Fırat Üniversitesi Müh. Bil. Dergisi
32(2), 369-380, 2020

Güncel İkili Optimizasyon Algoritmalarının Kısıtsız Kıyaslama Fonksiyonlarındaki Performans Karşılaştırmaları

Erkan TANYILDIZI[1] , Abdullah ÇELİK[2*]
[1,2] Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü, Elazığ
etanyildizi@gmail.com, acelik23@gmail.com

(Geliş/Received: 20/02/2020; Kabul/Accepted:18/06/2020)

Öz: İkili optimizasyon algoritmaları, 0 ve 1 gibi kesin sonuçlar ürettiğinden algoritma geliştiricileri için ilgi alanı olmuştur. Bu çalışmada ikili problemleri çözmek için geliştirilen algoritmalardan başarımı yüksek olan optimizasyonları belirlemek amacıyla kıyaslama fonksiyonları kullanarak test yapılmıştır. İkili optimizasyonlar, optimum çözümün garanti edilemediği algoritmaların aksine 0 ve 1 gibi sonuç üreterek kolaylık sağlamaktadır. Bu kolaylığı daha etkin kullanmak amacıyla aktif olarak kullanılan optimizasyonlar ile hibrid yöntemlerin gelişimi sağlanmıştır. Bu yöntemlerden başarılı olanların belirlenmesi çalışmalarımızda rehber olması açısından önem arz etmektedir. Bunu sağlamak için ikili optimizasyon problemleri, kullanılan yöntemler, modifikasyon teknikleri hakkında literatür taraması yapıldıktan sonra 13 adet kısıtsız kıyaslama fonksiyonu kullanarak İkili Parçacık Sürü Optimizasyonu (BPSO), İkili Gri Kurt Optimizasyonu (BGWO), İkili Yusufçuk Algoritması (BDA), İkili Yarasa Algoritması (BBA) ve hibrid BPSOGSA algoritmaları test edilmiş ve optimuma en yakın sonuç veren algoritmaların tespiti yapılmıştır. Elde edilen test sonuçlarına göre optimuma en yakın sonucu BBA vermiştir.

Anahtar kelimeler: İkili optimizasyon, kısıtsız kıyaslama fonksiyonları, modifikasyon teknikleri, başarım değerlendirmesi.

Performance Comparisons of Current Binary Optimization Algorithms in Unconstrained Benchmark Functions

Abstract: Binary optimization algorithms have been of interest for algorithm developers as they produce precise results such as 0 and 1. In this study, testing was done by using benchmark functions to determine the optimizations with high performance from algorithms developed to solve binary problems. Binary optimizations provide convenience by producing results like 0 and 1, unlike algorithms where the optimum solution cannot be guaranteed. In order to use this facility more effectively, the development of hybrid methods has been provided with the optimizations used actively. The determination of the successful ones is important in terms of being a guide in our studies. To achieve this, after searching the literature on binary optimization problems, methods, modification techniques, using 13 unconstrained benchmark functions, Binary Particle Swarm Optimization (BPSO), Binary Gray Wolf Optimization (BGWO), Binary Dragonfly Algorithm (BDA), Binary Bat Algorithm (BBA) and hybrid BPSOGSA algorithms were tested and algorithms that gave the best results were determined. According to the test results obtained, BBA gave the closest result to the optimum.

Key words: Binary optimization, unconstrained benchmark functions, modification techniques, performance evaluation.

1. Giriş

Optimizasyon, en iyiye ulaşma amacıyla verilen amaçlar için belirli kısıtlamalar kullanılarak en iyi çözümün elde edilme sürecidir. Bu nedenle karşılaşılan problemlerin çözümü için kullanılan metot ve toleransa bağlı olarak birçok optimizasyon teknikleri farklı alanlara uygulanmıştır [1]. Optimizasyon işlemine maliyetin azaltılıp karın artırılması, zamanın faydalı kullanılması, enerji verimliliğinin artırılması, seri üretimi gerçekleştirilen herhangi bir ürünün sayısının artırılması ya da elektronik olarak çalışan bir cihazın boyutunun küçülmesi gibi çok farklı alanlarda ihtiyaç duyulmaktadır. Bu nedenle optimizasyon algoritmaları ile ilgili çalışmalar yıllar önce başlamış ve popülerliği giderek artacak şekilde devam etmektedir [2].

Günümüzde, optimizasyonlar sosyal bilimlerden matematik bilimine, mühendislikten, fizibiliteye kadar uzanan geniş bir alanda kullanılmakta ve bu alanlardaki problemleri gerçeğe en yakın şekilde sürekli sonuçlar ile çözmektedir. Birçok gerçek hayat problemi sürekli optimizasyonlarla çözüldüğü gibi ikili optimizasyon yöntemleri ile daha net sonuçlar ortaya koyarak çözümlenebilir. Bu sayede yanıtların evet/hayır, seçme/seçmeme ya da 0-1 değerleri ile algoritmalarda kesin sonuçların alınmasını sağlamaktadır. Bu durum sağlandığında ise

[*] Sorumlu yazar: acelik23@gmail.com Yazarların ORCID Numarası:[1] 0000-0003-2973-9389, [2] 0000-0002-3430-4630

algoritmaların başarımının değerlendirilmesi ve test problemleri üzerinde denenmesi daha net sonuçlar vermektedir [3].

Literatürlerde ikili problemleri çözmek için sürekli bir algoritma gerektirecek farklı yöntemler vardır. Bu yöntemlerden bazıları algoritmanın yapısını değiştirirken, diğerleri algoritmanın mekanizmasını ikili arama alanlarında korur. Literatürlerde Genetik Algoritma (GA), Harmoni Arama (HS), Diferansiyel Evrim (DE), Parçacık Sürü Optimizasyonu (PSO) ve Yerçekimi Arama Algoritması (GSA) gibi stokastik optimizasyon algoritmaları ikili problemleri çözmek için uyarlanmıştır [4].

1960 ve 1970'lerde John Holland ve arkadaşları tarafından geliştirilen ve ikili optimizasyon algoritmaların ilki olarak bilinen Genetik Algoritma (GA), Charles Darwin'in doğal seleksiyon teorisine dayanan biyolojik evrimin bir modeli veya soyutlamasıdır. Adaptif ve yapay sistemlerin çalışmasında çaprazlama ve birey üretimi, mutasyon ve seçimi ilk kullanan Holland oldu [5]. Bu genetik operatörler, bir problem çözme stratejisi olarak genetik algoritmanın önemli bir bölümünü oluşturmaktadır. Algoritmada kullanılan parametreler ikili yapıda temsil edilmektedir. O zamandan beri, genetik algoritmaların birçok çeşidi geliştirilmiş ve grafik renklendirmeden desen tanımaya, seyahat eden satıcı sorunu gibi ayrık sistemlerden havacılık mühendisliğinde hava folyosunun verimli tasarımı gibi sürekli sistemlere ve finansal piyasadan çok amaçlı mühendislik optimizasyonuna kadar çok çeşitli optimizasyon problemlerine uygulanmıştır [6].

İkili HS algoritması, ikili optimizasyonu gerçekleştirmek için bir dizi uyum arama kaygısı ve perde ayarlama kuralları kullanır. Ayrıca, ikili DE ayrık problemleri çözmek için bir olasılık tahmin operatörü kullanır. İkili PSO ve İkili GSA mekanizması oldukça benzerdir, çünkü her ikisi de ikili problemleri çözmek için transfer fonksiyonlarını kullanır[4].

Bu çalışmanın ikinci bölümünde ikili optimizasyonların kullandığı modifikasyon teknikleri ve yöntemler anlatılmıştır. Üçüncü bölümde ikili optimizasyon kullanılarak hibridize edilmiş algoritmalar hakkında bilgi verilmiştir. Dördüncü bölümde optimizasyonlar arasındaki performans değerlendirmesini yapmak için kullanılacak kısıtsız kıyaslama fonksiyonları tanıtılmıştır. Beşinci bölümde ikili optimizasyonlar kullanılarak geliştirilen Hibrid optimizasyonlarının performans değerlendirmesi için test sonuçları oluşturulmuştur. Altıncı bölümde elde edilen sonuçların değerlendirilmesi yapılmıştır. Yedinci bölümde ise çalışmadan elde edilen sonuçların değerlendirmesi yapılmıştır.

2. İkili Optimizasyonlar İçin Kullanılan Modifikasyon Teknikleri

Literatürde, temel problemler için aday çözüm üretme amacıyla modifikasyon yöntemlerinin kullanıldığı görülmektedir. İkili problemlerin çözümü için çaprazlama ve mutasyon operatörleri ile bit dizilerini kullanarak aday çözüm ürettiğinden genetik algoritma gibi yöntemlerin kullanımı kolaylık sağlamaktadır. Genetik algoritma haricindeki algoritmalar, karar değişkenleri sürekli değişen değerler alan problemlerin çözümünde kullanılmak için tasarlandığından ikili değerlere dönüşümün sağlanması gerekmektedir. Bu değişiklikler yöntemin karakteristiğine ve çalışma şekline göre farklılık gösterir. Bu nedenle, aşağıdaki gibi bazı modifikasyon teknikleri önerilmiştir [7].

• **Transfer fonksiyonu:** Bu fonksiyon, konum vektörünün elemanlarını 0'dan 1'e değiştirme olasılığını tanımlar (veya tam tersi). Bu teknik ilk olarak PSO'ya uygulanmıştır. Transfer fonksiyonları parçacıkları ikili bir alanda hareket etmeye zorlar. Bir transfer fonksiyonunun aralığı, bir parçacığın konumunu değiştirmesi olasılığını temsil ettiklerinden, [0,1] aralığında sınırlanır. Hızın büyük bir mutlak değeri için pozisyon değiştirme olasılığı yüksek olur. Hızları için büyük mutlak değerleri olan parçacıklar en iyi çözümden uzaktır. Bu nedenle bir sonraki yinelemede konumlarını değiştirir[8].

• **Açı modülasyonu:** Bu teknik telekomünikasyon endüstrisinden türetilir ve sinyal işleme alanında kullanılır. Bu yöntemin ardında yatan fikir, ikili diziler oluşturmak için bir sinüs ve kosinüs fonksiyonunun kullanılmasıdır. Fonksiyonun dört gerçek değerli parametresi vardır ve her parametre seti sadece bir ikili diziyi temsil eder. Bu, ikili bir arama boşluğunun eşdeğer dört boyutlu sürekli boşluğa dönüştürülmesini kolaylaştırır[9].

• **Kuantumdan ilham alan bitler:** İki durumlu bir kuantum bilgisayarında saklanabilecek en küçük bilgi birimine kuantum biti denir. Kuantum bilgisayarlar, kuantumdan ilham alan bitlerini (Q-bitleri) manipüle ederek ve bunlara bir dizi kuantum operatörü uygulayarak çalışır. Bir Q-bit, "1", "0" durumlarını veya bu ikisinin üst üste binmesini sunabilir. Meta-sezgisel arama algoritması, yapılarını oluşturmak için kuantumdan ilham alan bitleri kullanır. [10].

• **Genetik operatörler:** Arama alanının ikili özelliği, evrimsel algoritmalarda ikili çapraz geçiş ve takas operatörü gibi genetik operatörlerin kullanılmasına izin verir. Örneğin Öztürk ve ark. genetik algoritmaların özel arama stratejileri tasarlamak için kullanıldığı ABC algoritmasının etkin bir ikili varyantını önermişlerdir[11,12].

• **İkili işleçler:** İkili uzayda, ikili optimizasyon problemlerini çözmek için, her vektörün elemanları mantıksal sayı olarak kabul edildiğinden XOR mantık operatöründen yararlanılır [13].

• **Farklılık ölçüsü:** İkili yapıların benzerlik/farklılığının ölçülmesi, kümeleme ve sınıflandırma gibi birçok problemde kritik rol oynar. İki nesne arasındaki benzerlik miktarını nicelemek, birbirlerine ne kadar benzediklerini veya birbirlerinden ne kadar uzakta olduklarını, iki yapının bitleri arasında ortak bir örüntüyü paylaşma derecesine benzer görülmesine göre bulunur [14].

• **Sezgisel yöntemler:** Bu yöntemlerde hedef problemin karmaşıklığı ve ölçeğine göre, uygun bir arama stratejisi tasarlamak için seçilir[3].

Hedef sorunların karmaşıklığına ve ölçeğine bağlı olarak, yukarıda bahsedilen yöntemlerden herhangi biri uygun bir arama stratejisi tasarlamak için seçilebilir. İlk iki yaklaşımı izleyen ikili algoritmalar, sürekli arama motorlarıyla çalışır. Açı modülasyon tekniği herhangi bir ikili problemi dört boyutlu bir sürekli probleme indirgese de, ikili problemin boyutu yüksekse, sürekli problemin çözümünün son derece doğru olması gerekir. Ancak bu yöntemler düşük ölçekli problemler için etkilidir. Diğer yandan, kuantumdan ilham alan bitler ve ikili veya genetik operatörler, yenilikçi modern ikili algoritmaların geliştirilmesinde kullanılan çok yönlü tekniklerdir. Ayrıca, ikili dizilerin farklılığını ölçmeye dayanan teknikler, ikili algoritmaların geleneksel arama stratejileri ile donatılmasını sağlar [11].

3. Materyal ve Metot

Gerçek yaşam ikili optimizasyon problemleri genellikle yüksek boyutlu, çok modlu ve çözülmesi zor problemlerdir. Bu, birçok çalışmayı modern ikili algoritmalar geliştirmek için motive etmiştir[11]. Bu amaçla ikili optimizasyon algoritmaları, İkili Parçacık Sürü Optimizasyonu (BPSO), İkili Karınca Kolonisi Optimizasyonu (BACO), İkili Gri Kurt Optimizasyonu (BGWO), İkili Balina Optimizasyon Algortiması (BWOA), İkili Yapay Arı Kolonisi (BABC), İkili Ateş Böceği Algoritması (BFA), İkili Diferansiyel Algoritması (BDE), İkili Yusufçuk Algoritması (BDA), İkili Kedi Sürüsü Optimizasyonu (BCSO), İkili Yarasa Algoritması (BBA), İkili Yerçekimi Arama Algoritması (BGSA) olarak geliştirilmiştir. Çalışmada BBA, BPSO, hibrid BPSOGSA, BGWO ve BDA ikili optimizasyon algoritmaları kısıtsız kıyaslama fonksiyonlarında kullanılmıştır.

3.1 İkili parçacık sürü optimizasyonu (BPSO)

İkili parçacık sürüsü optimizasyonu (BPSO), ikili optimizasyon görevlerini çözmek için önerilen parçacık sürüsü optimizasyonunun (PSO) ikili versiyonudur. PSO gibi BPSO da hız ve pozisyon güncellemesinde kişisel en iyi (en iyi) ve küresel en iyi (en iyi) çözümleri içerir. Her parçacık (çözelti) için hız olarak güncellenir [15]:

$$v_i^d(t+1) = wv_i^d(t) + c_1 r_1 \left(pbest_i^d(t) - x_i^d(t)\right) + c_2 r_2 \left(gbest^d(t) - x_i^d(t)\right) \quad (1)$$

Daha sonra hız, Denklem 2 kullanılarak olasılık değerine dönüştürülür ve parçacığın konumu Denklem 3'te gösterildiği gibi güncellenir [16]:

$$S\left(v_i^d(t+1)\right) = \frac{1}{1+\exp(-v_i^d(t+1))} \quad (2)$$

$$x_i^d(t+1) = \begin{cases} 1, if\ rand < S\left(v_i^d(t+1)\right) \\ 0, otherwise \end{cases} \quad (3)$$

3.2 PSOGSA'nın ikili sürümü (BPSOGSA)

Orijinal PSOGSA'da, ajanlar sürekli gerçek etki alanına sahip konum vektörlerine sahip oldukları için sürekli olarak arama alanı etrafında hareket edebilirler. Arama aracılarının ikili bir arama alanında hareket etmesini istemek için, konum güncellemesini değiştirmemiz gerekir. Bir maddenin pozisyonunu hız olasılığı ile değiştirmek için bir transfer fonksiyonuna da ihtiyaç duyulmaktadır. Transfer fonksiyonları hız değerlerini pozisyonların güncellenmesi için olasılık değerleriyle eşleştirir. Bir aktarım işlevi, hızın büyük bir mutlak değeri

için pozisyonu değiştirme olasılığının yüksek olmasını sağlamalıdır. Ek olarak, hızın küçük bir mutlak değeri için pozisyonun değiştirilmesi için küçük bir olasılık sunmalıdır. Ayrıca, bir transfer fonksiyonunun aralığı [0, 1] aralığında sınırlanmalı ve hızın artmasıyla arttırılmalıdır[4].

$$S\left(V_i^d(t)\right) = \left|\tanh(v_i^d(t))\right| \tag{4}$$

Bu denklemi, BPSOGSA'daki ajanların hızlarını konum vektörlerinin elemanlarını çevirme olasılıklarıyla eşleştirmek için kullanırız. Olasılıkları hesapladıktan sonra, ajanlar Denklem 5'te sunulan kurallara göre pozisyonlarını günceller[4].

$$\begin{cases} if\ rand < S\left(V_i^d(t+1)\right)\ then\ \ x_i^d(t+1) = complement\ (x_i^d(t)) \\ \qquad\qquad else\ \ x_i^d(t+1) = x_i^d(t) \end{cases} \tag{5}$$

3.3 İkili gri kurt optimizasyonu (BGWO)

Genel olarak, GWO sürekli optimizasyon problemlerini çözmek için tasarlanmıştır. Özellik seçimi gibi ikili optimizasyon sorunları için GWO'nun ikili sürümü gerekir. Bu yaklaşıma göre, kurt pozisyonunu Denklem 6'daki gibi güncellemek için geçiş operatörünü kullanır[17,18]:

$$X(t+1) = Crossover(Y_1, Y_2, Y_3) \tag{6}$$

Burada Crossover (Y1, Y2 ve Y3), çözeltiler arasındaki geçiş işlemidir ve Y1, Y2 ve Y3, sırasıyla alfa, beta ve delta kurtlarının hareketinden etkilenen ikili vektörlerdir. BGWO'da, Y1, Y2 ve Y3 sırasıyla Denklem 7, 10 ve 13 kullanılarak tanımlanır[17,18].

$$Y_1^d = \begin{cases} 1,\ if\ (X_\alpha^d + bstep_\alpha^d) \geq 1 \\ 0,\ otherwise \end{cases} \tag{7}$$

Burada X_α^d alfa pozisyonudur, d arama alanının boyutudur ve $bstep_\alpha^d$ şu şekilde ifade edilebilen ikili adımı temsil eder[17,18]:

$$bstep_\alpha^d = \begin{cases} 1,\ if\ \ cstep_\alpha^d \geq r_3 \\ 0,\ otherwise \end{cases} \tag{8}$$

Buradaki r3, [0, 1] 'de rastgele bir vektördür ve $cstep_\alpha^d$, Denklem 9' da olduğu gibi hesaplanabilen sürekli değerli adım boyutunu belirtir[17,18].

$$cstep_\alpha^d = \frac{1}{1+\exp(-10(A_1^d.D_\alpha^d-0.5))} \tag{9}$$

Burada A_1^d ve D_α^d, GWO'ya ait denklemler ile belirlenir[17,18].

$$Y_2^d = \begin{cases} 1,\ if\ (X_\beta^d + bstep_\beta^d) \geq 1 \\ 0,\ otherwise \end{cases} \tag{10}$$

Burada X_β^d beta konumudur, d arama alanının boyutudur ve $bstep_\beta^d$ şu şekilde ifade edilebilen ikili adımı temsil eder[17,18]:

$$bstep_{\beta}^{d} = \begin{cases} 1, \ if \ \ cstep_{\beta}^{d} \geq r_4 \\ 0, \ otherwise \end{cases} \tag{11}$$

Burada r4, [0, 1] 'de rastgele bir vektördür ve $\mathrm{cstep}_{\beta}^{\mathrm{d}}$, Denklem 12' de olduğu gibi hesaplanabilen sürekli değerli adım boyutunu belirtir[17,18].

$$cstep_{\beta}^{d} = \frac{1}{1+\exp(-10(A_1^d.D_{\beta}^d-0.5))} \tag{12}$$

Burada $\mathrm{A}_1^{\mathrm{d}}$ ve $\mathrm{D}_{\beta}^{\mathrm{d}}$, GWO'ya ait denklemler ile belirlenir[17,18].

$$Y_3^d = \begin{cases} 1, \ if \ (X_{\delta}^d + bstep_{\delta}^d) \geq 1 \\ 0, \ otherwise \end{cases} \tag{13}$$

Burada $\mathrm{X}_{\delta}^{\mathrm{d}}$ beta konumudur, d arama alanının boyutudur ve $\mathrm{bstep}_{\delta}^{\mathrm{d}}$ şu şekilde ifade edilebilen ikili adımı temsil eder[17,18]:

$$bstep_{\delta}^{d} = \begin{cases} 1, \ if \ \ cstep_{\delta}^{d} \geq r_5 \\ 0, \ otherwise \end{cases} \tag{14}$$

Burada r5, [0, 1] 'de rastgele bir vektördür ve $\mathrm{cstep}_{\delta}^{\mathrm{d}}$, Denklem 15' de olduğu gibi hesaplanabilen sürekli değerli adım boyutunu belirtir[17,18].

$$cstep_{\delta}^{d} = \frac{1}{1+\exp(-10(A_1^d.D_{\delta}^d-0.5))} \tag{15}$$

Burada $\mathrm{A}_1^{\mathrm{d}}$ ve $\mathrm{D}_{\delta}^{\mathrm{d}}$, GWO'ya ait denklemler ile belirlenir. Y1, Y2 ve Y3 elde edildikten sonra, kurdun yeni pozisyonu Denklem 16'daki gibi çaprazlama işlemi kullanılarak güncellenir[17,18]:

$$X^d(t+1) = \begin{cases} Y_1^d, if \ r_6 < \frac{1}{3} \\ Y_2^d, if \ \frac{1}{3} \leq r_6 \leq \frac{2}{3} \\ Y_3^d, otherwise \end{cases} \tag{16}$$

Burada d, arama alanının boyutu ve r6, [0, 1] arasında eşit olarak dağıtılmış rastgele bir sayıdır[17,18].

Başlangıçta, gri kurt popülasyonu rastgele başlatılır (bit 1 veya 0). Daha sonra, her kurdun zindeliği değerlendirilir. En iyi, ikinci en iyi ve üçüncü en iyi çözümler alfa, beta ve delta olarak tanımlanır. Her kurt için, Y1, Y2 ve Y3 sırasıyla Denklem 7,10 ve 13 kullanılarak hesaplanır. Daha sonra, kurt pozisyonu Y1, Y2 ve Y3 arasındaki geçit uygulanarak güncellenir. Daha sonra, her kurtun zindeliği değerlendirilir. Yinelemeli olarak alfa, beta ve delta konumları güncellenir. Algoritma, sonlandırılan kriter karşılanana kadar tekrarlanır. Sonunda, alfa çözümü en uygun özellik olarak seçilir[17,18].

3.4 İkili yarasa algoritması (BBA)

İkili bir optimizasyon algoritmasının arama ajanları (parçacıkları) sadece bu hiper küpün daha yakın ve daha uzak köşelerine çeşitli sayıda bit çevirerek kayabilir. Bu nedenle, BA'nın ikili versiyonunu tasarlarken, hız ve pozisyon güncelleme sürecinin bazı temel kavramları değiştirilmelidir. BA'nın sürekli versiyonunda, yapay yarasalar, sürekli gerçek alan içindeki konum ve hız vektörlerini (veya güncellenmiş konum vektörlerini) kullanarak arama alanı etrafında hareket edebilirler. Sonuç olarak, konum güncellemesi kavramı denklem kullanarak konumlara hız ekleyerek yarasalar için kolayca uygulanabilir [19].

$$S\left(v_i^k(t)\right) = \frac{1}{1+e^{-v_i^k(t)}} \tag{17}$$

Burada $v_i^k(t)$, t tekrarında k. boyutunda i parçacığının hızıdır[19].

Transfer fonksiyonlarını kullanarak olasılıkları hesapladıktan sonra, parçacıkların pozisyonunu Denklem 26'daki gibi güncellemek için yeni bir pozisyon güncelleme denklemi gereklidir [19]:

$$x_i^k(t+1) = \begin{cases} 0 & If \quad rand < S(v_i^k(t+1)) \\ 1 & If \quad rand \geq S(v_i^k(t+1)) \end{cases} \tag{18}$$

Bu yöntemin bir dezavantajı vardır, zira parçacıklar 0 veya 1 değerlerini almaya zorlanmaktadır. Böylece, hız değerleri arttığında parçacıklar pozisyonlarında değişmeden kalır. Bununla birlikte, bir transfer fonksiyonu tasarlamak için yukarıda belirtilen kavramlara göre, daha iyi bir yol, konumlarını değiştirmek için yüksek hızlı parçacıkları zorunlu kılmaktır. Bunu Denklem 19 ve 20'de olduğu gibi yapmak için v şeklinde bir aktarım işlevi ve konum güncelleme kuralı önerilmektedir[19].

$$V\left(v_i^k(t)\right) = \left|\frac{2}{\pi}\arctan(\frac{\pi}{2}v_i^k(t))\right| \tag{19}$$

$$x_i^k(t+1) = \begin{cases} (x_i^k(t))^{-1} & If \quad rand < V\left(v_i^k(t+1)\right) \\ x_i^k(t) & If \quad rand \geq V\left(v_i^k(t+1)\right) \end{cases} \tag{20}$$

3.5 İkili yusufçuk algoritması (BDA)

İkili bir optimizasyon probleminde, arama alanı hiper küp olarak kabul edilir, burada bir kişi konum vektörünün bir veya daha fazla bitini x = {x_1, x_2, ..., x_d} değiştirerek konumunu bir konumdan diğerine değiştirebilir. Orijinal Yusufçuk Algoritması (DA) sürekli optimizasyon problemlerini ele almak için tasarlandığından, bireyin pozisyonu mevcut vektörün vektörü adım vektörüne eklenerek güncellenir. Ancak bu mekanizma, özellik seçimi gibi bir ikili optimizasyon problemini ele almak için kullanılamaz[20].

Transfer fonksiyonları, mevcut iterasyonda, d. boyutunda i. arama maddesinin adım vektörünün (hız) değerine dayanarak bir konumun elemanlarını 0 veya 1 olarak değiştirme olasılığını üretmek için kullanılır. Denklem 21'de, sürekli konumların ikili olarak değiştirilme olasılığını hesaplamak için kullanılmıştır[20].

$$T\left(v_d^i(t)\right) = \left|v_d^i(t)/\sqrt{1+(v_d^i(t))^2}\right| \tag{21}$$

Denklem 29'dan elde edilen $T\left(v_d^i(t)\right)$ sonucu daha sonra Denklem 22'ye göre pozisyon vektörünün i. elemanını 0 veya 1'e dönüştürmek için kullanılır[20].

$$X(t+1) = \begin{cases} \neg X_t & r < T\left(v_k^i(t)\right) \\ X_t & r \geq T\left(v_k^i(t)\right) \end{cases} \tag{22}$$

4. Performans Karşılaştırması İçin Kullanılan Kısıtsız Kıyaslama Fonksiyonları

Çalışmamızda optimizasyonların performansını farklı açılardan test etmek için farklı özelliklere sahip iki test fonksiyonu grubu kullanılmıştır. Bunlar: Tek modlu (Tablo 1) ve çok modlu (Tablo 2) fonksiyonlardır. İsimlerinin ima ettiği gibi, tek modlu test fonksiyonlarının tek bir optimum özelliği vardır, böylece bir algoritmanın sömürüsü ve yakınsamasını kıyaslayabilirler [16].

Tablo 1. Tek modlu kıyaslama fonksiyonları[16]

Fonksiyon Adı	Fonksiyon Denklemi	Değişken Boyutu	Aralık	Optimum Değer
Sphere	$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-5.12,5.12]	0
Schwefel 2.22	$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
Schwefel 1.2	$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[-100,100]	0
Schwefel 2.21	$F_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[-100,100]	0
Generalized Rosenbrock	$F_5(x) = \sum_{i=1}^{n-1} [100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
Step	$F_6(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$	30	[-100,100]	0
Quartic	$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1.28,1.28]	0

Çok-modlu test fonksiyonlarının birden fazla optimum özelliği vardır, bu da onları tek-işlevli fonksiyonlardan daha zorlaştırır. Optimalardan birine küresel optimum, geri kalanına ise yerel optima denir. Bir algoritma tüm yerel optimanın küresel optimum yaklaşıma yaklaşmasını ve belirlemesini engellemelidir. Bu nedenle, algoritmaların keşfi ve lokal optimadan kaçınması, çok modlu test fonksiyonları ile test edilebilir [16].

Tablo 2.Çok modlu kalite fonksiyonları[16]

Fonksiyon Adı	Fonksiyon Denklemi	Değişken Boyutu	Aralık	Optimum Değer
Generalized Schwefel	$F_8(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[-500,500]	-418.98xn
Generalized Rastrigin	$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
Ackley	$F_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0
Griewank	$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
Penalized Functions 1	$F_{12}(x) = \frac{\pi}{n}\{\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\ [10sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10,100,4)$ $y_i = 1 + \frac{x_i + 1}{4}\ u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i > -a \end{cases}$	30	[-50,50]	0
Penalized Functions 2	$F_{13}(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\ [1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2\ [1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	30	[-50,50]	0

5. Kısıtsız Kıyaslama Fonksiyonları ile Yapılan Deneyler

Yapılan literatür araştırmasında elde edilen optimizasyon problemlerini kıyaslamak için kalite test fonksiyonları kullanılmıştır. Kısıtsız kıyaslama fonksiyonlarını uygulamak için BPSO, BBA, BPSOGSA, BGWO, BDA algoritmaları kullanılmıştır. Test sonuçlarının eşit şartlarda olması için iterasyon sayısı, parçacık sayısı,değişken boyutu ve kullanılan fonksiyonun türü aynı olması gerekir. Bu nedenle;

- Parçacık sayısı:50,
- İterasyon sayısı:1000,
- Değişken boyutu:100,

- Kullanılan fonksiyon türü sayısı:13 olarak belirlenmiştir.

Elde edilen test sonuçlarına göre tek modlu kısıtsız kıyaslama fonksiyonlarına ait sonuçlar Tablo 3, çok modlu kısıtsız kıyaslama fonksiyonlarına ait sonuçlar Tablo 4'de verilmiştir.

Tablo 3. Tek modlu kısıtsız kıyaslama fonksiyonlarına ait sonuçlar

Fonksiyon	İstatistik	BPSO	BBA	BPSOGSA	BGWO	BDA
F1	En iyi	18.00000	**0**	1.000000	32.000000	2.000000
	En kötü	43.00000	**39.000000**	41.000000	39.000000	36.00000
	Ortalama	18.12600	**0.289000**	7.687000	32.056000	13.53600
	Standart Sapma	1.426239	**1.849026**	11.253251	0.552421	9.791659
F2	En iyi	16.00000	**0**	1.000000	33.000000	3.000000
	En kötü	33.00000	**41.00000**	39.000000	37.000000	44.00000
	Ortalama	16.09900	**0.223000**	7.920000	33.035000	13.99300
	Standart Sapma	1.061285	**1.936239**	11.771753	0.312846	9.665214
F3	En iyi	10454.00	**0**	3.000000	42644.00	95.00000
	En kötü	59046.00	**50454.00**	45281.000	55664.00	45145.00
	Ortalama	10604.66	**83.74600**	5040.7520	42725.48	7938.714
	Standart Sapma	2109.9250	**1691.695**	9665.2969	949.8650	8703.521
F4	En iyi	1	1	1	1	1
	En kötü	1	1	1	1	1
	Ortalama	1	1	1	1	1
	Standart Sapma	0	0	0	0	0
F5	En iyi	2238.000	302.000000	1244.0000	**0**	709.000
	En kötü	3650.000	3748.00000	4147.0000	**4161.00000**	4151.00
	Ortalama	2244.306	405.058000	1774.9190	**15.677000**	1881.11
	Standart Sapma	76.86051	273.233173	875.84557	**203.049284**	935.763
F6	En iyi	57.000000	**25.000000**	27.000000	85.000000	27.000000
	En kötü	107.000000	**99.000000**	105.000000	101.000000	101.00000
	Ortalama	57.294000	**25.424000**	39.980000	85.300000	51.940000
	Standart Sapma	3.059236	**3.359689**	22.231280	2.044059	21.774263
F7	En iyi	592.000052	**0.001582**	17.000005	1577.00000	78.000002
	En kötü	1836.07257	**1820.43668**	2009.130262	2047.90624	1786.0955
	Ortalama	596.377988	**10.496406**	331.665571	1580.08861	611.65183
	Standart Sapma	58.766149	**78.522309**	541.723280	35.082173	469.17476

Tablo 3 incelendiğinde F4 ve F5 hariç tüm fonksiyonlar için BBA'nın en iyi sonuca ulaştığı görülmektedir. BBA'nın F1,F2,F3 ve F7 için optimum değeri elde ettiği, F4 için tüm algoritmaların aynı sonucu verdiği, F5'te ise BGWO'nun optimuma ulaşarak en iyi sonucu verdiği görülmektedir. F4 için tüm ikili optimizasyon algoritmaları aynı sonucu vermiştir.

Tablo 4.Çok modlu kalite fonksiyonlarına ait sonuçlar

Fonksiyon	İstatistik	BPSO	BBA	BPSOGSA	BGWO	BDA
F8	En iyi	-68.159150	**-84.147098**	**-84.147098**	**-84.147098**	-83.30563
	En kötü	-48.805317	-50.488259	-49.646788	-50.488259	-51.32973
	Ortalama	-68.071637	-83.504215	-78.217252	-83.985536	-74.79246
	Standart Sapma	0.958864	1.815894	9.925932	1.501577	6.834448
F9	En iyi	15.000000	**0**	1.000000	36.000000	1.000000
	En kötü	40.000000	**43.000000**	41.000000	41.000000	41.000000
	Ortalama	15.106000	**0.525000**	7.949000	36.040000	11.655000
	Standart Sapma	1.224853	**2.104710**	11.859133	0.422586	9.630718
F10	En iyi	1.711187	**0**	**0**	2.107583	0.557760
	En kötü	2.458254	**2.376360**	**2.170738**	2.319824	2.348296
	Ortalama	1.715371	**0.076740**	**0.536441**	2.110000	1.246636
	Standart Sapma	0.043084	**0.233549**	**0.850032**	0.020681	0.561815
F11	En iyi	0.236655	**0**	0.023147	0.459676	0.033550
	En kötü	0.466461	**0.487576**	0.522854	0.518773	0.541036
	Ortalama	0.237619	**0.004144**	0.098055	0.459908	0.161211
	Standart Sapma	0.012234	**0.023773**	0.129901	0.003005	0.110929
F12	En iyi	2.197151	**1.325359**	**1.325359**	2.960951	1.370520
	En kötü	3.483241	**3.498949**	**3.573562**	3.184790	3.483241
	Ortalama	2.201781	**1.337168**	**1.693344**	2.963178	1.884700
	Standart Sapma	0.062327	**0.092958**	**0.633338**	0.020818	0.469496
F13	En iyi	2.100000	0.100000	**0**	**0**	0.300000
	En kötü	3.900000	4.000000	**4.000000**	**3.800000**	3.700000
	Ortalama	2.110100	0.156400	**0.710800**	**0.010000**	1.432800
	Standart Sapma	0.103724	0.240285	**1.209391**	**0.153374**	0.984161

Tablo 4'de görüldüğü gibi BBA'nın F9,F10 ve F11'de, BPSOGSA'nın F10 ve F13'de BGWO'nun F13'de optimum sonuca ulaşmıştır. F7'de BBA, BPSOGSA ve BGWO'nun, F12'de BBA ve BPSOGSA'nın optimuma en yakın sonuca ulaşmıştır. Tek modlu kısıtsız kıyaslama fonksiyonlarında olduğu gibi Tablo 4'de de BBA en iyi performans göstermiştir.

6. Bulgular Ve Tartışma

Yapılan çalışmada ikili optimizasyon kullanılarak sunulan hibrid algoritmaların başarımını değerlendirmek için 13 adet kalite test fonksiyonu kullanılarak sonuçlar analiz edilmiştir. Analiz sonucunda elde edilen bulgulara göre 13 adet kalite test fonksiyonundan 8'inde Tablo 5'de belirtildiği gibi optimum sonuca ulaşılmıştır. Bunlardan BBA, 6 tanesinde optimum sonuca ulaşırken, BPSOGSA 2 tane, BGWO ise 2 tanesinde optimum sonuca ulaşmıştır.

Tablo 5. Kalite test fonksiyonları sonucu optimuma ulaşan algoritmalar

Optimuma Ulaşılan Fonksiyon	Optimuma Ulaşan Algoritmalar
F_1	BBA
F_2	BBA
F_3	BBA
F_5	BGWO
F_9	BBA
F_{10}	BBA,BPSOGSA
F_{11}	BBA
F_{13}	BPSOGSA,BGWO

Genel anlamda optimuma yakınlığı göz önüne alındığında BBA, 13 fonksiyondan 10'unde en iyi sonucu elde etmiştir. BPSOGSA 4 adet, BGWO 2 adet fonksiyonda optimuma en yakın sonuca ulaşmıştır.

Tablo Hata! Belgede belirtilen stilde metne rastlanmadı..Kalite test fonksiyonları sonucu optimuma veya en yakın sonuca ulaşan algoritmalar

Optimuma Ulaşılan Fonksiyon	Optimuma Ulaşan Algoritmalar
F_1	BBA
F_2	BBA
F_3	BBA
F_5	BGWO
F_6	BBA
F_7	BBA
F_8	BBA, BPSOGSA, BGWO
F_9	BBA
F_{10}	BBA,BPSOGSA
F_{11}	BBA
F_{12}	BBA,BPSOGSA
F_{13}	BPSOGSA,BGWO

Analiz sonucuna göre BPSOGSA, hibrid edilmesinde kullanılan BPSO ve BGSA'ya göre daha iyi sonuç elde etmiştir. Tüm test sonuçlarında en iyi sonucu BBA elde etmiştir. BGWO'nun en düşük standart sapmayla

kendi en iyi sonucuna en hızlı ulaşan algoritma olduğu görülmüştür. BPSO, BGSA ve BDA diğer algoritmalara göre iyi sonuç vermediği gözlenmiştir.

6. Sonuçlar

Gerçek hayat problemlerini çözmek için evet/hayır, seçme/seçmeme ya da 0-1 gibi net sonuçlar verdiği için ikili optimizasyon algoritmaları kullanılmaktadır. Literatürde ikili optimizasyon kullanılarak bir çok algoritma geliştirilmiştir. Bu çalışmada ikili optimizasyon ile ilgili literatür taraması yapılarak, bu optimizasyon için kullanılan modifikasyon teknikleri ve çözüm için kullanılan yöntemler hakkında bilgi verilmiştir. İkili optimizasyon ile hibrid edilmiş çalışmalardan BPSO, BPSOGSA, BGWO, BBA ve BDA'nın çalışma şekli ve kullanılan formüller ile ilgili bilgilendirme yapılarak bu alanlarda ne tür çalışmalar yapıldığı hakkında inceleme yapılmıştır.

Yapılan literatür taramasında optimizasyon algoritmalarının performans kıyaslamasını gerçekleştirmek için kullanılan kısıtsız kıyaslama fonksiyonları tanıtılmıştır. Bu amaçla 7 adet tek modlu ve 6 adet çok modlu olmak üzere 13 adet kısıtsız kıyaslama fonksiyonu kullanılmıştır. Test sonuçlarının eşit şartlarda olması için iterasyon sayısı, parçacık sayısı,değişken boyutu ve kullanılan fonksiyonun türü eşit olarak uygulandı.

Çalışmadan elde edilen sonuca göre kullanılan optimizasyonların tek ve çok modlu fonksiyonlarda kıyaslama için etkili sonuç verdiği ancak sabit boyutlu çok modlu fonksiyonlarda etkili sonuç vermediği gözlenmiştir. Bu testler sonucunda BBA'nın mevcut yöntemler içerisinde 13 adet kısıtsız kıyaslama fonksiyonundan 10'unde optimuma en yakın sonucu vermede daha başarılı olduğu görülmüştür. Ayrıca BPSOGSA da 4 adet kısıtsız kıyaslama fonksiyonunda optimuma en yakın sonucu vererek BBA'dan sonra en etkin algoritma olmuştur. BGWO, en düşük standart sapma ile kendi en iyi sonucuna en hızlı şekilde ulaşmıştır.

Gelecek çalışmada güncel optimizasyon algoritmalarından olan Altın Sinüs Algoritması (Gold-SA) için hibrid ikili optimizasyon algoritması gerçekleştirilerek mevcut ikili optimizasyon algoritmaları ile performans kıyaslaması yapılacaktır.

Kaynaklar

[1] Özsağlam Y, Çunkaş M. Optimizasyon problemlerinin çözümü için parçaçık sürü optimizasyonu algoritması. Politeknik Dergisi 2008; 11(4): 299-305.

[2] Doğan C. Balina Optimizasyon Algoritması Ve Gri Kurt Optimizasyonu Algoritmaları Kullanılarak Yeni Hibrit Optimizasyon Algoritmalarının Geliştirilmesi. Yüksek Lisans Tezi, Erciyes Üniversitesi, Kayseri, 2019.

[3] Aytimur A. İkili Optimizasyon Yöntemlerinin Araştırılması ve İkili Test Problemleri Üzerinde Başarımının Değerlendirilmesi. Yüksek Lisans Tezi, Erciyes Üniversitesi, Kayseri, 2019.

[4] Mirjalili S, Wang G, Coelho L. Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput & Applic 2014; 25: 1423-1435.

[5] Reeves CR. Genetic Algoritms. School of Mathematical and Information Sciences 2010; pp. 55-82.

[6] Yang XS. Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley & Sons. New Jersey, USA: Wiley, 2010.

[7] Korkmaz S. İkili optimizasyon problemlerinin çözümü için yapay alg algoritması tabanlı yeni yaklaşımlar. Doktora Tezi, Konya Teknik Üniversitesi, Konya, 2019.

[8] Mirjalili S, Lewis A. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm and Evolutionary Computation 2013; 9: 1-14.

[9] Pampara G, Engelbrecht AP, Franken N. Binary Differential Evolution. 2006 IEEE International Conference on Evolutionary Computation; 16-21 Temmuz 2006, Canada. 1873-1879.

[10] Nezamabadi-pour H. A Quantum-Inspired Gravitational Search Algorithm for Binary Encoded Optimization Problems. Engineering Applications of Artificial Intelligence 2015; 40: 62-75.

[11] Banitalebi A, İsmail M, Abdulaziz Z. A Self-Adaptive Binary Differential Evolution Algorithm For Large Scale Binary Optimization Problems. Information Sciences 2016; 367: 487-511.

[12] Ozturk C, Hancer E, Karaboga D. Dynamic clustering with improved binary artificial bee colony algorithm. Applied Soft Computing 2015; 28: 69-80.

[13] Kıran MS, Gündüz M. XOR-based artificial bee colony algorithm for binary optimization. Turkish Journal Of Electrical Engineering & Computer Sciences 2013; 21: 2307–2328.

[14] Kashan MH, Nahavandi N, Kashan AH. DisABC: A new artificial bee colony algorithm for binary optimization. Applied Soft Computing 2012; 12: 342-352.

[15] Too J, Abdullah A, Saad NM. A New Co-Evolution Binary Particle Swarm Optimization with Multiple InertiaWeight Strategy For Feature Selection. Informatics 2019; 6(2): 2-14.

[16] Mirjalili S. The Ant Lion Optimizer. Advances in Engineering Software 2015; 83: 80-98.

[17] Emary E, Zawbaa HM, Hassanien AE. Binary grey wolf optimization approaches for feature selection. Neurocomputing 2016; 172: 371–381.
[18] Too J, Abdullah AR, Saad NM, Ali NM, Tee W. A New Competitive Binary Grey Wolf Optimizer to Solve the Feature Selection Problem in EMG Signals Classification. Computers 2018; 7, 58: 2-18.
[19] Mirjalili S, Mirjalili SM, Yang X. Binary bat algorithm. Neural Comput & Applic 2013; 25: 663-681.
[20] Mafarja M, Aljarah İ, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S. Binary Dragonßy Optimization for Feature Selection using Time-Varying Transfer functions. Knowledge-Based Systems 2018; 161: 185-204.