Pamukkale Univ Muh Bilim Derg, 28(2), 313-323, 2022

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Evrimsel algoritmalar için yeni bir meta-iyileştirici: bipolar eşleşme eğilimi

A novel meta-optimizer for evolutionary algorithms: bipolar mating tendency

Mashar Cenk GENCAL[1], Mustafa ORAL[2]*

[1]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Ardahan Üniversitesi, Ardahan, Türkiye.
masharcenkgencal@ardahan.edu.tr

[2]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Çukurova Üniversitesi, Adana, Türkiye.
moral@eu.edu.tr

Geliş Tarihi/Received: 02.05.2021 Düzeltme Tarihi/Revision: 05.08.2021 doi: 10.5505/pajes.2021.29165
Kabul Tarihi/Accepted: 23.08.2021 Araştırma Makalesi/Research Article

Öz

Evrimsel Algoritmalar üzerine yapılan son araştırmalar, bu algoritmaların performansının, genellikle, uygun parametre yapılandırmalarının seçilmesine bağlı olduğunu göstermektedir. Araştırmacılar, ya literatürdeki benzer araştırma alanlarına bakarak ya da Izgara Arama'da (Grid Search) olduğu gibi uygun parametreleri tek tek deneyerek bu parametre yapılandırmalarını bulmaya çalışmışlardır. Ancak, parametrenin tek tek aranması zahmetli ve zaman alıcıdır; bu nedenle, meta-optimizasyon teknikleri, bir algoritmanın parametrelerini ayarlamak için yaygın olarak kullanılan yöntemler haline gelmiştir. Meta-optimizasyon teknikleri, yaygın olan iki biçimde sınıflandırılabilirler: çevrimdışı (algoritma başlamadan önce bir algoritmanın parametrelerini ayarlamak) ve çevrimiçi (çalışma sırasında parametreleri ayarlamak). Bu makalede, bir Genetik Algoritmanın (GA) seçim yöntemi olarak, Bipolar Eşleşme Eğilimi (BMT) algoritması seçilmiştir. Oluşan yeni algoritma, GA-BMT olarak adlandırılmış ve çevrimiçi meta-iyileştirici olarak ilk kez kullanılmıştır. Ayrıca makale, 17 test fonksiyonu için Standart GA'nın (SGA) en iyi parametre ayarlarının bulunmasında, iki arama algoritması (Izgara Arama, Kabadan İnceye Arama) ve üç meta-optimizasyon yöntemini (SGA, Parçacık Sürü Optimizasyonu, GA-BMT) kullanmakta ve sonuçlarını karşılaştıran bir çalışma sunmaktadır. Bununla birlikte, elde edilen sonuçları anlamlandırmak için istatistiksel testler, Friedman ve Wilcoxon İşaretli Sıralar, kullanılmıştır. Elde edilen tüm sonuçlar incelendiğinde, GA-BMT'nin makul bir başarı sunduğu aşikârdır.

Anahtar kelimeler: Genetik algoritmalar, Izgara arama, Kabadan inceye arama, Meta-Optimizasyonu, Parçacık sürü optimizasyonu.

Abstract

Recent studies show that the performance of Evolutionary Algorithms often depends on choosing appropriate parameter configurations. Thus, researchers have generally tuned these parameters either looking at the similar research areas in the literature or manually, e.g. Grid Search. However, searching the parameter manually is laborious and time-consuming; therefore, meta-optimization techniques have become commonly used methods to adjust parameters of an algorithm. These techniques can be classified in two widespread manners: off-line, tuning parameters of an algorithm before the algorithm initiates, and on-line, tuning the parameters while it is working. In this paper, Bipolar Matching Tendency (BMT) algorithm has been chosen as the selection method of a Genetic Algorithm (GA). The new obtained algorithm is named GA-BMT and has been used for the first time as an online meta-optimizer. In addition, the paper utilizes two search algorithms (Grid Search, Coarse to Fine Search) with three meta-optimization methods (Standard GA, Particle Swarm Optimization, GA-BMT) to investigate the best parameter settings of the Standard GA for 17 test functions, and offers a comparative work by comparing their results. Furthermore, non-parametric statistical tests, Friedman and Wilcoxon Signed Rank, were performed to demonstrate the significance of the results. Based on the all results that achieved, GA-BMT presents a reasonable achievement.

Keywords: Genetic algorithms, Grid search, Coarse to fine search, Meta-Optimization, Particle swarm optimization.

1 Giriş

Evrimsel Algoritmalar (EA'lar), doğadaki canlıların davranışlarından ilham alarak oluşturulmuş meta sezgisel algoritmalardır. EA'ların ilham aldıkları bu davranışlar taklit edilerek, optimizasyon problemlerinde kullanılmış ve başarılı sonuçlar sunmuşlardır. EA'ların elde ettiği bu başarılar yeni fikirlerin önünü açmış ve EA'lar literatürde yaygın kullanılan algoritmalar haline gelmiştir.

Bir EA'nın nasıl davrandığını etkileyen en önemli faktörlerden biri, uygun parametrelerin seçilmesidir: çaprazlama oranı, mutasyon oranı, turnuva büyüklüğü, popülasyon büyüklüğü vb. Birçok araştırmacı genellikle bu parametreleri "varsayılan parametre ayarları" [1] (çaprazlama oranı=0.7, turnuva büyüklüğü=3, mutasyon oranı=0.05) ile ayarlamışlardır. Ancak, bu yöntem tüm problem tipleri için istenen çözümleri sunmayabilir. Örneğin, seçilen bir parametre konfigürasyonu, problem durumlarından birinde optimum değeri bulabilir. Fakat bu parametre yapılandırması diğer problem örnekleri için iyi sonuçlar vermeyebilir; erken yakınsamaya veya yerel minimum değere sıkışıp kalmaya neden olabilir. Yaygın olarak kullanılan iki yaklaşım bu sorunlar için bir çare olmuştur: hiyerarşik arama teknikleri ve meta-optimizasyonu.

Hiyerarşik arama teknikleri, bir algoritmanın en iyi parametre ayarlarını bulmak için, rastgele ya da önceden belirlenmiş olan parametre setlerini kullanırlar. Kullanılan parametre setlerinin sayısı arttıkça, optimize edilmek istenilen algoritma için daha iyi sonuçlar elde edilebilir. Ancak, parametre setlerinin sayısının artması hesaplama bütçesini de doğru orantılı bir şekilde artıracaktır. Bu gibi durumlarda, hesaplama bütçesini azaltmak için meta-optimizasyonu tekniklerinden yararlanılmaktadır.

*Yazışılan yazar/Corresponding author

Optimizasyon problemlerinde kullanılan meta-optimizasyonu, bir algoritmanın parametrelerini ayarlayarak algoritmanın performansının iyileştirilmesi anlamına gelmektedir. Meta-optimizasyonu yöntemleri, algoritmanın verimliliğini arttırmanın yanı sıra, algoritmanın farklı problem türlerinde nasıl davrandığını anlamamıza da yardımcı olur. Bu yöntemleri iki ana grupta sınıflandırılabiliriz: çevrimdışı ve çevrimiçi.

Çevrimdışı yöntemler, algoritma çalışmaya başlamadan önce parametre ayarlarını tanımlar ve bir eğitim seti örneği üzerinde bazı parametre yapılandırmalarını analiz ederler. Ardından seçilen yapılandırma, üzerinde çalışılan tüm örneklere uygulanır. Seçilen örnekler, eğitim setindeki diğer örneklerle eşit yapıya sahipse, çevrimdışı yöntemlerin iyi çalışması beklenebilir. Ancak, örneklerin sınıfı heterojen olduğunda bu yöntemler işe yaramayabilir. Çünkü böylesi bir durumda, her bir örnek sınıfı için uygun parametre ayarlarının seçilmesi, büyük hesaplama süresine neden olacaktır.

Çevrimiçi yöntemler ise, algoritma problem durumlarından birini çözerken, aynı anda geri bildirimler toplarlar ve en iyi parametre ayarlarını bulmaya çalışırlar. Bu yöntemler, algoritma çalışırken parametre ayarlarını bulmaya çalıştığından, hesaplama süresinden tasarruf sağlamaktadır. Bu avantajlarının yanı sıra, yaptığımız literatür çalışması (bkz. 2. Bölüm) göstermiştir ki, bir algoritmanın parametrelerinin ayarlanması, algoritmanın performansını olumlu yönde etkilemektedir. Ayrıca, çevrimiçi yöntemler, çevrimdışı yöntemlere göre daha başarılı sonuçlar sunmaktadırlar. Yaptığımız bu çalışmadan elde ettiğimiz sonuçlar, makalemizde çevrimiçi bir yöntem seçmemizin nedeni olmuştur.

Bu sebeple, Standart bir Genetik Algoritma'nın (SGA) en iyi parametre yapılandırmasını bulmak için Bipolar Eşleşme Eğilimi (Bipolar Mating Tendency, BMT) algoritması, GA'nın seçim algoritması olarak alınmış ve oluşan yeni yöntem GA-BMT olarak adlandırılmıştır. Çevrim-içi bir yöntem (algoritma) olan GA-BMT, bu çalışma ile ilk kez bir meta-iyileştirici olarak kullanılmıştır.

Bu makalede, SGA'nın en önemli parametrelerinden olan; çaprazlama oranı, turnuva büyüklüğü ve mutasyon oranı parametrelerinin, en ideal değerlerini farklı durumlar altında bulmak amaçlanmıştır. Bu amacı gerçekleştirmek için, literatürde yaygın olarak kullanılan, 17 farklı test fonksiyonu kullanılmış ve her bir fonksiyon için bulunan en iyi sonuçlar ve bu sonuçlara ait parametre ayarları paylaşılmıştır. İlgili parametrelerin en iyi değerlerini bulmak amacıyla, BMT, SGA ve Parçacık Sürü Optimizasyonu meta-iyileştirici algoritmaları ile Izgara ve Kabadan İnceye arama algoritmaları kullanılmış ve elde edilen tüm sonuçlar karşılaştırmalı olarak sunulmuştur.

Makale şu şekilde düzenlenmiştir: 2. Bölümde, literatürde yer alan çalışmalar kısa incelemelerle sunulmuştur. 3. Bölümde, makalede kullanılan yöntemlerle ilgili bilgiler verilmiştir. 4. Bölüm yapılan testler hakkında bilgiler sunarken, 5. Bölümde bu testlerden alınan sonuçlar tartışılmıştır. Makale, 6. Bölüm ile sonlandırılmış ve gelecekteki fikirler ile ilgili bilgiler de sunulmuştur.

2 Literatür incelemesi

Meta-optimizasyonu alanında yapılan ilk çalışma 1978 yılındadır [2]. Mercer ve Sampson, bir GA kullanarak, başka bir GA'nın çaprazlama ve mutasyon oranları için en iyi parametre ayarlarını bulmaya çalışmışlardır. Ancak, bu yıllarda modern bir bilgisayar olmadığından yapılan çalışma sınırlı kalmış ve bu nedenle deneyde sadece bir test yapılabilmiştir. Birkaç yıl sonra, GA'nın parametrelerden çaprazlama oranı, mutasyon oranı, popülasyon büyüklüğü, nesil farkı, ölçeklendirme penceresi ve seçim stratejisi, bir çevrimdışı ve bir çevrimiçi olmak üzere iki farklı yöntem kullanılarak incelenmiştir [3]. Çevrimiçi yöntemin performansının, çevrimdışı yöntemin performansına göre daha iyi olduğu belirtilmiştir. Freisleben ve Hartfelder [4], bir GA kullanarak başka bir GA'nın popülasyon büyüklüğünü ayarlamaya çalışmıştır. Bu ayarlamayı yaparken, [3]'te olduğu gibi çaprazlama oranı, mutasyon oranı vb. gibi parametreleri kullanarak en uygun popülasyon büyüklüğünü test etmiştir. Elde ettiği sonuçlar, algoritma çalışıyorken parametrelerin düzenlenmesinin önemli olduğunu göstermiştir.

Öte yandan Back [5], "efendi-köle yaklaşımından" faydalandığı meta algoritma ile GA'nın parametrelerini ayarlamak için alternatif bir yol sunmuştur. Bu yaklaşımda, "efendi" algoritması, farklı işlemciler üzerinde paralel olarak çalışan işçilere, makaledeki adıyla "kölelere" (GA'lar), sahiptir. Her bir köle, bireylerin farklı parametre tanımlarını taşır. Köleler, efendilerinin her nesli için, efendileri olmaya aday yeni bireyleri değerlendirirler. Bireylerin uygunluk değerlerine bakarak, efendileri olacak yeni bireylere karar verirler.

Bahsedilen diğer yöntemlerden farklı olarak, Hinterdings [6], normal dağılmış rastgele bir değişkenden elde edilen Gauss gürültüsü (noise) kullanılarak, sadece mutasyon parametresini ayarlamıştır. Hinterdings, algoritma çalışıyorken parametrelerini ayarlamanın, algoritma için daha verimli olduğunu iddia etmiştir. Başka bir makalede, GA ve Benzetilmiş Tavlama (Simulated Annealing, SA) yöntemleri, çoklu uç değerli problemlerinde (multi-peak problems) kullanılabilecek en iyi GA parametre ayarlarını keşfetmek için uygulanmıştır [7]. Her iki yöntem de, varsayılan parametre değerlerinin kullanıldığı durumlardan daha iyi sonuçlar sunmuşlardır. Böylelikle, ayarlanmış parametrelerden elde edilecek sonuçların, varsayılan parametre ayarlarından elde edilen sonuçlara göre daha etkili olduğu çıkarımında bulunmuştur.

Cortez ve diğ. [8] zaman serisi tahmin problemlerinde (time series forecasting problems) parametre tahmini için bir meta-GA önermiştir. Hong ve diğ. [9] çift ağlı bir model (çevrim içi ve çevrim dışı) ile model öngörücü kontrolünü (Model Predictive Control, MPC) birleştiren bir çalışma sunmuşlardır. Çok Katmanlı Algılayıcı (Multilayer Perceptron, MLP) çevrimdışı modeli oluşturmak için kullanılırken, Yapay Sinir Ağları (Artificial Neural Networks, ANNs) çevrim içi model için kullanılmıştır. Çevrim dışı modelin tahmin doğruluğunu artırmak, en ideal ağ yapısını bulmak, parametrelerini ayarlamak ve eğitim algoritmasını seçmek için bir GA'dan yararlanılmıştır. Son yıllarda yapılan başka bir çalışmada ise, bir karar ağacı öğrenicisinin (decision tree learner) parametrelerini optimize etmek için Standart Genetik Algoritma (SGA) kullanılmıştır [10]. Meta-SGA'nın performansı, Izgara Arama'nın performansı ile karşılaştırılmış ve SGA'nın, Izgara Arama'ya göre baskın sonuçlar sunduğu iddia edilmiştir. Bir diğer çalışmada ise, Çoklu-sadakat Stratejisi (Multi-fidelity Strategy) adında yeni bir yaklaşımla, yaygın olarak kullanılan; Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO), Diferansiyel Evrim (Differential Evolution, DE) ve Bozkurt (Grey-wolf) algoritması gibi meta tekniklerin parametre optimizasyonu hızlandırılmak istenmiştir [11].

Bununla birlikte Lapa [12], Çok Amaçlı Popülasyon Tabanlı (Multi Objective Population Based) algoritmasının parametrelerini ayarlamak için yeni bir yaklaşım sunmuş ve

sunduğu yaklaşımın parametrelerini bir GA kullanarak optimize etmiştir. Magliani ve diğ. [13] bağlam tabanlı görüntü alma (Context-Based Image Retrieval, CBIR) alanında, kNN (k-Nearest Neighbor) grafiklerini kullanarak, difüzyon parametrelerinin en uygun yapılandırmasını aramak için bir GA kullanmıştır. Meta-iyileştirici olarak kullandıkları GA'dan aldıkları sonuçları test etmek için, bilgisayarlı görme alanında yaygın olarak kullanılan; Oxford5k, Paris6k ve Flickr1M veri setlerini uygulamışlardır. Ayrıca, yaptıkları son çalışma ile [14], [13]'teki çalışmalarını detaylı bir şekilde analiz etmişlerdir. GA'lar ayrıca, zamanlama problemlerinde (scheduling problems) kullanılan, PSO'nun parametre ayarlarını seçmek için de kullanılmıştır [15]. Ayarlanmış bir PSO'nun, standart bir PSO'ya göre daha başarılı olduğu gözlemlenmiştir.

GA'ların haricinde meta-optimizasyonu olarak kullanılan bir diğer EA da PSO'dur. PSO'nun meta-optimizasyon literatüründeki ilk kullanımı OPSO (Optimized Particle Swarm Optimization) adı verilen ve PSO'nun etkinliğini artırmak için uygulanan algoritmadır [16]. Bu metot, küçük organik moleküllerin kan-beyin bariyeri geçirgenliğini öngörmek için, uygun bir model tasarlamak amacıyla, sinir ağı kullanılarak gerçekleştirilmiştir. Bundan sonra, Pedersen ve Chipperfield [17], PSO'yu bir meta-iyileştirici (optimizer) olarak kullanmış ve algoritmanın kendisini basitleştirmeye çalışmışlardır. Basitleştirilmiş yöntem, yapay sinir ağı kanseri ve kart (ANN cancer and card) problemleri üzerinde uygulanmış ve standart bir PSO'dan daha iyi sonuçlar elde edilmiştir. Ayrıca, yapay sinir ağı kullanarak DE yönteminin parametrelerini ayarlamak için de yeni bir çevrimdışı yaklaşım sunulmuştur [18]. Bu yöntem, parametreleri önceden tanımlanmış olan standart DE ile karşılaştırıldığında herhangi bir avantaj sunmamıştır. Bu nedenle, DE üzerinde daha iyi bir performans elde etmek için, çevrimiçi meta-optimizasyon tekniklerinden yararlanılmasını önermişlerdir. Mason ve diğ. [19] yaygın olarak kullanılan hız güncelleme denklemlerinin (The Attractive Repulsive PSO, The Dissipative PSO, the Adaptive Velocity PSO vb.) parametrelerinin PSO üzerindeki etkisini görmek amacıyla, bu denklemleri optimize eden bir PSO'yu meta-iyileştirici olarak önermiş ve havza yönetimi (watershed management) problemleri aracılığıyla elde edilen sonuçlar test edilmiştir. Meta-iyileştiricinin kullanılmasının nedeni, en iyi parametre yapılandırmalarının hız güncelleme denklemlerinin her birinde kullanılarak, adil bir karşılaştırma yapıldığından emin olunmak istenmesidir.

Jafaari ve diğ. [20], Uyarlamalı Ağ Tabanlı Bulanık Çıkarım Sisteminin (Adaptive Neuro-Fuzzy Inference System, ANFIS) parametrelerini hem Bozkurt algoritması, hem de Biyocoğrafya Temelli Optimizasyon (Biogeography-based Optimization, BBO) algoritmasını kullanarak iyileştirmişlerdir. Öncelikle, eğitim veri setlerini ve C-means kümeleme algoritmasını kullanarak başlangıç bulanık çıkarım sistemini oluşturmuşlardır. Daha sonra, Bozkurt ve BBO algoritmalarını kullanarak ANFIS'in parametre değerlerinin çeşitli olasılıklarını test etmişlerdir. Meta-iyileştiriciler (Bozkurt ve BBO) ANFIS'e adapte edilerek melez iki yeni model oluşturulmuş ve bu iki melez model, heyelan dağılımı ve heyelan duyarlılığı konularında test edilmiştir. Ayrıca Chen ve diğ. [21] Balina Optimizasyonu Algoritması (Whale Optimization Algorithms) ile Bozkurt algoritmasını kullanarak ANFIS'in tahmin yeteneğini artırmayı amaçlamışlardır.

Birattari ve diğ. [22] istatistiki bir metoda dayanan F-Race algoritmasını tanıtmışlardır. Tüm olası parametre ayarlarını (adaylar) test etmek yerine, F-Race gelecek vaat eden sonuçlar sunan adaylara odaklanır. Bu çalışmada, Karınca Koloni Optimizasyonu (Ant Colony Optimization) algoritmasının parametrelerini yapılandırmak için gezgin satıcı problemi (traveling salesman problem) kullanılmıştır. Birkaç yıl sonra Balaprakash [23], F-Race algoritmasını yinelemeyi önermiştir. F-Race ve yinelenen (iterated) F-Race arasındaki tek fark, yinelenen F-Race'in üç ana adımdan oluşmasıdır: aday çözümlerin oluşturulması, adayların değerlendirilmesi ve en iyi aday çözümün güncellenmesi. [24]'te F-Race ve yinelenen F-Race detaylı bir şekilde açıklanmıştır. Bununla birlikte, son yıllarda yapılmış olan bir çalışmada, F-Race ve yinelenen F-Race algoritmaları, GA'nın çaprazlama oranı ve mutasyon oranı parametrelerini ayarlamak için kullanılmıştır [25]. Çalışmanın sonunda, GA'nın parametrelerinin ayarlanmasının zor bir iş olduğu belirtilmiş ve standart parametre ayarlarının daha iyi sonuçlar sunduğu söylenmiştir.

François ve Lavergne [26], EA'lar için en iyi parametre ayarlarını bulmak amacıyla, istatistiksel çevrimdışı bir yöntem önermişlerdir. Yöntem iki ana bölümden oluşmaktadır: problem seviyesi modeli ve global model. Problem seviyesi modeli bölümünde, EA'ların davranışlarını belirlemek için belirli bir problem üzerinde algoritma test edilir. Global modelde ise, EA'ların problem setleri üzerindeki davranışlarını tanımlamak için bir dizi problem kullanılır. Elde edilen sonuçlara göre, EA'lar için uygun parametre ayarları seçilir ve bu parametre ayarları, en uygun sonuçları almak için başka problemlere de uygulanır. Bir diğer yeni yöntem olan CRE'de (Calibration and Relevance Estimation), ajan tabanlı uygulamada (agent-based application) yüksek performans elde etmek için, EA'lar meta-iyileştirici olarak uygulanmıştır [27]. Çalışmada, ajanların evrimsel davranışlarının kendi başlarına ilerleme kaydedemeyeceği, bunun yerine, ayarlanmış parametrelerin kullanımının öneminden bahsedilmektedir.

Diaz ve Laguna [28], iki mekanizmadan oluşan CALIBRA tekniğini sundular: deneysel tasarımlar ve yerel arama. Deneysel tasarımlar, arama alanının gelecek vaat eden bölgesini keşfetmeye yardımcı olur. Böylece, ilk aşama için daha iyi bir başlangıç noktası sağlar. Yerel arama aşamasının amacı, bulunan alanda yerel minimum değere ulaşmaktır. Bunu yapmak için yöntem, parametre ayarlarının aralığını azaltarak gelecek vaat eden bölgeye odaklanır. Ancak, bir algoritma, beş parametreden daha fazla parametreye sahipse bu teknik kullanılamaz. Bu sebeple, bir yıl sonra, yerel bir arama yöntemi olan ParamILS algoritması tanıtılmıştır [29]. Bu algoritmanın CALIBRA'dan farkı, parametre sayısında sınırlama olmaması ve performansının daha iyi olmasıdır.

Sıralı Parametre Optimizasyonu (Sequential Parameter Optimization, SPO), bilinen istatistiksel yaklaşımlardan biridir [30]. SPO'nun verimliliğini analiz etmek için, algoritma üç farklı durum altında test edilmiştir: GA'ların kimyasını keşfetmek, PSO ve EA'larda parametre optimizasyonu. SPO, bu üç durumda da başarılı olmasına rağmen, sadece ondalık ya da tam sayı değerlerini kullanması gibi dezavantajları nedeniyle algoritmanın iyileştirilmesine ihtiyaç duyulmuştur. Bu nedenle, SPO'dan daha iyi performans sunan SPO+ (SPO'nun geliştirilmiş hali) önerilmiştir [31]. SPO ve SPO+'nın bir alternatifi olarak, "SMAC" (Sequential Model Based Algorithm Configuration) ve "ROAR" olarak adlandırılan (Randomized Online Aggressive Racing) yöntemler de sunulmuştur [32]. Bu yöntemlerin amacı, diğer sıralı model tabanlı optimizasyon tekniklerinin takıldığı, sayısal parametrelerde kısıtlama ve bir örnek için hedef algoritmanın performansını optimize etmek gibi problemleri çözmektir. Bu yeni yöntemler (ROAR ve

SMAC), bilinen diğer tekniklerle (örneğin ParamILS ve SPO gibi) karşılaştırılmıştır. Yerel arama (local search) ve ağaç arama (tree search) çözücülerindeki optimizasyon becerileri, SAT problemlerinde (the propositional satisfiability problems) kullanılarak test edilmiş ve yeni yöntemler, karşılaştırılan yöntemlere göre daha iyi sonuçlar sunmuşlardır.

Branke ve Elomari [33] EA'ların parametrelerini (popülasyon büyüklüğü ve yavru popülasyon büyüklüğü) ayarlamak için yeni bir yöntem olan Çevrimdışı Esnek Bütçe (Flexible Budget Offline) yöntemini önermişlerdir. Bu yöntemde, ilk olarak, EA'ların olası tüm parametre ayarları kendi içerisinde uygulanır. Alınan sonuçlara göre, her bir tekrarlama için, parametre ayarları 1 ile n (n, test edilecek parametre ayarlarının sayısı olmak üzere) arasında derecelendirilir (1 en uygun parametre ayarı, n ise uygun olmayan parametre ayarı). Böylelikle algoritma, ebeveyn seçimi için düşük dereceye sahip parametre yapılandırmalarını seçer. Geleneksel teknik olan Sabit Bütçe Yöntemindeyse (Fixed Budget Method), istenilen tekrarlama sayısına ulaştıktan sonra, elde edilen sonuçlara bakılarak parametre ayarları seçilmektedir. Çevrimdışı Esnek Bütçe yöntemi bir tekrarlamayla çalıştırılmasına rağmen, Sabit Bütçe Yöntemi ile karşılaştırıldığında, daha iyi sonuçlar verdiği belirtilmiştir.

Yukarıda açıklanan çalışmalara ek olarak, [34] ve [35]'te, çevrimdışı yöntemler ile çevrimiçi yöntemleri karşılaştırmışlardır. Bu araştırmalardan elde edilen sonuçlara göre, çevrimiçi teknikler, global değerlere çevrimdışı tekniklerden daha iyi yaklaşım sunmaktadırlar. Bunlara ek olarak, Meyer [36], meta-optimizasyonuna ilgi duyanlar için geniş bir inceleme sunarken, Bendre [37] ve arkadaşları bilgisayarlı görme alanında kullanılan yöntemlerle ilgili bir inceleme sunmuşlardır.

3 Kullanılan metotlar

3.1 Izgara arama

Izgara Arama (Grid Search), bir algoritmanın en ideal parametre ayarlarını keşfetmek için, bu algoritmanın ızgara (grid) içerisindeki tüm parametre ayarlarını test eden geleneksel bir tekniktir [10]. Sadece birkaç parametreyi ayarlamak gerektiğinde bu yöntem yeterlidir. Ancak, ayarlanması gereken parametre sayısının artması, büyük bir hesaplama bütçesine neden olacaktır. Büyük hesaplama maliyetleri istenmediğinden, araştırmacılar, bir algoritmanın en ideal parametre ayarlarını bulmak için genellikle meta-optimizasyon tekniklerini kullanmaktadırlar.

Popülasyon büyüklüğü, turnuva büyüklüğü, çaprazlama olasılığı ve mutasyon oranı, algoritmanın verimliliğini doğrudan etkilediği için, bu parametreler SGA'nın en önemli parametreleridir. Bu nedenle, belirtilen parametreler için uygun değerleri seçmek gerekir. SGA'nın parametre konfigürasyonu, Izgara Arama kullanılarak, şu limitlere göre yapılmıştır: popülasyon büyüklüğü={25, 50, 100, 200}, turnuva büyüklüğü={3, 4, 5, 6}, çaprazlama olasılığı 0.25'ten 1'e 0.05 adımlarla, mutasyon olasılığı 0.005 adım ile 0.025'den 0.1'e, 300 tekrarlama ile. p, popülasyon büyüklüğü; t, turnuva büyüklüğü; ç, çaprazlama olasılığı; m, mutasyon olasılığı olmak üzere, yöntemin sözde-kodu (pseudo-code) aşağıda verilmiştir:

popülasyon = {25, 50, 100, 200};

turnuva = {3, 4, 5, 6};

çaprazlama = {0,25, 0,3, 0,35, … 1};

mutasyon = {0,025, 0,03, 0,035, … 0.1};

p = popülasyon büyüklüğü;

t = turnuva büyüklüğü;

ç = çaprazlama büyüklüğü;

m = mutasyon büyüklüğü;

Tekrarlama sayısı = 300;

```
for i =1 : p
    for j =1 : t
        for k=1 : ç
            for l=1 : m
            -SGA'yı; popülasyon(i), turnuva(j), çaprazlama(k), mutasyon(l) parametreleriyle çalıştır
            -Çıkan sonucu kaydet
            end
        end
    end
end
```

Kaydedilen tüm sonuçları göster

3.2 Kabadan inceye arama

Izgara Arama yöntemi, yalnızca birkaç parametrenin ayarlanması gerektiğinde başarılıdır. Ancak, parametre sayısı arttıkça, hesaplama bütçesi de doğru orantılı olarak artacaktır. Bu durumda, hesaplama bütçesi sorununun üstesinden gelmek için Kabadan İnceye Arama (Coarse to Fine Search, CFS) yöntemi kullanılabilir [38].

Tüm olası parametre düzenlemelerini çalıştırmak yerine, bu yöntem, belirli sayıda rastgele parametre ayarlarını seçer ve seçilen bu parametre ayarlarından problem için en iyi olanları tanımlar. Sonraki aşamada, en iyilerin bulunduğu arama alanındaki bölgeye odaklanır. Bir kez daha, o bölgede daha iyi bir parametre ayarı olup olmadığını keşfetmek için, o bölgeden bazı parametre ayarlarını rastgele seçer. Daha iyi bir parametre yapılandırması keşfedilirse, keşfedilen bu yapılandırma, bulunan en iyi parametre ayarı olarak güncellenir. Yöntemin sözde kodu şu şekildedir:

Rastgele m tane parametre konfigürasyonu seç

Uygunluk değerlerini hesapla

En iyi olan parametre ayarlarını belirle

En iyilerin olduğu bölgeyi sınırlandır

Bu bölgede yer alan parametre konfigürasyonlarından n tanesini rastgele seç

Uygunluk değerlerini hesapla

En iyi sonucu göster

3.3 Standart genetik algoritma

GA'lar, 1975 yılında John Holland tarafından literatüre kazandırılmıştır [39]. Holland, Darwin'in Evrim Teorisinden esinlenmiş ve optimizasyon problemleri için yeni bir model oluşturmuştur. Özellikle arama uzayındaki başarısından dolayı [40], onun bu fikri birçok araştırmacıya ilham kaynağı olmuş ve GA'lar, optimizasyon problemlerinde sıklıkla kullanılan algoritmalar haline gelmiştir.

GA'lar, başlangıç bireylerini rastgele tanımlayarak ilk popülasyonu oluştururlar. Optimizasyon probleminde kullanılacak olan uygunluk fonksiyonu yardımıyla, popülasyonda bulunan her bir birey için uygunluk değeri hesaplanır. Bu uygunluk değeri, popülasyondaki bireylerden hangisinin seçileceği hakkında bilgi vereceğinden, seçim aşaması için çok önemlidir. Başlangıç popülasyonunun oluşturulmasından sonra sırasıyla üç temel aşama vardır: seçim, çaprazlama ve mutasyon.

Seçim aşamasının görevi, verilen optimizasyon problemi için en uygun bireyleri seçmek ve seçilen bu bireyleri çaprazlama aşamasına iletmektir. Seçim yöntemleri, genellikle, popülasyon içindeki en iyi bireyleri (en yüksek uygunluk değerine sahip bireyleri) seçme eğilimindedirler. Çaprazlama aşamasında ise, seçilen bireylerden yeni birey ya da bireyler elde edilir [41]. GA'ların en son aşaması olan mutasyon aşamasında bireylerin genetik yapıları değiştirilerek yeni bireyler elde edilir. Mutasyon işleminin temel amacı, genetik çeşitliliği korumak ve lokal optimuma erken yakınsamayı önlemektir [41]. Son olarak, elde edilen bireyler ile mevcut popülasyondaki en iyi bireyler belirli oranda karıştırılarak yeni bir popülasyon oluşturulur. Bir GA süreci, istenilen kritere ulaşana kadar ya da elde edilmek istenen sonuç bulunana kadar devam ettirilir, bkz bir GA'nın sözde koduna:

Başlangıç popülasyonunu rastgele belirle

Her bireyin uygunluk değerini hesapla

İstenilen kritere ulaşana kadar ya da elde edilmek istenen sonuç bulunana kadar

{

- Seçim yöntemi ile çaprazlamaya katılacak bireyleri belirle

- Çaprazlama ile yeni birey(ler) oluştur

- Seçilen bireyler için mutasyon uygula

- Oluşan yeni bireyler ile mevcut popülasyondaki en iyi bireyleri belirli oranda karıştırarak yeni bir popülasyon oluştur (Elitism)

- Yeni popülasyondaki bireylerin uygunluk değerlerini hesapla

}

Bulunan en iyi sonucu göster

Standart Turnuva (ST) seçim yöntemi, daha az zaman karmaşıklığı [O(n)] ve paralel programlama için uygulanabilirlik gibi avantajlara sahip olduğu için GA'larda yaygın olarak kullanılmaktadır [42]. ST'yi seçim yöntemi olarak kullanan bir GA, Standart Genetik Algoritma (SGA) olarak adlandırılmaktadır [43].

3.4 Parçacık sürü optimizasyonu

PSO'nun kökeni, Eberhart ve Kennedy [44] tarafından sunulmuştur. Algoritma, kuş ve balık sürülerinin sosyal davranışlarını taklit etmek için tasarlanmıştır. Ancak, yapılan incelemeler sonucunda, PSO'nun optimizasyon yaptığı anlaşılmıştır.

GA'larda olduğu gibi, PSO da rasgele çözümlerle ilk popülasyonu (sürü, swarm) oluşturarak sürecini başlatır. GA'ların aksine, her bir potansiyel çözüm (parçacık, particle), aynı zamanda arama alanında dolaşmak için bir hız parametresi (velocity) de içerir. Her bir parçacığın hareketi, o parçacığın hızına, en iyi konumuna (pBest) ve sürü içerisinde bulunan en iyi konuma (gBest) dayanır. Bir parçacık daha iyi bir yer keşfettiğinde, keşfedilen yerin durumuna göre, pbest, gbest veya her ikisi de güncellenip, istenilen kritere ya da kriterlere ulaşana kadar arama işlemi devam ettirilir. Algoritmanın sözde-kodu aşağıdaki gibidir:

Sürüdeki parçacık sayısı = N;

Sürü içerisindeki her bir parçacığa rastgele değerler belirle

pBest ve gBest değerlerini belirle

for i=1 : N (her bir parçacık için)

Uygunluk değerini hesapla

If (uygunluk değeri pBest'ten daha iyi)

Yeni değeri pBest olarak ata

If (pBest, gBest'ten daha iyi)

pBest değerini gBest olarak da ata

end

end

Parçacık hızını hesapla

Parçacığın konumunu güncelle

end

3.5 Bipolar eşleşme eğilimi

Bipolar Eşleşme Eğilimi (Bipolar Mating Tendency, BMT), SGA'da kullanılan ST yöntemine dayanır [45]. Başlangıçta BMT, popülasyondan iki gruba ayrılacak kişileri rastgele seçer (eş olmaya aday bireyler). İlk eş, ST de olduğu gibi, grup üyeleri arasındaki en uygun (en iyi) bireydir. İlk eş seçiminin aksine, ikinci eşin seçimi için kullanılacak olan uygunluk kriteri belirsizdir. Birinci eşin iki kutuplu (bipolar) olan psikolojik durumuna bağlı olarak, ikinci eş, grup üyelerinden en iyi birey veya en kötü birey olabilir. Yöntemin sözde kodu aşağıda verilmiştir:

Popülasyon büyüklüğü = N;

for i=1 : N

İlk eşi standart turnuva yöntemiyle belirle

İkinci eş olmaya aday bireyleri rastgele seç ve uygunluk değerlerini hesapla

Adaylardan uygunluk değeri en iyi ve en kötü olanları belirle

If (rastgele değer ≤ iki kutupluluk olasılığı)

İkinci eşi uygunluk değeri en iyi olan birey olarak seç

Eşleri çaprazla

Else

İkinci eşi uygunluk değeri en kötü olan birey olarak seç

Eşleri çaprazla

end

end

İki kutupluluk olasılığı (bipolarity probability), deneysel bir değerdir. 0.05 ile 1 arasında, 0.05 adımlarla test edilerek bulunmuştur [45]. Yapılan teste göre, 0.25 değeri, diğer değerlere göre daha iyi performans sunmuş ve böylece, iki kutupluluk olasılığı 0.25 olarak alınmıştır.

4 Yapılan testler

Makalede test edilen tüm yöntemler, Matlab 2017a kullanılarak oluşturulmuştur. Ayrıca, Eşitlik (1)'de verilen ve literatürde yaygın olarak kullanılan Tam Aritmetik Çaprazlama (Whole Arithmetic Crossover) [46] tekniği yeniden düzenlenerek, bu yönteme benzer yeni bir çaprazlama yöntemi, bkz. Denklem (2), kullanılmıştır.

$$ {}_i^oG = \lambda {}_i^1G + (1-\lambda) {}_i^2G \quad (1) $$

$$ {}_i^oG = \frac{{}_i^1G + {}_i^2G}{2} + \lambda \left| {}_i^2G - {}_i^1G \right| (2rand - 1) \quad (2) $$

Denklem (1) ve (2)'de verilen ${}_i^oG$ ifadesi i'nci tekrarlamada elde edilen bireyi temsil ederken ${}_i^1G$ ve ${}_i^2G$ ifadeleri ise sırasıyla, i'nci tekrarlamada elde edilen 1. ve 2. bireyleri temsil etmektedir. λ sabiti ise $0 \le \lambda \le 1$ aralığında bir değere sahip olmalıdır. Önceki çalışmamızda [45] yapılan testlerde en uygun λ değeri 0.6 olarak belirlenmiş ve bu çalışmada da aynı değer kullanılmıştır. Son olarak *rand* ifadesi, Matlab'da kullanılan özel bir fonksiyon olup, 0 ile 1 arasında rastgele bir reel sayı üretmeye yaramaktadır.

Bireylerin çaprazlanması sonucunda oluşan yeni birey (çocuk), ebeveynlerinin genetik yapısı içerisinde sıkışıp kalmamalı ve bulunduğu uzayı araştırmaya (exploration) devam etmelidir. Denklem (1) dikkatlice incelenirse, oluşacak çocuğun genetik materyali, ebeveynlerinin genetik materyallerinin sınırladığı uzay bölgesinden değerler alacaktır. Tüm çocukların, ebeveynlerinin genetik sınırları içerisinde oluşması, popülasyonun sürekli daha dar bir alana sıkışmasına yol açacak ve uzayın araştırılması mümkün olmayacaktır.

Bunu önlemek için düzenlenen Denklem (2)'de, ebeveynlerin genetik materyallerinin sınırladığı bölgenin orta noktasını merkez alan (birinci terim) ve bu merkez etrafında ebeveyn genetik materyallerinin değer aralığının ±λ katı (ikinci terim) ile sınırlandırılmış daha geniş bir aralıkta elde edilen çocuklarla, popülasyonun dar alana sıkışması önlenmiştir.

Diğer taraftan mutasyon işlemleri olarak Rastgele Mutasyon (Random Mutation) kullanılmıştır. Rastgele Mutasyon, seçilen bir bireyin geni ya da genlerini, verilen aralık içerisinden, rastgele bir değer seçerek değiştirmektedir.

4.1 Test fonksiyonları

Algoritmaların performanslarını çeşitli şartlarda incelemek için, Tablo 1'de verilen, literatürde yaygın olarak kullanılan [47] test fonksiyonlarından yararlanılmıştır. Bu fonksiyonlar iki türlüdür: tek ve çok biçimli [48]. Tek biçimliler, hassas fonksiyonlardan oluşur ve global değere yavaş yavaş yakınsarlar. Çok biçimli fonksiyonlar, adından da anlaşılacağı gibi, birden fazla yerel ekstremum (maksimum ya da minimum) noktasına sahip fonksiyonları içerir.

4.2 İstatistiksel testler

Hipotez testleri, algoritmaları karşılaştırırken çıkarım yapmak için yaygın bir şekilde kullanılmaktadır [49]. Ancak, çıkarımı doğru bir şekilde yapmak için, sıfır hipotezi (H_0) ile alternatif hipotezin (H_1) uygun bir biçimde tanımlanmış olması gerekir. Sıfır hipotezi, genellikle, karşılaştırılan algoritmalar arasında hiçbir fark olmadığını ifade eden bir önermedir. Diğer bir taraftan, alternatif hipotez, farklılıkların bir ifadesidir. Bizim durumumuzda;

H_0: Karşılaştırılan yöntemler arasında, performans açısından bir fark yoktur.

H_1: Karşılaştırılan yöntemler arasında, performans açısından bir fark vardır.

Tablo 1. Test fonksiyonları, türleri ve global değerleri.

Table 1. Test functions, their types and global values.

Fonksiyon	Adı	Türü	Global Değeri
f_1	Ackley	Çok Biçimli	0
f_2	Axis Parallel Hyper Ellipsoid	Tek Biçimli	0
f_3	Branins	Çok Biçimli	0
f_4	De Jong	Tek Biçimli	0
f_5	Fifth De Jong	Çok Biçimli	Çok (Bunlardan biri 0.998)
f_6	Drop wave	Çok Biçimli	0
f_7	Goldstein-Price	Tek Biçimli	0
f_8	Griewangk	Çok Biçimli	3
f_9	Langermann	Çok Biçimli	Çok
f_{10}	Michalewicz	Çok Biçimli	- 1.8013034
f_{11}	Rastrigin	Çok Biçimli	0
f_{12}	Rosenbrock's Valley	Tek Biçimli	0
f_{13}	Rotated Hyper-Ellipsoid	Tek Biçimli	0
f_{14}	Shubert	Çok Biçimli	Çok (Bunlardan biri -210.48)
f_{15}	Schwefel	Çok Biçimli	-837.9658
f_{16}	Six Hump Camel Back	Çok Biçimli	-1
f_{17}	Sum of Different Powers	Tek Biçimli	0

Ayrıca, istatistiksel bir testin olasılık değeri olan α (hangi hipotezin reddedilmesi gerektiğine karar verir), yaygın olarak 0.05 veya 0.1 olarak alınır. 0.05 değerinin, α değeri için bir denge değeri olduğu düşünüldüğünden [50], yapılan testlerde α değeri, 0.05 olarak kabul edilmiştir.

4.2.1 Friedman testi

Friedman [51] tarafından sunulan Friedman testi, parametrik olmayan istatistiksel bir testtir. Birden fazla yöntemin davranış farklılıklarını belirtmek için kullanılmaktadır. Friedman testinde, test durumları satırlar halinde temsil edilirken, karşılaştırılan yöntemlerin sonuçları sütunlar halinde yer alır.

Friedman testi, prosedürüne başlarken, karşılaştırılan yöntemlerin sonuçlarına bakarak her satırı derecelendirir (en iyiye 1, en kötüye test edilen algoritma sayısı). Ardından her sütun için toplam derece değerini hesaplar. Testin önemi X^2 (Chi-square) dağılımı ve *k-1* serbestlik derecesi (degrees of freedom, *df*) ile belirlenir. Burada *k* değeri, karşılaştırılan yöntemlerin sayısıdır. *df* değerine karşılık gelen uygun X^2değerleri [52]'de bulunabilir. Bulunan X^2değeri beklenenden büyükse, sıfır hipotezini reddetmek gerekir. X^2değeri Denklem (3) kullanılarak bulunabilir:

$$ {X^2}_{found} = \frac{12}{n.k.(k+1)} * \sum R^2 - 3n(k+1) \quad (3) $$

Bu eşitlikte verilmiş olan *n* değeri, test edilen algoritma sayısını göstermektedir. *R* değeri ise derecelerin toplamını ifade etmektedir

4.2.2 Wilcoxon işaretli sıralar testi

Wilcoxon İşaretli Sıralar testi, iki örnek veya iki algoritma arasındaki farkları belirtmek için kullanılan parametrik

olmayan istatistiksel bir testtir [53]. Bu test şu şekilde çalışmaktadır:

Başlangıçta, her bir yöntem için N defa yapılmış olan test sonuçları, boyutları Nx1 olan vektör haline getirilir. Fark vektörünü elde etmek için, karşılaştırılacak iki yöntemin vektörleri birbirinden çıkarılır. Daha sonra, fark vektörünün her bir satırı, 1'den N'ye (minimum değer 1 olmak üzere) derecelendirilir. Sonra, fark vektörünün pozitif veya negatif değerlerinin toplam derecelerini temsil eden R_+ ve R_- değerleri hesaplanır. Testin T değeri olarak adlandırılan değer, $min(R_+,R_-)$ olarak belirlenir. Böylece, bu testin p olasılık değeri T değeri kullanılarak hesaplanır.

5 Test sonuçları ve tartışmalar

5.1 Test fonksiyonlarının sonuçları

Testlerde, rassallığın etkisini en aza indirmek için, literatürde yaygın olarak kullanılan bir yöntem olan [47],[54], n defa tekrarlama uygulanmıştır. Bu çalışmamızda, kullanılan her bir algoritma için, n=25 seçilerek, 25 farklı rastgele tohum içeren 25 farklı işlemle tekrarlama yapılmıştır. Böylece toplam 31 milyon işlem (yaklaşık olarak) gerçekleştirilmiştir: popülasyon büyüklüğü için 4 tane parametre, turnuva büyüklüğü için 4 tane parametre, çaprazlama olasılığı için 16 tane parametre, mutasyon oranı için 16 tane parametre, 25 rastgele tohum ve 300 tekrarlama. Aslında, 4096 parametre düzenlemesi verimli bir şekilde test edilmiştir. Çünkü bu makalede, 25 farklı işlemle alınan sonuçların medyan değerleri temsil edilmiştir (tekrarlama sayısının sabit olduğu kabul edilmiştir).

Izgara Arama istenen sonuçları sunsa bile, büyük bir hesaplama bütçesine neden olmaktadır. Öte yandan CFS, SGA, PSO ve GA-BMT algoritmalarında hesaplama bütçesini düşürmek için sadece 400 parametre koşulu uygulanmıştır. Böylelikle, Izgara Arama'nın kullandığı parametre ayarının yaklaşık %10'u kullanılarak hesaplama bütçesinden tasarruf edilmiştir.

Algoritmalarda (CFS, SGA, PSO ve GA-BMT) kullanılan her bir birey, dört gen içeren (popülasyon büyüklüğü, turnuva büyüklüğü, çaprazlama olasılığı ve mutasyon oranı) ve her genin bir parametre değerini temsil ettiği bireyler olarak tanımlanırlar. Bu tanımlamaya uyacak şekilde rastgele seçilen 20 birey ile algoritmaların başlangıç popülasyonları oluşturulur. Başlangıç popülasyonu oluşturulan algoritmalar, SGA'yı uygunluk fonksiyonu (fitness function) olarak alıp 20 popülasyon büyüklüğü ve 20 tekrarlama (400 parametre koşulu) ile SGA'nın en iyi parametre ayarlarını bulmak için işlemlerini gerçekleştirirler. Çaprazlama ve mutasyon aşamaları sonrasında oluşan bireylerin gen değerleri, reel sayı değerleri olduğundan, Izgara Arama için kullanılan parametre değerlerinden farklı olabilmektedir. Bu durum, bireylerin gen yapılarının bozulduğu ve bozulan bu genlerin tamir edilmesi gerektiği anlamına gelir. Tamir operatörü aracılığıyla, bireyin bozuk olan genleri, bilinen en yakın parametre değerlerine yuvarlanarak, bireyin istenilen genetik yapıya sahip olması sağlanır.

Bu makalede, meta-iyileştirici olarak kullanılan SGA'nın parametrelerini ayarlamak için, "varsayılan parametre ayarları" [1] (çaprazlama oranı = 0.7, turnuva büyüklüğü=3, mutasyon oranı = 0.05) kullanılmıştır. Ayrıca, GA-BMT'nin parametre yapılandırması için de aynı parametre ayarları kullanılmış ve iki kutupluluk olasılığı 0.25 olarak alınmıştır. Bununla birlikte, yapılan tüm işlemlerde standart PSO ayarları kullanılmıştır [44]. 4096 parametre konfigürasyonunu test eden Izgara Arama'nın, 400 parametre konfigürasyonu kullanan yöntemlere kıyasla en iyi performansı sunması doğaldır. Sadece %10'luk bir hesaplama bütçesi kullanarak, Izgara Arama ile aynı sonuçları sunabilen (veya bu sonuçlara olabildiğince yaklaşan) yöntemlerin başarılı kabul edilebileceği aşikâr olduğundan, Tablo 2'de bu sonuçlar koyu olarak gösterilmiş ve böylece yöntemlerin hangi testlerde daha başarılı olduğu belirtilmek istenmiştir.

CFS ve GA-BMT, Izgara Arama'ya yakın sonuçlar verdiklerinden, bu algoritmalar SGA'nın parametrelerini ayarlamak için kullanılabilir. PSO'nun standart parametre ayarlarının kullanılması, PSO için dezavantaj olduğundan, tüm test örneklerinde etkili sonuçlar verememiş ve SGA'nın gerisinde kalmıştır. Diğer bir taraftan, GA-BMT, Griewangk ve Sum of Different Powers dışındaki her test fonksiyonunda güçlü çözümler sunmuştur. Bu nedenle de, SGA'nın parametrelerini ayarlamada, sağlam bir meta-iyileştirici olarak kendini kanıtlamıştır. Tablo 3, her bir test fonksiyonu için algoritmaların önerdiği parametre yapılandırmalarını göstermektedir: sırasıyla çaprazlama olasılığı, turnuva büyüklüğü ve mutasyon oranı. Tablo 3'e göre SGA'nın, çaprazlama olasılığı 0.6'ya eşit veya daha yüksek olduğunda en iyi performansını sunmaktadır.

Tablo 2. GA-BMT ve diğer meta-tekniklerden alınan sonuçlar.

Table 2. Results of GA-BMT and other meta techniques.

Fonksiyon	Izgara Arama	CFS	SGA	PSO	GA-BMT
f_1	8.8818e-16	7.9936e-15	1.1546e-14	1.1546e-14	**4.4409e-15**
f_2	1.7350e-153	7.5559e-30	**1.6744e-47**	5.3463e-32	**1.6744e-47**
f_3	0.3979	**0.3979**	**0.3979**	**0.3979**	**0.3979**
f_4	2.8184e-164	1.5615e-30	1.5615e-32	1.0516e-21	**1.5525e-34**
f_5	0.998	**0.998**	**0.998**	**0.998**	**0.998**
f_6	-1	**-1**	**-1**	**-1**	**-1**
f_7	0	**0**	0.0015	0.002	0.0013
f_8	3	**3**	**3**	**3**	**3**
f_9	-4.16	-4.1545	-4.1554	**-4.1557**	**-4.1557**
f_{10}	-1.8013	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**
f_{11}	0	**0**	**0**	**0**	**0**
f_{12}	0	2.5504e-06	1.2264e-06	1.2264e-06	**7.6574e-07**
f_{13}	3.2836e-150	1.7307e-24	**5.7415e-39**	4.5233e-29	**5.7415e-39**
f_{14}	-210.4823	**-210.4823**	**-210.4823**	**-210.4823**	**-210.4823**
f_{15}	-837.9658	**-837.9658**	**-837.9658**	**-837.9658**	**-837.9658**
f_{16}	-1.0316	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**
f_{17}	2.5614e-155	**1.0992e-85**	4.0342e-29	5.5179e-29	1.6431e-36

Tablo 3. Her bir test fonksiyonu için önerilen parametre ayarları.

Table 3. Recommended parameter settings for each test function.

Fonksiyonlar	Izgara Arama			CFS			SGA			PSO			GA-BMT		
f_1	0.6	3	0.025	0.7	3	0.1	0.8	3	0.035	0.8	3	0.035	0.85	3	0.08
f_2	0.8	3	0.025	0.8	3	0.08	0.9	3	0.04	0.8	3	0.1	0.9	3	0.04
f_3	0.65	3	0.025	0.65	3	0.025	0.65	3	0.025	0.75	3	0.05	1	3	0.1
f_4	0.7	3	0.045	0.8	3	0.03	0.8	3	0.03	1	3	0.095	0.9	3	0.06
f_5	0.65	3	0.08	0.65	3	0.08	0.65	3	0.08	0.65	3	0.08	0.65	3	0.08
f_6	0.8	3	0.06	0.75	3	0.09	0.8	3	0.06	0.8	3	0.06	0.9	3	0.06
f_7	0.6	3	0.025	0.6	3	0.025	0.85	3	0.095	1	5	0.04	0.9	4	0.075
f_8	0.95	3	0.055	0.95	3	0.055	0.95	3	0.055	0.95	3	0.055	0.8	3	0.08
f_9	1	3	0.08	0.55	5	0.085	0.95	4	0.05	1	5	0.09	1	5	0.08
f_{10}	0.95	3	0.025	0.95	3	0.025	0.95	3	0.025	1	4	0.085	0.85	3	0.03
f_{11}	0.8	3	0.025	0.8	3	0.025	0.8	3	0.025	0.8	3	0.035	0.8	3	0.025
f_{12}	0.7	3	0.025	0.7	4	0.05	0.9	4	0.075	0.9	4	0.075	0.85	4	0.095
f_{13}	0.85	3	0.045	0.85	3	0.025	0.9	3	0.04	0.9	3	0.1	0.9	3	0.04
f_{14}	0.85	3	0.045	0.85	3	0.045	0.85	3	0.045	0.9	3	0.07	0.9	3	0.095
f_{15}	0.7	3	0.055	0.7	3	0.055	0.6	3	0.085	0.55	3	0.08	0.65	3	0.075
f_{16}	0.75	3	0.025	0.75	3	0.025	0.6	3	0.05	0.8	3	0.035	0.8	3	0.05
f_{17}	0.9	3	0.035	0.85	5	0.075	0.9	3	0.085	0.75	3	0.1	0.8	3	0.025

Turnuva büyüklüğü için genellikle 3 alınabilirken, mutasyon oranı çeşitlidir. Tüm meta-optimizasyon teknikleri, popülasyon büyüklüğü 200 olduğunda en iyi performanslarını elde ettiğinden, popülasyon büyüklüklerinin bilgileri Tablo 3'te gösterilmemektedir.

5.2 İstatistiksel testlerin sonuçları

Friedman testinin sonuçları IBM SPSS Statistics 22 ile elde edilmiştir. Ayrıca, Wilcoxon İşaretli Sıralar Testi'nin sonuçlarına ulaşmak için Matlab 2017a'nın *signrank* fonksiyonu kullanılmıştır.

5.2.1 Friedman testinin sonuçları

Tablo 4'e bakarak hangi seçim yönteminin daha başarılı olduğunu anlamak mümkündür. Minimum sıra (derece) değerine sahip bir seçim yönteminin, diğerlerine göre daha iyi performans sunduğu söylenebilir. O halde, Tablo 4'e göre Izgara Arama daha iyi bir performans sunmaktadır. Ancak, Izgara Arama'nın kullanılması sonucu oluşacak toplam maliyet göz önünde bulundurulursa, GA-BMT'nin iyi bir alternatif olabileceği aşikârdır.

Tablo 4. Yöntemlerin ortalama sıraları.

Table 4. Mean ranks of the methods.

Metotlar	Sıra (Derece)
Izgara Araması	**2.35**
CFS	3.29
SGA	3.18
PSO	3.50
GA-BMT	**2.68**

df=4 ve α=0.05 olduğundan, [51] 'deki tablodan, X^2= 9.48 olmalıdır. Tablo 5'te bulunan X^2değeri (13.46405), beklenen değerden (9.48) daha yüksek olduğundan, sıfır hipotezi reddedilmelidir. Sıfır hipotezini reddetmek ise, "Karşılaştırılan yöntemler performans açısından farklıdır" anlamına gelmektedir. Ancak, bu test özellikle hangi seçim yönteminin diğerinden farklı olduğunu netleştirmez; bu nedenle, Wilcoxon İşaretli Sıralar Testi uygulanmıştır:

Tablo 5. İstatistiksel değerler.

Table 5. Statistical values.

İstatistiksel Değerler	Sonuçlar
X^2	13.46405
Olasılık Değeri	0.009217

5.2.2 Wilcoxon işaretli sıralar testinin sonuçları

Wilcoxon İşaretli Sıralar Testi çiftler halinde gerçekleştirilir: Bir yöntem, tablonun ilk sütunundan alınır ve sırasıyla sonraki tüm yöntemlerle karşılaştırılır.

p (olasılık) değerleri, karşılaştırılan algoritmalar arasındaki önemli farklılıkları açıklar. Sıfır hipotezi reddetmek için bu değerlerin α (0.05) değerinden daha düşük olması gerekir. Aksi takdirde, sıfır hipotezinin kabul edilmesi, yani "Karşılaştırılan yöntemlerin performans açısından benzerliği vardır." denilmesi, gerekir.

Tablo 6'da sunulan *p* değerlerinin neredeyse tamamı, α = 0.05 değerinden küçük olduğu için, karşılaştırılan yöntemlerin performans açısından benzerliğinin olmadığı aşikârdır. Ayrıca, GA-BMT'nin, Izgara Arama ve CFS'ye; SGA'nın da PSO'ya benzer bir performans gösterdiği görülebilmektedir.

Tablo 6. Wilcoxon işaretli sıralar testinin olasılık değeri.

Table 6. The probability values of wilcoxon signed rank test.

Metotlar	Izgara Arama	CFS	SGA	PSO	GA-BMT
Izgara Arama	-	0.0437	0.0195	0.0195	0.0531
CFS	-	-	0.0471	0.0195	0.0531
SGA	-	-	-	0.3125	0.0312
PSO	-	-	-	-	0.0156
GA-BMT	-	-	-	-	-

6 Sonuçlar

Evrimsel algoritmaların performansında, uygun parametre ayarlarının seçilmesi önemli bir rol oynamaktadır. Parametre ayarlarının manuel olarak yapılandırılması zaman alıcı olduğundan, algoritmaları etkin bir şekilde kullanmak için meta-optimizasyon teknikleri önerilmiştir.

Bu nedenle, standart bir Genetik Algoritmanın (SGA) performansını iyileştirmek için, bir Genetik Algoritmanın (GA) seçim yöntemi olarak Bipolar Eşleşme Eğilimi (BMT) algoritması kullanılmış ve oluşan yeni algoritma (GA-BMT) meta-iyileştirici olarak uygulanmıştır. GA-BMT'nin performansını değerlendirmek için, algoritmanın performansı, meta arama teknikleri (Izgara Arama ve Kabadan İnce'ye Arama) ile meta-optimizasyon algoritmalarının (SGA, Parçacık Sürü Optimizasyonu) performanslarıyla karşılaştırılmıştır. Elde edilen sonuçları anlamlandırmak için, parametrik olmayan istatistiksel testler, Friedman ve Wilcoxon İşaretli Sıralar, uygulanmıştır.

Yapılan test sonuçlarına göre, Izgara Arama en iyi sonuçları sunmaktadır. Ancak, bu yöntemin kullanılması, hesaplama maliyetini artıracağından, Izgara Arama'ya yakın sonuçlar veren bir meta-iyileştiricinin kullanılması daha mantıklı olacaktır.

GA-BMT, kullanılan üç fonksiyonda; Ackley, De Jong ve Rosenbrocks Valley tek kazanan, on yedi fonksiyondan on dördünde de uygun çözümler sunan bir meta-iyileştirici olmuştur. Bu nedenle, GA-BMT, SGA'yı ayarlamak için güçlü bir meta-iyileştirici olduğunu kanıtlamıştır.

GA-BMT, SGA'nın parametrelerini ayarlamada sağlam bir meta-iyileştirici olarak kendini kanıtlasa da, gerçek dünya mühendislik problemlerindeki performansının incelenmesi önerilmektedir.

7 Conclusions

Choosing the appropriate parameter settings plays an important role in the performance of Evolutionary Algorithms. Since manually configuring parameter settings is time consuming, meta-optimization techniques have been proposed in order to utilize these algorithms effectively.

Therefore, to improve the performance of a standard Genetic Algorithm (SGA), the Bipolar Mating Tendency (BMT) algorithm was used as the selection method of a Genetic Algorithm (GA), and the new obtained algorithm (GA-BMT) was applied as a meta-optimizer. To evaluate the performance of GA-BMT, the performance of the algorithm was compared with the performance of meta-search techniques (Grid Search and Rough-to-Fine Search) and meta-optimization algorithms (SGA, Particle Swarm Optimization). Non-parametric statistical tests, Friedman and Wilcoxon Signed Rank, were performed to demonstrate the significance of the results.

Based on the test results, Grid Search offers the best results. However, since using this method will increase the computational cost, it would be more logical to use a meta-optimizer that gives results close to Grid Search.

In addition to being the only winner in three functions; Ackley, De Jong, and Rosenbrocks Valley, GA_BMT presents appropriate solutions in all fourteen of the seventeen functions as a meta-optimizer. Therefore, GA-BMT has proven to be a powerful meta-optimizer for adjusting SGA.

Although GA-BMT has proven itself as a solid meta-optimizer in the manner of adjusting the parameters of SGA, it is recommended to examine its performance in real-world engineering problems.

8 Yazar katkı beyanı

Gerçekleştirilen çalışmada, Mashar Cenk GENÇAL ve Mustafa ORAL fikrin oluşmasına, elde edilen sonuçların değerlendirilmesine ve incelenmesine, yazım denetimi ve içerik açısından makalenin kontrol edilmesine birlikte katkı sunmuşlardır. Ayrıca, Mashar Cenk GENÇAL tüm algoritmaların ve fonksiyonların Matlab kodlarını yazmış ve testlerini gerçekleştirmiştir.

9 Etik kurul onayı ve çıkar çatışması beyanı

Hazırlanan makalede etik kurul izni alınmasına gerek yoktur. Hazırlanan makalede herhangi bir kişi/kurum ile çıkar çatışması bulunmamaktadır.

10 Kaynaklar

[1] Lobo FG, Goldberg DE. "The parameter-less genetic algorithm in practice". *Information Science*, 167(1-4), 217-232, 2004.

[2] Mercer RE, Sampson JR. "Adaptive search using a reproductive meta-plan". *Kybernetes*, 7(3), 215-228, 1978.

[3] Grefenstette JJ. "Optimization of Control Parameters for Genetic Algorithms". *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1), 122-128, 1986.

[4] Freisleben B, Härtfelder M. "Optimization of genetic algorithms by genetic algorithms". *International Conference on Artificial Neural Nets and Genetic Algorithms-ANNGA'93,* Innsbruck, Austria, 1-5 May 1993.

[5] Bäck T. "Parallel optimization of evolutionary algorithms". *The Third Conference on Parallel Problem Solving from Nature-PPSN III*, Jerusalem, Israel, 9-14 October 1994.

[6] Hinterding R. "Gaussian mutation and self-adaption for numeric genetic algorithms". *IEEE International Conference*, Perth, Australia, 29 November-1 December 1995.

[7] Keane AJ. "Genetic algorithm optimization of multi-peak problems: studies in convergence and robustness". *Artificial Intelligence Eng*ineering, 9(2), 75-83, 1995.

[8] Cortez P, Rocha M, Neves J. "A meta-genetic algorithm for time series forecasting". *10th Portuguese Conference on Artificial Intelligence*, Porto, Portugal, 1-3 December 2001.

[9] Hong SH, Cornelius J, Wang Y, Pant K. "Fault compensation by online updating of genetic algorithm-selected neural network model for model predictive control". *Applied Science*, 1(11), 1-16, 2019.

[10] Camilleri M, Neri F, Papoutsidakis M. "An algorithmic approach to parameter selection in machine learning using meta-optimization techniques". *WSEAS Transactions on systems*, 13(1), 203-212, 2014

[11] Li H, Huang Z, Liu X, Zeng C, Zou P. "Multi-Fidelity meta-optimization for nature inspired optimization algorithms". *Applied Soft Computing*, 2020. https://doi.org/10.1016/j.asoc.2020.106619.

[12] Łapa K. "Meta-optimization of multi-objective population-based algorithms using multi-objective performance metrics". *Information Science*, 489, 193-204, 2019.

[13] Magliani F, Cagnoni S, Sani L, Prati A. "Genetic algorithms for the optimization of diffusion parameters in content-based image retrieval". *Proceedings of the 13th International Conference on Distributed Smart Cameras*, Trento, Italy, 9-11 September 2019.

[14] Magliani F, Cagnoni S, Sani L, Prati A. "Diffusion Parameters Analysis in a Content-Based Image Retrieval Task for Mobile Vision". *Sensors*, 20(16), 4449-4472, 2020.

[15] Matrenin PV, Sekaev VG. "Particle Swarm optimization with velocity restriction and evolutionary parameters selection for scheduling problem". *2015 International Siberian Conference on Control and Communications*, Omsk, Russia, 21-23 May 2015.
[16] Meissner M, Schmuker M, Schneider G. "Optimized particle swarm optimization (OPSO) and its application to artificial neural network training". *BMC Bioinformatics*, 7(1), 1-11, 2006.
[17] Pedersen MEH, Chipperfield AJ. "Simplifying Particle Swarm Optimization". *Applied Soft Computing*, 10(2), 618-628, 2010.
[18] Pedersen MEH, Chipperfield AJ. "Tuning Differential Evolution for Artificial Neural Networks". Hvass Laboratories, Technical Report, HL0803, 2008.
[19] Mason K, Duggan J, Howley E. "A meta optimization analysis of particle swarm optimization velocity update equations for watershed management learning". *Applied Soft Computing*, 62, 148-161, 2018.
[20] Jaafari A, Panahi M, Pham BT, Shahabi H, Bui DT, Rezaie F. "Meta optimization of an adaptive neuro-fuzzy inference system with grey wolf optimizer and biogeography-based optimization algorithms for spatial prediction of landslide susceptibility". *Catena*, 175, 430-445, 2019.
[21] Chen W, Hong H, Panahi M, Shahabi H, Wang Y, Shirzadi A. "Spatial prediction of landslide susceptibility using GIS-based data mining techniques of ANFIS with whale optimization algorithm (WOA) and grey wolf optimizer (GWO)". *Applied Science*, 9(18), 3755-3787, 2019.
[22] Birattari M, Stützle T, Paquete L, Varrentrapp K. "A racing algorithm for configuring metaheuristics". *GECCO 2002 Proceedings of the Genetic and Evolutionary Computation Conference*, New York, USA, 9-13 July 2002.
[23] Balaprakash P, Birattari M, Stuetzle T. "Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement". *Hybrid Metaheuristics*, 4771, 108-122, 2007.
[24] Birattari M, Yuan Z, Balaprakash P, Stützle T. "F-Race and Iterated F-Race: An overview". Institut de Recherches Interdisciplinaireset de D´eveloppements en Intelligence Artificielle, Technical Report, TR/IRIDIA/2009-018, 2009.
[25] Trujillo L, González EÁ, Galván E, Tapia JJ, Ponsich A. "On the analysis of hyper-parameter space for a genetic programming system with iterated F-Race". *Soft Computing*, 24(19), 14757-14770, 2020.
[26] François O, Lavergne C. "Design of evolutionary algorithms - A statistical perspective". *IEEE Transactions on Evolutionary Computation*, 5(2), 129-148, 2001.
[27] Nannen V, Eiben AE. "A method for parameter calibration and relevance estimation in evolutionary algorithms". *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, New York, USA, 8-12 July 2006.
[28] Adenso-Díaz B, Laguna M. "Fine-Tuning of algorithms using fractional experimental designs and local search". *Operational Research,* 54(1), 99-114, 2006.
[29] Hutter F, Hoos HH, Stützle, T. "Automatic algorithm configuration based on local search". *BMC Bioinformatics*, 7, 1152-1157, 2007.
[30] Bartz-Beielstein T, Lasarczyk CWG, Preuß M. "Sequential parameter optimization". *2005 IEEE Congress*, Edinburgh, UK, 2-5 September 2005.
[31] Hutter F, Hoos HH, Leyton-Brown K, Stützle T. "ParamILS: An automatic algorithm configuration framework". *Artificial Intelligence Research*, 36, 267-306, 2009.
[32] Hutter F, Hoos HH, Leyton-Brown K. "Sequential model-based optimization for general algorithm configuration". *Learning and Intelligent Optimization-5th International Conference, LION 5*, Rome, Italy, 17-21 January 2011.
[33] Branke J, Elomari J. "Meta-Optimization for parameter tuning with a flexible computing budget". *Proceeding Fourteenth International Conference,* Philadelphia, USA, 7-11 July 2012.
[34] Francesca G, Pellegrini P, Stützle T, Birattari M. "Off-line and on-line tuning: a study on operator selection for a memetic algorithm applied to the QAP". *European Conference on Evolutionary Computation in Combinatorial Optimization*, Torino, Italy, 27-29 April 2011.
[35] Aburomman AA, Ibne Reaz M Bin. "A novel SVM-kNN-PSO ensemble method for intrusion detection system". *Applied Soft Computing,* 38, 360-372, 2016.
[36] Meyer-Nieberg S, Beyer HG. "Self-adaptation in evolutionary algorithms". *Computational Intelligence*, 54, 47-75, 2007.
[37] Bendre N, Marín HT, Najafirad P. "Learning from few samples: A survey". *arXiv*, 2020. https://arxiv.org/pdf/2007.15484.pdf.
[38] Staelin C. "Parameter Selection for Support Vector Machines". Hewlett-Packard Company, Technical Report, HPL-2002-354, 2003.
[39] Holland JH. *Adaptation in Natural and Artificial Systems*. 2nd ed. Michigan, USA, Ann Arbor, University of Michigan Press, 1975.
[40] Goldberg DE. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 3rd ed. Boston, USA, Addison Wesley Publishing Company, 1989.
[41] Mitchell M. *An Introduction to Genetic Algorithms*. 2nd ed. London, England, MIT Press, 1996.
[42] Blickle T, Thiele L. "A comparison of selection schemes used in evolutionary algorithms". *Evolutionary Computation*, 4, 361-394, 1996.
[43] Jiacheng L, Lei L. "A hybrid genetic algorithm based on information entropy and game theory". *IEEE Access*, 8, 36602-36611, 2020.
[44] Eberhart R, Kennedy J. "A new optimizer using particle swarm theory". *Micro Machine and Human Science, 1995 MHS'95*, *Proceedings of the Sixth International Symposium*, Nagoya, Japan, 4-6 October 1995.
[45] Oral M, Gençal MC. "Harmony between the best and the worst individuals in tournament selection". *Majlesi Journal of Mechatronic Systems,* 6(3), 8-25, 2017.
[46] Michalewicz Z, Janikow C. Z. "Handling constraints in genetic algorithms". *Adaptive Computing in Design and Manufacture*, 1991. https://doi.org/10.1007/978-0-85729-345-9_23.
[47] Mirjalili S, Lewis A. "Grey wolf optimizer". *Advances in Engineering Software*, 69, 46-61, 2014.
[48] Molga M, Smutnicki C. "Test functions for optimization needs". https://marksmannet.com/RobertMarks/Classes/ENGR5358/Papers/functions.pdf (23.01.2021)
[49] Conover WJ. *Practical Nonparametric Statistics*. 3rd ed. New Delhi, India, Wiley India Pvt. Limited, 2006.
[50] Everitt B, Skrondal A. *The Cambridge Dictionary of Statistics*. 4th ed. London, England, Cambridge University Press, 2002.

[51] Friedman M. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". *Journal of the American Statistical Association*, 32(200), 675-701, 1937.

[52] Sheskin DJ. *Handbook of Parametric and Nonparametric Statistical Procedures*. 5th ed. New York, USA, Chapman and Hall/CRC, 2003.

[53] Wilcoxon F. "Individual comparisons by ranking methods". *In Breakthroughs in Statistics*, 1(6), 196-202, 1992.

[54] Karaboga D, Akay B. "A comparative study of artificial bee colony algorithm". *Applied Mathematics and Computation*, Elsevier, Netherlands, 214, 108-132, 2009.