POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

Küresel optimizasyon problemlerinin çözümü için zamanla değişen rastgele atalet ağırlıklı jaya algoritması

Time-varying random inertia weighted jaya algorithm for the solution of global optimization

***Yazar(lar) (Author(s))**: Mehmet Fatih TEFEK*

ORCID: 0000-0003-3390-4201

Bu makaleye şu şekilde atıfta bulunabilirsiniz(To cite to this article): Tefek M. F., "Küresel optimizasyon problemlerinin çözümü için zamanla değişen rastgele atalet ağırlıklı jaya algoritması", ***Politeknik Dergisi***, 25(1): 123-135, (2022).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: **10.2339/politeknik.745819**

Küresel Optimizasyon Problemlerinin Çözümü İçin Zamanla Değişen Rastgele Atalet Ağırlıklı Jaya Algoritması

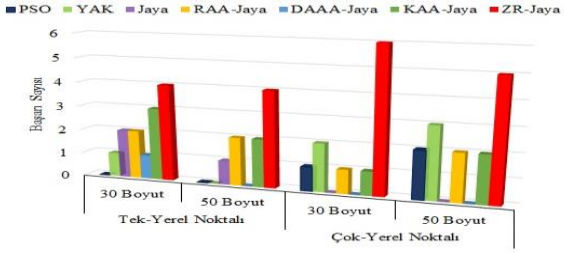
Time-varying Random Inertia Weighted Jaya Algorithm for the Solution of Global Optimization Problems

Önemli noktalar (Highlights)

- *Yeni bir atalet ağırlığı strateji önerisi / Proposing a new inertia weight strategy.*
- *Zamanla değişen rastgele atalet ağırlıklı Jaya (ZR-Jaya) algoritması geliştirilmesi / Time-varying random inertia-weighted Jaya (ZR-Jaya) algorithm proposed.*
- *Küresel optimizasyon problemlerinin çözümü / Solution of global optimization problems*

Grafik Özet (Graphical Abstract)

Geliştirilen ZR-Jaya yönteminin diğer algoritmalardan tek-yerel noktalı fonksiyonlarda toplam başarı sayısı oranı %75, çok-yerel noktalı fonksiyonlarda ise %61,11 olduğu görülmektedir. / It is seen that the proposed ZR-Jaya from other algorithms has a total success rate of 75% in unimodal functions and 61.11% in multimodal functions.



Şekil. Geliştirilen ZR-Jaya ile PSO, YAK ve Jaya'nın başarı sıralama sayıları **/Figure.** Success ranking points of the proposed ZR-Jaya and PSO, YAK and Jaya

Amaç (Aim)

Geliştirilen ZR-Jaya'daki atalet ağırlığının amacı, algoritmanın hesaplama maliyetinin azalması için en uygun bireylerin seçimini erken iterasyonlarda yapmaktır. / The purpose of the inertia weight in the proposed ZR-Jaya is to select the most suitable individuals in early iterations to reduce the calculation cost of the algorithm.

Tasarım ve Yöntem (Design & Methodology)

Jaya algoritmasının popülasyon güncelleme parametresine yeni bir atalet ağırlığı eklenmesi ve küresel optimizasyon problemlerinin çözümü. / Adding a new inertia weight to the population update procedure of the Jaya algorithm and solution of global optimization problems.

Özgünlük (Originality)

Zamanla değişen rastgele atalet ağırlığının tasarımı ve ilk kez Jaya algoritmasına uyarlanması. / Time-varying random inertia weight design and adaptation to Jaya algorithm for the first time

Bulgular (Findings)

Deneysel çalışmalarda geliştirilen ZR-Jaya'nın tek-yerel noktalı fonksiyonların toplam 8 tanesinde, çok-yerel noktalı fonksiyonlarda ise 11 tanesinde olmak üzere sırasıyla %75 ve %61,11 başarı oranı elde etmiştir. / In experimental studies, the proposed ZR-Jaya achieved 75% success rate in 8 functions and 61.11% success rate in 11 functions in unimodal and multimodal functions, respectively.

Sonuç (Conclusion)

Önerilen ZR-Jaya'daki atalet ağırlığının algoritmanın keşif ve sömürü dengesini başarılı ve etkili olarak sağladığı ve ZR-Jaya'nın uygulanabilir olduğu tespit edilmiştir./ It has been determined that the inertia weight in the proposed ZR-Jaya successfully and effectively ensures the balance of exploration and exploitation of the algorithm and that ZR-Jaya is feasible.

Etik Standartların Beyanı (*Declaration of Ethical Standards*)

Bu makalenin yazar(lar)ı çalışmalarında kullandıkları materyal ve yöntemlerin etik kurul izni ve/veya yasal-özel bir izin gerektirmediğini beyan ederler. / The author(s) of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

Küresel Optimizasyon Problemlerinin Çözümü İçin Zamanla Değişen Rastgele Atalet Ağırlıklı Jaya Algoritması

Araştırma Makalesi / Research Article

Mehmet Fatih TEFEK*

Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Osmaniye Korkut Ata Üniversitesi, Türkiye

(Geliş/Received : 31.05.2020 ; Kabul/Accepted : 10.11.2020 ; Erken Görünüm/Early View : 24.11.2020)

ÖZ

Jaya algoritması küresel optimizasyon problemlerini çözmek için son zamanlarda sıklıkla kullanılan popülasyon tabanlı bir optimizasyon algoritmasıdır. Bu çalışmada küresel optimizasyon problemlerinin çözümü için zamanla değişen rastgele atalet ağırlıklı Jaya (ZR-Jaya) algoritması geliştirilmiştir. Geliştirilen algoritmada Jaya'ya göre optimizasyon problemlerini daha erken iterasyonlarda çözmek, yakınsama süresini azaltmak ve daha iyi çözüm elde etmek amaçlanmıştır. ZR-Jaya deneysel çalışmalar için literatürde iyi bilinen on adet kıyaslama fonksiyonu ile bu fonksiyonların birleşiminden oluşan beş adet kompozit küresel optimizasyon problemlerine uygulanmıştır. ZR-Jaya algoritmasının bulduğu sonuçlar Yapay Arı Kolonisi (YAK), Parçacık Sürü Optimizasyon (PSO), Jaya algoritmaları ve Jaya'nın güncelleme prosedürüne eklenen rastgele atalet ağırlıklı Jaya (RAA-Jaya), doğrusal azalan atalet ağırlıklı Jaya (DAAA-Jaya) ve karmaşık atalet ağırlıklı Jaya (KAA-Jaya) ile karşılaştırılmıştır. Geliştirilen algoritmanın başarısı YAK, PSO, Jaya ve Jaya'nın diğer ağırlık stratejileriyle kıyaslanmış ve sonuçlar çizelgelerde verilmiş ve grafiklerle gösterilmiştir. Deneysel çalışma sonuçlarına göre ZR-Jaya'nın PSO, YAK, Jaya ve Jaya'nın diğer ağırlık stratejilerinden, tek-yerel noktalı fonksiyonlarda başarı performans sayısı oranı %75, çok-yerel noktalı fonksiyonlarda ise %61,11 olmuştur. Geliştirilen ZR-Jaya algoritmasında zamanla değişen rastgele atalet ağırlığı faktörünün oldukça etkili olduğu ve uygulanabilir olduğu deneysel çalışmalarla tespit edilmiştir.

Anahtar Kelimeler: Jaya, atalet ağırlığı, optimizasyon, küresel optimizasyon problemleri.

Time-varying Random Inertia Weighted Jaya Algorithm for the Solution of Global Optimization Problems

ABSTRACT

The Jaya algorithm is a population-based optimization algorithm that has been used recently to solve global optimization problems. In this study, the time-varying random inertia-weighted Jaya (ZR-Jaya) algorithm has been developed for the solution of global optimization problems. In the proposed algorithm, it is aimed to solve optimization problems in earlier iterations, reduce the convergence time and obtain a better solution according to Jaya. ZR-Jaya has been applied to the five composite spherical optimization problems, consisting of ten well-known bencmark functions in the literature for experimental studies and the combination of these functions. The results of the ZR-Jaya algorithm were compared with the Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), Jaya algorithms and three different weighting strategies applied to Jaya's update procedure. In order to compare the success of the developed ZR-Jaya algorithm with other weight strategies of Jaya, ABC, PSO, Jaya, random inertia weight Jaya (RIW-Jaya), linear decreasing inertia weight Jaya (LDIW-Jaya) and Chaotic inertia weight Jaya (CIW-Jaya). The results are given in tables and shown with graphics. According to the experimental study results, ZR-Jaya's PSO, ABC, Jaya and other Jaya inertia weight strategies have achieved a 75% success rate in unimodal functions and 61.11% in multimodal functions. The proposed ZR-Jaya algorithm, it has been determined by experimental studies that the time-varying random inertia weight factor is highly effective and feasible.

Keywords: Jaya, inertia weight, optimization, global optimization problems.

1. GİRİŞ (INTRODUCTION)

Optimizasyon problemlerinde belirli koşullar altında belirli bir sorun için en uygun parametreleri ve çözümü seçmek gereklidir. Optimizasyon problemlerinin çözüm yöntemlerinden önemli bir sınıfını sezgisel algoritmalar oluşturmaktadır [1]. Sezgisel algoritmalar optimizasyon problemlerinde optimum veya optimuma yakın bir çözüm elde etmeye çalışır. Sezgisel algoritmalar genellikle doğadaki canlıların davranışlarını taklit ederek oluşturulmuş yöntemlerdir. Örneğin kuşların besin aramasından esinlenerek geliştirilen parçacık sürü optimizasyon (PSO) [2] algoritması, arıların besin arayışı ve davranışlarından esinlenerek geliştirilen yapay arı kolonisi (YAK) [3] gibi algoritmalar çok bilinen sezgisel algoritmalardır. Literatürde PSO [2], YAK [3] gibi çok bilinen algoritmalarının yanında son zamanlarda güncel popülasyon tabanlı algoritmalarda optimizasyon

**Sorumlu Yazar (Corresponding Author)*
e-posta : mehmetfatihtefek@osmaniye.edu.tr

problemlerin çözümlerine uygulanmaktadır. Bu algoritmalardan bir tanesi de Jaya algoritmasıdır. Jaya algoritması Rao (2016) tarafından öne sürülmüş popülasyon tabanlı sezgisel optimizasyon algoritmasıdır [4]. Jaya, algoritmaya ait herhangi özel bir parametre içermeyen sade ve güçlü bir optimizasyon yöntemidir. Bu durum Jaya için bir avantajdır. Çünkü sezgisel optimizasyon yöntemlerinde algoritmaların performansı, algoritmaya özgü parametrelerden çok etkilenir [5]. Jaya için sadece popülasyon sayısı ve iterasyon sayısı gibi her sezgisel yöntemin temelinde olan parametreleri tanımlamak yeterlidir [4, 5].

Rao (2016) tarafından tasarlanan Jaya algoritması literatürde birçok çalışmada optimizasyon problemlerine uygulanmıştır. Bunlardan bazıları: ekonomik yük dağıtımı [6], fotovoltaik sistemin maksimum güç noktası takibi [7], büyük ölçekli bir kentsel trafik ışığı planlama probleminin çözümü [8], tabanında anahtar kesit bulunan bir betonarme konsol istinat duvarının optimum tasarımı [9] gibi çalışmalardır. Jaya yeni bir algoritma olmasının yanında bazı problemlerin çözümü için modifiye edilerek yeniden düzenlenmiş veya geliştirilmiştir. Bu çalışmalardan bazıları şunlardır: Gao ve ark. (2016) yeni iş ekleme ile esnek atölye planlama sorunun çözümü amacıyla Jaya algoritmasının keşif yeteneğini arttırmak için ayrık Jaya algoritmasını tasarlamışlarıdır [10]. Wang ve ark. (2018) fotovoltaik hücre modellerinin parametre tahmini için yeni bir elit zıtlık tabanlı Jaya algoritmasını öne sürmüşlerdir [11]. Migallon ve ark. (2020), verimli paralel ve hızlı yakınsama için kaotik Jaya algoritmalarını tasarlamışlardır [12]. Optimizasyonun nihai hedefi ya en iyi çözümü elde etmek için yakınsama süresini azaltmak ya da algoritmanın verimliliğini artırmaktır [1]. Düzenlenen veya geliştirilen Jaya algoritmalarındaki ana hedefte bu bağlamda yakınsama süresini azaltma, etkili sonuçlar üretme ve yerel arama yeteneğini geliştirme üzerine yoğunlaşmaktadır.

Bu çalışmada optimizasyon problemlerinin çözümünde yakınsama süresini azaltmak ve bu şekilde daha az iterasyon sayısında çözüme gidebilmek için Jaya algoritmasına her bir iterasyonda, değişen rastgele atalet ağırlığı eklenerek ZR-Jaya algoritması tasarlanmıştır. Geliştirilen ZR-Jaya algoritması deneysel çalışma amaçlı literatürde iyi bilinen küresel optimizasyon problemlerin çözümüne uygulanmıştır. Deneysel çalışma sonuçları için standart PSO, YAK ve Jaya algoritmalarının yanında Jaya'nın güncelleme prosedürüne rastgele atalet ağırlığı (RAA-Jaya), doğrusal azalan atalet ağırlığı (DAAA-Jaya) ve karmaşık atalet ağırlığı (KAA-Jaya) eklenmiş ve kıyaslamalar yapılmıştır. Deneysel çalışmaların sonuçları çizelgeler ve grafiklerle desteklenmiştir.

2. ATALET AĞIRLIĞI PARAMETRESİ (INERTIA WEIGHT PARAMETER)

Atalet Ağırlığı (w) Parametresi (AAP) genellikle parçacık sürü optimizasyonunun (PSO) küresel keşif ve yerel sömürü yeteneklerini kontrol etmek için öne sürülmüştür [13, 14]. PSO'da sürünün keşfi ve sömürü işleminin dengelenmesinde başlangıç hızının büyük bir etkisi olduğundan, hızı kontrol etmek için AAP kullanılmaktadır [14]. AAP, bir parçacığın önceki hızının geçerli zaman adımındaki hızına katkı oranını belirler [14]. Eberhart ve ark. (2001) PSO'da hızı kontrol etmek için rastgele atalet ağırlığını (RAA) önermişlerdir [15]. RAA'da atalet ağırlığı her bir iterasyonda 0.5 ile 1 arasında rastgele değişmektedir. Feng ve ark. (2007) PSO algoritmasında AAP için karmaşık atalet ağırlıklı (KAA) stratejisini önermişlerdir [16]. KAA'nın PSO'nun tercih edilen yakınsama hassasiyetine, hızlı yakınsama hızına ve daha iyi global arama yeteneğine sahip olmasını sağladığını belirtmişlerdir [16]. Xin ve ark. (2009) PSO için her bir iterasyonda maksimum iterasyona ve algoritmaya özgü minimum ve maksimum başlangıç ağırlık parametreleri değerlerine bağlı olarak zamanla değişen doğrusal azalan atalet ağırlığını (DAAA) tasarlamışlardır [13]. Shi ve ark. (1998) sabit atalet ağırlıklı (SAA) PSO'yu önermişlerdir [17]. Algoritmaya özgü ağırlık parametre değerinin 0.9 ile 1.2 arasında sabit bir değer olması gerektiğini öne sürmüşlerdir. SAA'da optimum sonuca yaklaşılmasına rağmen algoritma çalışma sayısına bağlı ortalama değerde aşırı sapmalar olmaktadır [14, 17]. Arumugam ve ark. (2008) PSO'da atalet ağırlığını local ve küresel en iyi değerlerine göre belirlemişler ve küresel-yerel en iyi atalet ağırlığı olarak adlandırmışlardır [18]. Nickbadi ve ark. (2011) atalet ağırlık stratejilerini sabit, zamanla değişen ve uyarlanabilir olmak üzere üç farklı şekilde sınıflamışlar ve PSO için yeni bir uyarlamalı atalet ağırlığı (UAA) önermişlerdir [19]. Fan ve ark. (2007) büyük boyutlu problemlerde çözüme hızlı yakınsayan doğrusal olmayan zamanla azalan atalet ağırlığı (DOZA) stratejisini önermişlerdir [20]. Alataş ve ark. (2007) PSO'nun hız ve konum güncellemesi prosedürüne on iki farklı kaotik haritalı PSO (KHPSO) atalet ağırlıklarını önermişlerdir. Kaotik haritalardaki KHPSO7 ve KHPSO8 yöntemlerinin standart PSO'ya göre çözüm kalitesini arttırdığını tespit etmişlerdir [21]. Aydilek (2018) sabit, rasgele, doğrusal azalan, küresel yerel en iyi, benzetimli tavlama ve kaotik atalet ağırlığı stratejileri kombinasyonunu kullanarak yeni bir topluluk atalet ağırlığını önermiştir [22].

AAP, PSO'da kullanıldığı gibi son zamanlarda diğer optimizasyon algoritmalarında da bu parametre algoritmaların popülasyon güncelleme prosedüründe, probleme etkili çözüm bulmak ve yakınsama hızını dengelemek amacıyla kullanmaktadır. Rauf ve ark. (2020), yarasa algoritmasında yarasaların hızını etkin bir şekilde artırmak için uyarlanabilir bir atalet ağırlığı eklemiştir. Uyarlanabilir atalet ağırlığı, yarasaların çok boyutlu arama alanında açık hareketlerini yinelemeli olarak kısıtlamalarına yardımcı olmakta, av ararken en iyi ve en kötü çözümlerinden bağımsız olarak hızlarını değiştirmektedir [23]. Aynı şekilde Shukla ve ark. (2020) uyarlamalı atalet ağırlıklı öğretme-öğrenme temelli optimizasyon (ÖÖTO) algoritmasını öne sürmüşlerdir. Atalet ağırlığının, keşif ve sömürüyü kontrol etmek için çok önemli bir etken olduğunu ve deneysel çalışmalarda

atalet ağırlığının ÖÖTO performansını güçlü bir şekilde etkilediğini belirtmişlerdir. Yue ve ark. (2020), yarasa algoritmasına akıllı atalet ağırlığı ve genetik çaprazlama operatörü ekleyerek modifiye edilmiş yarasa algoritmasını öne sürmüşlerdir. Önerdikleri AAP yöntemi doğrusal olarak azalan ile uygunluk fonksiyonun minimum ve maksimum değerlerine göre değişen iki parçadan oluşmaktadır [24]. Ekinci ve ark. (2019) sinüs-kosinüs algoritmasının konum güncelleme prosedüründe Gauss fonksiyonuna dayalı doğrusal olmayan ağırlık parametresini yeniden düzenlemişlerdir [25]. Ramli ve ark. (2019) yarasa algoritmasında iter asyön ilerledikçe sömürü yeteneğinin düştüğünü tespit etmişler ve bunun üstesinden gelebilmek için hız güncelleme aşamasına yarasa konumlarına bağlı atalet ağırlığı eklemişlerdir [26]. Gan ve ark. (2018) yinelemeli erel arama ve stokastik (rastgele değişen) atalet ağırlığına dayalı yeni bir yarasa algoritmasını tasarlamışlardır. Stokastik atalet ağırlığı hız güncelleme prosedürüne uygulanmış ve yarasa popülasyonunun çeşitliliğini ve esnekliğini artırdığını tespit etmişlerdir [27]. Yılmaz ve ark. (2013) zamanla doğrusal azalan atalet ağırlığı parametresini yarasa algoritmasına uygulamışlardır [28]. Toz ve ark. (2019) geri izleme arama optimizasyonu algoritmasının yerel arama yeteneğini arttırmak için doğrusal olmayan üstel atalet ağırlığı parametresini tasarlamışlardır [29]. Wu ve ark. (2015), doğrusal olmayan atalet ağırlıklı ÖÖTO'yu önermişlerdir. Önerdikleri yöntemi çok bilinen küresel optimizasyon problemlerinden benchmark test fonksiyonlarına uygulamışlardır. Önerilen yöntem, temel ÖÖTO ve diğer bazı algoritmalardan daha hızlı yakınsama hızına ve daha iyi performansa sahip olduğunu göstermişlerdir [30]. Çizelge 1'de PSO'da ve diğer optimizasyon algoritmalarında kullanılan bazı AAP stratejileri ve geliştirilen ZR-Jaya verilmiştir [13-20]. Çizelge 1'de en

Çizelge 1. Geliştirilen ZR-Jaya ile literatürde kullanılan bazı AAP stratejileri, formülleri ve kriterleri (Some AAP strategies , formulas and criteria used in the literature with the proposed ZR-Jaya) [13-20]

AAP Stratejisi	Formülü	Kriter
Karmaşık atalet ağırlığı (KAA) [16]	$z = 4\ x\ z\ x\ (1-z)$ $w = (w_{maks} - w_{min})\ x\ \frac{Maks\dot{I}ter - \dot{I}ter}{Maks\dot{I}ter} + w_{min}\ x\ z$	En iyi ortalama değer
Rastgele atalet ağırlığı (RAA) [15]	$w = 0.5 + \frac{rand()}{2}$	En iyi ortalama iterasyon sayısı
Doğrusal azalan atalet ağırlığı (DAAA) [13]	$w_{iter} = w_{maks} - \frac{w_{maks} - w_{min}}{Maks\dot{I}ter} x\ \dot{I}ter$	En iyi minimum değer
Sabit atalet ağırlığı (SAA) [17]	$w = c$ $(0.9 < c < 1.2)$	En iyi minimum değerine rağmen en kötü ortalama iterasyon sayısı
Karmaşık rastgele atalet ağırlığı (KRAA) [14]	$z = 4\ x\ z\ x\ (1-z)$ $w = 0.5xrand() + 0.5xz$	En kötü minimum değer ve ortalama hata değeri
Küresel-Yerel en iyi atalet ağırlığı (KYAA) [18]	$w = 1.1 - \frac{g_{best}}{p_{best\ i_{average}}}$	Küresel en iyi ile yerel en iyi ortalama değer
Uyarlanabilir atalet ağırlığı (UAA) [19]	$w_i = w_{initial}(1\frac{dis_i}{max\_dist}),$ $dist_i = \sqrt{\sum_{d=1}^{D}(gbest_d - x_{i,d})^2}$	Küresel en iyi değere hızlı yakınsama
Doğrusal olmayan zamanla azalan atalet ağırlığı (DOZA) [20]	$w(iter) = (\frac{2}{iter})^{0.3}$	Büyük boyutlu problemlerde hızlı yakınsama
Geliştirilen Zamanla değişen rastgele atalet ağırlığı (ZR-Jaya)	$w_{min}, w_{maks} \in rand()$ $(w_{maks} > w_{min}\ ve\ rand() \in (0,1))$ $w = w_{maks} + (w_{maks} - w_{min}) * \frac{(Maks\dot{I}ter - \dot{I}ter)}{Maks\dot{I}ter}$	Doğrusal azalan ve rastgele değişen atalet ağırlıkları kombinasyonu ile en iyi minimum değer ve en iyi ortalama değer

iyi ortalama sonuçları için karmaşık atalet ağırlığı (KAA) [16], en iyi ortalama iterasyon sayısında yani daha erken yakınsamada bulunan çözüm için rastgele atalet ağırlığı (RAA) [15] ve en iyi minimum değer için doğrusal azalan atalet ağırlığı (DAAA) [13], sabit atalet ağırlığı (SAA) [17] ile küresel-yerel en iyi atalet ağırlığı (KYAA) [18] formülleri verilmiştir. KAA'da [16], Karmaşık rastgele atalet ağırlığı (KRAA) [14] değerindeki z, w_1 ve w_2 değerleri kullanıcı tarafından tanımlanan (0,1) aralığındaki parametrelerdir. Aynı şekilde DAAA'da [13] w_{min} ve w_{maks} değerleri de kullanıcı tarafından girilen (0,1) aralığındaki değerlerdir. RAA'da [15], (KRAA) [14] ve geşitirilen ZR-Jaya'da $rand()$ ise (0,1) aralığında değişen rastgele değerlerdir. KYAA [18] ve UAA [19]'da p_{best} lokal en iyi, g_{best} ise global en iyi değerlerdir. Algoritmaların her bir çalışmadaki iterasyon sayısı *İter*, maksimum iterasyon sayısı aynı zamanda durdurma kriteri ise *Maksİter* olarak tanımlanmıştır.

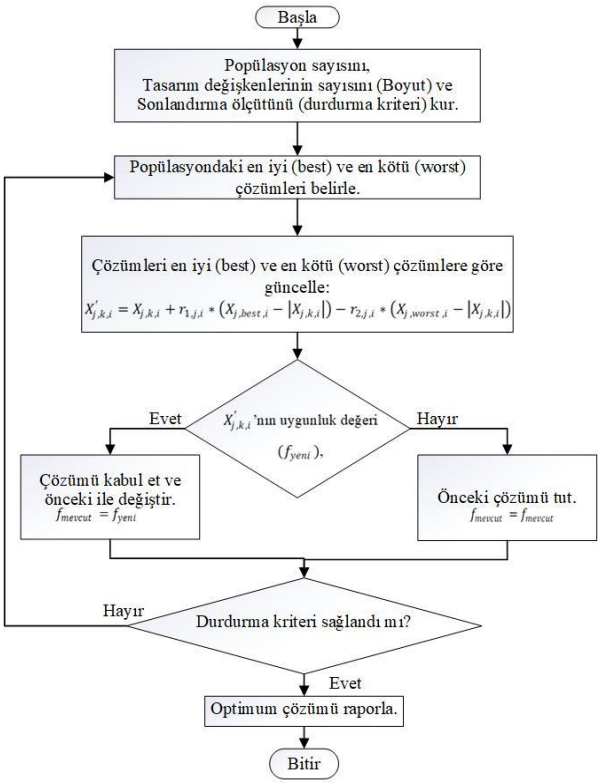
Literatürde [13, 19, 20], mevcut atalet ağırlıklarının çoğu, keşif ve sömürü arasındaki dengeyi sağlamak için hem doğrusal artan hem de doğrusal olmayan azalma gerçekleştirmektedir. Mevcut doğrusal ve doğrusal olmayan atalet ağırlığının özelliklerini dahil etmek, algoritmaların arama kapasitesini arttırmak ve yakınsama oranını korumada fayda sağlamaktadır [13, 24, 25, 31]. Bu çalışmada Çizelge 1'de hem en iyi minimum değer hem de ortalama için sırasıyla doğrusal azalan ve rastgele atalet ağırlıkları Jaya algoritmasının farklı bir formu öne sürülerek ZR-Jaya algoritması geliştirilmiştir.

3. JAYA ALGORİTMASI (JAYA ALGORITHM)

Jaya, Rao (2016) tarafından kısıtlı ve kısıtsız sürekli optimizasyon problemlerini çözmek için önerilen popülasyon tabanlı bir algoritmadır [4]. Jaya ismi Hintçe'de "zafer" anlamına gelen bir kelimedir ve nihai sonuca ulaşarak muzaffer olma manasında kullanılmaktadır. Jaya algoritmasının ana amacı, verilen bir sorunun en iyi çözüme doğru ilerlemesini sağlamak için ödüllendirmek ve en kötü çözümden kaçınmak için cezalandırmaktır. Bunun için her iterasyonda en iyi ve en kötü bireyler Denklem 1'deki gibi güncellenmekte ve amaç fonksiyonun maksimim ya da minimum yapılması amacıyla uygulanmaktadır.

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i} * \left(X_{j,best,i} - \left|X_{j,k,i}\right|\right) - r_{2,j,i} * \left(X_{j,worst,i} - \left|X_{j,k,i}\right|\right) \quad (1)$$

Burada, herhangi bir i iterasyonunda "m" tasarım değişken sayısı (problem boyutu) $(j = 1,2,\dots,m)$, "n" aday çözüm sayısı (popülasyon sayısı) $(k = 1,2,\dots,n)$ olarak kabul edilsin. $X_{j,best,i}$ en iyi (best) aday çözüm için j tasarım değişkeninin değeridir. $X_{j,worst,i}$ en kötü aday (worst) çözüm için j tasarım değişkeninin değeridir. $X'_{j,k,i}$, $j.$ tasarım değişkeni için $i.$ iterasyonda $X_{j,k,i}$' nin güncellenmiş değeridir. $r_{1,j,i}$ ve $r_{2,j,i}$ $j.$ tasarım değişkeni için $i.$ iterasyonda [0, 1] aralığında değişen iki farklı rasgele sayıdır [4]. Jaya'nın temel çalışma prensibi şu şekildedir: $i.$ iterasyonda güncellenen $X'_{j,k,i}$ değerinin uygunluk fonksiyonunda bulduğu çözüm (f_{yeni}), $X_{j,k,i}$ değerinin bulduğu mevcut çözümden (f_{mevcut}) daha iyi ise yeni çözümü mevcut çözüm olarak al ($f_{mevcut} = f_{yeni}$). Aksi takdirde mevcut çözümü (f_{mevcut}) al ve üretilen yeni popülasyonu Denklem 1'deki gibi güncelle. Bu işlemi durdurma kriteri sağlanıncaya kadar devam ettirilir. Jaya algoritmasının akış şeması Şekil 1'de verilmiştir.



Şekil 1. Jaya Algoritmasının Akış Diyagramı (Flow Chart of Jaya Algorithm) [4]

4. GELİŞTİRİLEN ZR-JAYA ALGORİTMASI (THE PROPOSED ZR-JAYA ALGORITHM)

Bu çalışmada Jaya algoritmasına popülasyon güncelleme sırasında her bir iterasyonda, değişen rastgele atalet ağırlığı eklenerek, **Z**amanla değişen **R**astgele atalet ağırlıklı **Jaya** (ZR-Jaya) algoritması geliştirilmiştir. ZR-Jaya algoritmasında popülasyondaki zamanla değişen rastgele atalet ağırlığı için Algoritma 1'deki kaba kod verilmiştir.

Algoritma 1'de Jaya'nın Denklem 1'deki güncelleme prosedürüne $\boldsymbol{W}$ atalet ağırlığı eklenmiştir. Çizelge 1'deki AAP stratejilerindeki karmaşık atalet ağırlığından [16] ve doğrusal azalan atalet ağırlığından [13] farklı olarak geliştirilen ZR-Jaya yönteminde ağırlık değerleri rastgele belirlenmektedir. Bu durumda ZR-Jaya'nın performansının, algoritmaya özgü parametrelerden etkilenmemesini sağlamıştır. Geliştirilen ZR-Jaya'daki atalet ağırlığının bir amacı, algoritmanın hesaplama maliyetinin azalması için en uygun bireylerin seçimini erken iterasyonlarda yapılması diğer amacı ise

başlangıçtan beri daha hızlı keşif ve sömürü dengesinin sağlanmasıdır.

Algoritma 1. Geliştirilen ZR-Jaya Algoritması ile yeni popülasyon güncelleme prosedürü (New Population Update Procedure with the proposed ZR-Jaya Algorithm)

Algorithm 1. Geliştirilen ZR-Jaya'nın Algoritmik Tablosu

Girdi
Problemin tasarım değişkenlerini (boyutunu) belirle (m)
Birey popülasyon sayısını belirle (n)
Fonksiyon değerlendirme sayısını (Fes) belirle ($İter$)
Durdurma kriterini belirle ($Maksİter$)
Çıktı
Amaç fonksiyonun minimum ya da maksimum (optimum) değeri al.
1. Başlangıç popülasyonunu üret
2. Problem tanımındaki arama uzayında m-boyutundaki n adet rastgele aday çözümleri üret: $X(n,m)$
3. **WHILE** $İter <= MaksİterMaksİter$
4. $F_{Best} = F_{Obj}(X(n,m))$; // Aday çözümleri amaç fonksiyonuna uygula
5. **FOR** Her aday çözüm ($X(n,m)$)
6. Denklem 1.'deki yeni güncelleme prosedürünü aşağıdaki gibi belirle
7. **FOR** i=1 **TO** n //n: Popülasyon sayısı
8. **FOR** j=1 **To** m // m: Tasarım değişken sayısı (Boyut)
9. $r = rand(1,2)$ // r: (0,1) aralığında rastgele
10. //üretilen iki sayı
11. $w_1(\boldsymbol{i},\boldsymbol{j}) = r(1);$ //w_1, w_2: rastgele üretilen değerler
12. $w_2(\boldsymbol{i},\boldsymbol{j}) = r(2);$
13. **IF** $(w_1(\boldsymbol{i},\boldsymbol{j}) > w_2(\boldsymbol{i},\boldsymbol{j}))$ // maksimum ve minimum
14. //ağırlık değerlerini belirle.
15. $w_{maks}(\boldsymbol{i},\boldsymbol{j}) = (w_1(\boldsymbol{i},\boldsymbol{j}))/n$
16. $w_{min}(\boldsymbol{i},\boldsymbol{j}) = (w_2(\boldsymbol{i},\boldsymbol{j}))/n$
17. **ELSE**
18. $w_{maks}(\boldsymbol{i},\boldsymbol{j}) = (w_2(\boldsymbol{i},\boldsymbol{j}))/n$
19. $w_{min}(\boldsymbol{i},\boldsymbol{j}) = (w_1(\boldsymbol{i},\boldsymbol{j}))/n$
20. **END IF**
21. $$\boldsymbol{W}(\boldsymbol{i},\boldsymbol{j}) = w_{maks}(\boldsymbol{i},\boldsymbol{j}) + (w_{maks}(\boldsymbol{i},\boldsymbol{j}) - w_{min}(\boldsymbol{i},\boldsymbol{j})) * \frac{(MaksİterMaksİter - İter)}{MaksİterMaksİter}$$
22. $$X'_{i,j} = \boldsymbol{W}(\boldsymbol{i},\boldsymbol{j}) * X_{i,j} + r_{1,i,j} * \left(X_{i,best,j} - \left|X_{i,j}\right|\right) - r_{2,i,j} * \left(X_{i,worst,j} - \left|X_{i,j}\right|\right)$$
23. **END FOR**
24. **END FOR**
25. Amaç fonksiyon ($F_{yeni,Best}$**)** değerini $X'_{i,j}$ için hesapla
26. **IF** $((F_{yeni,Best})< F_{Best})$
27. $X(n,m) = X'_{i,j}$
28. $F_{Best} = F_{new,Best}$
29. **END IF**
30. $İter = İter + 1;$
31. **END FOR**
32. En iyi uygunluk fonksiyon değerine sahip aday çözümü al ($X(n,m)$).
33. **IF** Durdurma kriteri sağlanmadıysa
34. Adım 3'e git
35. **ELSE**
36. Bulunan en iyi çözümü al (F_{Best}).
37. **END WHILE**

5. DENEYSEL ÇALIŞMA VE SONUÇLARI (EXPERIMENTAL STUDY AND RESULTS)

Geliştirilen ZR-Jaya literatürde iyi bilinen 10 adet kıyaslama fonksiyonu (F1-F10) ile bu fonksiyonların birleşiminden oluşan 5 adet kompozit fonksiyona (F11-F15) uygulanmıştır [32]. F11-F15 arasındaki kompozit fonksiyonlar sırasıyla: hibrit kompozit fonksiyonu (hybrid composition function), F11'in döndürülmüş hibrit kompozit fonksiyonu (rotated version of hybrid composition function F11), F12'nin gürültüyle döndürülmüş hibrit kompozit fonksiyonu (F12 with Noise in Fitness), döndürülmüş hibrit kompozit fonksiyonu (rotated hybrid composition function), global optimum için dar bir havza ile döndürülmüş hibrit kompozisyon fonksiyonu (rotated hybrid composition function with a narrow basin for the global optimum) olarak bilinmektedir [32]. Çizelge 2'de bu fonksiyonlar ve özellikleri verilmiştir.

Çizelge 2. Kıyaslama Fonksiyonları (Benchmark Functions)

No	Fonksiyon	Arama Uzayı	Fonksiyonun Özelliği			
			Tek-Yerel Noktalı	Çok-Yerel Noktalı	Ayrıştırılabilir Evet	Ayrıştırılabilir Hayır
F1	Sphere	[-100,100]^{B*}	✓		✓	
F2	SumSquares	[-10,10] ^{B}	✓		✓	
F3	Step	[-100,100] ^{B}	✓		✓	
F4	Rosenbrock	[-30,30] ^{B}	✓			✓
F5	Rastrigin	[-5.12,5.12] ^{B}		✓	✓	
F6	Dixon&Price	[-10,10] ^{B}	✓			✓
F7	Griewank	[-600,600] ^{B}	✓		✓	
F8	Schwefel 2.26	[-500,500] ^{B}		✓		✓
F9	Ackley	[-32,32] ^{B}		✓		✓
F10	Levy	[-10,10] ^{B}		✓		✓
F11	Kompozit Fonksiyon 1	[-5,5] ^{B}		✓	✓	
F12	Kompozit Fonksiyon 2	[-5,5] ^{B}		✓		✓
F13	Kompozit Fonksiyon 3	[-5,5] ^{B}		✓		✓
F14	Kompozit Fonksiyon 4	[-5,5] ^{B}		✓		✓
F15	Kompozit Fonksiyon 5	[-5,5] ^{B}		✓		✓

*B: Boyut

Çizelge 2'de bir fonksiyonda birden fazla lokal optimum varsa, bu fonksiyon çok-yerel noktalı (multimodal) olarak adlandırılır. Çok-yerel noktalı fonksiyonlar algoritmanın global arama yeteneğini test etmek amacıyla kullanılır.

Tek-yerel noktalı (unimodal) fonksiyonlar sadece bir lokal optimum noktalıdır. Tek-yerel noktalı fonksiyonlar sezgisel algoritmaların yerel aramadaki sömürü yeteneğini test etmek amaçlı kullanılırlar. N adet değişkene sahip bir fonksiyon, bir değişkenin n fonksiyonun toplamı olarak yazılabilirse, bu ayrıştırılabilir fonksiyon olarak adlandırılır. Ayrıştırılamaz fonksiyonlarda ise fonksiyonların değişkenleri arasında karşılıklı ilişki olduğundan n fonksiyonun toplamı olarak yazılamaz [33].

Deneysel çalışma sonuçlarını karşılaştırma amacıyla geliştirilen ZR-Jaya algoritması PSO, YAK, Jaya algoritması ve Çizelge 1'deki Jaya'nın güncelleme prosedürüne eklenen karmaşık atalet ağırlıklı Jaya (KAA-Jaya), rastgele atalet ağırlıklı Jaya (RAA-Jaya) ve doğrusal azalan atalet ağırlıklı Jaya (DAAA-Jaya) oluşturulmuş ve Çizelge 2'deki benchmark fonksiyonlarına uygulanmıştır. Test amaçlı tüm algoritmalar 30 ve 50 boyutta, 20 bireyli popülasyon, 1000 iterasyonda, fonksiyon değerlendirme sayısı (Fes) 20.000 (Fes=Popülasyon X İterasyon) olarak alınmıştır. Ayrıca standart PSO algoritması kontrol parametreleri $c1 = 2,\ c2 = 2$ olarak [34, 35], YAK için kontrol parametreleri gözcü arı ve işçi arı sayısı popülasyon sayısının yarısı, limit ise gözcü arı ile problem boyutunun çarpımı olarak alınmıştır [36]. KAA-Jaya'da z değeri (0,1) arasında değişen algoritma içerisinde değişen rastgele bir değer olarak belirlenmektedir [21]. Aynı zamanda KAA-Jaya ve DAAA-Jaya'da $w_{maks} = 0.9, w_{min} = 0.4$ olarak alınmaktadır [13, 21, 35]. Tüm algoritmalar aynı şartlar altında 30 kez çalıştırılmıştır.

Çizelge 3 ve Çizelge 4'te sırasıyla 30 ve 50 boyutlu fonksiyonlar için ZR-Jaya ile PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya çalışma sonuçları verilmiştir. Çizelge 3 ve Çizelge 4'te her bir fonksiyon için hesaplanan en iyi ve en iyi ortalama (Ort.) değerler kalın yazı tipiyle gösterilmiştir. Şekil 2 ve Şekil 3'te 30 ve 50 boyutlu fonksiyonların yakınsama grafikleri verilmiştir. Şekil 4'te ZR-Jaya ile PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın performans başarı sayı grafiği verilmiştir.

Çizelge 3'te 30 boyutlu fonksiyonların karşılaştırma sonuçlarına göre PSO, sadece F13 fonksiyonda iyi bir sonuç elde etmiştir. YAK hem en iyi hem de ortalama sonuçlarda F10 ve F12 fonksiyonunda diğerlerinden iyi bir sonuç verirken F6 fonksiyonunda en iyi ortalama sonucu vermiştir. Jaya algoritması F4 fonksiyonunda hem en iyi hem de en iyi ortalama sonucu vermiş aynı zamanda F6 fonksiyonunda en iyi sonucu hesaplamıştır.

RAA-Jaya, F3, F5 ve F7 fonksiyonunda optimum sonuç ve en iyi ortalama sonucu vermiştir. F9 fonksiyonunda ise sadece en iyi sonucu hesaplamıştır. DAAA-Jaya, sadece F3 fonksiyonunda optimum sonucu bulmuştur. KAA-Jaya F3, F5,F7 ve F9 fonksiyonlarında hem optimum sonucu hem de ortalama en iyi sonucu hesaplamıştır. ZR-Jaya ise F1, F2, F3, F5, F7, F8 ve F9 fonksiyonlarında hem en iyi sonucu hem de en iyi ortalama sonucu vermiştir. Aynı zamanda ZR-Jaya F3, F5, F7 ve F9 fonksiyonlarında optimum değeri elde etmiştir. ZR-Jaya F11, F14 ve F15 fonksiyonlarında diğer yöntemlerden daha iyi sonuçlar hesaplamıştır. Şekil 2'de ZR-Jaya ve diğer PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın 30 boyut da en iyi sonuçlarına göre yakınsama eğrisi verilmiştir. Şekil 2'de ZR-Jaya'nın en iyi sonuç için PSO, YAK ve Jaya algoritmalarından daha az fonksiyon değerlendirme sayısında (Fes) dolayısıyla daha erken iterasyonlarda çözüme ulaştığı görülmektedir.

Çizelge 4'te 50 boyutlu fonksiyonların karşılaştırma sonuçlarına göre ZR-Jaya hem en iyi hem de ortalama sonuçlara göre F1, F2, F3, F5, F7, F9, F11, F14 ve F15 fonksiyonlarında PSO, YAK ve Jaya'dan daha iyi sonuçlar elde etmiştir. PSO F12 ve F13 fonksiyonlarında diğer yöntemlerden daha iyi sonuç vermiştir. YAK, F8, F10 ve F12, Jaya ise F4 fonksiyonlarında hem en iyi hem de en iyi ortalama sonucu elde etmişlerdir. RAA-Jaya F3, F5,F7 ve F15'de en iyi sonuçları vermiştir. DAAA-Jaya hiçbir fonksiyonda başarı elde edememiştir. KAA-Jaya F3, F5, F7 ve F9 fonksiyonlarında optimum sonucu bulmuştur. Şekil 3'te 2'de ZR-Jaya ve diğer PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın 50 boyut da en iyi sonuçlarına göre yakınsama eğrisi verilmiştir. Şekil 3'te geliştirilen ZR-Jaya'nın standart Jaya, PSO ve YAK algoritmalarından daha erken ve hızlı yakınsadığı görülmektedir.

Çizelge 3 ve Çizelge 4'te ZR-Jaya ve atalet ağırlığı eklenen diğer RAA-Jaya, DAAA-Jaya ve KAA-Jaya stratejilerine bakıldığında DAAA-Jaya hariç diğer atalet ağırlığı stratejileri standart PSO, YAK ve Jaya'dan daha iyi sonuçlar vermektedir. Şekil 2 ve Şekil 3'te geliştirilen yöntem ile RAA-Jaya, KAA-Jaya'nın daha erken fonksiyon değerlendirme sayılarında yakınsadığı görülmektedir. Şekil 4'te PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya, KAA-Jaya ve geliştirilen ZR-Jaya yöntemlerinin başarı sayıları verilmiştir. Başarı sayıları Çizelge 3 ve Çizelge 4'teki sırasıyla 30 ve 50 boyutta tek ve çok yerel noktalı fonksiyonlarda algoritmaların en iyi değerleri baz alınarak elde edilmiştir. Çizelge 2'de tek-tek yerel noktalı 6, çok-yerel noktalı 9 fonksiyon bulunmaktadır. Algoritmaların başarıları 30 ve 50 boyut için başarı sayısının toplamının her iki boyuttaki toplam fonksiyon sayısına oranı (ZR_Jaya başarı sayısı/toplam başarı sayısı) ile hesaplanmıştır. Dolayısıyla 30 ve 50 boyutta tek-yerel noktalı fonksiyonlarda toplam başarı sayısı 12, çok-yerel noktalı fonksiyonlarda toplam başarı sayısı 18 olarak belirlenmiştir.

Şekil 4'te geliştirilen ZR-Jaya'nın tek-yerel noktalı 30 ve 50 boyutlu deneysel çalışmada toplamda 8 fonksiyonda başarılı olmuş, Jaya toplamda 3 fonksiyonda, DAAA-Jaya ve YAK toplamda 1 fonksiyonda başarı göstermiştir. RAA-Jaya ve KAA-Jaya toplamda sırasıyla 4 ve 5 fonksiyonda başarılı olmuştur. PSO ise hiçbir başarı elde edememiştir. Geliştirilen ZR-Jaya yönteminin tek-yerel noktalı fonksiyonlarda 30 ve 50 boyutta toplam başarı sayısına göre diğer yöntemlerden %75 oranı ile daha başarı olduğunu göstermektedir.

Çok-yerel noktalı 30 ve 50 boyutlu deneysel çalışmalarda ZR-Jaya toplamda 11 fonksiyonda, YAK toplamda 5 fonksiyonda başarılı sonuçlar elde ederken RAA-Jaya, KAA-Jaya ve PSO toplamda 3 fonksiyonda, Jaya ve DAAA-Jaya ise hiçbir başarı elde edememiştir. Geliştirilen ZR-Jaya'nın çok-yerel noktalı fonksiyonlarda diğer yöntemlere göre toplam başarı sayısı oranı %61,11 olmuştur.

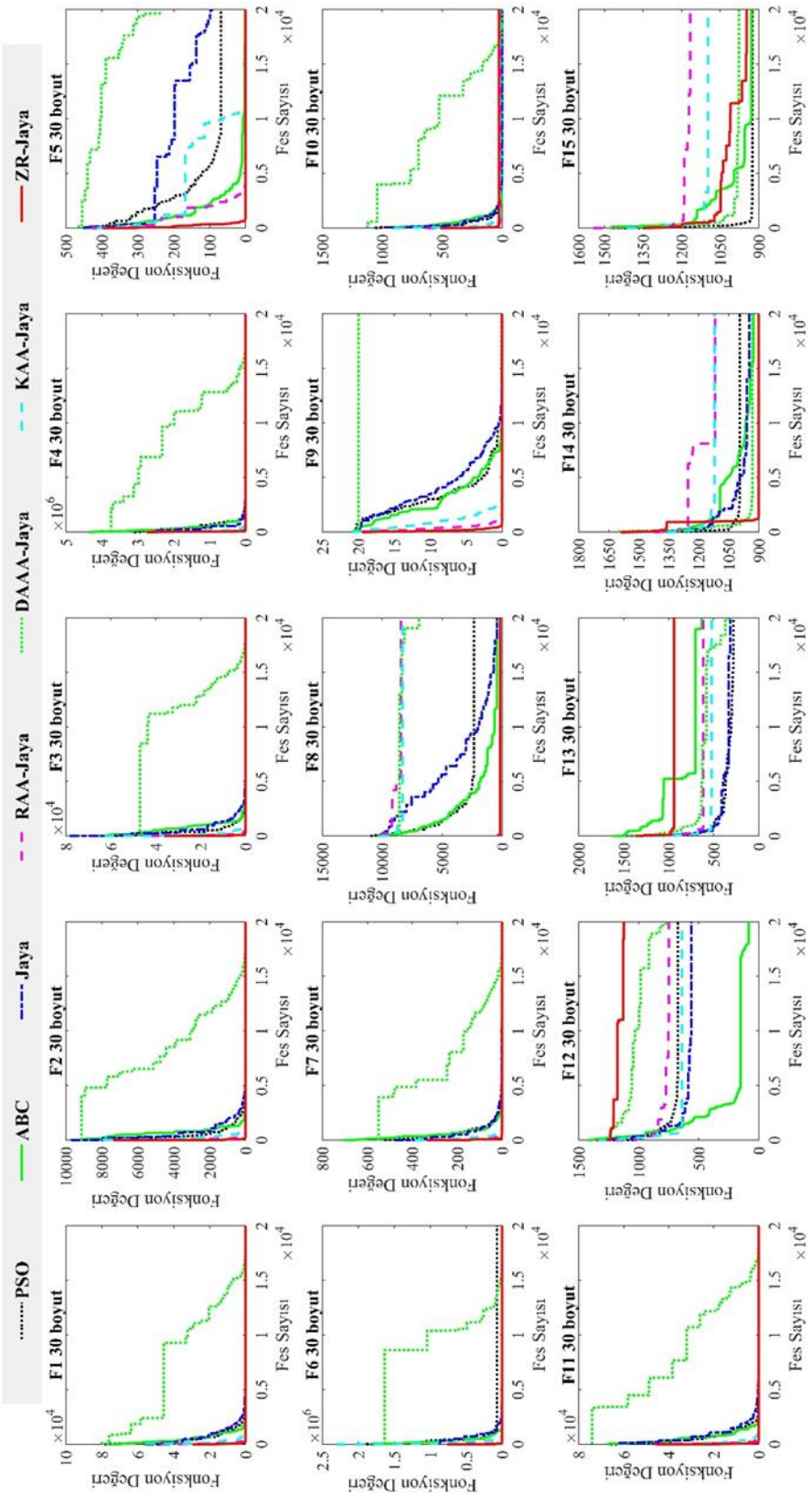
6. SONUÇLAR (CONCLUSION)

Bu çalışmada Jaya algoritmasının popülasyon güncelleme aşamasına yeni bir atalet ağırlığı yöntemi eklenmiştir. Eklenen atalet ağırlıklı Jaya algoritması, zamanla değişen rastgele atalet ağırlıklı Jaya (ZR-Jaya) olarak adlandırılmıştır. ZR-Jaya, deneysel çalışma amaçlı küresel optimizasyon problemlerinin çözümüne, literatürde iyi bilinen ve farklı özelliklere sahip on beş adet kıyaslama fonksiyonuna 30 ve 50 boyut için uygulanmıştır. Geliştirilen yöntemin başarısını test etmek amaçlı Jaya'ya rastgele atalet ağırlığı (RAA-Jaya), doğrusal azalan atalet ağırlığı (DAAA-Jaya) ve karmaşık atalet ağırlığı (KAA-Jaya) eklenmiştir. ZR-Jaya'nın sonuçları standart PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya ile karşılaştırılmıştır. Karşılaştırma sonuçları çizelgeler ve grafiklerle verilmiştir.

Çizelge 3 ve Çizelge 4'e bakıldığında geliştirilen ZR-Jaya'nın sonuçlarının standart PSO, YAK, Jaya ve Jaya'nın popülasyon güncelleme prosedürüne eklenen diğer üç farklı RAA-Jaya, DAAA-Jaya ve KAA-Jaya'dan daha iyi olduğu görülmektedir. Şekil 3 ve Şekil 4'teki yakınsama grafiklerine bakıldığında ise geliştirilen ZR-Jaya'nın optimum sonuca daha hızlı yakınsadığı bu durumda algoritmanın hesaplama maliyetinin azalması için en uygun bireylerin seçimini erken iterasyonlarda yaptığı söylenebilir. Şekil 4'te ise tek-yerel noktalı fonksiyonlarda ZR-Jaya'nın %75 oranındaki başarı sayısının yerel aramadaki sömürü yeteneğinin başarılı olduğu, çok-yerel noktalı fonksiyonlarda ZR-Jaya'nın %61,11 oranındaki başarı sayısıyla da global aramada keşfetme yeteneğinin başarılı olduğu sonucuna ulaşılmıştır. Bu şekilde geliştirilen ZR-Jaya'daki atalet ağırlığının algoritmanın keşif ve sömürü dengesini başarılı ve etkili olarak sağladığı ve ZR-Jaya'nın uygulanabilir olduğu tespit edilmiştir.

Çizelge 3. Geliştirilen ZR-Jaya ile PSO, YAK , Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın 30 boyutta karşılaştırma sonuçları (Comparison results of proposed ZR-Jaya with PSO, YAK , Jaya, RIW-Jaya, LDIW-Jaya and CIW-Jaya algorithms in 30 dimensions)

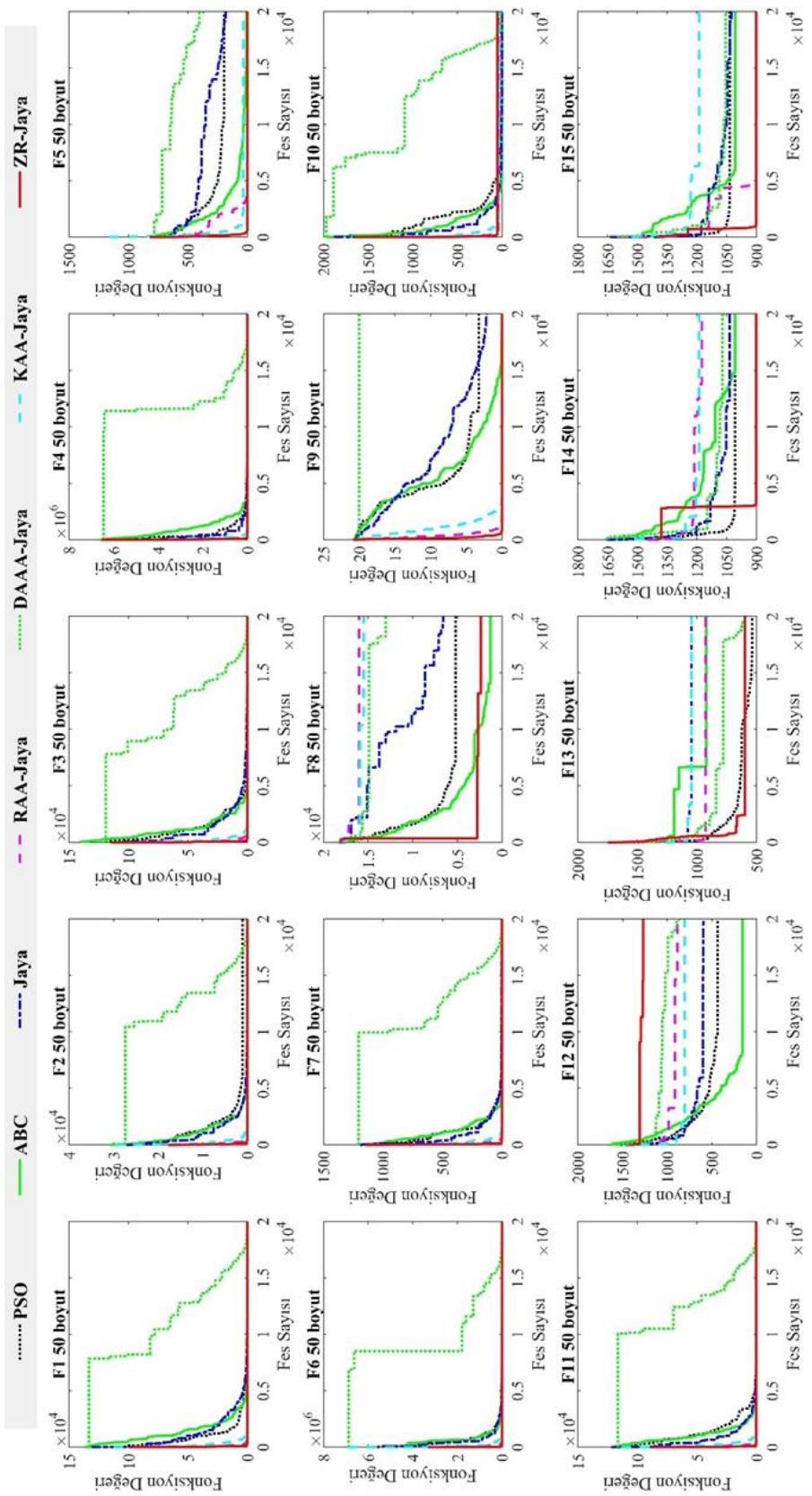
		PSO	YAK	Jaya	RAA-Jaya	DAAA-Jaya	KAA-Jaya	ZR-Jaya
	En İyi	1.7181E-08	7.0055E-12	1.7783E-06	3.66E-86	1.96E-01	6.12E-97	**1.8247E-173**
F1	Ort.	1.6667E+03	3.1766E-10	1.1923E-05	2.53E-83	9.14E-01	9.672E-94	**1.2337E-160**
	Std.	3.7268E+03	7.2065E-10	1.393E-05	4.34E-83	0.16E+01	1.19E-93	4.6023E-160
	En İyi	3.3815E-08	4.6991E-13	6.0834E-08	6.85E-88	3.67E-02	4.85E-98	**1.6543E-172**
F2	Ort.	4.2335E+02	8.6470E-12	2.3275E-06	3.54E-84	1.83E-01	1.359E-94	**2.1380E-164**
	Std.	4.3256E+02	8.5855E-12	2.8931E-06	6.07E-84	1.56E-01	1.853E-94	0
	En İyi	0	2	0	**0**	**0**	**0**	**0**
F3	Ort.	1.0025E+03	5.7	3.733	**0**	1.8	**0**	**0**
	Std.	2.9997E+03	3.4559	2.8394	0	2.638	0	0
	En İyi	1.7946	9.153E-01	**6.5102E-03**	2.91 E+01	3.006E+01	2.861E+01	2.88678E+01
F4	Ort.	8.4170E+02	1.03168E+01	**1.04870E+01**	2.89 E+01	3.39E+01	2.865E+01	2.89389E+01
	Std.	2.4697E+03	1.03412E+01	2.1883E+01	2.90E-01	3.39	0.0383	2.21078E-02
	En İyi	6.7710E+01	9.970E-01	9.3642E+01	**0**	2.38E+02	**0**	**0**
F5	Ort.	1.4013E+02	4.8248	1.9955E+02	**0**	2.59E+02	**0**	**0**
	Std.	3.4865E+01	1.8959	4.2143E+01	0	13.509	0	0
	En İyi	6.674E-01	2.331E-02	**6.8268E-06**	6.66E-01	9.44E-01	6.672E-01	8.108E-01
F6	Ort.	5.1718E+04	**5.34E-02**	1.1703	6.67E-01	1.406	6.795E-01	9.771E-01
	Std.	8.3228E+04	3.35E-02	1.4308	3.93E-06	0.292	1.15E-02	4.63E-02
	En İyi	4.4674E-07	1.9757E-10	5.1780E-06	**0**	2.566E-01	**0**	**0**
F7	Ort.	1.80677E+01	3.3207E-03	4.07E-02	**0**	6.72E-01	**0**	**0**
	Std.	3.60624E+01	6.2722E-03	1.019E-01	0	0.334	0	0
	En İyi	2.3551E+03	4.13195E+02	4.2535E+02	8.44E+03	6.93E+03	8.29E+03	**1.72469E+02**
F8	Ort.	3.5725E+03	8.70854E+02	5.8301E+03	9.21E+03	7.458E+03	9.065E+03	**4.69905E+02**
	Std.	5.83706E+02	2.4712E+02	1.9383E+03	2.58E+02	3.627E+02	4.52E+02	1.2846E+02
	En İyi	4.09E-02	5.8317E-06	5.2406E-04	0	1.995E+01	**0**	**0**
F9	Ort.	1.1831E-01	2.8799E-05	3.0457	1.33E-15	1.996E+01	**0**	**0**
	Std.	7.3666	2.0346E-05	5.6952	1.08E-15	1.263E-03	0	0
	En İyi	5.1318E-08	**7.0858E-11**	2.9344E-10	2.71E+01	3.964	1.56E+01	2.63222E+01
F10	Ort.	4.15966E-01	**1.3495E-08**	2.729E-01	2.96E+01	6.516	1.777E+01	3.01816E+01
	Std.	4.83763E-01	3.0351E-08	1.0323	2.61E+01	2.402	1.2958	1.1864
	En İyi	3.02E-05	2.82E-11	3.71E-06	3.81E-84	0.709	5.81E-96	**4.561E-170**
F11	Ort.	4.00E+03	4.25E-11	1.14E-05	2.933E-82	1.863	3.6E-94	**4.35E-162**
	Std.	4.89E+03	9.38E-12	7.90E-06	5.20E-82	1.407	4.09E-94	8.60E-162
	En İyi	6.73E+02	**0.814E+02**	5.62E+02	7.49E+02	7.632E+02	6.398E+02	1.125E+03
F12	Ort.	7.32E+02	**1.27E+02**	6.593E+02	9.0295E+02	8.284E+02	8.395E+02	1.19E+03
	Std.	57.77	39.98	66.61	81.31	53.366	1.108E+02	39.684
	En İyi	**2.81E+02**	6.33E+02	3.13E+02	6.15E+02	3.35E+02	5.222E+02	9.386E+02
F13	Ort.	**4.22E+02**	6.88E+02	4.56E+02	7.33E+02	4.32E+02	5.88E+02	1.097E+03
	Std.	1.088E+02	0.39E+02	1.41E+02	0.957E+02	0.939E+02	0.518E+02	1.026E+02
	En İyi	9.93E+02	9.25E+02	9.46E+02	1.118E+03	9.302E+02	1.11E+03	**9.00E+02**
F14	Ort.	1.041E+03	9.45E+02	9.68E+02	1.16E+03	9.90E+02	1.13E+03	**9.40E+02**
	Std.	34.338	15.22	36.81	25.43	33.38	15.72	20.62
	En İyi	9.24E+02	9.28E+02	9.46E+02	1.16E+03	9.759E+02	1.096E+03	**9.00E+02**
F15	Ort.	1.007E+03	1.004E+03	9.763E+02	1.19E+03	1.003E+03	1.132E+03	**9.342E+02**
	Std.	83.93	5.60	16.05	13.38	19.18	23.23	13.41



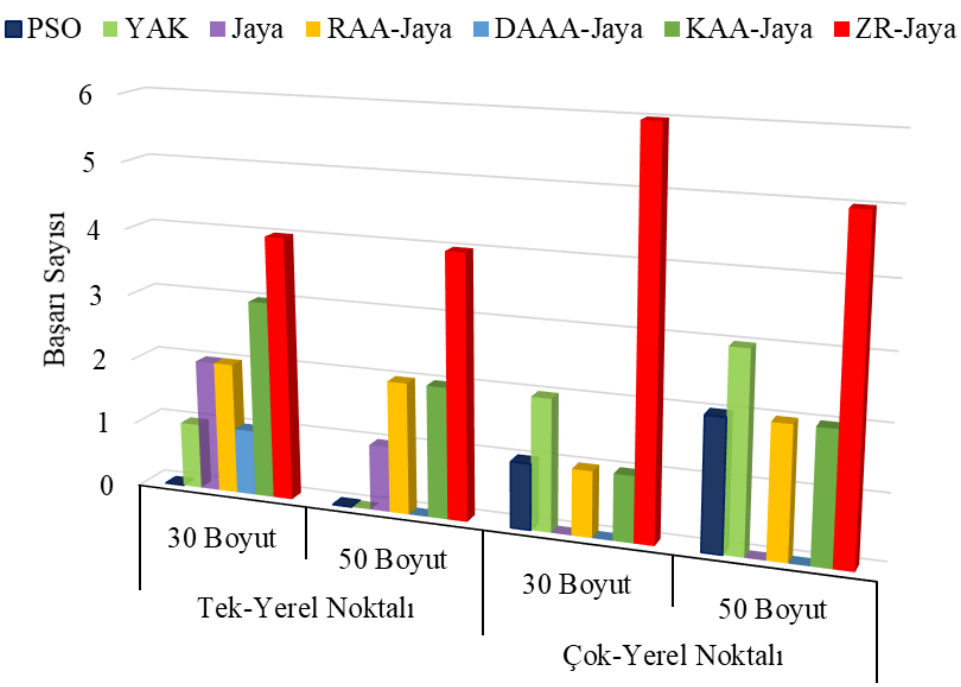
Şekil 2. Geliştirilen ZR-Jaya ile PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın 30 boyutlu fonksiyonlarda da en iyi sonuçlara göre yakınsama eğrisi (Convergence curve of proposed ZR-Jaya and PSO, YAK , Jaya, RIW-Jaya, LDIW Jaya and CIW-Jaya algorithms for 30-dimensional functions according to the best results)

Çizelge 4. Geliştirilen ZR-Jaya ile PSO, YAK , Jaya RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın 50 boyutta karşılaştırma sonuçları (Comparison results of proposed ZR-Jaya with PSO, YAK , Jaya, RIW-Jaya, LDIW-Jaya and CIW-Jaya algorithms in 50 dimensions)

		PSO	YAK	Jaya	RAA-Jaya	DAAA-Jaya	KAA-Jaya	ZR-Jaya
F1	En İyi	1.6350	1.8926E-06	1.751E-01	1.43E-74	13.94	4.233E-90	**1.5999E-164**
	Ort.	3.0018E+04	2.7932E-04	1.2792	3.62E-71	22.68	7.252E-87	**3.0597E-156**
	Std.	7.4570E+03	9.2967E-04	1.3048	7.07E-71	9.086	1.359E-86	1.6025E-155
F2	En İyi	1.0011E+03	1.6607E-06	4.71E-02	2.01E-74	5.044	1.127E-89	**1.7476E-169**
	Ort.	4.0928E+03	2.9763E-05	3.364E-01	7.23E-73	9.9979	1.298E-88	**7.6801E-162**
	Std.	2.1203E+03	8.3324E-05	4.37E-01	7.06E-73	4.234	1.246E-88	3.6927E-161
F3	En İyi	2	11	15	**0**	23	**0**	**0**
	Ort.	9.7042E+03	3.10667E+01	1.87011E+02	**0**	50.20	**0**	**0**
	Std.	8.3587E+03	1.38153+E01	4.091103+E02	0	35.27832195	0	0
F4	En İyi	1.6353E+02	8.2368	**1.0408**	4.84E+01	7.444E+01	4.851E+01	4.88507E+01
	Ort.	6.6136E+04	4.5024E+01	**1.81738E+01**	4.859E+01	1.193E+02	4.866 E+01	4.89360E+01
	Std.	2.3676E+05	3.4295E+01	3.02423E+01	0.112	25.775	1.124E-01	3.12E-02
F5	En İyi	1.9167E+02	7.2423	1.792826E+02	**0**	3.695E+02	**0**	**0**
	Ort.	3.26322E+02	2.18773E+01	3.03105E+02	**0**	5.177E+02	**0**	**0**
	Std.	5.38178E+01	7.9964	8.91580E+01	0	8.6E+01	0	0
F6	En İyi	1.2837E+01	**4.113E-01**	6.429E-01	6.667E-01	6.1492	0.673	7.064E-01
	Ort.	2.0154E+05	3.772	6.91	**6.68E-01**	15.625	6.7346E-01	9.867E-01
	Std.	8.0621E+05	1.9301	4.0974	6.999E-06	6.4391	5.65E-02	5.24E-02
F7	En İyi	0.3159	7.1344E-06	0.2271	**0**	1.1679	**0**	**0**
	Ort.	8.82163E+01	8.5E-03	0.6560	6.394E-02	1.367	2.692E-02	**3.3941E-04**
	Std.	9.13317E+01	1.47E-1	2.093E-01	7.01E-03	0.13087	9.41E-03	1.8E-03
F8	En İyi	5.1660E+03	**1.2760E+03**	6.5911E+03	1.60E+04	1.299E+04	1.549E+04	2.354162E+0
	Ort.	7.2270E+03	**2.3137E+03**	1.2361E+04	1.63E+04	1.405E+04	1.61E+04	2.802234E+0
	Std.	1.0632E+03	4.70449E+02	2.4081E+03	4.61E+02	7.104E+02	3.661E+02	1.8384E+02
F9	En İyi	3.2199	2.631E-03	2.190	2.224E-15	19.963	**0**	**0**
	Ort.	15.9103	4.27E-02	6.1597	2.224E-15	19.968	**0**	**0**
	Std.	4.9282	3.39E-02	4.4193	**0**	0.73E-02	0	0
F10	En İyi	9.262E-01	**1.0023E-06**	5.14E-04	38.55	12.364	33.871	4.934E+01
	Ort.	1.8711E+02	**7.6565E-05**	1.8242	42.58	16.0523	38.035	5.1887E+01
	Std.	1.3097E+02	8.5653E-05	5.2256	3.053	2.2503	3.29166	2.2970
F11	En İyi	6.7012E-01	5.823E-06	9.138E-02	8.255E-75	16.31	2.68E-90	**5.553E-166**
	Ort.	1.002E+04	2.6299E-05	1.73123	5.67E-72	43.17	3.436E-85	**9.936E-163**
	Std.	1.2675E+04	2.61E-05	1.7988	6.817E-72	17.312	6.63E-85	2.2227E-162
F12	En İyi	**4.339E+02**	1.576E+02	5.922E+02	8.858E+02	8.675E+02	8.039E+02	1.2681E+03
	Ort.	8.114E+02	**1.736E+02**	7.1944E+02	9.792E+02	8.867E+02	9.83E+02	1.3007E+03
	Std.	2.092E+02	22.682	1.040E+02	69.584	20.379	98.757	34.86
F13	En İyi	**5.33E+02**	9.223E+02	5.5122E+02	9.268E+02	5.952E+02	1.045E+03	5.945E+02
	Ort.	**6.582E+02**	1.072E+03	6.594E+02	1.047E+03	6.594E+02	1.221E+03	7.22E+02
	Std.	1.031E+02	82.558	75.9339	92.684	37.717	1.563E+02	79.24
F14	En İyi	1.008E+03	1.0077E+03	1.0345E+03	1.174E+03	1.0655E+03	1.189E+03	**9.0E+02**
	Ort.	1.054E+03	1.0389E+03	1.0803E+03	1.191E+03	1.1197E+03	1.21E+03	**9.69E+02**
	Std.	37.514	23.996	30.70	12.249	42.85	20.75	13.8
F15	En İyi	1.0331E+03	1.0068E+03	1.025E+03	**9.0E+02**	1.0542E+03	1.188E+03	**9.0E+02**
	Ort.	1.0726E+03	1.0405E+03	1.0567E+03	1.056E+03	1.098E+03	1.208E+03	**9.0E+02**
	Std.	49.159	23.487	25.23	1.30E+02	35.148	16.15	0



Şekil 3. Geliştirilen ZR-Jaya ile PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın 50 boyutlu fonksiyonlarda da en iyi sonuçlara göre yakınsama eğrisi (Convergence curve of proposed ZR-Jaya and PSO, YAK , Jaya, RIW-Jaya, LDIW-Jaya and CIW-Jaya algorithms for 50-dimensional functions according to the best results)



[5]

Şekil 4. Geliştirilen ZR-Jaya ile PSO, YAK, Jaya, RAA-Jaya, DAAA-Jaya ve KAA-Jaya'nın başarı sıralama sayıları (Success ranking points of the proposed ZR-Jaya with PSO, YAK, RIW-Jaya, LDIW-Jaya and CIW-Jaya and Jaya)

ETİK STANDARTLARIN BEYANI (DECLARATION OF ETHICAL STANDARDS)

Bu makalenin yazarı çalışmada kullandığı materyal ve yöntemlerin etik kurul izni ve/veya yasal özel bir izin gerektirmediğini beyan eder.

YAZARLARIN KATKILARI (AUTHORS' CONTRIBUTIONS)

Mehmet Fatih TEFEK: Deneyleri yapmış, sonuçlarını analiz etmiştir ve makalenin yazım işlemini gerçekleştirmiştir.

ÇIKAR ÇATIŞMASI (CONFLICT OF INTEREST)

Bu çalışmada herhangi bir çıkar çatışması yoktur.

KAYNAKLAR (REFERENCES)

[1] Yılmaz S. and Küçüksille E. U., "A new modification approach on bat algorithm for solving optimization problems", ***Applied Soft Computing***, 28: 259-275, (2015).

[2] Kennedy J. and Eberhart R., "Particle swarm optimization", ***Proceedings of ICNN'95 - International Conference on Neural Networks***, Perth, 4: 1942-1948, (1995).

[3] Karaboga D. and Akay B., "A Survey: Algorithms Simulating Bee Swarm Intelligence", ***Artificial Intelligence Review***, 31: 68-85, (2009).

[4] Venkata R. R. , "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems", ***International Journal of Industrial Engineering Computations***, 7: 19-34, (2016).

Venkata R. R. and Saroj A., "A self-adaptive multi-population based Jaya algorithm for engineering optimization", ***Swarm and Evolutionary Computation***, 37: 1-26, (2017).

[6] Bhoye M., Pandya M. H., Valvi S., Trivedi I. N., Jangir P., and Parmar S. A., "An emission constraint Economic Load Dispatch problem solution with Microgrid using JAYA algorithm", ***2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS)***, Nagercoil, 497-502, (2016).

[7] Huang C., Wang L., Yeung R. S., Zhang Z., Chung H. S., and Bensoussan A., "A Prediction Model-Guided Jaya Algorithm for the PV System Maximum Power Point Tracking", ***IEEE Transactions on Sustainable Energy***, 9 (1): 45-55, (2018).

[8] Gao K., Zhang Y., Sadollah A., Lentzakis A., and Su R., "Jaya, harmony search and water cycle algorithms for solving large-scale real-life urban traffic light scheduling problem", ***Swarm and Evolutionary Computation***, 37: 58-72, (2017).

[9] Öztürk H. T. and Türkeli E., "Tabanında Anahtar Kesiti Bulunan Betonarme İstinat Duvarlarının Jaya Algoritmasıyla Optimum Tasarımı", ***Politeknik Dergisi***, 22 (2) 2147-9429, (2019).

[10] Gao K., Sadollah A., Zhang Y., Su R., and Li K. G. J., "Discrete Jaya algorithm for flexible job shop scheduling problem with new job insertion", ***14th International Conference on Control, Automation, Robotics and Vision (ICARCV)***, Singapore, 1-5, (2016).

[11] Wang L. and Huang C., "A novel Elite Opposition-based Jaya algorithm for parameter estimation of photovoltaic cell models", ***Optik***, 155: 351-356, (2018).

[12] Migallón H., Jimeno-Morenilla A., Sánchez-Romero J. L., and Belazi A., "Efficient parallel and fast convergence chaotic Jaya algorithms", ***Swarm and Evolutionary Computation***, 56: 1-17, (2020).

[13] Xin J., Chen G., and Hai Y., "A Particle Swarm Optimizer with Multi-stage Linearly-Decreasing Inertia Weight", ***International Joint Conference on Computational Sciences and Optimization***, Sanyan, 505-508, (2009).

[14] Bansal J. C., Singh P. K., Saraswat M., Verma A., Jadon S. S., and Abraham A., "Inertia Weight strategies in Particle Swarm Optimization", ***Third World Congress on Nature and Biologically Inspired Computing***, Salamanca, 633-640, (2011).

[15] Eberhart R. C. and Yuhui S., "Tracking and optimizing dynamic systems with particle swarms", ***Proceedings of the 2001 Congress on Evolutionary Computation***, Seul, 94-100, (2001).

[16] Feng Y., Teng G., Wang A., and Yao Y., "Chaotic Inertia Weight in Particle Swarm Optimization", ***Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)***, Kumamato, 475-475, (2007).

[17] Shi Y. and Eberhart R., "A modified particle swarm optimizer", ***IEEE World Congress on Computational Intelligence***, Anchorage, 69-73, (1998).

[18] Arumugam M. S. and Rao M. V. C., "On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems", ***Applied Soft Computing,*** 8 (1): 324-336, (2008).

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

[20] Fan S.K. S. and Chiu Y.Y., "A decreasing inertia weight particle swarm optimizer", ***Engineering Optimization***,39 (2): 203-228, (2007).

[21] Alatas B., Akin E., and Ozer A., "*Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları*", ***XII. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi***, Eskişehir, (2007).

[22] Aydilek İ. B., "An Ensemble inertia Weight Calculation Strategy in Particle Swarm Optimization Algorithm", ***Selçuk Üniversitesi Mühendislik, Bilim ve Teknoloji Dergisi (SUJEST)***, 6 (4): 544-558, (2018).

[23] Rauf H. T., Malik S., Shoaib U., Irfan M. N., and Lali M. I., "Adaptive inertia weight Bat algorithm with Sugeno-Function fuzzy search", ***Applied Soft Computing***,90:106159, (2020).

[24] Yue X. and Zhang H., "Modified hybrid bat algorithm with genetic crossover operation and smart inertia weight for multilevel image segmentation", ***Applied Soft Computing***, 90: 106157, (2020).

[25] Ekinci S., Hekimoğlu B., Demirören A., and Eker E., "Speed Control of DC Motor Using Improved Sine Cosine Algorithm Based PID Controller", ***3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)***, Ankara, 1-7, (2019).

[26] Ramli M. R., Abas Z. A., Desa M. I., Abidin Z. Z., and Alazzam M. B., "Enhanced convergence of Bat Algorithm based on dimensional and inertia weight factor", ***Journal of King Saud University - Computer and Information Sciences***, 31 (4): 452-458, (2019).

[27] Gan C., Cao W., Wu M., and Chen X., "A new bat algorithm based on iterative local search and stochastic inertia weight", ***Expert Systems with Applications***, 104: 202-212, (2018).

[28] Yılmaz S. and Kucuksille E. U., "Improved Bat Algorithm (IBA) on Continuous Optimization Problems", ***Lecture Notes on Software Engineering***, 1 (3): 279-283, 2013.

[29] Toz G., Yücedağ İ., and Erdoğmuş P., "A fuzzy image clustering method based on an improved backtracking search optimization algorithm with an inertia weight parameter," ***Journal of King Saud University - Computer and Information Sciences***, 31 (3): 295-303, (2019).

[30] Wu Z.S., Fu W.P., and Xue R., "Nonlinear Inertia Weighted Teaching-Learning-Based Optimization for Solving Global Optimization Problem", ***Computational Intelligence and Neuroscience***, 2015: 1-15, (2015).

[31] Shukla A. K., Singh P., and Vardhan M., "An adaptive inertia weight teaching-learning-based optimization algorithm and its applications", ***Applied Mathematical Modelling***, 77: 309-326, (2020).

[32] Suganthan P. N., Hansen N., Liang J. J., Deb K., Chen Y.P., Auger A., Tiwari S.., "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization", ***Natural Computing***, 341-357, (2005).

[33] Kıran M. S. and Gündüz M., "A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems", ***Applied Soft Computing***, 13(4):2188-2203, (2013).

[34] Wang D., Tan D., and Liu L., "Particle swarm optimization algorithm: an overview", ***Soft Computing***, 22 (2): 387-408, (2018).

[35] Shi Y. and Eberhart R. C., "Empirical study of particle swarm optimization", ***Proceedings of the 1999 Congress on Evolutionary Computation***, Washington, 3:1945-1950, (1999).

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