	*p* -değeri	0.1915	0.5999	0.6583	0.5038	0.7343
	R^+	169	207	211	200	216
	R^-	296	258	254	265	249
	K	-	-	-	-	-
ASA / K-ASA2	*p*-değeri	0.0285	0.1109	0.1986	0.2802	0.5999
	R^+	126	155	170	180	258
	R^-	339	310	295	285	207
	K	-	-	-	-	+
ASA / K-ASA3	*p*-değeri	0.4653	0.2989	0.8774	0.5857	0.4779
	R^+	197	182	240	206	267
	R^-	268	283	225	259	198
	K	-	-	+	-	+
ASA / K-ASA4	*p*-değeri	0.0428	0.2289	0.1986	0.8612	0.6583
	R^+	134	174	295	241	211
	R^-	331	291	170	224	254
	K	-	-	+	+	-
ASA / K-ASA5	*p*-değeri	0.6143	0.1650	0.7189	0.7499	0.5577
	R^+	208	165	250	248	261
	R^-	257	300	215	217	204
	K	-	-	+	+	+
ASA / K-ASA6	*p*-değeri	0.9918	0.1064	0.4284	0.6143	0.7655
	R^+	233	154	194	208	247
	R^-	232	311	271	257	218
	K	+	-	-	-	+
ASA / K-ASA7	*p*-değeri	0.5038	0.1306	0.5440	0.9918	0.6143
	R^+	200	159	262	233	257
	R^-	265	306	203	232	208
	K	-	-	+	+	-
ASA / K-ASA8	*p*-değeri	0.0068	0.0428	0.7971	0.9426	0.4165
	R^+	101	134	245	236	272
	R^-	364	331	220	229	193
	K	-	-	+	+	+
ASA / K-ASA9	*p*-değeri	0.0148	0.0196	0.3709	0.0519	0.7971
	R^+	114	119	189	138	220
	R^-	351	346	276	327	245
	K	-	-	-	-	-
ASA / K-ASA10	*p*-değeri	0.3493	0.4284	0.4779	0.7499	0.1306
	R^+	187	194	198	248	306
	R^-	278	271	267	217	159
	K	-	-	-	+	+

Tablo 5.1'deki test sonuçlarından F5 fonksiyonunun çözümünde K-ASA10 algoritmasının, F12 fonksiyonunun çözümünde K-ASA1 algoritmasının, F13 fonksiyonunun çözümünde K-ASA5 algoritmasının, F19 ve F20 fonksiyonlarının çözümünde K-ASA6 algoritmasının optimum noktaya en yakın sonuçlar verdikleri görülmektedir.

Tablo 5.2'deki Wilcoxon işaretli sıralar testi karşılaştırma sonuçlarına göre K-ASA1 ASA'ya 5/0, K-ASA2 ASA'ya 4/1, K-ASA3 ASA'ya 3/2, K-ASA4 ASA'ya 3/2, K-ASA6 ASA'ya 3/2, K-ASA7 ASA'ya 3/2, K-ASA9 ASA'ya 5/0, K-ASA10 ASA'ya 3/2 üstünlük sağlamaktadır. ASA K-ASA5'e 3/2, K-ASA8'e 3/2 üstünlük sağlamaktadır.

5.4. Sonuç

Bu bölümde ASA'nın performansını iyileştirmek için on kaotik harita kullanılmıştır. Kalite testi fonksiyonları üzerinde testler yapılarak sonuçlar Wilcoxon işaretli sıralar testi ile desteklenmiştir. Sonuçlar, kaotik haritaların ASA'nın hem arama hem de sömürme evrelerini iyileştirme kabiliyetine sahip olduğunu kanıtlamaktadır. Ayrıca sonuçlar, sinüzoidal haritanın en iyi harita olduğunu göstermektedir.

Kaotik haritalar *p* değerini aniden değiştirdiğinden keşif aşamasını güçlendirir ve yerel minimuma sıkışmaktan kurtulmaya yardımcı olur. Kaotik haritalar, önerilen yeni yaklaşımla keşif ve sömürünün adaptif olarak dengelenmesine izin verir. Başka bir deyişle, önerilen yöntem K-ASA'nın keşif evresinden sömürü evresine aşamalı olarak geçmesine yardımcı olur.

Gelecek çalışmalarda, gerçek dünya mühendislik problemlerinin çözümü için K-ASA algoritmalarının kullanılması amaçlanmaktadır. Ayrıca, diğer kaotik haritaların da ASA'ya uygulanması hedeflenmektedir.

6. SONUÇ

Bu çalışmada; optimizasyon problemlerinin çözümü için yeni bir popülasyon tabanlı optimizasyon algoritması olan Sinüs Kosinüs Algoritması (SKA) tanıtılmaktadır. SKA, sinüs ve kosinüs fonksiyonlarına dayalı bir matematiksel model ile içe veya dışa doğru dalgalanma yaparak en iyi çözümü bulmaya çalışır. Birkaç rastgele ve adaptif değişken de arama uzayının keşfini ve sömürülmesini garanti etmek için algoritmaya entegre edilmiştir.

Sürekli iyileşme ve daima daha iyiyi arama felsefesi altında SKA üzerinde iyileştirmeler yapılarak Adaptif Sinüs Kosinüs Algoritması (ASKA) geliştirilmiştir. Ayrıca bu tez çalışması kapsamında optimizasyon problemlerinin çözümü için matematik tabanlı yeni bir metasezgisel yöntem olan Altın Sinüs Algoritması (ASA) önerilmektedir. ASA popülasyon tabanlı yeni bir arama algoritması olarak geliştirilmiştir. Geliştirilen bu algoritmanın ilham kaynağı trigonometrik bir fonksiyon olan sinüstür. Çözüm uzayı altın kesit ile daraltılarak tüm çözüm uzayının taranması yerine sadece iyi sonuç vereceği düşünülen alanlar taranarak optimum sonuca en hızlı şekilde ulaşılması hedeflenmiştir ve test sonuçlarından ASA'nın karşılaştırılan algoritmalara karşı üstün geldiği görülmüştür. ASA'nın da yine bu çalışma kapsamında ilk kez önerilen normalize ve hibrit yapıdaki kaotik harita yöntemi ile iyileştirilmesi sağlanmıştır.

SKA'nın ve geliştirilen yeni algoritmaların performansını değerlendirmek için literatürde yaygın olarak kullanılan kalite testi fonksiyonları üzerinde testler yapılmıştır. Algoritmalar, kısıtlı mühendislik tasarım problemlerinin çözümünde kullanılarak algoritmaların kısıtlı problemler üzerindeki etkinliği de test edilmiştir.

KAYNAKLAR

[1] Ebrahimi, A., Khamehchi, E., 2016. Sperm whale algorithm: An effective metaheuristic algorithm for production optimization problems. *Journal of Natural Gas Science and Engineering*, 29, 211-222.

[2] Li, M. D., Zhao, H., Weng, X. W., Han, T., 2016. A novel nature-inspired algorithm for optimization: virüs colony search. Advances in Engineering Software, 92, 65-88.

[3] Prakasam, A., Savarimuthu, N.,2015. Metaheuristic algorithms and polynomial turing reductions: a case study based on ant colony optimization. Procedia Computer Science 46:388 – 395.

[4] Mirjalili S, Mirjalili S. M., Lewis A.,2014. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61.

[5] Demir, G., Alataş B., 2016. Lig şampiyonası algoritması ile gezgin satıcı probleminin çözümü. International Conference on Engineering Technology and Applied Sciences (ICETAS), 1:793-800.

[6] Fister, I.J., Yang, X.S., Fister, I., Brest J., Fister, D., 2013. A brief review of nature-inspired algorithms for optimization. Elektrotehniski Vestnik 80(3):1-7.

[7] Holland, J.H., 1975. Adaptation in natural and artificial systems: an ıntroductory analysis with applications to biology, control, and artificial ıntelligence. U Michigan Press.

[8] Storn, R., Price, K., 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4):341–359.

[9] Simon, D., 2008. Biogeography-based optimization. Evolutionary Computation, IEEE Transactions on, 12(6):702–713.

[10] Shi, Y., 2011 An optimization algorithm based on brainstorming process. International Journal of Swarm Intelligence Research (IJSIR), 2(4):35–62.

[11] Kaveh, A., Farhoudi, N., 2013. A. new optimization method: Dolphin echolocation. Advances in Engineering Software, 59:53–70.

[12] Parpinelli, R.S, Lopes, H.S., 2011. An eco-inspired evolutionary algorithm applied to numerical optimization. In Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on, pages 466–471.

[13] Yang, X.S., 2012. Flower pollination algorithm for global optimization. Unconventional Computation and Natural Computation, pages 240–249.

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

[15] Hedayatzadeh, R., Salmassi, F.A., Keshtgari, M., Akbari, R., Ziarati, K., 2010. Termite colony optimization: A novel approach for optimizing continuous problems. Electrical Engineering (ICEE), 2010 18th Iranian Conference.

[16] Yan, G.W., 2013 A novel optimization algorithm based on atmosphere clouds model. International Journal of Computational Intelligence and Applications, 12(1).

[17] Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In Neural Networks, 1995. Proceedings., IEEE International Conference on, volume 4, pages 1942–1948.

[18] Dorigo, M., 1992. Optimization, learning and natural algorithms. Ph. D. Thesis, Politecnico di Milano, Italy.

[19] Kevin, M.P., 2002. Biomimicry of bacterial foraging for distributed optimization and control. Control Systems, IEEE, 22(3):52–67.

[20] Yang, X.S., 2010. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 65–74.

[21] Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization, 39(3):459–471.

[22] Tang, R, Fong, S., Yang, X.S., Deb, S., 2012. Wolf search algorithm with ephemeral memory. In Digital Information Management (ICDIM), 2012 Seventh International Conference on, pages 165–172..

[23] Yang, X.S., Deb, S., 2009. Cuckoo search via l´evy flights. In Nature & Biologically Inspired Computing, 2009. World Congress on, pages 210–214.

[24] Yang, X.S., 2010. Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation, 2(2):78–84.

[25] Mirjalili, S., 2015. The ant lion optimizer. Adv Eng Softw. 83:80-8.

[26] Mirjalili, S., Mirjalili S.M., Lewis, A., 2014. Grey wolf optimizer. Adv Eng Softw. 69:46-61.

[27] Mirjalili, S, Mirjalili, S.M., 2016. The whale optimization algorithm. Adv Eng Softw. 95:51-67.

[28] Zandi, Z., Afjei, E., Sedighizadeh, M., 2012. Reactive power dispatch using big bang-big crunch optimization algorithm for voltage stability enhancement. In Power and Energy (PECon), 2012 IEEE International Conference on, pages 239–244. IEEE.

[29] Abdolreza, H., 2012. Black hole: A new heuristic optimization approach for data clustering. Information Sciences.

[30] Formato, R.A., 2007. Central force optimization: A new metaheuristic with applications in applied electromagnetics. Progress In Electromagnetics Research, 77:425–491.

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

[32] Cuevas, E., Oliva, D., Zaldivar, D., Cisneros, M.P., Sossa, H., 2012. Circle detection using electromagnetism optimization. Information Sciences, 182(1):40–55.

[33] Rashedi, E., Pour, H.N., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Information sciences, 179(13):2232–2248.

[34] Geem, Z.W., Kim J.H., Loganathan, G.V., 2001 A new heuristic optimization algorithm: Harmony search. Simulation, 76(2):60–68.

[35] Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M., 2012. Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers & Structures.

[36] Benasla, L., Belmadani, A., Rahli, M., 2014. Spiral optimization algorithm for solving combined economic and emission dispatch. International Journal of Electrical Power & Energy Systems, 62, 163–174.

[37] Shayeghi, H., Dadashpour, J., 2012. Anarchic society optimization based pid control of an automatic voltage regulator (avr) system. Electrical and Electronic Engineering, 2(4):199–207.

[38] Gargari, E.A., Lucas, C., 2007. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, pages 4661–4667.

[39] Ramezani, S., Lotfi, S., 2013. Social-Based Algorithm. Applied Soft Computing;13:2837–2856.

[40] Kashan, A.H., 2009. League championship algorithm: a new algorithm for numerical function optimization. In Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of, pages 43–48. IEEE.

[41] Moosavian, N., Roodsari, B.K., 2014. Soccer league competition algorithm, a new method for solving systems of nonlinear equations. Int. J. Intell. Sci. 4(1), 7-16.

[42] Osaba, E., Diaz, F., Onieva, E., 2014. Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. Applied Intelligence.

[43] Salem, S.A., 2012. BOA: A novel optimization algorithm. In International Conference on Engineering and Technology (ICET) (pp. 1–5). Egypt: IEEE.

[44] Mirjalili, S., 2016. SCA: A sine cosine algorithm for solving optimization problems. Knowledge-Based Systems; 96:120–133.

[45] Altunbey, F., Alataş, B., 2015. Sosyal ağ analizi için sosyal tabanlı yapay zekâ optimizasyon algoritmalarının incelenmesi. Int. J. Pure Appl. Sci. 1: 33-52.

[46] Demir, G., Tanyıldızı, E., 2017. Optimizasyon problemlerinin çözümünde sinüs kosinüs algoritması (SKA)'nın kullanılması. Fırat Üniv. Müh. Bil. Dergisi. 29(1), 225-236.

[47] Derrac, J., García, S., Molina, D., Herrera, F., 2011 A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary andswarm intelligence algorithms, Swarm Evol. Comput. 1:3–18.

[48] Coello, C.A.C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng; 191:1245–87.

[49] Nasseri, S.H., Alizadeh, Z., Taleshian, F., 2012. Optimized solution of pressure vessel design using geometric programming. The Journal of Mathematics and Computer Science Vol. 4 No. 3:344 – 349.

[50] Tanyıldızı, E., Demir, G., 2017. Golden sine algorithm: a novel math-ınspired algorithm. Advances in Electrical and Computer Engineering Vol 17. No:2, 71-78.

[51] Arora, R.K., 2015. Optimization Algorithms and Applications. ISBN-13: 978-1-4987-2115-8. 46-47.

[52] Mirjalili, S., Gandomi, A. H., 2017. Chaotic gravitational constants for the gravitational search algorithm. Vol.53 s.407-419.

[53] Alatas, B., Akin, E., Ozer, A. B., 2009. Chaos embedded particle swarm optimization algorithms. Vol. 40 No.4 s. 1715-1734.

[54] Alatas, B., 2010. Chaotic bee colony algorithms for global numerical optimization. Vol. 37. No:8. s. 5682-5687.

[55] Jun-feng, Y., Chi, M., Xiao-qi, P., Zhi-kun, H., Jun, H. A new optimization approach-chaosgenetic algorithm, Syst. Eng. 1 (2001) 105.

[56] Zhenyu, G., Bo, C., Min, Y., Binggang, C., 2006. Self-adaptive chaos differential evolution, Adv. Nat. Comput., 972–975.

[57] Mingjun, J., Huanwen, T., 2004. Application of chaos in simulated annealing, Chaos Solitons Fractals 21, 933–941.

[58] Gandomi, A.H., Yang X.S., Talatahari S., Alavi A.H., 2012. Firefly algorithm with chaos,Commun. Nonlin. Sci. Numer. Simulat.

[59] Wang, G.G., Guo, L., Gandomi, A.H., Hao, G.S., Wang, H., 2014. Chaotic krill herd algorithm, Information Sciences 274, 17–34.

[60] Saremi, S., Mirjalili, S., Lewis, A., 2014. Biogeography-based optimisation with chaos, Neural Comput. Appl. 25, 1077–1097.

[61] Talatahari, S., Azar, B.F., Sheikholeslami, R., Gandomi, A.H., 2012. Imperialist competitive algorithm combined with chaos for global optimization, Commun. Nonlin. Sci.Numer. Simulat. 17, 1312–1319.

[62] Gandomi, A.H., Yang, X.S., 2014. Chaotic bat algorithm, J. Comput. Sci. 5, 224–232.

[63] Wang, G.G., Deb, S., Gandomi, A.H., Zhang, Z., Alavi, A.H., 2016. Chaotic cuckoo search,Soft Comput. 20, 3349–3362.

ÖZGEÇMİŞ

1991 Elazığ doğumlu Gökhan Demir ilk ve orta öğrenimini Elazığ'da tamamladı. 2011 yılında başladığı Fırat Üniversitesi Yazılım Mühendisliği Bölümü'nden 2015 yılında mezun oldu. Aynı yıl eğitim hayatına ara vermeden Fırat Üniversitesi Fen Bilimleri Enstitüsü Yazılım Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başladı. 2017 yılı Nisan ayında TEDAŞ Bilgi Teknolojileri Daire Başkanlığı'na yazılım geliştirici (mühendis) olarak atandı ve halen görevine devam etmektedir.