NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., Erken Görünüm / InPress
Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences
Araştırma makalesi / Research article
www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Meta sezgisel arama algoritmalarının çok seviyeli görüntü eşikleme performanslarının karşılaştırılması

Comparison of multi-level image thresholding performances of meta-heuristic search algorithms

Asuman Günay Yılmaz[1,*], Samoua Alsamoua[2]

[1] *Karadeniz Teknik Üniversitesi, Mühendislik Fakültesi, Yapay Zeka ve Veri Mühendisliği Bölümü, Trabzon, Türkiye*
[2] *Karadeniz Teknik Üniversitesi, Of Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, Trabzon, Türkiye*

Öz

Görüntü segmentasyonu alanında sıklıkla kullanılan eşiklemeye dayalı segmentasyon yöntemleri, az sayıdaki eşik değerinin belirlenmesinde başarılıdır. Fakat eşik seviye sayısı arttıkça hesaplama karmaşıklığında üstel artış meydana gelmektedir. Bu nedenle problemin çözümünde meta sezgisel arama algoritmaları tercih edilmektedir. Bu çalışmada görüntü segmentasyonu literatüründe en çok kullanılan 11 meta sezgisel algoritmanın (yapay arı kolonisi, guguk kuşu arama, diferansiyel evrim, gri kurt optimizasyonu, harris şahini optimizasyonu, güve-alev optimizasyonu, parçacık sürüsü optimizasyonu, sinüs kosinüs algoritması, simbiyotik organizmalar arama, salp sürü algoritması, balina optimizasyon algoritması) performansı kıyaslanmıştır. m=2,3,4,5,6,7,8 seviyeli segmentasyon ile algoritmaların düşük ve orta seviyelerdeki başarımı değerlendirilmiştir. 100 görüntü üzerinde yapılan deney sonuçları Friedman istatistiksel analiz yöntemi ile analiz edilmiştir. Elde edilen sonuçlar en başarılı ilk iki algoritmasın güve-alev optimizasyonu ve parçacık sürüsü optimizasyonu olduğunu göstermiştir. Geniş bir veri seti üzerinde çeşitli seviyelerde eşikleme ve farklı sonlandırma kriterleri ile yapılan bu çalışmanın sonuçları, görüntü segmentasyonu alanında çalışan araştırmacılara algoritma seçimi konusunda önemli bilgiler sağlamaktadır.

Anahtar kelimeler: Çok seviyeli eşikleme, Görüntü segmentasyonu, Meta sezgisel optimizasyon

Abstract

Thresholding based segmentation methods are successful in determining small number of threshold values. As the computational complexity is high for multilevel thresholding, meta-heuristic search algorithms are used. In this study, the performance of 11 meta-heuristic search algorithms (artificial bee colony, cuckoo search, differential evolution, gray wolf optimization, harris hawk optimization, moth-flame optimization, particle swarm optimization, sine-cosine algorithm, symbiotic organisms search, salp swarm algorithm, whale optimization algorithm) were compared for image segmentation problem. The low and medium level (m=2,3,4,5,6,7,8) segmentation performances of algorithms were evaluated. The experimental results on 100 images were analyzed with Friedman statistical analysis method. The obtained results showed that the first two most successful algorithms were moth-flame optimization and particle swarm optimization. The results of this study, conducted on a large dataset with various levels of thresholding and different termination criteria, provide important information on algorithm selection for researchers working in the field of image segmentation.

Keywords: Multi level thresholding, Image segmentation, Meta heuristic optimization

1 Giriş

Görüntü segmentasyonu, bir görüntünün şekil, renk, parlaklık ve benzeri ortak özelliklere sahip çok sayıdaki homojen parçalara bölünmesi işlemidir. Görüntünün karmaşıklığını azaltan ve böylece görüntünün analizini kolaylaştıran segmentasyon işlemi, görüntü tabanlı uygulamalara çok önemli bilgiler sağlar. Nesne sınıflandırma, tıbbi görüntüleme, uzaktan hedef algılama, biyometrik tanıma ve benzeri bilgisayarla görme uygulamalarında sıklıkla kullanılan görüntü segmentasyonu, oldukça aktif bir araştırma alanıdır. Görüntü işleme literatüründe özellikle son birkaç on yılda çok sayıda algoritma önerilmiştir. Bu algoritmalar, histogram tabanlı, kümeleme tabanlı, kenar tabanlı ve bölge büyütme tabanlı olarak sınıflandırılabilir. Segmentasyon algoritmasının ve seviye sayısının seçimi, üzerinde çalışılan problemin karakteristiğine ve kullanılan görüntülerin yapısına bağlıdır.

Segmentasyonu algoritmaları içerisinde histogram tabanlı yöntem, basitliği ve hesaplama etkinliği nedeniyle yaygın kullanıma sahiptir. Yöntemde görüntü histogramından uygun eşik değerlerini belirlenerek segmentasyon işlemi gerçekleştirilir. Bu algoritma ile görüntüyü eşiklemenin iki yolu, iki seviyeli eşikleme ve çok seviyeli eşiklemedir. İki seviyeli eşikleme, görüntünün histogramından tek bir eşik değerinin belirlenmesi ve bu eşik değerinden küçük renk seviyelerinin siyah, büyüklerin beyaz

* Sorumlu yazar / Corresponding author, e-posta / e-mail: gunaya@ktu.edu.tr (A. Günay Yılmaz)
Geliş / Recieved: 07.08.2024 Kabul / Accepted: 07.11.2024 Yayımlanma / Published: xx.xx.20xx
doi: 10.28948/ngumuh.1529614

olarak sınıflandırılması ile ikili görüntü üretilmesi işlemidir. Çok seviyeli eşiklemede ise görüntü histogramından m ≥ 2 adet eşik değeri belirlenerek görüntü m+1 bölgeye ayrılır.

Görüntü histogramından optimum eşik değerinin belirlenmesinde çeşitli yaklaşımlar önerilmiştir. Bunlardan en çok kullanılanlar Kapur entropisi [1] ve Otsu kriteridir [2]. Kapur entropisi, gri seviyelerin olasılık dağılımını kullanarak segmente sınıfların histogram entropilerini maksimize eden eşik değerlerini belirler. Otsu kriteri ise sınıflar arasındaki varyansı maksimize eden eşik değerlerini seçer. Sezgin ve Sankur eşik değeri belirleme yöntemlerini kıyaslamış ve Kapur entropisinin diğer fonksiyonlardan daha iyi performans gösterdiğini göstermiştir [3]. Gri seviye görüntü segmentasyonunda tek renk kanalı için eşikleme fonksiyonları kullanılarak eşik değerleri belirlenmektedir. Renkli görüntülerin segmentasyonunda ise eşik değerlerinin belirlenmesi işleminin tüm renk kanalları için (kırmızı, yeşil, mavi) ayrı ayrı yapılması gerekmektedir.

İki seviyeli segmentasyon probleminin çözümünde başarılı ve etkin bir şekilde kullanılan eşikleme yaklaşımları, çok seviyeli segmentasyon problemi söz konusu olduğunda hesaplama karmaşıklığında ve maliyetinde üstel artışa neden olmaktadır [4]. Bu nedenle araştırmacılar son on yıldır hesaplama karmaşıklığını azaltmak ve segmentasyon başarımını artırmak amacıyla çeşitli meta-sezgisel algoritmaların (MSA) kullanımını önermiştir. MSA'lar, bir çözüm uzayında etkin bir arama yapabilmek ve lokal optimum noktasından kurtulabilme yeteneğine sahip bir süreç oluşturmak için yerel iyileştirme prosedürleri ve üst düzey stratejiler arasında bir etkileşim organize eden optimizasyon teknikleridir. Çok sayıda doğadan ilham alan MSA, çok seviyeli görüntü eşikleme probleminin çözümüne uygulanmıştır. Tüm problemleri başarılı şekilde çözebilecek bir algoritma bulunmadığı/bulunamayacağı için, bu alanda halen çok sayıda çalışma yapılmakta, çeşitli algoritmalar geliştirilmektedir.

Literatürdeki meta-sezgisel optimizasyona dayalı çok seviyeli görüntü eşikleme çalışmalarını iki gruba ayırmak mümkündür. Birinci gruptaki çalışmalar mevcut bir MSA'nın problemin çözümüne uygulanması ve genelde yazarlar tarafından seçilen rakip algoritmalarla kıyaslamasını içermektedir [5-15]. İkinci gruptaki çalışmalarda ise mevcut bir MSA'nın modifiye edilmesi ile yeni bir algoritma geliştirilmesi ve böylece görüntü segmentasyonu performansının artırılması hedeflenmiştir [16-24]. Fakat her bir algoritmanın kendine özgü kısıtları vardır. Bu nedenle araştırmacılar, algoritmadaki arama süreci stratejisini değiştirerek yöntemi modifiye etme ya da algoritmaları birlikte kullanarak hibrid yöntemler geliştirme yoluyla performansı artırmaya ve karmaşıklığı azaltmaya çalışmaktadır.

İlk olarak Yin [5] minimum çapraz entropiye dayalı parçacık sürüsü optimizasyonu (PSO) algoritmasını, çok seviyeli eşikleme ile segmentasyon probleminin çözümünde kullanmıştır. Dört gri seviyeli görüntü üzerinde m=2, 3, 4 eşik seviyesi için yapılan deneylerde PSO için iterasyon sayısı 200 olarak uygulanmıştır. Horng [6] Kapur entropisine dayalı bal arısı çiftleşme optimizasyonu (HBMO) algoritmasını çok seviyeli görüntü eşikleme problemi için önermiştir. 5 gri seviye görüntü için m=2, 3, 4, 5 seviyeli eşik değerlerinin belirlenmesinde iterasyon sayısı 200 seçilmiş, fakat elde edilen sonuçlardan 100 iterasyonun yeterli olduğu görülmüştür. Horng ve Liou [7] bir sonraki çalışmada eşik değerlerini seçmek için minimum çapraz entropiye dayalı ateşböceği algoritmasını (FA) önermiştir. Akay [8] PSO ve yapay arı kolonisi (ABC) yöntemlerinde amaç fonksiyonu olarak Otsu kriterini ve Kapur entropisini kullanmanın, çok seviyeli görüntü eşikleme problemi çözüm performansına etkisini araştırmıştır. Deneyler 12 gri seviyeli doğal görüntü üzerinde m=2, 3, 4, 5 seviyeli eşikleme için 100 iterasyon kullanılarak gerçekleştirilmiştir. Bahandari vd. [9] gri seviyeli uydu görüntülerinin çok seviyeli eşikleme ile segmentasyonu probleminde guguk kuşu arama (CS) ve rüzgâr güdümlü optimizasyon (WDO) yöntemlerinin kullanımını önermiştir. Çalışmalarında 12 gri uydu görüntüsü m=2, 3, 4, 5 seviyeli eşikleme ile segmente edilmiştir. Amaç fonksiyonu olarak Kapur entropisi kullanılmıştır. Pare vd. [10] optimizasyon işleminin süresini kısaltmayı hedefledikleri çalışmalarında, optimum eşik seviyelerinin etkili ve doğru üretilmesi için, çok seviyeli minimum çapraz entropi fonksiyonunun CS algoritması ile minimize edilmesini önermiştir. Çalışmada önerilen yöntemle 3 renkli doğa ve 3 renkli uydu görüntüsü m=2, 5, 8, 16 seviyeli eşiklenerek segmente edilmiş, elde edilen sonuçlar ABC, bakteriyel yiyecek arama algoritması (BFA), WDO ve diferansiyel evrim (DE) algoritmaları ile kıyaslanmıştır. Khairuzzaman ve Chaudhury [11] çok seviyeli eşiklemeye dayalı segmentasyon probleminin çözümünde gri kurt optimizasyon (GWO) yöntemini kullanmıştır. Kapur entropisi, sınıflar arası varyans eşikleme fonksiyonlarının kullanıldığı çalışmada m=2, 3, 4, 5 seviyeli eşikleme yapılmıştır. Yöntemin başarımı 8 gri görüntü üzerinde, PSO'nun geliştirilmiş versiyonu ve BFO yöntemleriyle kıyaslanmıştır. El Aziz vd. [12] optimal eşik değerlerinin belirlenmesinde balina optimizasyon algoritması (WOA) ve güve-alev optimizasyonu (MFO) algoritmalarını kullanmıştır. Algoritmalar sınıflar arası varyans fonksiyonu kullanılarak, 100 iterasyonda m=2, 3, 4, 5 seviyeli eşiklemede test edilmiştir. 8 gri görüntü üzerinde yapılan segmentasyon sonuçları sinüs kosinüs algoritması (SCA), harmoni arama algoritması (HSO), sosyal örümcek optimizasyonu (SSO), algoritmalarının sonuçlarıyla karşılaştırılmıştır. Küçükuğurlu ve Gedikli [13], simbiyotik organizmalar arama (SOS) algoritmasını görüntü eşikleme problemine uygulamıştır. Sınıflar arası varyans ve Kapur entropisi fonksiyonlarını kullanan yöntem PSO, FA, ABC, genetik algoritma (GA), GWO algoritmaları ile karşılaştırılmıştır. Deneylerde 8 gri görüntü m=2, 3, 4, 5 seviyeli eşiklenerek sonuçlar değerlendirilmiştir. Sathya vd. [14] döviz piyasası algoritması (EMA) yöntemini görüntü segmentasyonu probleminin çözümüne uygulamıştır. Segmentasyon işlem hızını artırmayı amaçlayan çalışmada, sınıflar arası varyans, Kapur entropisi ve minimum çapraz entropi amaç fonksiyonları kullanılmış, m= 4, 5, 6,7 seviyeli eşikleme geçekleştirilmiştir. Deneyler 5 renkli standart test görüntüsü üzerinde gerçekleştirilmiş ve sonuçlar, krill sürü algoritması (KHA), öğretme-öğrenme tabanlı optimizasyon (TLBO) ve CS algoritmaları ile kıyaslanmıştır. Tüm

algoritmalar için iterasyon sayısı 100 seçilmiştir. Houssein vd. [15], eşikleme probleminin çözümünde kara dul optimizasyonu (BWO) algoritmasını kullanmıştır. Çalışmada sınıflar arası varyans ve Kapur entropisi eşikleme fonksiyonları ile 10 gri seviye görüntü üzerinde *m*=2, 3, 4 ,5 seviyeli eşikleme yapılmıştır. İterasyon sayısı 350 olarak seçilmiş, yöntem GWO, MFO, WOA, SCA, SSO algoritmaları ile kıyaslanmıştır.

Bahandari vd. [16] uydu görüntülerinin daha hızlı ve verimli segmentasyonu için değiştirilmiş ABC algoritmasını önermiştir Yöntemde *m*=2, 3, 4, 5 seviyeli eşikleme için sınıflar arası varyans, Kapur entropisi ve Tsalli entropisi fonksiyonları kullanılmıştır. Yöntem 5 gri uydu görüntüsü üzerinde test edilmiş ve sonuçlar ABC, PSO ve GA algoritmaları ile karşılaştırılmıştır. He ve Huang [17] renkli görüntülerin çok seviyeli eşikleme ile segmentasyonu probleminin daha etkin çözümü için, standart FA yönteminin performansını artırarak değiştirilmiş FA yöntemini önermiştir. Yöntem, Kapur entropisi, sınıflar arası varyans, Tsalli entropisi fonksiyonları ile 10 renkli görüntünün *m*=4, 5, 6, 7 seviyeli eşiklenmesinde kullanılmıştır. Anihta vd. [18] eşiklemeye dayalı görüntü segmentasyonu probleminin çözümü için değiştirilmiş WOA algoritmasını önermiştir. Algoritma 8 renkli görüntü ile 2 uygu görüntüsünün üzerinde *m*= 2, 3, 4, 5 seviyeli eşikleme için test edilmiştir. İterasyon sayısı 150 olarak belirlenmiş ve önerilen yöntemin sonuçları GA, PSO, ABC ve CS algoritmalarıyla kıyaslanmıştır. Rahkar vd. [19] FA ve PSO algoritmalarını kullanarak hibrid bir segmentasyon yöntemi önermiştir. Sınıflar arası varyans ve Kapur entropisi eşikleme fonksiyonları ile 6 gri seviyeli görüntü üzerinde *m*=2, 3, 4, 5 seviyeli segmentasyon gerçekleştirilmiştir. Yöntemin sonuçları PSO, GA, FA ile kıyaslanmıştır. Çalışmada tüm algoritmalar için iterasyon sayısı 200 olarak kullanılmıştır. Abd Elaziz vd. [20], görüntü segmentasyonu problemi için kuantum deniz yırtıcıları algoritmasını geliştirmiştir. Deneyler bulanık entropi eşikleme fonksiyonu kullanılarak 10 gri seviye görüntü üzerinde *m*=6, 8, 15, 17, 19, 25 seviyeli eşikleme için gerçekleştirilmiş ve iterasyon sayısı 200 olarak kullanılmıştır. Rakip algoritmalar ise deniz yırtıcıları algoritması (MPA), WOA, SCA, MFO, GWO, çekirge optimizasyon algoritması (GOA), salp sürü algoritması (SSA) olarak seçilmiştir. Qiao vd. [21] aritmetik optimizasyon algoritmasını (AOA) harris şahini optimizasyonu (HHO) ile kullanan bir segmentasyon yöntemi önermiştir. Yöntemde ilk 50 iterasyon AOA, sonraki 50 iterasyon ise HHO en iyi çözümü aramaktadır. Deneyler 8 görüntü üzerinde *m*=2, 3, 4, 5, 15, 16, 32 seviyeli eşikleme için gerçekleştirilmiştir. Rakip algoritmalar AOA, HHO, WOA, Çekirge optimizasyon algoritması (GOA) olarak seçilmiştir. Gharehchopogh ve İbrikçi [22] görüntü segmentasyonu için, geliştirilmiş bir Afrika akbabaları optimizasyon algoritması (AVOA) önermiştir. Üç farklı eşikleme fonksiyonu için deneyler yapılmış 8 uzay görüntüsü *m*=5, 7, 9, 11 seviyeli segmente edilmiştir. Abualigah vd. [23] ise segmentasyon probleminin çözümü için güçlendirilmiş Aquila aritmetik optimizasyonu yöntemini geliştirmiştir. Covid-19'a yakalanmış 4, sağlıklı 4 X-ray görüntüsü, 4 renkli ve 4 gri seviye görüntü üzerinde *m*=3, 4, 5, 6 seviye eşikleme yapılmış, önerilen yöntem SSA, WOA, PSO, AO (Aquila Optimizasyonu) ile kıyaslanmıştır. Wang vd. [24] eşikleme problemi için birleştirilmiş mutasyon ve benzerliğin kaldırılmasına dayalı WOA algoritmasını geliştirmiştir. 10 gri görüntü için *m*=2, 4, 6, 8 seviyeli segmentasyon yapılmış, iterasyon sayısı 300 kullanılmıştır.

Literatürde her yıl onlarca MSA tanıtılmaktadır. Buna bağlı olarak pek çok MSA ve değiştirilmiş versiyonları çoklu eşikleme ile segmentasyon probleminin çözümüne uygulanmaktadır. Meta-sezgisel yöntemler doğaları gereği kullandıkları parametre sayısı, iterasyon sayısı ve iterasyon işlem adımlarında farklılıklar göstermektedir. Bir algoritma bir iterasyon adımında bir komşuluk hesaplarken, diğer bir algoritma birden fazla komşu çözüm hesaplayabilir. Bu durumda algoritma amaç fonksiyonunu bir iterasyonda birden fazla çağırmaktadır. Bu ise adaletli bir kıyaslama yapmaya engel olmaktadır. Çok seviyeli segmentasyon konusunda yapılan çalışmalarda genellikle sonlandırma kriteri olarak iterasyon sayısı kullanılmıştır ve bu iterasyon sayıları farklılıklar göstermektedir. Oysaki sonlandırma kriterinin optimizasyon sürecinin sonuçları üzerinde önemli bir etkisi vardır. Bu parametrenin değeri çözümlerin kalitesini ve en iyi çözümü bulan algoritmaları değiştirebilmektedir. Algoritmaların adil olarak kıyaslanabilmesi için bu çalışmada sonlandırma kriteri olarak maksimum amaç fonksiyonu değerlendirme sayısı, *maxFEs*=500**d* (*d*: eşik seviye sayısı) ve *maxFEs*=1000**d* kullanılmıştır.

Eşiklemeye dayalı görüntü segmentasyon problemleri için çok farklı fonksiyonlar geliştirilmiş olsa da literatürde genel olarak Kapur entropisi [1] ve Otsu kriteri [2] en çok kullanılan yöntemlerdir. Kapur entropisinin genel kullanımı ve başarımı nedeniyle çalışmada eşikleme fonksiyonu olarak Kapur entropisi tercih edilmiştir.

Segmentasyon probleminin çözümünde önerilen MSA'lar sınırlı sayıda (2-8 arası) rakip algoritma ile kıyaslanmıştır. Her çalışma farklı rakiplerle kıyaslama yaparak kendi yönteminin daha iyi olduğunu belirtmektedir. Fakat genel olarak bu algoritmaların neden seçildiği ile ilgili bir bilgiye rastlanılmamaktadır. Bu çalışmada görüntü segmentasyonu literatüründeki çalışmalarda modifiye edilme ya da rakip olma açısından en çok kullanılan algoritmaların başarımları kıyaslanmıştır. Literatürde her yıl çok sayıda MSA önerilmektedir. Bu algoritmaların artık geçerliliği ve başarımları da sorgulanmaya başlanmıştır. Bu nedenle çalışmada segmentasyon literatüründe en çok kullanılmış, geçerliliği kabul edilmiş, pek çok farklı problemin çözümünde de başarılı olmuş 11 MSA'nın görüntü segmentasyondaki başarımları değerlendirilmiştir.

Çalışmalarda genellikle az sayıdaki görüntüler üzerinden sonuçlar rapor edilmektedir. Bu ise büyük veri kümeleri söz konusu olduğunda algoritmaların gerçek başarımları ile ilgili bilgi sağlamamaktadır. Çalışmada Berkeley segmentasyon veri setinin test kümesinde bulunan 100 görüntü kullanılarak deneyler gerçekleştirilmiştir. Böylece algoritmaların gerçek segmentasyon performanslarının değerlendirilmesi sağlanmıştır. Bu açıdan bu çalışma yapılacak görüntü

segmentasyonu çalışmalarında algoritma seçimi konusunda araştırmacılara ışık tutacak niteliktedir.

Literatürdeki çalışmalardaki diğer bir problem ise görüntülerin segmentasyonunda kullanılan eşik seviye sayısıdır. Çalışmaların çoğunda düşük sayıda eşik seviyesi (2, 3, 4, 5) kullanılmıştır. Dolayısıyla algoritmanın orta ya da yüksek eşik seviye sayısı kullanıldığında nasıl bir performans göstereceği bilinmemektedir. Bu çalışmada belirlenen algoritmalarla 100 görüntünün *m*=2, 3, 4, 5, 6, 7, 8 seviyeli segmentasyonu gerçekleştirilmiştir.

Bu çalışmanın ana katkıları aşağıdaki gibidir.

-Çok seviyeli görüntü eşiklemeye dayalı görüntü segmentasyonu literatüründe en çok kullanılan ABC [25], CS [26], DE [27], GWO [28], HHO [29], MFO [30], PSO [31], SCA [32], SOS [33], SSA [34], WOA [35] algoritmalarının performansları karşılaştırılmıştır.

-Algoritmaların adil şekilde kıyaslanabilmesi için sonlandırma kriteri olarak maksimum amaç fonksiyonu değerlendirme sayısı, *maxFEs*=500**d* (*d*:eşik seviye sayısı) ve *maxFEs*=1000**d* kullanılmıştır.

-Algoritmaların performansı Berkeley segmentasyon veri setinin Test klasöründe bulunan 100 görüntü üzerinde değerlendirilmiştir.

-*m*=2, 3, 4, 5, 6, 7, 8 seviyeli segmentasyon ile algoritmaların düşük ve orta seviye eşiklerdeki başarımı değerlendirilmiştir.

-Friedman istatistiksel analiz yöntemi ile algoritmaların performansı değerlendirilmiştir.

Yazı şu şekilde organize edilmiştir. İkinci bölümde çok seviyeli eşikleme problemi, üçüncü bölümde MSA'ların genel yapısı ve deneylerde kullanılan algoritmalar açıklanmıştır. Kullanılan veri seti, deneysel çalışmalar ve bulgular dördüncü bölümde verilmiş, elde edilen sonuçlar beşinci bölümde tartışılmıştır.

2 Materyal ve metot

2.1 Çok seviyeli eşikleme problemi

Eşikleme gri ya da renkli görüntülerdeki piksellerin, görüntü segmentasyonu amacıyla, parlaklık değerlerine göre ayrı sınıflara sınıflandırılması işlemidir. Görüntüdeki sınıfların doğru seçilebilmesi için optimal eşik değerinin belirlenmesi gerekir. Bu eşik değeri kullanılarak görüntü Denklem (1) ile iki sınıfa ayrılabilir.

$$\begin{aligned} C_1 &\leftarrow p \quad \text{if} \quad 0 \le p < th \\ C_2 &\leftarrow p \quad \text{if} \quad th \le \mathrm{p} < L \end{aligned} \tag{1}$$

Denklemde L görüntüdeki gri seviye sayısını, p mxn boyutlu görüntüdeki herhangi bir pikseli, *th* belirlenen eşik değerini ifade etmektedir. Burada C_1 ve C_2 *th* eşik değerine göre ayrılan iki sınıfı temsil etmektedir. Bir eşik değerine göre yapılan ve o eşik değerinden küçük olan pikselleri C_1, eşik değerine eşit ve büyük olanları ise C_2 sınıfına atayan bu yöntem iki seviyeli eşikleme olarak adlandırılmaktadır. Bu yapı çok sayıda eşik değeri kullanılarak, çok sayıda sınıf içeren görüntü oluşturulmasını sağlayan çoklu eşiklemeye temel oluşturur (Denklem (2)).

$$\begin{aligned} C_1 &\leftarrow p \quad \text{if} \quad 0 \le p < th_1 \\ C_2 &\leftarrow p \quad \text{if} \quad th_1 \le p < th_2 \\ C_i &\leftarrow p \quad \text{if} \quad th_i \le p < th_{i+1} \\ C_n &\leftarrow p \quad \text{if} \quad th_n \le p < L \end{aligned} \tag{2}$$

Denklemde *th*'ler farklı eşik değerlerini ifade etmektedir. Buradaki amaç, iki ya da çok seviyeli eşikleme kullanarak görüntüde farklı karakteristiğe sahip bölgeleri en iyi şekilde sınıflandıracak eşik değerlerinin belirlenmesidir. Renkli görüntülerde bu eşik değerleri 3 kanal içinde ayrı ayrı işlem yapılarak belirlenir.

Görüntüdeki nesneler ve arka plan arasında iyi bir ayrım yapacak eşik değerlerinin belirlenmesinde çeşitli yöntemler önerilmiştir. Bunlardan en başarılı olan ve eşik değerlerinin belirlenmesinde sıklıkla parametrik olmayan amaç fonksiyonu olarak kullanılan Kapur entropisi aşağıda açıklanmaktadır

Kapur entropisi [1], segmente edilen sınıfların entropi değerlerine göre optimal eşik değerlerini belirleyen bir eşikleme yöntemidir. Kapur entropisi görüntü histogramındaki gri seviyelerin olasılık dağılımlarına bağlı olarak hesaplanmaktadır. $[th_1, th_2, \ldots . th_n]$'nin görüntüyü sınıflara ayırmak için kullanılacak eşik değerleri olduğu varsayılırsa, Kapur yöntemindeki amaç fonksiyonu Denklem (3)'teki gibidir.

$$\begin{aligned} H(th_1, th_2, \ldots . th_n) &= H_0 + H_1 + \cdots + H_n \\ H_0 &= -\sum_{j=0}^{th_1-1} \frac{p_j}{\omega_0} \ln \frac{p_j}{\omega_0}, \quad \omega_0 = \sum_{j=0}^{th_1-1} p_j \\ H_1 &= -\sum_{j=th_1}^{th_2-1} \frac{p_j}{\omega_1} \ln \frac{p_j}{\omega_1}, \quad \omega_1 = \sum_{j=th_1}^{th_2-1} p_j \\ H_n &= -\sum_{j=th_n}^{L-1} \frac{p_j}{\omega_n} \ln \frac{p_j}{\omega_n}, \quad \omega_n = \sum_{j=th_n}^{L-1} p_j \end{aligned} \tag{3}$$

Denklemlerde $H_1, H_2 \ldots H_n$ sınıf entropilerini, $\omega_1, \omega_2 \ldots \omega_n$, sınıf olasılıklarını, p_j j. renk değerinin olasılığını, L ise maksimum renk değerini ifade etmektedir. Amaç fonksiyonunun maksimize edilmesi ile optimal eşik değerleri elde edilmektedir. Problemin yukarıda verilen denklemlerle çözümünün işlem karmaşıklığı, eşik sayısının artışına bağlı olarak üstel olarak artmaktadır. Bu nedenle denklemde verilen amaç fonksiyonunu maksimize edecek eşik değerlerinin belirlenmesi *n* boyutlu bir optimizasyon problemi olarak ele alınabilir ve Denklem (4) ile ifade edilir.

$$t^* = \arg\max\left(\sum_{i=1}^{n} H_i\right) \tag{4}$$

2.2 Meta-sezgisel algoritmalar

Popülasyona dayalı arama yöntemleri olan MSA'lardaki arama süreci genel olarak iki ana adımdan oluşur. İlk adımda, benimsenen seçim yöntemine bağlı olarak popülasyondan çözüm adayları (arama uzayındaki referans konumları) seçilir. İkinci adımda seçilen referans konumları kullanılarak arama işlemi gerçekleştirilir. İkinci adım

MSA'ların arama operatörleri tarafından gerçekleştirilir. Arama operatörlerinin görevi, sömürü (exploitation) ve keşif (exploration) gereksinimlerini karşılamaktır. Referans konumlarına (seçim sürecindeki çözüm adayları) yakın bir aramaya komşuluk araması adı verilir. Amaç, seçilen çözüm adaylarının yakınındaki en iyi pozisyonu (çözüm) keşfetmektir. Çeşitlilik, arama sürecindeki iyileşmenin tamamen durduğu veya başarılı çözümlerin elde edilemediği durumlarda, popülasyonu yerel çözüm tuzaklarından kurtarma görevidir. Çeşitlilik görevini yerine getirmek için referans konumlarında etkili bir değişiklik sağlamak amacıyla mutasyon gibi bir işlem uygulanır. Çeşitlilik süreci, arama alanındaki referans konumlarından daha başarılı olan konumların keşfedilmesine olanak sağlamalıdır.

MSA'ların farklı arama özellikleri vardır. Bazı MSA'lar rakiplerine göre daha hızlı arama yapabilmekte ve yerel çözümleri kolaylıkla bulabilmektedir. Ancak çoğu MSA, birçok yerel çözüm tuzağı olan arama uzaylarında çeşitlilik sağlayamaz ve erken yakınsama sorunu yaşar. Karmaşık arama uzaylarında çeşitliliği başarıyla sağlayan ve sömürü-keşif dengesini koruyan algoritmalar, etkin bir şekilde küresel çözüme yakınsayabilir. MSA'lar optimizasyon sürecinde benzer adımlardan oluşan bir döngüyü takip ederler. Bu işlem adımları Algoritma 1'de verilmiştir.

Algoritma 1. MSA algoritmalarında arama sürecinin genel adımları [36]

1) Optimizasyon probleminin tanımlanması (amaç fonksiyonu, tasarım parametreleri vd.)
2) Algoritma parametrelerinin tanımlanması ve çözüm adayları topluluğunun oluşturulması
3) Aday çözümlerin uygunluk değerlerinin hesaplanması
4) Arama Süreci Yaşam Döngüsü
   a. Seçim süreci
   b. Komşuluk araması ve çeşitliliğin sağlanması
   c. Çözüm adayı topluluğunun güncellenmesi
5) Sonlandırma kriteri sağlandı mı?
   a. Hayır (Adım 4'e dön)
   b. Evet (Arama sürecini sonlandır ve en iyi çözüm adayını kaydet)

Bu çalışmada çok seviyeli görüntü eşiklemeye dayalı görüntü segmentasyonu literatüründe en çok kullanılan ABC [25], CS [26], DE [27], GWO [28], HHO [29], MFO [30], PSO [31], SCA [32], SOS [33], SSA [34], WOA [35] algoritmalarının performansları karşılaştırılmıştır. Bu algoritmalar sonraki alt bölümlerde açıklanmıştır.

2.2.1 *Yapay arı kolonisi (ABC) algoritması*

ABC algoritması arıların yiyecek arama davranışlarının modellenmesi ile geliştirilmiştir [25]. Yiyecek arama süreci, yiyecek kaynağının değeri, türü, yuvaya yakınlığı, nektar konsantrasyonu veya nektarın elde edilmesinin kolaylığı gibi birçok faktöre bağlıdır. ABC algoritmasında koloni üç grup arıdan oluşur: işçi arılar, kaşif arılar ve gözcü arılar. İşçi arılar, daha önceden keşfedilen kaynaklardan kovana yiyecek getirilmesinden sorumludur. Her besin kaynağı için yalnızca bir yapay işçi arının olduğu varsayılmaktadır. İşçi arılar ziyaret ettikleri kaynağın kalitesi ve yeriyle ilgili bilgiyi diğer arılara iletirler. Besin kaynağı terk edilen işçi arı kaşif yada gözcü arı olur. Kaşif arılar yeni besin kaynağı bulmak için arayışa başlar. Gözcü arılar ise işçi arıların davranışlarını izlerler ve işçi arıların paylaştığı bilgiyi kullanarak yeni bir kaynağa yönelirler. Arı kolonilerinin yiyecek kaynaklarının konumları çözülmek istenen problemin muhtemel çözümlerine, nektar miktarı ise çözümün kalitesini ifade etmektedir.

ABC algoritmasında başlangıç değerlerinin üretilmesi, SN yiyecek kaynağı sayısı ve M optimize edilecek parametre sayısı olmak üzere, Denklem (5) ile gerçekleştirilir.

$$x_{ij} = x_j^{min} + rand(0,1) * (x_j^{max} - x_j^{min}), \quad i=1, \ldots, SN \quad j=1, \ldots, M \tag{5}$$

x_i çözümünün görevli arısı için yeni kaynak üretimi Denklem (6) ile hesaplanırken, bu kaynağın maliyet değeri, $f(v_i)$, Denklem (7)'de yerine koyularak uygunluk değeri elde edilir. Gözcü arılar ise seçme işlemini Denklem (8)'de verilen olasılık değerine göre yapar.

$$v_{ij} = x_{ij} + \phi_{ij} * (x_{ij} - x_{kj}) \tag{6}$$

$$uygunluk_i = \begin{cases} 1/(1+f_i), & f_i \geq 0 \\ 1 + abs(f_i), & f_i < 0 \end{cases} \tag{7}$$

$$p_i = \frac{uygunluk}{\sum_{i=1}^{SN} uygunluk} \tag{8}$$

2.2.2 *Guguk kuşu arama (CS) algoritması*

Yang ve Deb [26] tarafından önerilen guguk kuşu algoritması, guguk kuşlarının kuluçka paraziti davranışından ilham alan meta-sezgisel bir algoritmadır. Guguk kuşu bir seferde bir yumurta bırakır ve bunu rasgele olarak diğer kuş türlerinin yuvalarına bırakır. Ev sahibi kuşlar, guguk kuşu yumurtasını kuluçkaya yatırabilir ve yumurtadan çıkarabilir veya izinsiz giren yumurtayı $P_\alpha \in (0,1)$ olasılıkla tanıyıp yuvayı terk edebilir. Guguk kuşu algoritması adımları aşağıda verilmiştir.

Öncelikle her bir guguk kuşu x_i^{t-1} yuvasını rasgele ve x_i^t yuvasını Levy flight kullanarak seçer. Daha sonra bu yuvalar F fonksiyonu ile değerlendirilir. Eğer $F(x_i^t) > F(x_i^{t-1})$ ise guguk kuşu x_i^{t-1}'yi x_i^t ile değiştirir ve x_i^t yuvasına bir yumurta bırakır. Daha sonra *t*. adımdaki tüm yuvaları sıralayarak en iyi yuvayı bulur. Bu sırada bazı kötü yuvalar P_α olasılığı ile terk edilir ve yenileri inşa edilir. Yeni bir yuva (ya da çözüm) x_i^{t+1} ise Denklem (9) ile hesaplanır. Denklemde $step\_length$ Levy dağılımıyla ilişkili adım uzunluğu, α ise ağırlık katsayısıdır.

$$x_i^{t+1} = x_i^t + \alpha \times step\_length \tag{9}$$

2.2.3 *Diferansiyel evrim (DE) algoritması*

DE algoritması [27], genetik algoritmalardaki çaprazlama, mutasyon ve seçim gibi operatörleri kullanan popülasyona dayalı bir algoritmadır. Genetik algoritmalarda daha iyi çözümler oluşturmada çaprazlama kullanılırken, DE'de mutasyon işlemi kullanılır. Bu temel işlem,

popülasyondaki rastgele örneklenen çözüm çiftlerinin farklılıklarına dayanır.

D parametreden oluşan bir optimizasyon görevi *D* boyutlu bir vektörle temsil edilebilir. DE'de başlangıçta NP çözüm vektörlerinden oluşan bir popülasyon rastgele oluşturulur. Bu popülasyon mutasyon, çaprazlama ve seçim operatörleri uygulanarak başarılı bir şekilde iyileştirilir.

Her $x_{i,G}$ hedef vektörü için mutant vektörü Denklem (10) ile üretilir.

$$v_{i,G+1} = x_{i,G} + K.\left(x_{r1,G} - x_{i,G}\right) + F.\left(x_{r2,G} - x_{r3,G}\right) \quad (10)$$

Denklemde $i, r_1, r_2, r_3 \in \{1, 2, \dots, NP\}$ rasgele seçilmeli birbirinden farklı olmalıdır. *F* ölçekleme *K* ise kombinasyon faktörüdür. Çaprazlama işleminde ana vektör, mutasyona uğramış vektörle karıştırılarak $u_{ji,G+1}$ deneme vektörleri üretilir (Denklem (11)).

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} \; if \; \left(rnd_j \le CR\right) or \; j = rn_i \\ q_{ji,G} \quad if \; \left(rnd_j > CR\right) or \; j \ne rn_i \end{cases} \quad (11)$$

Denklem 11'de $j = 1, 2, \dots, D$; $r_j \in [0,1]$ rasgele bir sayı, $CR \in [0,1]$ çaprazlama sabiti ve $rn_i \in (1, 2, \dots, D)$ rasgele seçilen bir indekstir.

Popülasyondaki tüm çözümler, uygunluk değerlerine bağlı olmaksızın ebeveyn olarak seçilme şansına sahiptir. Mutasyon ve çaprazlama işlemlerinden sonra üretilen çocuk değerlendirilir. Daha sonra, çocuk vektörünün ve ebeveyninin performansı karşılaştırılır ve daha iyi olan seçilir. Ebeveyn hala daha iyiyse, popülasyonda tutulur.

2.2.4 *Gri kurt optimizasyon (GWO) algoritması*

GWO algoritması [28], gri kurtların sosyal hiyerarşik yapısından ve akıllı avlanma yönteminden esinlenmiştir. Gri kurtların yaşamında katı bir sosyal hiyerarşi vardır. Bir gri kurt sürüsünün liderleri alfadır. Bireyler alfa tarafından verilen karara uymalıdır. Sonraki seviye betadır. Beta, alfanın danışmanıdır ve sürüyü disiplin altına almaktan sorumludur. Sürüdeki en zayıf seviye, omegadır. Omega seviyesindeki kurtlar, diğer bireylerin emirlerine uymak zorundadır ve yiyecek yemelerine izin verilen son kurtlardır. Alfa, beta ve omega dışındaki kalan kurtlar, delta olarak adlandırılır. Delta seviyesindeki kurtlar, alfa ve beta kurtlarına itaat eder ve omega kurtlarına hükmeder.

Gri kurtların avlanması üç ana bölümden oluşur: (1) Avı izleme, kovalama ve yaklaşma. (2) Avı takip etme, kuşatma ve hareket etmeyi bırakana kadar taciz etme. (3) Ava saldırma. Avlanma sırasında en iyi çözüm α olarak kabul edilir. İkinci ve üçüncü en iyi çözümler sırasıyla β ve δ olarak kabul edilir ve diğer çözümün ω olduğu varsayılır. Avlanma (optimizasyon) α, β ve δ tarafından yönlendirilir ve ω ise onları takip eder. Avın konumunu keşfetmek için birbirlerinden ayrılırlar ve sonra ava saldırmak için birleşirler. Gri kurtların ayrışmasını matematiksel olarak modellemek için $\vec{A}$ kullanılabilir. $\vec{A}$, arama ajanını avdan ayrılmaya zorlamak için 1'den büyük veya ava yaklaştırmak için 1'den küçük değer alan rastgele bir vektördür ve bu da GWO'daki küresel aramayı vurgular. $\vec{A}$ >1 olduğunda, gri kurt daha iyi çözümler aramak için avdan uzaklaşmaya (yerel optimum) zorlanır. $\vec{A}$ <1 olduğunda ise aday çözümler avda birleşir. Bu süreç durdurma kriterlerinin karşılanıncaya kadar devam eder. Algoritmadaki av arama süreci Denklem (12) ile modellenmiştir.

$$\vec{D} = \left|\vec{C}.\vec{X}_p(t) - \vec{X}(t)\right| \qquad \vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \quad (12)$$

$\vec{X}_p(t)$ avın pozisyonunu, $\vec{X}(t)$ kurdun pozisyonunu, $\vec{D}$ ise kurdun yeni pozisyonunun hesaplanmasında kullanılan vektörü ifade etmektedir. $\vec{A}$ ve $\vec{C}$ katsayı vektörleri olup Denklem (13) ile hesaplanır.

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a} \qquad \vec{C} = 2.\vec{r_2} \quad (13)$$

Burada $\vec{a}$ iterasyonlar boyunca 2'den 0'a lineer olarak azalan bir vektördür. $\vec{r_1}$ ve $\vec{r_2}$ ise [0,1] aralığında rasgele vektörlerdir.

2.2.5 *Harris şahini optimizasyon (HHO) algoritması*

HHO algoritması [29], Harris şahinlerinin avını keşfetme, sürpriz saldırı ve farklı saldırı stratejilerinden esinlenerek önerilmiştir. HHO, popülasyona dayalı, eğimsiz bir optimizasyon tekniğidir. HHO'da, Harris şahinleri rastgele bazı konumlara tüner ve iki stratejiye dayalı olarak bir av tespit etmeyi bekler. Her tüneme stratejisi için eşit bir q şansı düşünülürse, diğer aile üyelerinin konumlarına ve tavşana göre ($q < 0.5$), ya da rastgele uzun ağaçlara ($q \ge 0.5$) tünerler (Denklem (14)).

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1\left|X_{rand}(t) - 2r_2X(t)\right| & q \ge 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - r_3\left(LB + r_4(UB - LB)\right) & q < 0.5 \end{cases} \quad (14)$$

Denklemde $X(t+1)$ şahinin sonraki iterasyondaki konumunu, $X_{rabbit}(t)$ tavşanın konumunu, r_1, r_2, r_3, r_4 (0,1) aralığında rasgele sayıları, *LB* ve *UB* ise değişkenlerin alt ve üst sınırlarını ifade etmektedir. $X_{rand}(t)$ mevcut popülasyondan rastgele seçilen bir şahin ve $X_m(t)$ ise mevcut şahin popülasyonunun ortalama konumudur.

HHO algoritması keşiften sömürüye geçebilir ve ardından avın kaçma enerjisine bağlı olarak farklı sömürücü davranışlar arasında geçiş yapabilir. $E = 2E_0(1 - t/T)$ ile modellenen avın enerjisi, kaçma davranışı sırasında önemli ölçüde azalır. Burada, *E* avın kaçma enerjisi, *T* maksimum yineleme sayısı ve E_0 ise enerjisinin başlangıç durumudur. HHO'da, E_0 her yinelemede (−1, 1) aralığı içinde rastgele değişir. E_0'ın değeri 0'dan -1'e düştüğünde tavşan fiziksel olarak zayıflıyor, 0'dan 1'e çıktığında ise tavşan güçleniyor anlamına gelir. Avlar her zaman tehdit edici durumlardan kaçmaya çalışırlar. Av ne yaparsa yapsın, şahinler avı yakalamak için sert veya yumuşak bir kuşatma gerçekleştirecektir. $E < 0.5$ olduğunda yumuşak kuşatma, $E \ge 0.5$ olduğunda ise sert kuşatma gerçekleşir.

2.2.6 *Güve-alev optimizasyonu (MFO) algoritması*

MFO algoritması, güvelerin navigasyon davranışından esinlenilerek geliştirilmiştir [30]. Güvelerin geceleri özel bir navigasyon yöntemleri vardır. Güveler ay ışığını kullanarak geceleri uçmak üzere evrimleşmişlerdir. Navigasyon için enine yönelim adı verilen bir mekanizma kullanırlar. Bu yöntemde, bir güve, uzun mesafeleri düz bir yolda kat etmek için çok etkili bir mekanizma olan aya göre sabit bir açıyı koruyarak uçar. Ay, güveden çok uzakta olduğundan, bu mekanizma düz bir çizgide uçmayı garanti eder.

MFO algoritmasında, aday çözümlerin güveler olduğu ve problemin değişkenlerinin güvelerin uzaydaki konumları olduğu varsayılır. Bu nedenle güveler, konum vektörlerini değiştirerek 1-B, 2-B, 3-B veya hiper boyutlu uzayda uçabilirler. Algoritmada güveler ve alevlerin her ikisi de çözümdür. Aralarındaki fark, her yinelemede ele alınma ve güncellenme şekilleridir. Güveler, arama alanında hareket eden arama ajanlarıdır, alevler ise güvelerin o ana kadar elde ettiği en iyi konumdur. Her güve bir alevin etrafında arama yapar ve daha iyi bir çözüm bulması durumunda onu günceller. Bu mekanizma ile bir güve asla en iyi çözümünü kaybetmez. Güvelerin güncelleme mekanizması logaritmik bir spiral olup Denklem (15)'le modellenmiştir.

$$S(M_i, F_i) = D_i e^{bt}\cos(2\pi t) + F_j$$
$$D_i = |F_j - M_i| \qquad (15)$$

Denklemde D_i i. güvenin j. aleve olan uzaklığını, b logaritmik spiralin şeklini tanımlayan sabiti, t ise [-1,1] aralığında rasgele bir sayıyı ifade eder. Denklemden görüldüğü gibi güvenin sonraki konumu aleve göre belirlenmektedir. t parametresi, güvenin bir sonraki pozisyonunun aleve ne kadar yakın olması gerektiğini tanımlar (t = -1 aleve en yakın pozisyondur, t = 1 ise en uzak pozisyonu gösterir)

2.2.7 *Parçacık sürüsü optimizasyonu (PSO) algoritması*

PSO'da [31] bir dizi basit varlık (parçacıklar) bir problemin veya fonksiyonun arama alanına yerleştirilir ve her biri mevcut konumunda hedef fonksiyonu değerlendirir. Daha sonra her parçacık, kendi mevcut ve en iyi (en iyi uyum) konumlarını, sürünün bir veya daha fazla üyesinin konumlarıyla birleştirerek arama alanındaki hareketini belirler. Bu sayede sürünün bir bütün olarak, uyum fonksiyonunun bir optimumuna doğru hareket etmesi sağlanır.

Parçacık sürüsündeki her birey, üç D-boyutlu vektörden oluşur. Bunlar, geçerli konum $\vec{x_i}$, önceki en iyi konum $\vec{p_i}$ ve hız $\vec{v_i}$'dir. D arama alanının boyutudur. Algoritmanın her yinelemesinde, geçerli konum bir problem çözümü olarak değerlendirilir. Bu konum, şu ana kadar bulunanlardan daha iyiyse, koordinatlar ikinci vektör olan $\vec{p_i}$'de saklanır. Şimdiye kadarki en iyi fonksiyon sonucunun değeri, daha sonraki yinelemelerde karşılaştırma için $pbest_i$'de saklanır. Amaç, daha iyi konumlar bulmaya ve $\vec{p_i}$ ve $pbest_i$'yi güncellemeye devam etmektir. Yeni noktalar, $\vec{x_i}$'ye $\vec{v_i}$ koordinatları eklenerek seçilir ve algoritma, adım boyutu olarak görülebilen $\vec{v_i}$'yi ayarlayarak çalışır.

Algoritmada her parçacık için,optimizasyon uygunluk fonksiyonu değerlendirilir. Bu değer parçacığın $pbest_i$'si ile karşılaştırılır. Mevcut değer $pbest_i$'den daha iyiyse, $pbest_i$ mevcut değere, $\vec{p_i}$ D boyutlu uzaydaki mevcut konum $\vec{x_i}$'ye eşitlenir. Şimdiye kadar komşuluk içindeki en iyi başarıya sahip parçacık belirlenir ve endeksini g değişkenine atanır. Daha sonra parçacığın hızı ve konumu Denklem (16)'ya göre değiştirilir.

$$\begin{cases} \vec{v_i} \leftarrow \vec{v_i} + \vec{U}(0,\phi_1)\otimes(\vec{p_i} - \vec{x_i}) + \vec{U}(0,\phi_2)\otimes(\vec{p_g} - \vec{x_i}) \\ \vec{x_i} \leftarrow \vec{x_i} + \vec{v_i} \end{cases} \qquad (16)$$

2.2.8 *Sinüs kosinüs algoritması (SCA)*

SCA algoritması [32], optimizasyon problemlerini çözmek için sinüs/kosinüs matematiksel fonksiyonlarına dayanmaktadır. Stokastik popülasyon tabanlı optimizasyon algoritmalarının ortak özelliği optimizasyon sürecinin iki faza bölünmesidir: keşif ve sömürü. İlk fazda, optimizasyon algoritması, arama alanının umut verici bölgelerini bulmak için, çözümler kümesindeki rastgele çözümleri yüksek bir rastgelelik oranıyla birleştirir. Sömürü fazında ise, rastgele çözümlerde kademeli değişiklikler olur ve rastgele varyasyonlar keşif fazındakilerden çok daha azdır. SCA algoritmasında, her iki faz için Denklem (17)'deki konum güncelleme ifadeleri önerilmiştir.

$$X_i^{t+1} = X_i^t + r_1 \times \sin(r_2) \times |r_3 P_i^t - X_i^t|$$
$$X_i^{t+1} = X_i^t + r_1 \times \cos(r_2) \times |r_3 P_i^t - X_i^t| \qquad (17)$$

Denklemde X_i^tmevcut çözümün i. boyutta t. iterasyondaki konumunu, r_1, r_2, r_3 rasgele sayıları, P_i hedefin i. boyuttaki pozisyonunu ifade etmektedir. Bu iki denklem [0,1] aralığında rasgele değer alan r_4 e göre Denklem (18)'deki gibi birleştirilir.

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times \left|r_3 P_i^t - X_i^t\right| & r_4 < 0.5 \\ X_i^t + r_1 \times \cos(r_2) \times \left|r_3 P_i^t - X_i^t\right| & r_4 \geq 0.5 \end{cases} \qquad (18)$$

Yukarıdaki denklemlerin gösterdiği gibi, SCA'da dört ana parametre vardır: r1, r2, r3 ve r4. r1 parametresi, çözüm ile hedef arasındaki veya dışındaki alanda olabilen bir sonraki konumun bölgesini (veya hareket yönünü) belirler. r2 parametresi, hareketin hedefe veya hedefin dışına doğru ne kadar uzakta olması gerektiğini tanımlar. r3 parametresi, hedefin mesafeyi tanımlamadaki etkisini stokastik olarak vurgulamak (r3 > 1) veya önemsizleştirmek (r3 < 1) için hedefe rastgele bir ağırlık katar. Son olarak, r4 parametresi sinüs ve kosinüs bileşenleri arasında geçiş yapılmasını sağlar.

2.2.9 *Simbiyotik organizmalar arama (SOS) algoritması*

Doğadaki canlılar yaşamak için diğer canlılara ihtiyaç duyarlar. İki tür arasında var olan karşılıklı işbirliği simbiyotik olarak adlandırılır. SOS [33] algoritması doğadaki bu simbiyotik ilişkiden esinlenilerek geliştirilmiştir. Doğadaki simbiyotik bağların bazıları, karşılıkcılık (mutualism), ortakçılık (commensalism) ve asalaklık (parasitism)'tır. SOS, optimum global çözümü

arama sürecinde, arama alanındaki olası aday çözümlerden oluşan bir popülasyonu yinelemeli olarak kullanır. SOS, ekosistem adı verilen bir başlangıç popülasyonuyla başlar. Başlangıç ekosisteminde, bir grup organizma arama alanında rastgele oluşturulur. Her organizma, karşılık gelen soruna bir aday çözümü temsil eder.

SOS'ta X_i, ekosistemin i. üyesiyle eşleştirilmiş bir organizmadır. Daha sonra ekosistemden rastgele seçilen başka bir organizma X_j, X_i ile etkileşime girer. Her iki organizma da ekosistemde karşılıklı hayatta kalma (mutualism) avantajını artırma amacıyla karşılıklı bir ilişkiye girer. X_i ve X_j, için yeni aday çözümler, (Denklem (19)), iki organizma arasındaki karşılıklı simbiyoza dayanarak hesaplanır.

$$\begin{aligned} X_{inew} &= X_i + rand(0,1) * (X_{best} - Mutual\_vector * BF_1) \\ X_{jnew} &= X_j + rand(0,1) * (X_{best} - Mutual\_vector * BF_2) \\ Mutual\_vector &= (X_i + X_j)/2 \end{aligned} \quad (19)$$

Denklemde BF_1 ve BF_2 fayda faktörleri olup rastgele 1 veya 2 olarak belirlenir. Bu faktörler her organizma için fayda düzeyini, yani bir organizmanın etkileşimden kısmen veya tamamen faydalanıp faydalanmadığını temsil eder. Organizmalar yalnızca yeni uygunlukları etkileşim öncesi uygunluklarından daha iyiyse güncellenirler.

Ortakçılıkta (commensalism) benzer şekilde ekosistemden rastgele seçilen bir organizma X_j, X_i ile etkileşime girer. Bu durumda, X_i etkileşimden faydalanmaya çalışır. Ancak, organizma X_j'nin kendisi ilişkiden ne faydalanır ne de zarar görür. X_i'nin yeni aday çözümü, organizma X_i ve X_j arasındaki ortak yaşam simbiyoza göre Denklem (20) ile hesaplanır.

$$X_{inew} = X_i + rand(-1,1) * (X_{best} - X_j) \quad (20)$$

Asalaklıkta ise organizma X_i'ye "Parasite_Vector" adı verilen yapay bir parazit yaratılır. Organizma X_j, ekosistemden rastgele seçilir ve parazit vektörüne konak görevi görür. Parasite_Vector, ekosistemde X_j'nin yerini almaya çalışır. Parasite_Vector daha iyi bir uygunluk değerine sahipse, organizma X_j'yi öldürür ve ekosistemdeki konumunu alır. X_j'nin uygunluk değeri daha iyiyse, X_j parazite karşı bağışıklık kazanır ve Parasite_Vector artık o ekosistemde yaşayamaz.

2.2.10 zSalp sürüsü algoritması (SSA)

Salplar, Salpidae familyasına ait şeffaf, fıçı biçimli gövdeye sahip, dokuları deniz analarına benzeyen canlılardır. Salpların en ilginç davranışlarından biri, sürü halinde hareket etmeleridir. Derin okyanuslarda, salplar genellikle salp zinciri adı verilen bir sürü oluştururlar. Bu davranışın matematiksel modellenmesi ile SSA algoritması geliştirilmiştir [34]. Salp zincirlerini matematiksel olarak modellemek için, popülasyon önce iki gruba ayrılır: lider ve takipçiler. Lider, zincirin önündeki salp iken, geri kalan salplar takipçi olarak kabul edilir.

Diğer sürü tabanlı yöntemlere benzer şekilde, salpların pozisyonu n boyutlu bir arama alanında tanımlanır; burada n, verilen bir problemin değişken sayısıdır. Bu nedenle, tüm salpların pozisyonu x adlı iki boyutlu bir matriste saklanır. Ayrıca, arama alanında sürünün hedefi olarak F adlı bir yiyecek kaynağının olduğu varsayılır. Liderin pozisyonunu güncellemek için Denklem (21) kullanılır.

$$x_j^1 = \begin{cases} F_j + c_1\left((ub_j - lb_j)c_2 + lb_j\right) & c_3 \geq 0 \\ F_j - c_1\left((ub_j - lb_j)c_2 + lb_j\right) & c_3 < 0 \end{cases} \quad (21)$$

Denklemde x_j^1 lider salpin j. boyuttaki pozisyonunu , F_j yiyecek kaynağının poziyonunu, ub_j, lb_j sırasıyla j. boyuttaki üst ve alt sınırı, c_1, c_2, c_3 rasgele sayıları ifade eder. Takipçilerin konumu ise Denklem (22)'ye göre güncellenir. Denklemde i≥2, i. takipçi salpin j. boyuttaki konumunu, t zamanı, v_0 başlangıç hızını, ifade ederken $a = v_{final}/v_0$ ve $v = (x - x_0)/t$ dir.

$$x_j^i = \frac{1}{2}at^2 + v_0 t \quad (22)$$

2.2.11 Balina optimizasyon algoritması (WOA)

WOA [35], kambur balinaların kabarcık ağı avlanma stratejisine dayalı olarak geliştirilen meta-sezgisel algoritmalardan biridir. Kambur balinalar genellikle su yüzeyine yakın küçük balık sürüleri tarafından beslenir. Kambur balinalar avlarına yaklaşırken hava kabarcıkları oluştururlar, böylece avlarını bir arada tutabilirler ve avlarına görülmeden yaklaşabilirler. Balina optimizasyon algoritması temel olarak 3 bölümden oluşur: avı çevreleme, kabarcık ağı avlanma ve av aramadır.

Kambur balinalar avın yerini tanıyabilir ve onları çevreleyebilir. Optimizasyon problemlerinde önceden bilinmeyen en iyi çözümler olarak, en iyi çözüm veya en iyi çözüme yakın bir nokta en iyi çözüm olarak kabul edilir. En iyi çözüm tanımlandıktan sonra, diğer çözümlerin konumları en iyiye göre güncellenir. Balinaların çevreleme davranışının matematiksel modeli Denklem (23)'te sunulmuştur.

$$\begin{aligned} \vec{D} &= \left|\vec{C}.\vec{X}_{best}(t) - \vec{X}(t)\right| \\ \vec{X}(t+1) &= \vec{X}_{best}(t) - \vec{A}.\vec{D} \end{aligned} \quad (23)$$

$\vec{X}_{best}(t)$ en iyi çözümü, $\vec{A}$ ve $\vec{C}$ katsayı vektörlerini ifade etmektedir. Bu vektörler Denklem (24) ile hesaplanır. Burada $\vec{a}$ iterasyonlar boyunca 2'den 0'a lineer olarak azalan bir vektördür. $\vec{r}$ ise [0,1] aralığında rasgele vektördür.

$$\begin{aligned} \vec{A} &= 2\vec{a}.\vec{r} - \vec{a} \\ \vec{C} &= 2.\vec{r} \end{aligned} \quad (24)$$

Av bulunduktan sonra av iki farklı hareketle, yani Küçülen Çevreleme Mekanizması ve Spiral Güncelleme Pozisyonu ile hareket eder. Spiral hareket için en iyi ajan ile arama ajanı arasındaki fark Denklem (25) ile hesaplanır. Burada b logaritmik spiral için sabiti, l ise [-1, 1] aralığındaki rastgele bir sayıyı ifade eder. D⃗' en iyi arama ajanı ile mevcut arama ajanı arasındaki farktır.

$$X(t+1) = \overrightarrow{D'}.e(bl).\cos(2\pi l) + \vec{X}_{best}(t) \quad (25)$$

3 Deneysel bulgular

3.1 Veri seti

Berkeley görüntü segmentasyonu veri seti [37] herkesin kullanımına açık olarak 200 eğitim ve 100 test verisi olmak üzere 300 görüntüden oluşmaktadır (BSD300). Veri seti gri seviyeli ve renkli görüntüler içermektedir. Bu veri seti doğal görüntülerden bitki, insan, yiyecek vb. nesneye özgü görüntülere kadar geniş bir yelpazedeki görüntülerden oluştuğu için segmentasyon çalışmalarında sıklıkla kullanılmaktadır. Bu çalışmada MSA'ların çok seviyeli görüntü segmentasyonu performanslarının karşılaştırılması amacıyla BSD300 veri setinin 100 renkli görüntüden oluşan test kümesi, BSD100 kullanılmıştır. BSD100 veri setinden rasgele seçilmiş görüntüler ve görüntülere ait histogramlar Şekil 1'de verilmiştir. Şekilden görüldüğü gibi görüntüler çeşitli parlaklık ve renk dağılımlarına sahiptir. Bu özellikler deneylerdeki hesaplama karmaşıklığını ve problem çeşitliliğini oluşturmaktadır.

3.2 *Deneysel ayarlar*

Deneylerde çok seviyeli görüntü eşiklemeye dayalı görüntü segmentasyonu literatüründe en çok kullanılan ABC, CS, DE, GWO, HHO, MFO, PSO, SCA, SOS, SSA, WOA algoritmalarının performansları karşılaştırılmıştır. Algoritmaların adil şekilde kıyaslanabilmesi için sonlandırma kriteri olarak maksimum amaç fonksiyonu değerlendirme sayısı, *maxFEs* kullanılmıştır. 100x3 problem (her bir görüntü ve görüntünün her kanalı ayrı bir optimizasyon problemidir) *maxFEs*=500**d* ve *maxFEs*=1000**d* için çözülerek algoritmaların başarımları değerlendirilmiştir. Her problemin optimizasyonu sürecinde algoritmalar 25 kez koşturulmuştur. *m*=2, 3, 4, 5, 6, 7, 8 seviyeli segmentasyon ile algoritmaların düşük ve orta seviye eşiklerdeki başarımı incelenmiştir. Deneyler, 12 GB RAM'e sahip ve Windows 11 çalıştıran AMD Ryzen 7 6800H 3.2 GHz CPU kullanılarak gerçekleştirilmiştir. Çalışmada 11 MSA, 100 görüntü, 3 renk kanalı, 7 farklı eşik seviyesi, 2 farklı sonlandırma kriteri için toplamda 46200 deney yapılmış, elde edilen sonuçlar Friedman istatistiksel analiz yöntemi ile değerlendirilmiştir. Deneylerde MSA'ların orijinal ayarları kullanılmıştır. Bu ayarlar Tablo 1'de verilmiştir.

3.3 İstatistiksel analiz

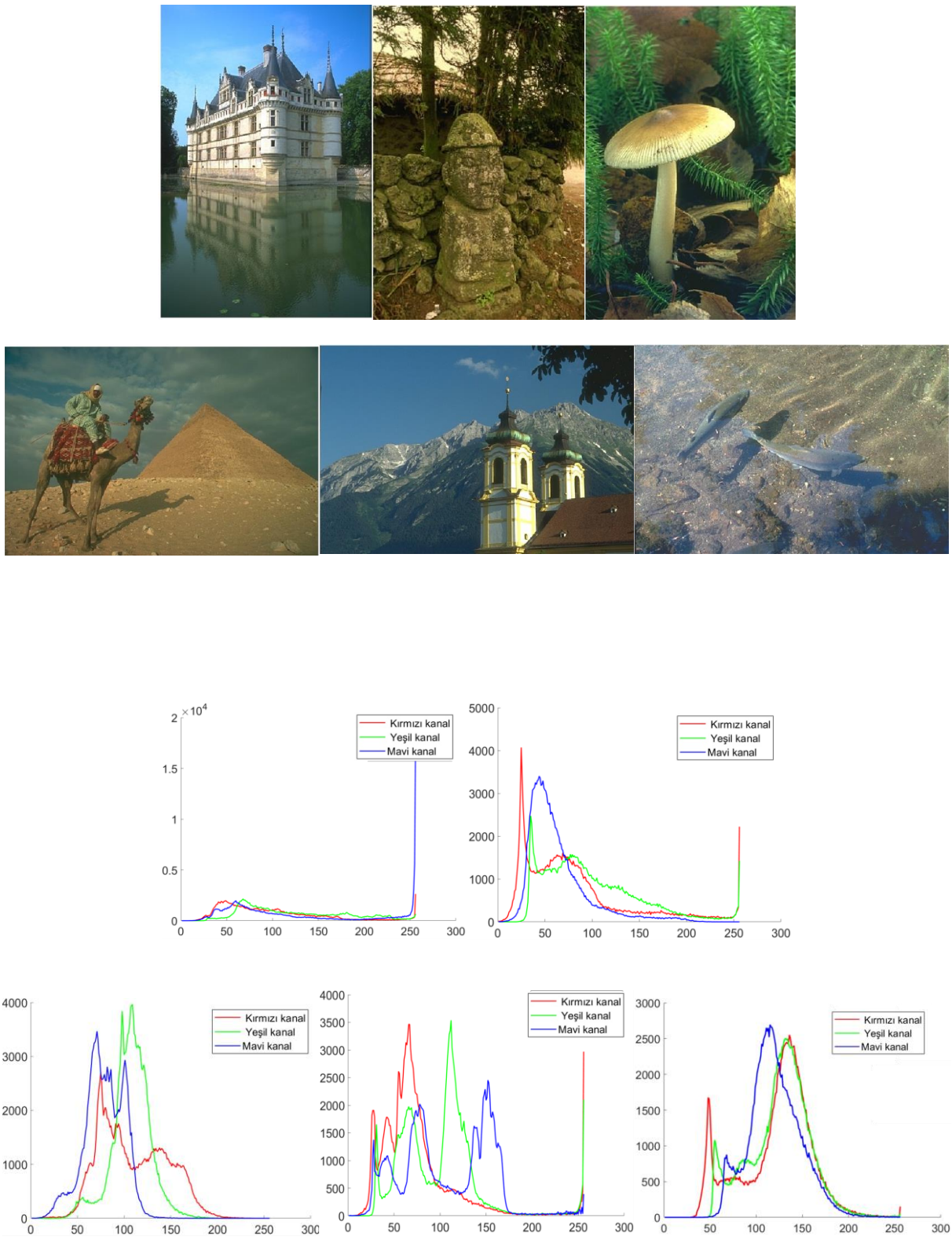
Friedman testi, çoklu test denemeleri arasındaki farklılıkları tespit etmek için kullanılan parametrik olmayan bir istatistiksel analiz yöntemidir. Bu test özellikle aynı denek üzerinde farklı yöntemlerden, algoritmalardan veya koşullardan birden fazla ölçümün alındığı durumlarda kullanışlıdır. Çalışmada istatistiksel analizler, 11 algoritma, 100 görüntü, 25 çalıştırmayı içermektedir. Görüntüler renkli olduğu için R, G ve B kanalları için ayrı ayrı çözümler üretilmiş ve ortalaması alınmıştır. Deneylerde kullanılan algoritmaların *m*=2, 3, 4, 5, 6, 7, 8 seviyeli segmentasyon için, sonlandırma kriteri *maxFEs*=500**d* ve *maxFEs*=1000**d* seçildiğinde elde dilen deneysel verilerine ait Friedman sıralamaları ve skorları sırasıyla Tablo 2 ve Tablo 3'te görülmektedir.

Tablo 1. MSA ayarları

Algoritma	Ayarlar
ABC	koloni boyutu = 50, SN = koloni boyutu/ 2, limit = D * SN
CS	yuva sayısı = 25, olasılık = 0.25, beta = 1.5
DE	popülasyon = 50, çaprazlama olasılığı = 0.2
GWO	arama ajan sayısı = 30
HHO	N = 30, Bfid = 1, nD = 10, Jr = 0.25
MFO	güve sayısı = 30
PSO	sürü boyutu = 30, bilişsel ve sosyal sabit = 2
SCA	arama ajan sayısı = 30, a = 2
SOS	ekosistem boyutu = 50
SSA	popülasyon = 30
WOA	popülasyon = 30

Tablo 2'deki sonuçlar incelendiğinde *maxFEs*=500**d*'de tüm eşik seviyeleri için MFO algoritmasının rakiplerinden daha üstün başarı gösterdiği görülmektedir. MFO'yu yine tüm eşik seviyelerinde ikinci olarak PSO takip etmektedir. PSO'dan sonra başarılı olan algoritmalar ABC, GWO ve SSA'dır. Fakat eşik seviye sayısına göre bu algoritmaların sıralamadaki yeri değişmektedir. Örneğin m=2, 3, 4 ,5 için SSA üçüncü başarılı algoritma olmuşken, m=6, 7, 8 için yerini ABC ye bırakmış ve sıralamada beşinciliğe düşmüştür. *maxFEs*=500**d* için m=2 eşik seviyesi hariç dördüncü sırada GWO algoritması yer almaktadır. Bu boyutta rakiplerine kıyasla en başarısız algoritmaların DE, SOS ve SCA olduğu görülmektedir.

Tablo 3'teki sonuçlar ise algoritmalara biraz daha şans verildiğinde bazı eşik seviyelerindeki sıralamaların değişebileceğini göstermektedir. *maxFEs*=1000**d* boyutunda da MFO algoritması 7 farklı durumdan 4'ünde en başarılı algoritma olmayı başarmıştır. *m*=3, 4, 5, ve 6 seviyeli eşiklemede MFO birinci, PSO ikinci en başarılı algoritma olmayı başarmıştır. *m* = 7, 8 seviyeli eşiklemede bu algoritmalar yer değiştirmiş, *m*=2 seviyeli eşiklemede ise ABC'den sonra ikinci ve üçüncü başarılı algoritmalar olmuştur. Tablodaki sonuçlar ABC algoritmasının *maxFEs*=1000**d* boyutunda, 500*d boyutuna göre sıralamada daha üste çıktığını göstermektedir. *maxFEs*=500*d boyutunda genelde dördüncü ve beşinci sırada yer alan ABC algoritması *maxFEs*=1000*d boyutunda *m*=2 seviyeli eşiklemede en başarılı, geri kalan tüm durumlarda ise en başarılı üçüncü algoritma olmayı başarmıştır. Bu boyutta da ilk beşe giren algoritmalar değişmemiş ve MFO, PSO, ABC, SSA ve GWO olarak belirlenmiştir. En başarısız algoritmalar ise yine *maxFEs*=500**d* boyutundaki gibi DE, SOS ve SCA'dır.



Şekil 1. Berkeley veri setinden örnek görüntüler ve histogramları (soldan sağa: '102061.jpg', '55073.jpg', '208001.jpg', '299086.jpg', 126007.jpg', '86068.jpg')

Tablo2. *maxFEs*=500**d için* MSA'ların Friedman skorları ve sıralamaları

m=2		m=3		m=4		m=5		m=6		m=7		m=8	
MFO	**3.59**	**MFO**	**2.55**	**MFO**	**2.44**	**MFO**	**2.52**	**MFO**	**2.59**	**MFO**	**2.63**	**MFO**	**2.76**
PSO	**3.69**	**PSO**	**2.80**	**PSO**	**2.83**	**PSO**	**2.90**	**PSO**	**2.91**	**PSO**	**2.82**	**PSO**	**2.85**
SSA	4.34	SSA	3.96	SSA	4.24	SSA	4.35	GWO	4.41	ABC	4.32	ABC	4.13
ABC	4.36	GWO	4.52	GWO	4.45	GWO	4.46	ABC	4.46	GWO	4.43	GWO	4.42
GWO	4.74	ABC	4.92	ABC	4.85	ABC	4.62	SSA	4.46	SSA	4.55	SSA	4.62
WOA	5.53	WOA	5.67	WOA	5.72	WOA	5.68	WOA	5.61	WOA	5.64	WOA	5.58
DE	7.14	CS	7.73	CS	7.37	CS	7.18	CS	7.07	HHO	6.95	HHO	6.78
HHO	7.37	HHO	7.74	HHO	7.61	HHO	7.41	HHO	7.18	CS	7.00	CS	6.90
SOS	7.50	DE	7.99	DE	7.81	DE	7.77	DE	7.77	DE	7.81	DE	7.87
CS	7.91	SOS	8.00	SOS	8.41	SOS	8.69	SOS	8.97	SOS	9.14	SOS	9.28
SCA	9.84	SCA	10.10	SCA	10.26	SCA	10.43	SCA	10.56	SCA	10.70	SCA	10.80

Tablo3. *maxFEs*=1000**d için* MSA'ların Friedman skorları ve sıralamaları

m=2		m=3		m=4		m=5		m=6		m=7		m=8	
ABC	4.81	**MFO**	**3.40**	**MFO**	**3.00**	**MFO**	**2.94**	**MFO**	**2.86**	**PSO**	**2.89**	**PSO**	**2.85**
MFO	**4.94**	**PSO**	**3.63**	**PSO**	**3.33**	**PSO**	**3.17**	**PSO**	**3.06**	**MFO**	**2.98**	**MFO**	**3.19**
PSO	**5.04**	ABC	3.75	ABC	4.08	ABC	4.22	ABC	4.24	ABC	4.11	ABC	3.90
GWO	5.26	SSA	4.27	SSA	4.29	SSA	4.35	GWO	4.32	GWO	4.40	GWO	4.46
SSA	5.27	GWO	4.42	GWO	4.36	GWO	4.36	SSA	4.53	SSA	4.64	SSA	4.80
DE	5.32	CS	5.94	CS	6.21	CS	6.16	CS	6.20	CS	6.15	CS	6.04
CS	5.45	WOA	6.58	WOA	6.49	WOA	6.36	WOA	6.23	WOA	6.15	WOA	6.08
SOS	5.98	HHO	7.25	HHO	7.03	HHO	6.91	HHO	6.67	HHO	6.52	HHO	6.29
WOA	6.62	SOS	8.11	DE	8.18	DE	8.08	DE	8.11	DE	8.14	DE	8.19
HHO	7.09	DE	8.24	SOS	8.59	SOS	8.90	SOS	9.12	SOS	9.24	SOS	9.36
SCA	10.22	SCA	10.43	SCA	10.43	SCA	10.54	SCA	10.67	SCA	10.77	SCA	10.85

Deneylerde her algoritma her bir problemin çözümü için 25 kez çalıştırılmış ve bu çalışmaların ortalama değerlerine göre analiz gerçekleştirilmiştir. Algoritmaların her bir eşik seviyesi için 100 görüntü üzerinde ortalama, maksimum minimum amaç fonksiyon değerleri ve bunların standart sapmaları hesaplanmıştır. Tüm sonuçların yazıda verilmesi okunaklığı azaltacağı için, Tablo 4'te çalışmadaki en zor problemler olan 7 ve 8 seviyeli eşikleme için elde edilen sonuçlar verilmiştir. Bu sonuçlara göre karmaşık segmentasyon problemlerinde MFO ve PSO algoritmaları diğerlerine kıyasla daha başarılı olmuştur.

Deneylerde koşturulan 11 algoritmanın tüm eşik seviyelerinde elde ettikleri Friedman skorlarının ortalaması alınmış ve buna göre algoritmaların başarım sıralaması Tablo 5'te verilmiştir. Buna göre sonlandırma kriterinin değeri değiştirilse de 100 görüntünün 7 farklı seviyede segmentasyonu için eşik değerlerinin belirlenmesinde en başarılı algoritma MFO'dur. MFO'dan sonra en başarılı ikinci algoritma PSO olmuştur. *maxFEs*=500**d* için en başarılı üçüncü algoritma SSA iken *maxFEs*=1000**d* için en başarılı üçüncü algoritma ABC olmuştur. Bu iki algoritma haricinde GWO algoritması her iki boyutta da dördüncü başarılı algoritma olmuştur. Kıyaslanan algoritmalar içerisinde her iki boyutta da en son sıralarda SOS ve SCA yer almıştır.

Tablo 5. Tüm eşik seviyeleri için MSA'ların Friedman skorlarının ortalamaları ve sıralamaları

maxFEs=500**d*		*maxFEs*=1000**d*	
MFO	**2.73**	**MFO**	**3.33**
PSO	**2.97**	**PSO**	**3.43**
SSA	4.36	ABC	4.16
GWO	4.49	GWO	4.51
ABC	4.52	SSA	4.59
WOA	5.64	CS	6.02
HHO	7.29	WOA	6.36
CS	7.31	HHO	6.82
DE	7.74	DE	7.75
SOS	8.57	SOS	8.47
SCA	10.38	SCA	10.56

Tablo 4. m=7 ve m=8 seviyeli eşikleme için 11 algoritmanın ortalama, maksimum, minimum amaç fonksiyonu değerleri ve bu değerlerin standart sapmaları

	m=8				m=7			
Alg.	**Ortalama**	**Maksimum**	**Minimum**	**Standart Sapma**	**Ortalama**	**Maksimum**	**Minimum**	**Standart Sapma**
***maxFEs*=1000*d**								
MFO	41.65331993	41.65587	41.65084	0.001130394	38.23118	38.23287	38.22897897	0.001029826
PSO	41.60941586	41.62171	41.59296	0.007431853	38.19773	38.20793	38.18189274	0.006950524
SSA	41.5843474	41.60567	41.56166	0.01017246	38.16061	38.17695	38.14113213	0.010501344
GWO	41.55595642	41.57991	41.53393	0.009269151	38.15412	38.16889	38.14088046	0.009005806
ABC	41.5619203	41.58368	41.52593	0.01183809	38.16718	38.18059	38.15384417	0.007058496
WOA	41.50074068	41.5175	41.48246	0.009370277	38.10939	38.12928	38.09305349	0.010601274
HHO	41.46251154	41.48834	41.43519	0.014847444	38.05685	38.08395	38.02856409	0.015136133
CS	41.57828216	41.5873	41.57226	0.003312661	38.17621	38.18201	38.17012933	0.003322286
DE	41.42230015	41.43144	41.41507	0.004295581	38.06616	38.07373	38.05778563	0.003638793
SOS	41.26618141	41.27488	41.25511	0.004757537	37.94974	37.96448	37.92856	0.007057516
SCA	40.74269401	40.77262	40.7075	0.016734962	37.56756	37.59464	37.54990202	0.013026707
***maxFEs*=500*d**								
MFO	38.1909059	38.20433	38.17594	0.007298532	41.60074	41.61391	41.577946	0.007731648
PSO	38.14199881	38.16432	38.11619	0.011826951	41.55368	41.58117	41.52137043	0.015437733
SSA	38.09651742	38.12051	38.07246	0.009808443	41.4858	41.52193	41.45777903	0.015207616
GWO	38.11646677	38.13597	38.10041	0.010294678	41.49625	41.51316	41.45418226	0.011920535
ABC	38.18535472	38.1885	38.18122	0.002252595	41.59216	41.59867	41.58793297	0.002389206
WOA	38.04812857	38.07017	38.01155	0.010926424	41.4236	41.44126	41.3902492	0.012552967
HHO	37.93275982	37.97205	37.89467	0.018413705	41.30365	41.35109	41.25536863	0.020864752
CS	38.03106156	38.04142	38.01559	0.006131919	41.39986	41.4105	41.38835807	0.005758005
DE	37.97516332	37.9844	37.96256	0.004927146	41.31267	41.32692	41.29685073	0.008065821
SOS	37.83039607	37.84897	37.81315	0.007562737	41.10877	41.125	41.09285678	0.010480436
SCA	37.41279261	37.43207	37.39356	0.011854887	40.56236	40.59452	40.5266354	0.017762358

4 Tartışma ve sonuçlar

Bu çalışmada ABC, CS, DE, GWO, HHO, MFO, PSO, SCA, SOS, SSA, WOA algoritmalarının çok seviyeli görüntü segmentasyonu performansları karşılaştırılmıştır. Algoritmaların sonlandırma kriteri olarak maksimum amaç fonksiyonu değerlendirme sayısı, *maxFEs*, kullanılmış, problemler *maxFEs*=500**d ve maxFEs*=1000**d* için çözülmüştür. BSD100 veri setindeki görüntülerin *m*=2, 3, 4, 5, 6, 7, 8 seviyeli segmentasyonu gerçekleştirilmiştir.

Görüntü segmentasyonu literatüründe pek çok MSA ve değiştirilmiş versiyonları problemin çözümüne uygulanmaktadır. Meta-sezgisel yöntemler doğaları gereği her iterasyonda işlem adımları açısından farklılıklar gösterir. Bir algoritma bir iterasyon adımında amaç fonksiyonunu bir kez çağırırken diğeri birden fazla çağırabilir. Literatürdeki çalışmalar incelendiğinde genelde sonlandırma kriteri olarak iterasyon sayısının kullanıldığı görülmektedir. Bu çalışmada sonlandırma kriteri olarak maksimum amaç fonksiyonu değerlendirme sayısının kullanılması algoritmaların daha adil kıyaslanmasını sağlamış, *maxFEs*=500**d* ve *maxFEs*=1000**d* için algoritmaların başarı sıralamasında değişimler olduğu görülmüştür. Bu ise sonlandırma kriterinin optimizasyon sürecinin sonuçları üzerinde önemli bir etkisi olduğunu göstermektedir. Bu parametrenin değeri çözümlerin kalitesini ve en iyi çözümü bulan algoritmaları değiştirebilmektedir.

Literatürdeki çalışmalarda önerilen algoritmaların az sayıda rakiple kıyaslandığı ve bu algoritmaların seçim sebebi ile ilgili herhangi bir bilgi verilmediği görülmüştür. Ayrıca her yıl geliştirilen onlarca algoritmanın seçilen 5-10 görüntüden oluşan kısıtlı bir veri kümesi üzerindeki başarısı genel bir başarım olarak değerlendirilememektedir. Bu nedenle bu çalışmada segmentasyon literatüründe en çok kullanılmış, 11 MSA'nın görüntü segmentasyondaki başarımları değerlendirilmiştir. Deneyler Berkeley segmentasyon veri setinin test klasöründeki 100 görüntü üzerinde gerçekleştirilmiştir. Böylece algoritmaların gerçek segmentasyon performanslarının değerlendirilmesi sağlanmıştır. Çalışmaların çoğunda düşük sayıda eşik seviyesi (2, 3, 4, 5) kullanılmıştır. Dolayısıyla algoritmanın orta ya da yüksek eşik seviye sayısı kullanıldığında nasıl bir performans göstereceği bilinmemektedir. Bu çalışmada 100

görüntünün *m*=2, 3, 4, 5, 6, 7, 8 seviyeli segmentasyonu gerçekleştirilmiştir.

Elde edilen sonuçlarda en başarılı iki algoritmanın MFO, PSO olduğu görülmüştür. MFO'nun keşiften sömürüye geçerek erken aşamada hızlı bir yakınsamaya olanak sağlaması yöntemin verimliliğini artırmaktadır [30]. PSO ise global optimumları bulmada daha yüksek verimliliğe, karmaşık problemleri çözmek için doğru matematiksel modeller oluşturma yeteneğine ve hızlı yakınsama hızına sahip bir algoritmadır [31]. Yeterli şans verildiğinde ABC'nin üçüncü başarılı, daha hızlı çözüm elde edilmek istendiğinde SSA'nın üçüncü en iyi algoritma olduğu görülmüştür. 11 algoritma içerisinde en başarısız olanlar DE, SOS ve SCA olarak belirlenmiştir. Bu açıdan bu çalışma yapılacak görüntü segmentasyonu çalışmalarında algoritma seçimi konusunda araştırmacılara ışık tutacak niteliktedir.

Çıkar çatışması

Yazarlar çıkar çatışması olmadığını beyan etmektedir.

Benzerlik oranı (iThenticate): %10

Kaynaklar

[1] J. N. Kapur, P. K Sahoo and A. K. Wong, A new method for gray-level picture thresholding using the entropy of the histogram. Computer vision, graphics and image processing, 29 (3), 273-285, 1985. https://doi.org/10.1016/0734-189X(85)90125-2.

[2] N. Otsu, A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9, 62-66, 1979 https://doi.org/10.1109/TSMC.1979.4310076.

[3] M. Sezgin and B.L. Sankur, Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic imaging, 13(1), 146-168, 2004. https://doi.org/10.1117/1.1631315.

[4] P. Y. Yin and L. H. Chen. A fast iterative scheme for multilevel thresholding methods. Signal processing, 60(3), 305-313, 1997. https://doi.org/10.1016/S0165-1684(97)00080-7.

[5] P. Y. Yin, Multilevel minimum cross entropy threshold selection based on particle swarm optimization. Applied mathematics and computation, 184(2), 503-513, 2007. https://doi.org/10.1016/j.amc.2006.06.057.

[6] M. H. Horng, A multilevel image thresholding using the honey bee mating optimization. Applied Mathematics and Computation, 215(9), 3302-3310, 2010. https://doi.org/10.1016/j.amc.2009.10.018.

[7] M. H. Horng and R. J. Liou, Multilevel minimum cross entropy threshold selection based on the firefly algorithm. Expert Systems with Applications, 38(12), 14805-14811, 2011. https://doi.org/10.1016/j.eswa.2011.05.069.

[8] B. Akay, A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. Applied Soft Computing, 13(6), 3066-3091, 2013. https://doi.org/10.1016/j.asoc.2012.03.072.

[9] A. K. Bhandari, V. K. Singh, A. Kumar, and G. K. Singh, Cuckoo search algorithm and wind driven optimization-based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. Expert Systems with Applications, 41(7), 3538-3560, 2014. https://doi.org/10.1016/j.eswa.2013.10.059.

[10] S. Pare, A. Kumar, V. Bajaj, and G. K. Singh, An efficient method for multilevel color image thresholding using cuckoo search algorithm based on minimum cross entropy. Applied Soft Computing, 61, 570-592, 2017. https://doi.org/10.1016/j.asoc.2017.08.039.

[11] A. K. M. Khairuzzaman, and S. Chaudhury, Multilevel thresholding using grey wolf optimizer for image segmentation. Expert Systems with Applications, 86, 64-76, 2017. https://doi.org/10.1016/j.eswa.2017.04.029.

[12] M. Abd El Aziz, A. A. Ewees, and A. E. Hassanien, Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Systems with Applications, 83, 242-256, 2017. https://doi.org/10.1016/j.eswa.2017.04.023.

[13] B. Küçükuğurlu, and E. Gedikli, Symbiotic organisms search algorithm for multilevel thresholding of images. Expert Systems with Applications, 147, 113210, 2020. https://doi.org/10.1016/j.eswa.2020.113210.

[14] P. D. Sathya, R. Kalyani, and V. P. Sakthivel, Color image segmentation using Kapur, Otsu and minimum cross entropy functions based on exchange market algorithm. Expert Systems with Applications, 172, 114636, 2021. https://doi.org/10.1016/j.eswa.2021.114636.

[15] E. H. Houssein, B. E. D. Helmy, D. Oliva, A. A. Elngar, and H. Shaban, A novel black widow optimization algorithm for multilevel thresholding image segmentation. Expert Systems with Applications, 167, 114159, 2021. https://doi.org/10.1016/j.e.swa.2020.114159.

[16] A. K. Bhandari, A. Kumar, and G. K. Singh, Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur's, Otsu and Tsallis functions. Expert systems with applications, 42(3), 1573-1601, 2015. https://doi.org/10.1016/j.eswa.2014.09.049.

[17] L. He, and S. Huang, Modified firefly algorithm based multilevel thresholding for color image segmentation. Neurocomputing, 240, 152-174, 2017. https://doi.org/10.1016/j.neucom.2017.02.040.

[18] J. Anitha, S. I. A. Pandian, and S. A. Agnes, An efficient multilevel color image thresholding based on modified whale optimization algorithm. Expert Systems with Applications, 178, 115003, 2021. https://doi.org/10.1016/j.eswa.2021.115003.

[19] T. Rahkar Farshi, and A. K. Ardabili, A hybrid firefly and particle swarm optimization algorithm applied to multilevel image thresholding. Multimedia Systems, 27(1), 125-142, 2021. https://doi.org/10.1007/s00530-020-00716-y.

[20] M. Abd Elaziz, D. Mohammadi, D. Oliva, and K. Salimifard, Quantum marine predators algorithm for addressing multilevel image segmentation. Applied Soft Computing, 110, 107598, 2021. https://doi.org/10.1016/j.asoc.2021.107598.

[21] L. Qiao, K. Liu, Y. Xue, W. Tang, and T. Salehnia, A multi-level thresholding image segmentation method using hybrid Arithmetic Optimization and Harris Hawks Optimizer algorithms. Expert Systems with Applications, 241, 122316, 2024. https://doi.org/10.1016/j.eswa.2023.122316.

[22] F. S. Gharehchopogh, and T. Ibrikci, An improved African vultures optimization algorithm using different fitness functions for multi-level thresholding image segmentation. Multimedia Tools and Applications, 83(6), 16929-16975, 2024. https://doi.org/10.1007/s11042-023-16300-1.

[23] L. Abualigah, N. K. Al-Okbi, E. M. Awwad, M. Sharaf, and M. S. Daoud, Boosted Aquila Arithmetic Optimization Algorithm for multi-level thresholding image segmentation. Evolving Systems, 15, 1399-1426, 1-28, 2024. https://doi.org/10.1007/s12530-023-09566-1.

[24] J. Wang, J. Bei, H. Song, H. Zhang, and P. Zhang, A whale optimization algorithm with combined mutation and removing similarity for global optimization and multilevel thresholding image segmentation. Applied Soft Computing, 137, 110130, 2023. https://doi.org/10.1016/j.asoc.2023.110130.

[25] D. Karaboga, and B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. InInternational Fuzzy Systems Association World Congress, pp. 789-798, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-72950-1_77.

[26] X. S. Yang, and S. Deb, Cuckoo search via Lévy flights. In 2009 World Congress On Nature & Biologically Inspired Computing (NaBIC), pp. 210-214, 2009. https://doi.org/10.1109/NABIC.2009.5393690.

[27] R. Storn, and K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11(4), 341-359, 1997. https://doi.org/10.1023/A:1008202821328.

[28] S. Mirjalili, S. M. Mirjalili, and A. Lewis, Grey wolf optimizer. Advances in engineering software, 69, 46-61, 2014. https://doi.org/10.1016/j.advengsoft.2013.12.007.

[29] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, Harris hawks optimization: Algorithm and applications. Future generation computer systems, 97, 849-872, 2019. https://doi.org/10.1016/j.future.2019.02.028.

[30] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-based systems, 89, 228-249, 2015. https://doi.org/10.1016/j.knosys.2015.07.006.

[31] R. Poli, J. Kennedy, and T. Blackwell, Particle swarm optimization. Swarm intelligence, 1(1), 33-57, 2007. https://doi.org/10.1007/s11721-007-0002-0.

[32] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems. Knowledge-Based Systems, 96, 120-133, 2016. https://doi.org/10.1016/j.knosys.2015.12.022.

[33] M. Y. Cheng, and D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm. Computers & Structures, 139, 98-112, 2014. https://doi.org/10.1016/j.compstruc.2014.03.007.

[34] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114, 163-191, 2017. https://doi.org/10.1016/j.advengsoft.2017.07.002.

[35] S. Mirjalili, and A. Lewis, The whale optimization algorithm. Advances in engineering software, 95, 51-67, 2016. https://doi.org/10.1016/j.advengsoft.2016.01.008.

[36] H. T. Kahraman, S. Aras, and E. Gedikli, Fitness-distance balance (FDB): A new selection method for metaheuristic search algorithms. Knowledge-Based Systems, 190, 105169, 2020. https://doi.org/10.1016/j.knosys.2019.105169.

[37] The Berkeley Segmentation Dataset and Benchmark, https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/