T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİĞİN DİRENÇLİ ORTAMDA HAREKETİNİ TEMEL ALAN YENİ BİR META SEZGİSEL ALGORİTMA TASARIMI

DOKTORA TEZİ

Hüseyin DEMİRCİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Bilim Dalı

HAZİRAN 2023

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİĞİN DİRENÇLİ ORTAMDA HAREKETİNİ TEMEL ALAN YENİ BİR META SEZGİSEL ALGORİTMA TASARIMI

DOKTORA TEZİ

Hüseyin DEMİRCİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Bilim Dalı

Tez Danışmanı: Doç. Dr. Nilüfer YURTAY

HAZİRAN 2023

Hüseyin Demirci tarafından hazırlanan "Elektriğin Dirençli Ortamda Hareketini Temel Alan Yeni Bir Meta Sezgisel Algoritma Tasarımı" adlı tez çalışması 20.06.2023 tarihinde aşağıdaki jüri tarafından oy birliği ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı **Bilgisayar Mühendisliği** Bilim Dalı'nda Doktora tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı :	**Doç. Dr. Nilüfer YURTAY** (Danışman) Sakarya Üniversitesi	.....
Jüri Üyesi :	**Doç. Dr. Gökçen ÇETİNEL** Sakarya Üniversitesi	.....
Jüri Üyesi :	**Dr. Öğr. Üyesi Ali GÜLBAĞ** Sakarya Üniversitesi	.....
Jüri Üyesi :	**Doç. Dr. Çiğdem ACI** Mersin Üniversitesi	.....
Jüri Üyesi :	**Dr. Öğr. Üyesi Mehmet ACI** Mersin Üniversitesi	.....

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum "Elektriğin Dirençli Ortamda Hareketini Temel Alan Yeni Bir Meta Sezgisel Algoritma Tasarımı" başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(31/05/2023).

Hüseyin DEMİRCİ

Eşime ve oğluma

TEŞEKKÜR

Doktora eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen değerli danışman hocam Doç. Dr. Nilüfer YURTAY'a teşekkürlerimi sunarım.

En büyük şansım sevgili eşim ve canım ailem, bu uzun ve zorlu yolculuğumda her zaman yanımda olduğunuz için hepinize minnettarım, sabrınız ve fedakârlıklarınız için çok teşekkür ederim.

Bu çalışmanın maddi açıdan desteklenmesine olanak sağlayan Sakarya Üniversitesi Bilimsel Araştırma Projeleri (BAP) Komisyon Başkanlığına (Proje No: 2019-7-25-278) teşekkür ederim.

Hüseyin Demirci

İÇİNDEKİLER

Sayfa

KISALTMALAR

A* : A-Star Algoritması
AGKA : Artımlı Genetik K-ortalama Algoritması
BAPSO : bulanık uyarlamalı parçacık sürü optimizasyonu
CEC : Congress on Evolutionary Computation
DAA : Diferansiyel Arama Algoritması
DB-Index : Davies-Bouldin index
DYA : Deniz Yırtıcısı Algoritması
EA : Evrimsel Algoritma
EAA : Elektrik Arama Algoritması
GA : Genetik Algoritma
GİA : Geri İzleme Arama Algoritması
GKA : Genetik K-Ortalamalar Algoritması
HCV : Hepatit C Virüsü
HGKA : Hızlı Genetik K-Means Algoritması
H-GKA : Hibrit Genetik K-ortalama Algoritması
HKA : Hibrit k-medoid algoritması
HKPSO : Hızlandırılmış kaotik parçacık sürü optimizasyonu
HŞO : Harris Şahinleri Optimizasyon Algoritması
IEEE : Institute of Electrical and Electronics Engineers
KHO : K-Harmonik Ortalamalar
KK : Kaba Küme
KKO : karınca kolonisi optimizasyonu
KOA : K-Ortalama Kümeleme Optimizasyon Algoritması
NP : Non-deterministic polynomial-time
PSO : Parçacık Sürü Optimizasyonu

SİMGELER

p	: Olasılık değeri
α	: Duyarlık değeri
D	: Difüzyon katsayısı
t	: Mevcut iterasyon
Δx	: Uzaklık
π	: Pi sayısı

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ELEKTRİĞİN DİRENÇLİ ORTAMDA HAREKETİNİ TEMEL ALAN YENİ BİR META SEZGİSEL ALGORİTMA TASARIMI

ÖZET

"Optimizasyon" kelimesinin Türk Dil Kurumu'ndaki karşılığı "en uygun duruma getirme" olarak belirtilmiştir. Hepimiz günlük hayatta bilinçsizce bir optimizasyon süreci yaşamaktayız. Yolculuk yaparken rotayı seçerken veya basit alışverişlerimizi yaparken, mevcut duruma göre en uygun seçenekleri tercih etmeye çalışmaktayız.

Küçük ölçekli optimizasyon problemlerini beyin gücümüzü kullanarak veya geleneksel algoritmalarla çözebilmekte, ancak büyük veya karmaşık problemlerin çözümü için bilgisayarlardan faydalanmaktayız.

Bilgisayar destekli optimizasyon problemlerini çözmek için birçok algoritma tasarlanmıştır. Geleneksel algoritmalar, basit yapılarına rağmen yavaş çalışma süreleri ve esnek olmayan uygulanabilirlikleri nedeniyle sınırlıdır. Geleneksel algoritmalar sadece belirli tipte problemleri çözebilme yeteneğine sahiptirler.

İlerleyen bilim ve teknoloji sayesinde, daha hızlı çalışan ancak yine problem bağımlı olan algoritmalar geliştirilmiştir. Bu algoritmalar genellikle sezgisel algoritmalar olarak adlandırılır ve belirli bir problem türünün çözümü için kullanılmaktadır. Ancak, bu algoritmalar geleneksel algoritmalardan daha hızlı olmalarına rağmen problem çeşitliliği konusunda esnek değillerdir, bu nedenle eksik kalmışlardır ve gelişim ve değişim ihtiyacı ortaya çıkmıştır. Bu ihtiyaçtan yola çıkarak meta-sezgisel algoritmalar geliştirilmiştir. "Meta" kelimesi antik Yunanca ‘da "ötesi" veya "sonrası" anlamına gelir. Bu nedenle, meta-sezgisel algoritmaları "sezgisel ötesi" veya "sezgisel sonrası" algoritmalar olarak düşünebiliriz. Meta-sezgisel algoritmalar, sezgisel algoritmalardan daha hızlı çalışırken, bir probleme veya belirli bir problem türüne bağımlı olmadan çalışabilme özelliğine sahiptirler.

Bu tez çalışmasında, Elektriksel Arama Algoritması (EAA) adı verilen yeni bir meta sezgisel algoritması önerdik. Önerilen algoritma, ahşap, cam ve gazlar gibi yüksek dirençli alanlarda elektriğin hareketine dayanmaktadır. EAA, arama uzayının alt ve üst sınırlarında yalnızca bir ajanın başlattığı ve kutup adı verilen yapılar oluşturan benzersiz bir başlatma şemasına sahiptir. Bu aşamadan sonra EAA, arama yapmak için benzersiz küresel ve yerel arama stratejileri kullanmaktadır. Arama mekanizması, zıt kutuplara hareket eden elektronlara dayanmaktadır. EAA algoritmasının ilk başlatma şeması, kutup arama mekanizması ve en iyi çözümlerin güncelleme stratejisi ile karşılaştırıldığında diğer meta sezgisel yöntemlerden farklıdır. EAA, IEEE-CEC-2019'daki "100 Basamak Yarışması " test fonksiyonları, literatürde sıklıkla kullanılan dört test fonksiyonu ve bir np-hard kümeleme problemi ile test edilmiştir. Kümeleme problemi için iyi literatürde iyi bilinen dört veri seti kullanılmıştır. Bu veri setleri İris, Wine, Seeds ve Hepatit C Virüsü (HCV) veri setleridir. EAA, bu iyi bilinen test fonksiyonları üzerinde yedi farklı meta sezgisel algoritma ile karşılaştırılmış ve

kümeleme probleminin sonuçları ise X-Ortalamalar algoritması ile karşılaştırılmıştır. Ek olarak, sonuçların anlamlılığını göstermek için Friedman İşaretli Sıra testi ve olay sonrası Wilcoxon Testi yapılmıştır. İyi bilinen test fonksiyonlarının tümünde, EAA ya en iyi sonuçları vermiş ya da diğer karşılaştırılan algoritmalara benzer sonuçlar vermiştir. EAA'nın IEEE-CEC-2019 test fonksiyonlarındaki puanı, düşük iterasyon sayılarıyla bile EAA'nın rakip algoritmalara benzer sonuçlar elde edebildiğini göstermektedir. Sonuçlar, EAA'nın yerel noktalara takılmamak için sağlam bir mekanizmaya sahip olduğunu ve yavaş ama kalıcı bir hızda hareket ettiğini göstermektedir.

DESIGN OF A NEW METAHEURISTIC ALGORITHM BASED ON THE MOVEMENT OF ELECTRICITY IN HIGHLY RESISTANT ENVIRONMENT

SUMMARY

The term "optimization" refers to a group of mathematical ideas and techniques that are applied across various fields to tackle quantitative issues. We all unconsciously experience an optimization process in our daily life. While choosing the route or doing our shopping, we try to choose the most suitable options according to the current situation.

We can solve small-scale optimization problems using our brain power or traditional algorithms, but we use computers to solve large or complex problems.

Many algorithms have been designed to solve computer-aided optimization problems. Despite their simple nature, traditional algorithms are limited by their slow running times and limited applicability. Traditional algorithms are only capable of solving certain types of problems.

Thanks to advancing science and technology, algorithms that work faster but are still problem-dependent have been developed. These algorithms are often called heuristic algorithms and are used to solve a particular type of problem. However, although these algorithms are faster than traditional algorithms, they are not flexible in a variety of problems. Hence, they are incomplete, and the need for improvement and change has arisen. Based on this need, meta-heuristic algorithms have been developed. The word "meta" means "beyond" or "after" in ancient Greek. Therefore, we can think of meta-heuristic algorithms as "post-heuristic" or "post-heuristic" algorithms. While meta-heuristic algorithms work faster than heuristic algorithms, they can work without being dependent on a problem or a particular problem type.

In this thesis, a new nature-inspired metaheuristic optimization algorithm called EAA has been developed. It is based on how electricity moves through areas of high resistance. Like other metaheuristic algorithms, the population is necessary for EAA to start a search. Search agents with a memory and a lifetime make up this population. The EAA also has a unique structure known as negative and positive poles. Suppose we translate these terms into electrical terminology. In that case, we can say that the population is voltage, the search agents are electrons, and the negative and positive pole structures are electrical polarity structures. As a result, electrons at opposing poles tend to migrate in the same direction. The search algorithm used by EAA is based on this migration trend.

EAA was put to the test using four test functions and four different datasets to tackle the NP-hard clustering issue to ascertain the suggested algorithm's efficacy and dependability. The test functions and data sets used in the thesis work are widely used in the literature. The scenarios tested with all the test functions were compared on

seven different algorithms: BSA, DS, MPA, KO, GA, PSO, and HHO algorithms are algorithms with search mechanisms similar to the proposed EAA algorithm. As a result, the proposed EAA algorithm exhibits better global search, local search, and convergence properties than GA, PSO, BSA, and DS algorithms. It yielded similar results to the MPA, KO, and HHO algorithms. Additionally, the EAA method outperforms the GA, PSO, BS, and DS algorithms with a notable convergence rate. Compared to other algorithms, EAA typically finds accurate solutions in various settings. Additionally, statistical results demonstrate that EAA outperforms different compared algorithms and validate the test results. The 100 Digit Challenge test results indicate that, despite its limits, our system is capable of solving challenging tasks. Furthermore, the EAA algorithm outperformed the X-Means approach regarding clustering performance and demonstrated better clustering capability.

Ten times with the corresponding settings, each test case was executed. In 10, 20, and 30 dimensions, the Rosenbrock, Ackley, Griewank, and Rastrigin functions are assessed. We can observe that in every test case, the suggested algorithm, EAA, produces nearly identical results to those of MPA, KO, and HHO. In every one of the ten rounds of six of the twelve instances, the suggested algorithm discovered an exact solution. In addition, in all ten repeating runs in six of the twelve scenarios, the HHO and the KO found a conclusive resolution. MPA, on the other hand, only discovered five precise answers.

For the performance examination of the clustering problem, the Davies-Bouldin index (DB-Index) was used. The number of clusters "k" ranges from two to 10 for all datasets. Thirty-six test scenarios are available in this instance. For each EAA and X-Means method test case, the Davies-Bouldin index was computed and compared.

We can see that EAA has better clustering performance than the X-Means algorithm in all test cases. Also, if we look at the dataset basis, ESA clustered better than the X-Means algorithm in each dataset. The EAA and X-Means algorithms have the same DB-Index in test cases where K is two. This outcome shows that EAA performs the same as the popular clustering algorithm X-Means algorithm in solving the clustering problem. As a result, it has been proven that the proposed algorithm can solve the clustering problem.

DB-Index values are used to determine the optimum "k" number and to perform clustering analysis. Smaller DB-Index values indicate that the clusters created are compact and far apart.

The EAA algorithm obtained a lower DB-Index value for each cluster size in all data sets. The low DB-Index value indicates that the clusters produced by the algorithm have a tighter structure and a higher discreteness than other clusters.

We observe that increasing the time and iteration limit for problem-solving will enable our suggested algorithm to produce better outcomes. The proposed algorithm's architecture causes the search mechanism to operate sluggishly but persistently to look for worthwhile search areas. Our algorithm mainly employs short steps for global search instead of giant steps. To obtain the precise solution, the number of iterations must be increased. Even when faced with complex problems, the suggested algorithm never becomes stuck at local places because of the failure mechanism and initialization strategy. The results demonstrate that the proposed algorithm can approach the exact answer but may occasionally fail because of iteration constraints. Compared to existing

metaheuristic algorithms, the suggested algorithm differs in terms of initialization method, pole search mechanism, and update method of the best solutions.

A constant coefficient was used in our studies for global and local search movements. The algorithm can use a dynamic coefficient to overcome slow motion and late convergence.

The diffusion coefficient of wood was applied in our studies. Future research could compare the effects of various diffusion coefficients on the algorithm. Additionally, different baseline population first initiation techniques can be investigated.

1. GİRİŞ

Optimizasyon kelimesinin Türk Dil Kurumu'ndaki karşılığı "en uygun duruma getirme" olarak verilmiştir [1]. Hepimiz gün içerisinde ister istemez bir optimizasyon işlemi yaparız. Gerek bir yerden bir yere giderken güzergâh seçiminde gerekse günlük basit alışverişlerimizde bile istemeden de olsa mevcut duruma göre bize en uygun olan durumu seçmeye çalışırız.

Basit optimizasyon problemlerini çok rahatlıkla beynimizi kullanarak veya geleneksel algoritmalar yardımıyla çözebilsek dahi büyük veya karmaşık problemlerin çözümlerinde bilgisayarlardan yararlanmaktayız.

Bilgisayar yardımıyla optimizasyon problemlerini çözmek için birçok algoritmalar tasarlanmıştır. Geleneksel algoritmalar yapıca basit olmalarına karşın çalışma zamanı olarak yavaş ve uygulanabilirlik açısından esnek algoritmalar değillerdir. Geleneksel algoritmalar sadece belirli tipte problemleri çözebilmektedirler. Bu tür bir algoritmaya Macar Algoritması örnek olarak verilebilir [2]. İlerleyen bilim ve teknoloji sayesinde ilerleyen zamanlarda çalışma zamanı olarak daha hızlı ancak yine probleme bağlı olan algoritmalar geliştirilmiştir. Bu algoritmalar genel olarak sezgisel algoritmalar olarak bilinmekte ve yine sadece belirlenen bir tipteki problemlerin çözümleri için kullanılmaktadır. Geliştirilen bu algoritmalara örnek olarak A-Star (A*) algoritması gösterilebilir [3,4]. Ancak bu algoritmalar geleneksel algoritmalara göre daha hızlı olsalar dahi problem çeşidi bakımından esnek olmadıklarından dolayı eksik kalmışlar, bu sebeple de gelişim ve değişim ihtiyacı doğmuştur. Bu ihtiyacın bir sonucu olarak meta sezgisel algoritmalar geliştirilmiştir. Meta kelimesi antik Yunancada "ötesi", "sonrası" anlamlarına gelmektedir. Buradan yola çıkarak meta sezgisel algoritmaları sezgisel ötesi veya sezgisel sonrası algoritmalar olarak düşünebiliriz. Sezgisel algoritmaların aksine meta sezgisel algoritmalar hem daha hızlı çalışmakta hem de bir probleme veya bir problem türüne bağlı olmadan çalışabilmektedirler. Günümüzde literatürde mevcut 300'den fazla meta sezgisel algoritma geliştirilmiştir. Meta sezgisel algoritmalara örnek olarak Parçacık Sürü Optimizasyonu (PSO) [5], Genetik Algoritma (GA) [6], Diferansiyel Arama Algoritması (DAA) [7], Geri İzleme Arama

Algoritması (GİA) [8], Deniz Yırtıcısı Algoritması (DYA) [9], K-Ortalama Kümeleme Optimizasyon Algoritması (KOA) [10], Harris Şahinleri Optimizasyon Algoritması (HŞO) [11] gösterilebilir.

Bu tez çalışmasında elektriğin dirençli ortamda hareketini temel alan bir meta sezgisel algoritma geliştirilmiştir. Bu algoritma Elektriksel Arama Algoritması (EAA) olarak adlandırılmıştır. Geliştirilen algoritma dört adet test fonksiyonu üzerinde test edilmiş ve sonuçları yedi adet meta sezgisel algoritma ile kıyaslanmıştır. Ayrıca NP-Zor olarak sınıflandırılmış olan kümeleme problemi üzerinde çalıştırılmış ve elde edilen sonuçlar X-Ortalamalar algoritması ile kıyaslanmıştır. Bunun yanı sıra önerilen algoritma IEEE CEC 2019 yarışmasındaki "100 basamak problemi" üzerinde test edilmiştir.

Tezin bölümleri şu şekilde organize edilmiştir. Giriş bölümünün devamında alt başlıklar olarak Optimizasyon, Meta Sezgisel Algoritmalar ve Kümeleme Problemi detaylı olarak açıklanmıştır. İkinci bölümde literatür taraması verilmiştir. Üçüncü bölümde algoritma tasarımına yer verilmiştir. Dördüncü bölümde ise geliştirilen algoritmanın test ortamı hakkında bilgi verilmiştir. Beşinci bölümde geliştirilen algoritmanın test sonuçları verilmiş ve kısa açıklamalar yapılmıştır. Son olarak altıncı bölümde ise sonuçlar ve önerilere değinilmiştir.

1.1. Optimizasyon ve İlgili Tanımlar

Kelime olarak "en iyi duruma getirme" anlamına sahip olan optimizasyon, temelde sayısal problemleri çözmek için kullanılan matematiksel ilke ve yöntemlerin toplamı olarak ifade edilebilir. Basitçe ifade etmek gerekirse sayısal bir problemin minimum veya maksimum uygunluk değeri, bu değere karşılık gelen değişken veya değişkenlerin bulunması işlemine optimizasyon diyebiliriz.

Sayısal bir probleme örnek olarak bir makinanın maksimum verim için çalışma ve bakım zamanlarının düzenlenmesi, en kısa veya en hızlı yol kullanılarak bir noktadan bir noktaya güzergah çıkarılması, kapsama alanı en geniş ve maliyeti en az olacak şekilde telefon vericilerin yerleştirilmesi gibi problemler gösterilebilir. Örneklerden de anlaşılacağı üzere bir problemin çözümü için bir veya birden fazla amaç belirlenmeli ve bu amaçların matematiksel tanımları yapılmalıdır. Problemin tanımlandığı alana çözüm uzayı denilir. Sonsuz bir çözüm uzayında istenilen bir problemin çözümünün

yapılması imkansızdır. Bu sebeple çözülecek problemde birtakım sınırlamalar olmalıdır. Bu sınırlamalar eşitlik veya eşitsizlikler cinsinden verilmelidir. Bir optimizasyon probleminde problemin türüne göre amaç fonksiyonunda sınırlamalar dahilinde maksimum veya minimum değeri veren noktalar vardır. Bu noktalardan amaç fonksiyonumuza göre en yüksek veya en düşük değeri veren noktalar küresel çözüm, diğer çözümler ise yerel çözümlerdir.

Optimizasyon problemleri tanımlarına göre sınıflandırılırlar. Bir problemde eğer amacımız en yüksek değeri bulmak ise o problem bir maksimizasyon problemidir. Tersi şekilde en düşük değeri bulmamız isteniyorsa o problem minimizasyon problemi olur. Eğer problemde sadece değişken sınırları varsa bu tür problemler sınırlamasız problemler olarak adlandırılırken amaç fonksiyonu sınırlayan başka fonksiyonlar varsa sınırlamalı fonksiyon adını alır. Yine problemin tasarımında kullanılan değişkenlerin türü problemin sınıflandırılmasında rol oynar. Ayrık nesnelerin sınıflandırılması, sıralanması, seçilmesi veya gruplanması bu problemi ayrık veya kesikli optimizasyon problemi yapar. Tersi şekilde değişkenler belirli bir aralıkta sürekli değer alabiliyorsa bu tür problemlere sürekli optimizasyon problemi diyebiliriz. Bir problemde tek bir küresel çözüm var ancak hiç yerel çözüm yoksa bu problem tek modludur. Eğer problem bünyesinde tek bir küresel çözüm ve bir veya birden fazla yerel çözüm barındırıyorsa çok modludur. Aynı şekilde problem bünyesindeki yerel çözüm miktarına bakılmaksızın birden fazla küresel çözüm bulunduruyorsa yine çok modludur denilir.

1.2. Meta Sezgisel Algoritmalar

Bir gerçek dünya optimizasyon problemi karmaşık ve çözümü zor olarak nitelendirilen bir problemdir. Bu tür problemleri geleneksel yöntemlerle çözmeye çalışmak zaman alıcıdır. Ayrıca bir tür problemi çözmek için tasarlanmış bir geleneksel yöntem benzer başka bir problemde çalışmayabilir. Bu sorunu aşmak için yürütme zamanı açısından daha hızlı ancak tam sonucu garanti etmeyen algoritmalar geliştirilmiştir. Bu tür algoritmalara meta sezgisel algoritmalar denilir.

Meta sezgisel algoritmalar genellikle doğadaki olgulardan esinlenerek tasarlanmıştır. Örnek olarak PSO algoritması doğadaki sığırcık kuşu sürülerinin yiyecek arama

davranışından, GA algoritması yine doğadaki en güçlü olanın hayatta kalması ilkesinden yola çıkarak üremedeki gen birleşimlerinden, DAA algoritması ise göç eden canlıların yeni yaşanabilir bir yerleşim yeri bulması davranışından esinlenmiştir.

Meta sezgisel algoritmaları sınıflamak gerekirse esinlenildikleri olgulara göre dört büyük kategoriye bölebiliriz. Bunları biyoloji olgularından esinlenen algoritmalar, sürü zekası olgularından esinlenen algoritmalar, sosyal olgulardan esinlenen algoritmalar ve fiziksel-kimyasal olgulardan esinlenen algoritmalar olarak sınıflayabiliriz [12].

Genel olarak meta sezgisel algoritmaların iki temel arama yapısı bulunmaktadır. Bunlar küresel arama ve yerel aramadır. Küresel arama aşamasında algoritma arama uzayında büyük adımlarla hareket ederken, yerel arama aşamasında daha küçük adımlarla hareket etmektedirler. Her meta sezgisel algoritmanın bu aşamaları farklı hareket formüllerinden oluşmaktadır.

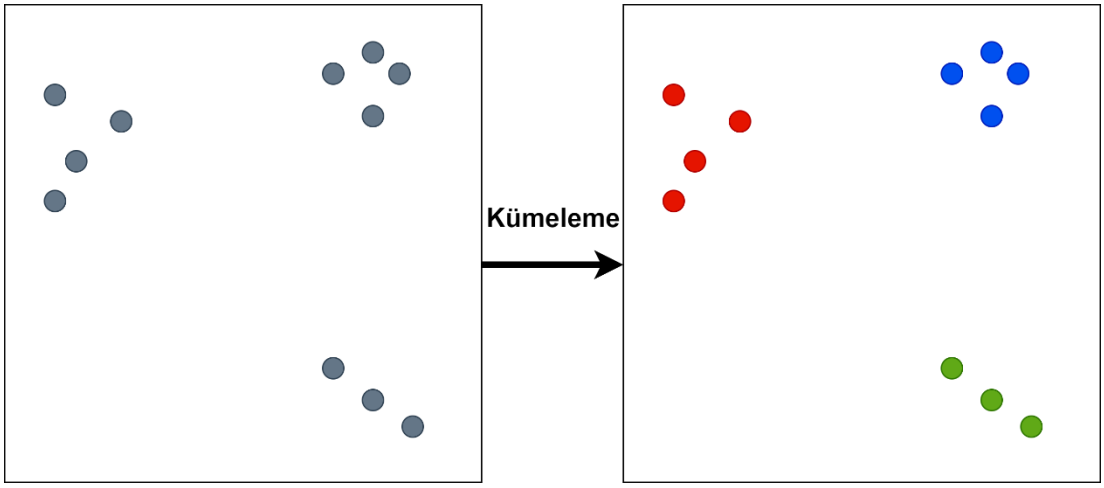
Bütün meta sezgisel algoritmaların yapılarını genelleyecek olursak, meta sezgisel algoritmaların yapısı beş aşamadan oluşur. Bunlar sırasıyla başlangıç çözümü oluşturma, küresel arama, yerel arama, yerel ekstremumlardan kurtulma ve aramayı durdurma mekanizmasıdır. Algoritma çalışırken bu sırayı takip eder eğer ki aramayı durdurma şartını sağlanmamışsa algoritma küresel arama aşamasına dönerek bu döngüyü tekrar başlatır [13]. Bu aşamaları ise şu şekilde açıklayabiliriz. Herhangi bir meta sezgisel algoritma bir problemin çözümüne başlamak için arama uzayında bir noktadan aramaya başlamalıdır. İşte bu aramaya başlanılacak noktanın belirlenme stratejisi meta sezgisel algoritmanın başlangıç çözümü oluşturma mekanizmasıdır. Bu mekanizma birbirlerine benzer veya tamamen farklı olabilir. Küresel arama ve yerel arama mekanizmaları ise meta sezgisel algoritmanın esinlendiği olgunun matematiksel modeline göre belirlenir. Genelde tamamen özgün yapıda olan bu mekanizmalar zaman zaman birbirlerine benzer özellikler gösterebilir. Bu arama mekanizmaları meta sezgisel algoritmanın özgün stratejisi ve temel bileşenidir. Bütün algoritmalar zor problemlerde yerel ekstremum noktalarına takılabilirler. Önemli olan algoritmanın bu noktanın bir yerel ekstremum olduğunun farkına varması ve bu noktalardan uzaklaşarak arama uzayında farklı noktalara yoğunlaşabilmesidir. Bu mekanizma algoritmanın performansını belirleyen kilit mekanizmalardan birisidir. Yine bu

mekanizmada diğer sezgisel algoritmaların mekanizmalarına benzer stratejik özellikler gösterebilir ancak bu yerel ekstremum noktalarından kurtulmak için kullandıkları fonksiyonlar eşsiz özelliktedir. Son olarak bir meta sezgisel algoritmada bulunana aramanın sonlandırılması kriteri algoritmanın ne zaman duracağını belirler. Bu mekanizma algoritmanın arama yapısını çok etkilememekle birlikte algoritmanın çalışma zamanını etkilemektedir. Nitekim yanlış belirlenen bir durdurma mekanizması aramanın çok erken sonlanmasına veya algoritma sonuca ulaşsa dahi gereksiz yere çalışma zamanını kullanmasına sebebiyet verebilir. Durdurma kriteri olarak genelde algoritmalar birbirlerine benzer yapıda stratejiler kullanmaktadırlar. Bu stratejiler genel olarak belirli bir iterasyon kadar çalışma, belirli bir sonuca ulaşınca durma veya belirli bir sonuca belirli bir hata payı dahilinde yaklaşınca durma veya bu stratejilerin bir birleşim kombinasyonu şeklinde seçilmektedir.

Meta sezgisel algoritma işleminde yukarıda bahsedilen yapıları kullanan elemanlara arama ajanı denilmektedir. Örnek olarak PSO algoritmasında arama işlemini yapan ajan kuş iken GA algoritmasında arama işlemini yapan ajan kromozomdur. Bu arama ajanının ismi algoritmanın yapısına göre farklı isimlerle anılsa da temelde arama işleminden sorumlu elemanlardır. Bu arama ajanlarının oluşturduğu topluluk ise popülasyon kavramı olarak adlandırılmaktadır. Bir ajanın yapısı, birbirleri ve popülasyon ile olan iletişimi bir meta sezgisel algoritmayı diğerinden ayıran yapan yapı taşlarından birisidir. Bir meta sezgisel algoritmadaki popülasyon boyutu algoritmanın çalışma hızını ve elde ettiği başarımı çok etkilemektedir. Popülasyon boyutunun çok büyük seçilmesi arama işleminin çok doğru bir sonuç vereceğini anlamına gelmez ancak algoritmanın çok yavaş çalışmasına sebep olabilir. Aynı şekilde popülasyon boyutunun çok küçük olarak seçilmesi algoritmanın çok hızlı çalışmasını sağlarken çok kötü bir başarım elde edileceği anlamına gelmez. Çünkü popülasyon içi arama ajanlarının birbirleri ile olan iletişimi ve popülasyonun geneli ile olan iletişimi arama mekanizmasını iyi veya kötü yönde etkileyebilir. Popülasyon boyutu ele alınan problemin boyutuna ve algoritmanın mekanizmalarının hesaplama karmaşıklıklarına göre deneysel çalışmalar yapılarak belirlenebilir [13].

1.3. Kümeleme Problemi

Veri kümeleme işlemi, tek tip bir veri kümesindeki verilerin belirlenen adet kadar kümelere bölümlenmesi işlemidir. Kümeleme problemindeki amaç, etiketlenmemiş bir veri setini doğru bir şekilde kümelere ayırabilen otomatik bir süreç tasarlamaktır. Veri kümeleme işlemi verileri anlamlı kümelere bölerek organize etme tekniğidir. Kümelenen verilerin kendi kümesi içerisindeki diğer elemanlara olan farklılıkları en az ve diğer kümelerdeki elemanlarla olan farklılıkları ise en çok olacak şekilde kümelere atanmaları sağlanır. Her küme birbirleri ile ilişkili ancak diğer kümelerden farklı veriler içerir. Şekil 1.1'de örnek bir kümeleme işlemi görselleştirilmiştir.



Şekil 1.1. Kümeleme işleminin temsili bir görselidir.

Kümeleme işlemini matematiksel olarak açıklayacak olursak, N-boyutlu bir Öklid uzayında, belirli bir n adet nokta kümesini, bazı benzerlik / benzemezlik ölçüsüne dayalı olarak grupların (veya kümelerin) bir sayısına, örneğin K'ye bölme işlemidir. Bir grup n noktası $\{x_1, x_2, \ldots, x_n\}$ S ile ve K küme ise $C_1, C_2, \ldots C_K$ ile temsil edilsin. Bu durumda veri seti, küme merkezleri ve buna bağlı kısıtlar denklem 1.1'de tanımlanmıştır.

$$C_i \neq \emptyset \qquad i = 1, \ldots, K$$

$$C_i \cap C_j = \emptyset, \qquad i = 1, \ldots, K, \qquad j = 1, \ldots, K, \qquad i \neq j \qquad \bigcup_{i=1}^{K} C_i = S \tag{1.1}$$

Kümeleme probleminin NP-zor bir problem olduğu kanıtlanmıştır. Tüm kümeleme algoritmaları üç kategoriye ayrılabilir. Bunlar sırasıyla hiyerarşik, parçalı ve örtüşen algoritmalardır. Hiyerarşik kümeleme algoritmaları, her veriyi bir küme olarak alarak ele alır. Daha sonra benzer verileri aynı kümeye üzerinde birleştirir. Bu sayede her adımda küme sayısını azaltarak önceden tanımlanmış küme sayısına ulaşılana kadar çalışır. Örtüşen kümeleme algoritmalarını bulanık kümeleme olarak ifade edebiliriz. Bu tür kümeleme stratejisinde tüm veriler, mevcut kümelerin belirli bir üyelik derecesine sahip üyesidir. Yani tüm veriler tüm kümelere belirli bir oranda aittir. Algoritma her veriyi üyelik derecesi en yüksek kümeye atayarak yani verileri sadece bir kümeye ait olacak şekilde yerleştirerek kümeleme işlemini gerçekleştirir. Parçalı kümeleme algoritmaları ise verileri alır ve veri ile küme merkezi arasındaki benzerliğe göre kümelere atar. Parçalı kümeleme yöntemi, uygunluk fonksiyonuna göre bir veri setini birkaç kümeye böler. Uygunluk fonksiyonu, küme oluşumunun doğasını doğrudan etkiler. Etkili bir uygunluk fonksiyonu seçildiğinde, örneğin, farklı kümeler arası mesafenin birbirine en uzak olması veya ilgili küme merkezleri ile ilgili kümeye ait verilerin mesafesinin birbirlerine en yakın olması temelinde bir uygunluk fonksiyonu seçildiğinde, kümeleme işi bir optimizasyon problemine dönüştürülmüş olur [14].

Parçalı kümeleme yöntemi, MacQueen'in k-ortalamalar algoritmasını geliştirmesinden bu yana en popüler yaklaşım olmuştur [15]. Bu yaklaşım ile algoritmalar büyük veri kümelerini kolayca kümeleyebilmektedir. Bu nedenle, çeşitli alanlardan araştırmacılar bu tür algoritmaları kullanmaktadır. Bu alanlar arasında sinyal ve görüntü işleme, kablosuz sensör ağı kapsamı, robotik, web madenciliği, örüntü tanıma, ekonomide tüketici tanımlama ve tıp bilimlerinde hastalık tanımlama yer almaktadır [14].

2. LİTERATÜR TARAMASI

2.1. Parçacık Sürü Optimizasyonu Algoritması

Parçacık Sürü Optimizasyonu (PSO) [5], sürü zekası tabanlı bir optimizasyon algoritmasıdır. Yiyecek kaynağı arayan bir kuş sürüsünün davranışı bu algoritmaya ilham vermiştir. Algoritmanın temel prensibi, sürüdeki kuşların yeni keşfedilen gıda kaynakları hakkında bilgi paylaşmasıdır. Kuşlar yeni keşfedilen kaynaklar arasından en iyi gıda kaynağına doğru hareket eder. En iyi gıda kaynağına doğru göç ederken, kuşlar yeni gıda kaynakları için arama uzayını inceler ve algoritmanın alternatif gıda kaynakları için arama uzayının çoğunu taramasına ve potansiyel olarak yaklaşık en iyi çözümü belirlemesini sağlar. Algoritmada, sürüdeki her bir kuşa parçacık adı verilir. Bu parçacıklar algoritmanın arama yapan ajanlarıdır. Algoritmanın ajan yapısında hafıza yer almaktadır. Bu hafıza her bir ajanın kendi en iyi çözümünü ve bu çözümün değişken değerlerini tutmaktadır. Algoritmada küresel ve yerel arama adımları birbirine bağlı olarak çalışarak ajanın bir sonraki konumunun hesaplanmasını sağlamaktadır. Bu sebeple her bir ajanın kendi hafızasında tuttuğu en iyi konum bilgisi yer değiştirmede kilit bir rol oynamaktadır. Algoritmada yine küresel olarak bütün sürüye ait ve bütün ajanlar tarafından paylaşılan ve güncellenebilen bir hafıza yapısı da vardır. Küresel hafızada tüm ajanlar arasından o ana kadar bulunmuş en iyi çözüm ve bu çözüme ait değişken değerleri tutulmaktadır. Bu yapı yine aynı şekilde bir sonraki konumun hesaplanmasında kilit rol oynamaktadır.

2.2. Genetik Algoritma

Genetik algoritma 1975 yılında Holland tarafından tanıtılmıştır [6]. Genetik algoritma, biyolojik üreme mekanizmasına dayanan evrimsel bir algoritmadır. Genetik algoritmada her bir optimizasyon ajanı bir gen olarak tanımlanır ve bu genlerin toplamı bir popülasyon olarak ifade edilir. Algoritmada arama yapan ajanlara gen ismi verilmiştir. Bu algoritmada iki benzersiz adım vardır. Birincisi, yeni bir gen oluşturmak için iki geni birleştiren çaprazlama aşamasıdır. İkinci aşama ise yeni oluşan genin farklı bölgelerini rastgele değiştiren bir mutasyondur. Bu benzersiz adımların

yardımıyla, rastgele oluşturulan popülasyonlar iyi bir çözüme yakınsayabilir. Bu algoritmanın ajanlarının kendilerine ait bir bellekleri yoktur. Bütün ajanların güncelleme yapabildiği bir hafıza yapısı vardır. Bu hafıza yapısında yine tüm ajanlar arasından o ana kadar bulunmuş en iyi çözüm ve değişken değerleri tutulmaktadır. Ancak algoritmanın orijinalinde bu hafıza yeni bir arama yapma aşamasında kullanılmamaktadır. Zaman içinde genetik algoritma gelişmiştir. Bu gelişmelerle birlikte algoritmanın aramasında kullanılan bazı stratejilerde bu küresel hafızadan yararlanılmıştır. Yeni geliştirilen çaprazlama operatörleri [16-18], mutasyon operatörleri [19] ve gen seçim mekanizmaları [20] genetik algoritmanın gelişimine katkı sağlamış ve literatüre eklenmiştir [21-24].

2.3. Diferansiyel Arama Algoritması

DAA'nın geliştirilmesinin arkasındaki teori, verimli bir geçim kaynağı arayan çeşitli hayvanların mevsimsel göç hareketleridir. Tüm organizmalar bir süper organizma oluşturmak için bir araya gelir ve verimli yaşam alanları aramaya başlamaktadırlar. Algoritmada arama yapan ajanlara süper organizma denilmiştir. Algoritmanın işleyiş şekli şu şekildedir. Süper organizmanın bireyleri, seyahatleri sırasında rastgele seçilen yerlerin geçici kriterlerine uyup uymadıklarını incelemektedirler. Süper organizmanın bireyleri, yolculuk sırasında bir mola için herhangi bir yer uygunsa, hızlı bir şekilde o bölgede yuvalanırlar ve o bölgeden yolculuklarına devam ederler. Bir mola sırasında, süper organizma, organizmalar arasında kalan yerleri keşfetmek için Brownian benzeri rastgele yürüyüşe benzeyen rastgele bir süreç kullanır. Daha sonra tüm süper organizmaların bireyleri yeniden düzenlenerek donörler oluşturulur. Bu donörler, mola yerlerini bulmak için idealdirler. Süper organizmanın içerisinden rastgele seçilen ajanlar, mola alanını etkili bir şekilde bulmak için donörün konumuna doğru göç eder. Alandaki bu değişiklik, süper organizmanın küresel minimuma doğru ilerlemeye devam etmesine izin verir. DAA, ele alınan bir problem için mümkün olan en iyi çözümü doğrudan seçmeyi tercih etmediği için çok modlu optimizasyon problemleri için uygundur. DAA, eldeki göreve bağlı olarak iki ince ayarlı kontrol değişkeni içermektedir. DAA hakkında ayrıntılı bilgi literatür [7] içerisinde bulunabilir.

2.4. Geri İzleme Algoritması

Evrimsel Algoritmalara (EA'lar) dayalı GİA [8], kontrol parametrelerine karşı yüksek hassasiyet ve erken yakınsama gibi EA'lardaki yaygın sorunların üstesinden gelmek için tasarlanmıştır. GİA, geleneksel EA ile aynı beş adımı izlemektedir. Bunlar sırasıyla başlangıç çözümü oluşturma, seçim-I, mutasyon, çaprazlama ve seçim-II aşamalarıdır. Seçim-I sırasında GİA, arama yolunda bir işaretçi olması için geçmiş popülasyonu hesaplamaktadır. Her iterasyonun başında, algoritma geçmiş popülasyonu yeniden tanımlayabilmektedir. Rastgele seçilen bir önceki nesilden bir ajan, değiştirilene kadar bir hafıza gibi davranmaktadır. GİA, EA'dan ve onun geliştirilmiş formlarından farklı bir mutasyon ve geçiş stratejisine sahiptir. Mutasyon aşamasında, deneme popülasyonları üretilirken arama yönü matrisinin genişliğini yönetmek için yalnızca bir parametre kullanılmaktadır. Son deneme popülasyonu iki yol kullanılarak oluşturulabilir. İlk yöntem, bir karışım oranı kullanarak bir denemede mutasyona uğrayacak ajanların sayısını kontrol etmektir. Buna karşılık, ikinci yöntem ise her denemede yalnızca rastgele seçilen bir ajanın mutasyona uğramasını sağlamaktır. GİA'nın ikinci aşamasında, denenen popülasyonda yalnızca yüksek uygunluk değerlerine sahip ajanların kullanıldığı açgözlü seçim prensibi kullanılarak popülasyon güncellenmektedir. Basit oluşumuna rağmen, algoritmanın ikili popülasyon yaklaşımını kullanması, hesaplama için zaman ve bellek tüketebilmektedir.

2.5. Deniz Yırtıcısı Algoritması

Deniz Yırtıcı Algoritmasının (DYA) arkasındaki fikir, okyanus yırtıcılarının avlarını avlarken yaptıkları hareketlerdir. DYA arama ajanları hem Lévy uçuşunu hem de Brownian hareketini kullanmaktadırlar. Bu algoritmanın üç ana aşaması bulunmaktadır. Birinci aşama, avın avcıdan daha hızlı hareket ettiği zamandır. İkinci aşama hem avcının hem de avın neredeyse aynı hızda hareket ettiği zamandır ve üçüncü aşama, avcının avdan daha hızlı hareket ettiği zamandır. Birinci aşamada, küresel arama öne çıkmakta ve Brownian hareketi kullanılmaktadır. Daha sonra, ikinci aşamada hem Lévy uçuşu hareketi hem de Brownian hareketi, küresel ve yerel arama için kullanılmaktadır. Bundan sonra, üçüncü aşamada, yerel arama için yalnızca Lévy

uçuşu kullanılmaktadır. Algoritma, büyük mesafede hareketler için Brownian hareketini, küçük mesafede hareketler için ise Lévy uçuş stratejisini [9] kullanmaktadır.

2.6. K-Ortalama Kümeleme Optimizasyon Algoritması

K-Means Kümeleme Optimizasyon Algoritmasının (KOA) ana fikri, arama uzayını üç kümeye ayırarak daha iyi potansiyel arama uzayları bulmaktır. KOA, diğer optimizasyon algoritmalarından farklı olarak, arama ajanlarının kendisini değil, arama uzayını iyileştirmeye odaklanmaktadır. KOA, başlangıçtaki popülasyonunu yalnızca söz konusu belirli popülasyona atanan arama ajanı tarafından güncellemektedir. Yani her bir ajanın sorumlu olduğu bölge farklıdır. Ayrıca, KOA'daki popülasyon sayısı, en kötü çözümleri eleyerek başlangıçtaki popülasyon sayısı N'den dörde kadar küçültmektedir. Arama ajanları, iterasyonlar boyunca hesaplanan ve daraltılan arama uzayı içindeki küme alanlarında büyük ölçüde arama yapar. Ayrıca ajanlar, belirlenen formülasyonun sınırları dahilinde tanımlı arama alanlarının dışında arama yapabilirler. Her iterasyonda, küme merkezleri yeniden hesaplanır ve küme merkezleri, algoritmanın son aşamalarında küresel en iyi konuma yaklaşır. Arama ajanları arama stratejilerini bir sonraki iterasyonda bir formülasyona ve bir eşik değerine göre seçerler. KOA hakkında daha ayrıntılı bilgi [10] literatürde mevcuttur.

2.7. Harris Şahinleri Optimizasyon Algoritması

Harris Şahinleri Optimizasyon algoritması (HŞO), yırtıcı kuş Harris Şahinlerinin sürüleri ve aile üyeleriyle birlikte olan yiyecek arama davranışlarından esinlenmiştir. Doğada, diğer yırtıcı kuşlar avlarını tek başlarına arar ve avlarlar. Buna karşılık, Harris Şahinleri benzersiz bir işbirlikçi yiyecek arama davranışına sahiptir. Bu algoritmadaki arama ajanları Harris şahinleridir. HŞO'nun iki küresel arama ve iki yerel arama stratejisi vardır. Küresel arama stratejileri, rastgele yer tüneği veya diğer aile üyelerinin yer tünekleri üzerine kuruludur ve bu iki stratejide eşit derecede seçilme şansına sahiptirler. Küresel arama aşamasından yerel arama aşamasına geçiş, iterasyonlar boyunca avın kaçması sonucunda azalan enerjisine dayanmaktadır. Algoritmanın ilk aşamalarında avın kaçış enerjisi yüksek iken arama stratejisi küresel

arama olarak seçilmiştir. Algoritmanın sonraki aşamalarında, avın kaçış enerjisinin azalması nedeniyle arama stratejisi küresel aramadan yerel aramaya doğru değişir. Yerel arama aşaması için HŞO'nun dört stratejisi vardır. Bunlar yumuşak kuşatma, sert kuşatma, aşamalı hızlı dalışlarla yumuşak kuşatma ve aşamalı hızlı dalışlarla sert kuşatma olarak adlandırılmıştır. HHO hakkında daha ayrıntılı bilgi [11] literatürde mevcuttur.

2.8. Meta Sezgisel Algoritmalar ile Kümeleme Probleminin Çözülmesi

Kümeleme problemini bir optimizasyon problemi olarak ele alabildiğimiz için bu problemi meta sezgisel algoritmalar ile çözebilmekteyiz. Kümeleme problemlerinin meta sezgisel yöntemlerle çözümüne yönelik birçok çalışma yapılmıştır. Kümeleme problemlerini çözmek için yaygın olarak kullanılan meta sezgisel algoritmalar, Genetik Algoritmalar (GA'lar) ve parçacık sürüsü optimizasyon algoritması (PSO) gibi sürü tabanlı optimizasyon algoritmalarıdır.

Bir çalışmada [25], denetimsiz kümeleme sırasında oluşturulan kümeleri optimize etmek için genetik algoritma kullanılır. Bu çalışmanın amacı kümelere verileri atamaktır. Bu algoritmanın performansı başlangıç popülasyonuna bağlıdır. Sarkar ve ark. çalışmalarında [26] kümeleme için evrimsel programlama kullanmıştır. Kümeleme problemini çözmek için kullanılan metodoloji ve amaç fonksiyonları bizim kullandığımız yaklaşıma benzemektedir. [27] numaralı referans çalışmasında, kümeleme problemlerini çözmek için başka bir genetik algoritma biçimi önerilmiştir. Ancak, ilk küme merkezlerini kümelenecek verilerden seçtikleri için [25] numaralı referanstan farklı yapıda bir algoritmadır. Başka bir çalışmada [28], kümeleme problemini çözmek için başka bir genetik algoritma varyasyonu kullanılmaktadır. Bu çalışmada, gen dizisinin yapısı, atanan kümelerin bir temsilinden oluşur. Yine başka bir çalışmada [29], genetik algoritmanın kromozom yapısı, küme merkezlerinin gerçek sayı değerleri dizileri olarak temsil edilir. Genetik K-Ortalamalar Algoritması (GKA) [30], atanmış veri noktalarını kodlanmış dizilerden alan ve bunları en yakın küme merkezlerine yeniden atayan tek adımlı K- Ortalamalar operatörüne sahip bir algoritmadır. Hibrit bir k-medoid algoritması (HKA) [31], k-medoid ve yerel arama meta sezgisellerinden oluşur ve algoritmanın meta sezgisel arama kısmı GA ile

hibritleşmiştir. Hızlı Genetik K-Means Algoritması (HGKA) [32], [30] numaralı çalışmadan esinlenmiştir. HGKA ve GKA arasındaki fark, HGKA'nın geçersiz dizilere izin vermesi ve GKA'nın geçersiz dizileri ortadan kaldırmaya çalışmasıdır. Artımlı Genetik K-ortalama Algoritması (AGKA) [33], HGKA'nın [32] bir uzantısıdır. Mutasyon olasılığının küçük olduğu durumlarda IGKA, HGKA'dan daha iyi çalışma performansına sahiptir. AGKA'nın arkasındaki fikir, mutasyon olasılığı küçük olduğunda, objektif değeri hesaplamak ve ağırlık merkezlerini artımlı olarak kümelemektir. Her iki algoritma da aynı sonuca yakınsamaktadır, ancak AGKA, HGKA'dan daha hızlı sonuç vermektedir. Aynı makalede yazarlar, her iki algoritmanın en iyi kısımlarını birleştiren AGKA ve HGKA'nın hibritleştirilmiş bir versiyonu olan Hibrit Genetik K-ortalama Algoritmasını (H-GKA) da tanıtmışlardır. COWCLUS [34] adlı bir algoritma GA ve yokuş tırmanma algoritmalarının bir kombinasyonudur. COWCLUS algoritmasındaki genler, atanmış bir veri noktaları kümesi olarak temsil edilmektedir. COWCLUS, standart bir GA algoritması gibi davranır, ancak son yinelemede, algoritma tepe tırmanma algoritmasıyla yerel bir arama gerçekleştirir. COWCLUS, kümeleri belirlemek için amaç fonksiyonu olarak Varyans Oranı Kriterini kullanmaktadır.

Daha önce bahsedilen makaleler, kümeleme problemini sabit küme sayılarında ele almaktadır. Bununla birlikte, gerçek dünyada, çoğu durumda, küme boyutu ön bilgi olmadan belirlenemez. Bu sorunun üstesinden gelmek için küme boyutunu belirtmeye gerek duymayan algoritmalar geliştirilmiştir.

Örneğin bir çalışmada [35] otomatik kümeleme önerilmiştir. Bu çalışmanın ana fikri, küme büyüklüğünü belirlemede kullanılan küme içi uzaklıktan kümeler arası uzaklığın çıkarılması olarak tanımlanan amaç fonksiyonundaki küme içi mesafeye eklenen bir ağırlıktır. Küçük bir ağırlık değeri, kümelerin çok sayıda ve kompakt olmasına neden olurken, daha yüksek bir ağırlık değeri, az sayıda ve daha geniş kümelerin oluşmasına neden olur. [36] referans numaralı makalede, küme sayısının daha önceden sabit olarak belirlenmemiş olan kümeleme problemleri için gerçek kodlu GA kullanılmıştır. Makalenin temel fikri, her bir genin, gen yapısında gerçek kodlanmış küme merkezleri içermesi ve farklı bir uzunluğa sahip olabilmesidir. Bu, her genin farklı sayıda küme merkezini temsil edebileceği anlamına gelir. Aynı yazarların başka bir çalışmasında

[37] otomatik kümeleme için başka bir girişimde bulunulmuştur. Bu çalışma aynı şekilde gerçek kodlu GA kullanmakta ve gen yapısı yine gerçek kodlu küme merkezleri içermektedir. Her genin uzunluğu, maksimum sayıda küme merkezine sabitlenmiştir. Makalenin kritik unsuru, gen kodlamasında "umursama" anlamında temsil edilen bir simgenin bulunmasıdır; bu, genlerin sabit olmayan sayıda küme merkezini temsil edebileceği anlamına gelir.

Kümeleme problemini çözebilen sürü tabanlı meta-sezgisel algoritmalar da vardır. Popüler sürü tabanlı meta sezgisel algoritmalardan biri PSO'dur.

Bir çalışmada PSO, kümeleme problemini çözmek için kullanılmıştır [38]. Makale, standart PSO algoritmasının kümeleme problemini çözebileceğini göstermektedir. Bu çalışmada başlangıç çözümünü K-ortalamalar algoritmasından alan bir PSO hibriti geliştirilmiştir. Başka bir çalışmada PSO ile kümeleme için farklı bir yaklaşım yapılmıştır [39]. Diğer yaklaşımların aksine bu yaklaşımda bir parçacık yalnızca bir küme merkezini temsil etmekte ve veri noktalarından etkilenmektedir. Her parçacık, çözümün bir kısmına sahiptir. Kümeleme problemini çözmek için tüm parçacıklar birleştirilmeli ve nihai çözüm olarak tutulmalıdır. Hızlandırılmış kaotik parçacık sürü optimizasyonu (HKPSO) algoritması kümeleme problemini çözmek için önerilmiştir [40]. Bu çalışma, hızlı yakınsama için standart PSO hız güncelleme formülasyonu üzerinde kaotik haritaları kullanmaktadır.

Kümeleme problemini çözmek için saf PSO algoritmalarının yanı sıra hibrit algoritmalar da kullanılmaktadır.

PSO ve K-Harmonik Ortalamalar (KHO) algoritmasının hibrit bir versiyonu kümeleme problemini çözmek için önerilmiştir [41]. PSOKHO adlı algoritma sırasıyla PSO ve KHO olarak çalışmaktadır. İlk sekiz yineleme PSO olarak çalışmakta ve sekizinci yinelemeden sonra algoritma dört yineleme boyunca KHM olarak çalışmaktadır. Bu dönüşümlü çalışma işlemi iterasyon limitinin sonuna kadar devam etmektedir. Bir diğer çalışmada PSO ve Kaba Küme (KK) kombinasyonu önerilmiştir [42]. Diğer yaklaşımlardan farklı olarak bu çalışmada veri noktasının her bir boyutu bir küme üyeliğine sahiptir. Bir veri noktası bir küme için belirli bir üyelik miktarına ulaşırsa, o veri noktası o kümeye atanmaktadır. Başka bir çalışmada, bulanık

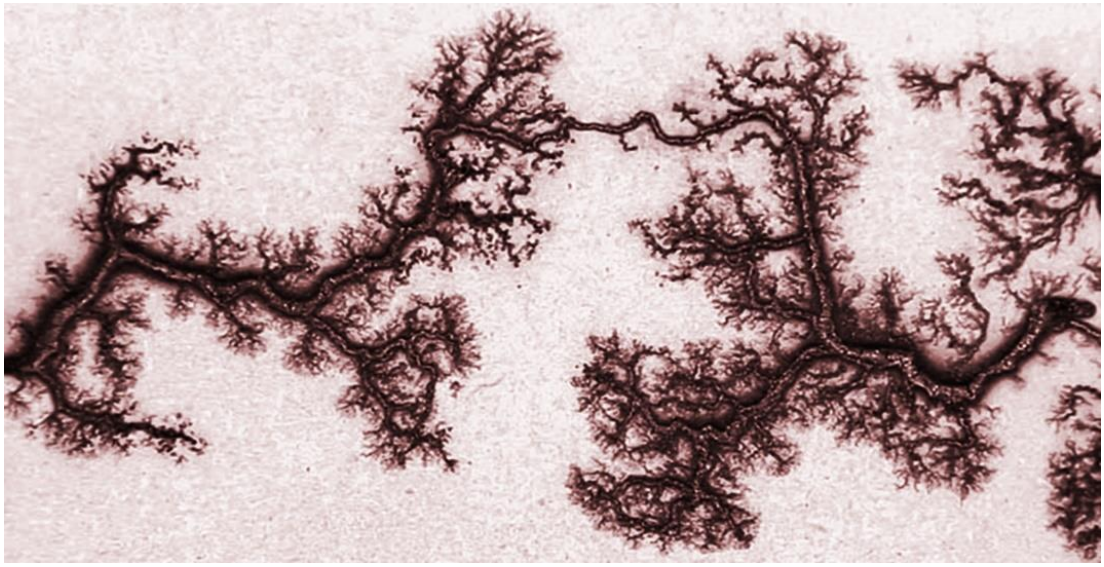
uyarlamalı parçacık sürü optimizasyonu (BAPSO), karınca kolonisi optimizasyonu (KKO) ve K-ortalamalar algoritmasının kombinasyonundan oluşan BAPSO-KKO–K adlı hibrit bir algoritma önerilmiştir [43]. Algoritmanın, temel PSO'dan farkı, tüm parçacıkların kendilerine ait, KKO algoritmasının iz yoğunluğu mekanizması aracılığıyla seçilebilen ve güncellenebilen küresel en iyi çözüm yapısına sahip olmasıdır. Algoritma durdurma kriterine ulaştıktan sonra, nihai küresel en iyi çözüm, K-ortalamalar algoritmasının ilk çözümü olarak kabul edilir ve K- ortalamalar algoritması kümeleme işlemine başlar. Hesaplama sonunda bulunan çözüm küresel en iyi çözümden daha iyi ise algoritma K-Ortalamalar çözümünün çıkış sonucunu dikkate alır. Eğer daha iyi değilse, algoritma PSO-KKO bölümünde elde edilen çıktı sonucunu dikkate alır.

Sonuç olarak, son yirmi yılda bu np-zor problemini çözmek için kapsamlı çalışmalar yapılmıştır. Yukarıda bahsedilen çalışmalardan faydalanılarak bu tez çalışmasında kullanılacak olan kümeleme probleminin amaç fonksiyonu, arama ajanları yapısındaki küme merkezi temsillerinin kodlanma şeması ve kümeleme işleminin performansının değerlendirilme kriterleri belirlenmiştir.

3. ALGORİTMANIN TASARIMI

3.1. Elektriğin Hareketi

Elektrik doğada meydana gelen bir olgudur. Elektrik doğada şimşek, insan sinir sistemi ve bazı hayvanların savunma sistemi gibi birçok sistem içerisinde bulunmaktadır. Elektriğin hareketine elektrik akımı denir ve basitçe atomun valans elektronunun hareketi olarak tanımlanır. Kutuplarında bağlı bir iletkene sahip mevcut bir potansiyel enerji kaynağının, bir elektronu çekecek veya itecek kadar büyük bir enerjiye sahip olduğunu varsayalım. Bu durumda bu kaynaktaki elektronlar, iletkenin atomundaki valans elektronları yardımıyla enerji kaynağının negatif kutbundan pozitif kutbuna doğru geçebilirler. Elektrik doğası gereği, düşük dirençli alanlarda hareket etmeyi sever. Elektriğin hareketi esnasında daha az dirençli alanlar, daha yüksek dirençli alanlardan daha az enerji gerektirdiğinden, elektrik hareket etmek için en az dirençli alanları arar. Daha az dirençli alanları ararken, elektrik Brownian hareketi benzeri hareketler sergileyerek hareket eder ve arkasında Lichtenberg desenlerini bırakır. Önerilen algoritmanın arama modelinin matematiksel altyapısı, yüksek dirençli alanlarda elektriğin hareketi esnasında ürettiği Lichtenberg figürlerinin oluşumuna dayanmaktadır [44]. Şekil 3.1'de, Lichtenberg figürünün bir modelini görebiliriz. Bu şekiller rastgele desenlere sahip olsa da dallanan ağaçlara benzerler. Bu ağaç benzetiminden dolayı bu şekillere Brown Ağaçları da denir.



Şekil 3.1. Ahşap üzerine yüksek voltajlı elektrikle oluşturulan Lichtenberg figürüdür.

Valans elektronunun yüksek dirençli bir alanda nasıl hareket ettiğinin matematiksel ifadesi denklem 3.1'de verilmiştir [44].

$$p(x,t) = \frac{N}{\sqrt{4\pi Dt}} \times exp\left(\frac{-x^2}{4Dt}\right) \tag{3.1}$$

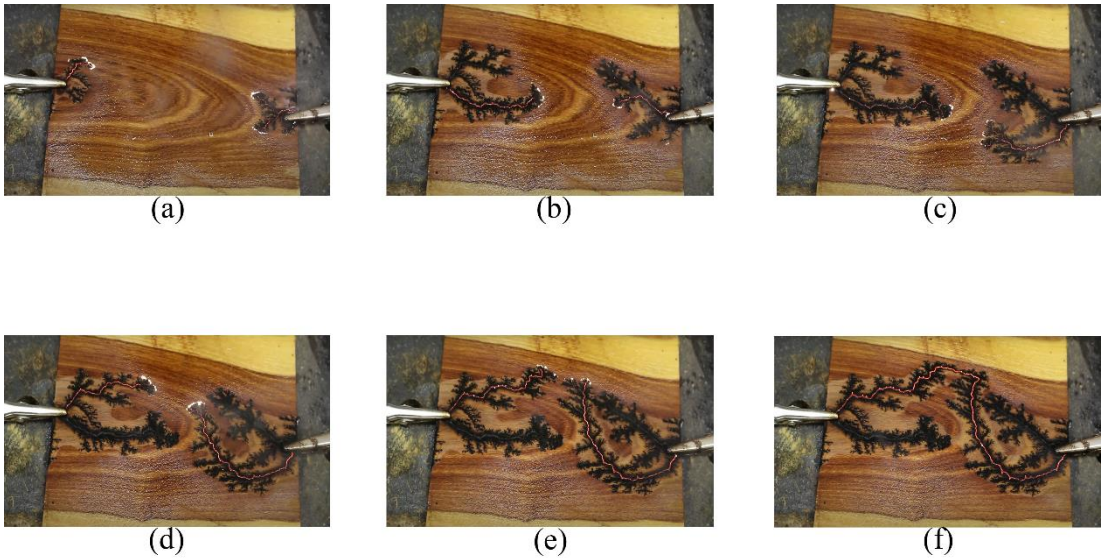
Burada N toplam parçacık sayısını, D ortamın difüzyon katsayısını, t zamanı ve x konumunu temsil etmektedir.

Elektrik, düşük dirençli alanlar aracılığıyla döngüyü tamamlamak için karşı kutba doğru hareket eder. Bu tez çalışmasının ana ilham kaynağı, elektriğin döngüyü tamamlamaya çalışırken ki hareketi ve en kısa yollar yerine, daha düşük dirençli yolları aramasıdır.

Lichtenberg şekillerini incelersek, elektriğin elektriksel döngüyü en kısa yoldan tamamlama şansı olsa bile düşük dirençli alanlara yöneldiğini görmekteyiz. Elektrik yüksek dirençli bölgelerin çevresinde hareket ederek düşük dirençli bölgelere doğru dallanmaktadır. Elektrik kutuplarının başlangıç noktası ahşap yüzeyin kenarıdır ve merkeze doğru düşük dirençli alanları aramaktadır. Ahşap yüzeyi bir arama uzayı olarak düşünürsek, önerilen algoritmanın başlatma şeması rastgele bir nokta olmamalıdır. Meta sezgisel algoritmaların çoğundan farklı olarak algoritma iki farklı ve birbirine en uzak noktadan başlamak zorundadır. Elektrik hareket ederken, tüm

ahşap yüzeyi yavaş ama istikrarlı bir şekilde aramaktadır. Bu istikrar, algoritmamızın erken yakınsama sorununu çözmesini sağlamaktadır. Elektrik hareketi kutuplar üzerinden yapılmıştır.

Önerdiğimiz algoritmada, popülasyonun ilk yarısı arama uzayını karşı kutba doğru aramaktadır. Bu arada, kutbun en iyi konumu ve tüm arama uzayının mevcut en iyi konumu güncellenmektedir. Bundan sonra popülasyonun diğer yarısı en iyi konumların güncellenmiş değerleri ile arama yapmaya başlamaktadır. Arama her kutupta sadece bir ajan ile başlar ve yerel arama ile yeni arama ajanları oluşturulmaktadır. Ajanlar oluşturulurken en iyi konumlar da güncellenmektedir. Bu mekanizma algoritmamıza dinamik bir yapı kazandırmaktadır. Şekil 3.2'de yüksek voltajlı alternatif akım elektriğin ahşap üzerindeki hareket adımları gösterilmiştir.



Şekil 3.2. Yüksek voltajlı elektrik kullanılarak ahşap üzerinde oluşturulan Lichtenberg figürü yanık izleridir.

Şekil 3.2'yi incelersek, Şekil 3.2 (a) 'da elektriğin rastgele kollarla hareket etmeye başladığını görebiliriz. Alternatif akım, elektrik kaynağının negatif ve pozitif kutuplarının sürekli dönüşüm halinde olması nedeniyle sırasıyla her iki kutupta da elektrik hareketinin başlamasına neden olur. Standart bir elektrik şebekesi, ülkeye bağlı olarak 50 veya 60 Hz'de frekansında salınım yapmaktadır. Bu kutupların birbiri arasında değişmesinin her 0,02 veya 0,0167 saniyede bir gerçekleştiği anlamına gelir. Bu değişim işlemi çok hızlı olduğu için insan gözü bu hareketi göremez ve elektrik her

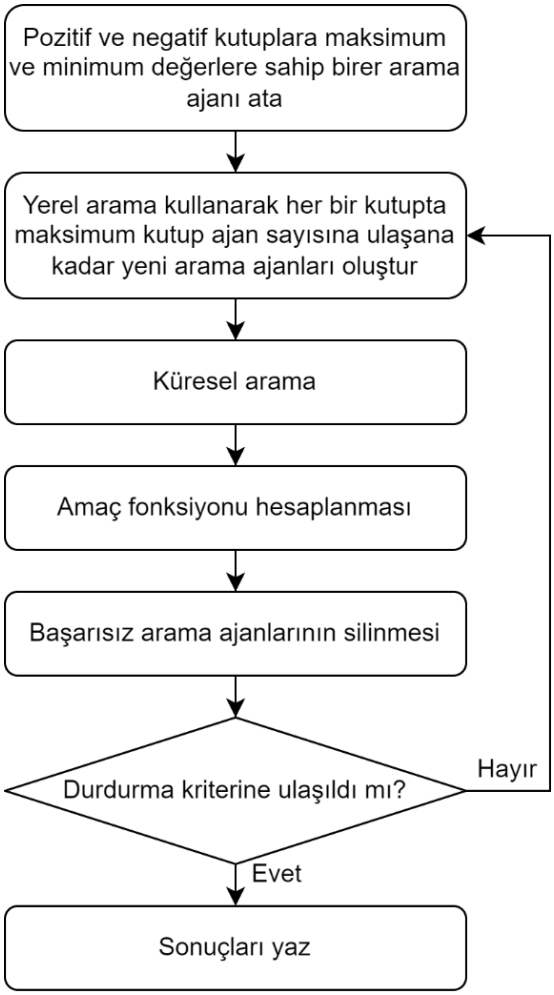
iki kutuptan aynı anda hareket ediyormuş gibi görünebilmektedir. Ancak sırasıyla hareket eder. Yine Şekil 3.2 (a)'ya bakacak olursak, ahşabın sağ tarafında elektrik hareketi iki ana kol ile, sol tarafı ise üç farklı kollara ayrılmış ancak sadece birinden hareket devam etmektedir. Daha sonra Şekil 3.2 (b)'de ahşabın sol tarafında elektriğin ana yolu terk ettiğini ve yeni bir ana yol oluşturulduğunu, sağ tarafında ise ilerlemek için ana yolu seçtiğini görebiliriz. Soldaki kutupta Şekil 3.2 (c) ve Şekil 3.2 (d)'ye baktığımızda, elektrik neredeyse elektriksel döngüyü tamamlıyor gibi görünebilmektedir, ancak yüksek dirençli alanlar nedeniyle sol taraftaki ana yol terk edilmiştir. Şekil 3.2 (b)'de oluşturulan önceki ana yoldan yeni bir yol başlamakta ve ahşabın sağ tarafındaki yol kendini bu yola göre ayarlayarak ve ahşabın sol tarafından gelen yola ulaşmak için yukarı doğru kıvrılmaktadır. Şekil 3.2 (e) ve (f) görüntülerinde, her iki kutbun da ahşapta düşük dirençli alanları aradığını ve birbirine doğru hareket ettiğini görebiliriz. Her iki kutbun başlangıç noktasına bakacak olursak en kısa yol lineer çizgidir fakat elektrik hareket etmek ve döngüyü tamamlamak için daha az enerji tüketmeyi seçmektedir. Arama mekanizmasının bu yapısı, algoritmamızın arama modelini oluşturur. Önerilen algoritmamız bu doğal olayı taklit eder ve yüksek dirençli alanları terk edip enerjiyi daha değerli alanlara aktarırken tüm arama uzayını dallarla aramaya çalışır.

3.2. Geliştirilen Algoritmanın Adımları

Önerilen algoritma, elektriğin yüksek dirençli bir alan veya yüzey üzerindeki hareketi olayını taklit etmektedir. Elektrik gibi, algoritmamızın da negatif ve pozitif olmak üzere iki kutbu vardır. Algoritma bu kutuplarda arama işlemine başlar. Negatif kutuptaki arama ajanı, arama uzayı boyutlarının minimum değerlerini referans alarak bir arama başlatır. Pozitif kutup için arama ajanı, arama uzayı boyutlarının maksimum değerleri ile bir arama başlatır. Arama, sırayla her iki kutuptaki bir ajanla başlar ve ajan sayısı arama işlemi ile artarken, zıt kutuplardaki ajanlar birbirlerini çeker. Her kutup için oluşabilecek maksimum ajan sayısı eşittir ve algoritmanın başında önceden tanımlanmıştır. Ajanlar, hem zıt kutuptaki ajanın en iyisine hem de karşı kutupta veya aynı kutupta olabilen küresel en iyi ajana doğru çekilir. Şekil 3.1'de gösterildiği gibi, çok yüksek dirençle karşılaşılan birçok dal, zamanından önce sonlandırılır ve karşı

kutba hareket etmeye devam etmez. Bu dalların aramada başarısız olduğunu veya yerel ekstremum noktalarda sıkışıp kaldığını söyleyebiliriz. Bu mekanizmayı taklit etmek için ajanlara bir başarısızlık değişkeni eklenmiştir. Bu değişken, bir ajan mevcut problem için daha iyi bir çözüm sağlayamadığında artacak şekilde tanımlanmıştır. Bu, ajanların kendilerine ait en iyi çözümü hatırlayabilen bir hafıza özelliğine sahip olduğu anlamına gelmektedir.

Bir ajan başarısızlık değişkeninin sınırını geçtikten sonra, algoritma ajanı siler ve silinen ajanın atanmış olduğu kutbunda yeni bir ajan oluşturulur. Bu mekanizma, yerel ekstremum noktasında takılma probleminin çözümüdür. Bunun bir sonucu olarak, popülasyon daha iyi bir çeşitlilik kazanmaktadır. Önerilen algoritmanın akış şeması Şekil 3.3'te verilmiştir.



Şekil 3.3. Önerilen algoritmanın akış şemasıdır.

Önerilen algoritma küresel arama için denklem 3.2'yi ve yerel arama için denklem 3.3'ü kullanmaktadır.

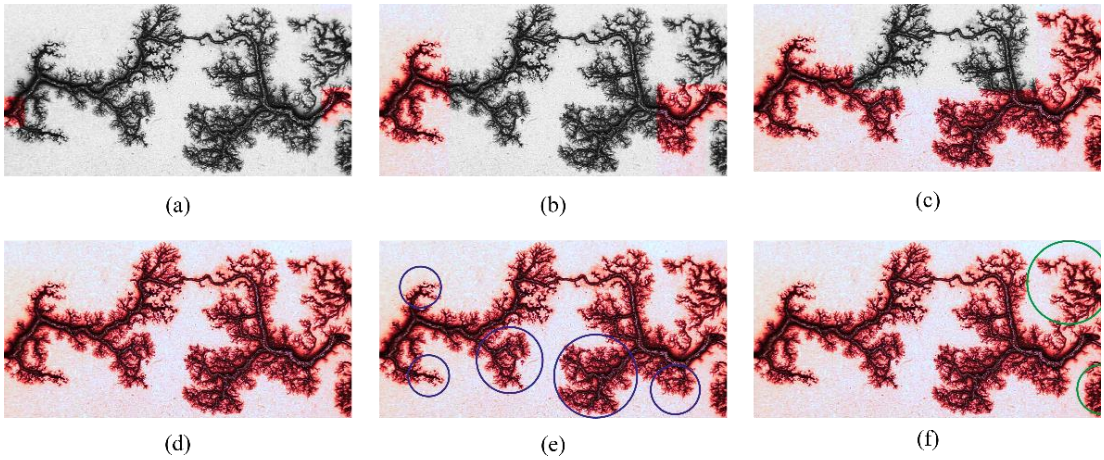
$$x_i(t+1) = 0.7 \times \left(r_1 \times x_i + (1 - r_1) \times x_{eniyi_i}\right) + 0.3 \times \left(r_2 \times x_i + (1 - r_2) \times x_{kutup_i}\right) \tag{3.2}$$

$$x_i(t+1) = x_i + \frac{0.3 \times \Delta x_{eniyi} + 0.7 \times \Delta x_{kutup}}{\sqrt{4\pi Dt}} \times exp\left(\frac{-1}{4Dt}\right) \tag{3.3}$$

Denklem 3.3'te x_i değişkeni, *i* ajanının değerini belirtir. r_1 ve r_2 değişkenleri [0.0,1.0] aralığında oluşturulan rastgele sayılardır. Son olarak, x_{eniyi_i} ve x_{kutup_i} değişkenleri, sırasıyla arama uzayındaki en iyi çözüme sahip ajanın ve o iterasyondaki zıt kutbun en iyi ajanının değerlerini ifade eder. Denklem 3.3 zıt kutupların çekim gücünü taklit etmektedir. x_{eniyi_i} ve x_{kutup_i} değişkenlerinin değerleri her iterasyonda değişir ve bu değerler aktif iterasyona özeldir. Bu değerlerdeki değişiklik, kutup değişimi meydana geldiğinde ve kutup içinde yeni bir ana yol oluşturulduğunda yapılan düzenlemeyi taklit etmektedir. 0.7 ve 0.3 sabit değerleri, algoritma geliştirilirken çok sayıda deneyle seçilen çekim katsayılarıdır. Küresel aramanın küçük ve büyük adımlarla gerçekleşmesini istediğimiz için, zıt kutup çekimini 0,3 ve en iyi çözüm çekimini 0,7 ile sınırlıyoruz. Çekim adımlarının iki olasılığı vardır: arama ajanına göre en iyi çözümün karşı kutupta ve en iyi çözümün aynı kutupta oluşudur. En iyi çözüm arama ajanına göre karşı kutupta oluştuğunda, arama ajanı büyük adımlarla karşı kutba doğru hareket eder. En iyi çözüm arama ajanına göre aynı kutupta meydana gelirse, arama ajanı küçük adımlarla karşı kutba doğru hareket eder. Her iki durumda da ajanlar her zaman zıt kutuplara çekilir ve zıt kutuplara doğru hareket etme eğilimindedir.

Denklem 3.3, denklem 3.2 ile aynı olarak, x_i değişkeni, *i* ajanının geçerli değerini belirtir. *D* ve *t* değişkenleri, sırasıyla difüzyon katsayısını ve yineleme sayısını temsil eder. Son olarak, Δx_{eniyi} ve Δx_{kutup} değişkenleri, sırasıyla mevcut ajanın konumu ile arama uzayının en iyi ajanı ve karşı kutbun en iyi ajanı arasındaki mesafeyi gösterir. Denklem 3.2'nin yardımıyla önerilen algoritma küresel en iyi çözüme ulaşmak için büyük adımlarla ilerler ve denklem 3.3 ile algoritma karşı kutbun en iyi çözümüne yaklaşmak için küçük adımlarla ilerler. EAA'nın tüm özellikleri ve aşamaları Şekil

3.4'te görselleştirilmiştir. Şekil 3.4'ün (a) bölümünde, ESA'nın başlatılması gösterilmektedir. (b) ve (c) bölümlerinde genel ve yerel arama başlatılır. Son olarak (d) kısmında arama işlemi tamamlanır. Şekil 3.4 (e) bize algoritmanın başarısız olan dallarını gösterir. (f) altbölümünde, başarısız dalların iterasyonlar yoluyla silinmesinden kaynaklanan rastgele yeni dalların ve dalların oluşumunu görebiliriz.



Şekil 3.4. Önerilen algoritmanın adımları ve özelliklerinin temsilidir.

Denklem 3.3'te elektriğin dallanma yapısını taklit etmesini görmekteyiz. $0.3 \times \Delta x_{eniyi} + 0.7 \times \Delta x_{kutup}$ ifadesi yönü ve $1/\sqrt{4\pi Dt} \times exp((-1)/4Dt)$, adımın büyüklüğünü belirler. Algoritmanın yerel arama aşaması, ajanın küresel arama aşamasından sonraki elektriğin dallanması hareketinden oluşur. 0,7 ve 0,3 sabit değerleri, denklem 3.2'deki sebeplerden dolayı dallanmayı büyük ve küçük adımlarla sınırlamak için seçilmiştir. Ayrıca bu yerel arama aşamasının iki olasılığı vardır: En iyi çözüm arama ajanına göre karşı kutupta oluştuğunda, arama ajanı büyük adımlarla karşı kutba doğru dallanır ve en iyi çözüm arama ajanına göre aynı kutupta bulunuyorsa, arama aracısı küçük adımlarla karşı kutba doğru dallanır.

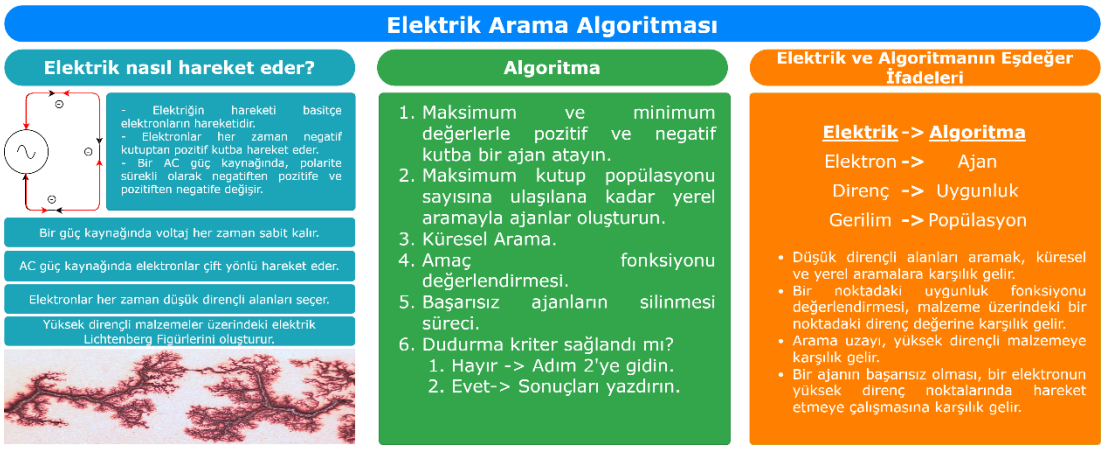
0,7 ve 0,3 sabit değerleri çözülmek istenilen problem için ayarlanabilir, ancak bu katsayıların değiştirilmesini tavsiye edilmemektedir. Çünkü bu katsayıları en iyiye ayarlamak başlı başına ayrı bir optimizasyon problemidir.

Başarısızlık limiti, olaylarda ana yolların terk edilmesini ve enerjinin farklı bir alana aktarılmasını taklit etmektedir. Bu yapı, algoritmanın yerel ekstremum noktalarına yakalanmamasına yardımcı olur. Başarısızlık limiti, maksimum iterasyon sayısının

‰3'ü olarak ile belirlenmiştir. Ayrıca, başarısızlık limiti ayarlanabilir. Başarısızlık sınırı, ajanların yerel aramaya mı yoksa küresel aramaya mı zaman ayıracağını belirler. Daha fazla küresel arama için başarısızlık limiti azaltılabilir ve daha fazla yerel arama için başarısızlık limiti arttırılabilir. Önerilen algoritmanın sözde kodu şekil 3.5'te ve algoritmanın grafiksel olarak özeti ise şekil 3.6'da verilmiştir.

```
Parametreleri tanımla:
    pop_boyutu
    maks_iterasyon
    Difüzyon_Katsayısı
    silme_limiti
Arama uzayının maksimum ve minimum değerleri ile bir ajanı pozitif ve negatif kutba ata
while t < maks_iterasyon do
    Maksimum kutup popülasyonu sayısına ulaşılana kadar yerel arama ile ajanlar oluştur
    Küresel arama
    Her ajan için uygunluk değerini hesaplayın
    Güncelle:
        ajanların başarısızlık sayısı
        en iyi ajan çözümü
        her kutuptaki en iyi ajan çözümü
    başarısızlık sayısı ≥ silme_limiti olan ajanları sil
    t = t + 1
end while
Bitir:
    En iyi çözüm değeri
    En iyi çözüm değişkenleri
```

Şekil 3.5. Önerilen algoritmanın sözde kodudur.



Şekil 3.6. Önerilen algoritmanın grafiksel özetidir.

3.3. Meta Sezgisel Algoritma ile Kümeleme Probleminin Çözümü

Kümeleme işlemine ait tanımlar ve kısıtlamalar denklem 1.1'de verilmiştir. Buradan yola çıkarak kümeleme probleminde optimize edeceğimiz değişken küme merkezleridir.

Algoritma başlangıçta çalışma şekline uygun olarak minimum ve maksimum noktalardan başlayarak rastgele küme merkezleri atamaktadır daha sonra bu küme merkezlerine en yakın olan elemanlar ilgili kümeye atanmaktadır. Problemin çözümünde algoritmanın adımları genel olarak aynıdır ancak uygunluk değeri hesaplandıktan sonra her kümeye ait olan elemanlar hesaplanmakta ve o kümeye ait olan elemanların orta noktası o kümenin yeni merkezi olmaktadır. Problemimiz genel olarak bir minimizasyon problemidir. Problem, her bir küme merkezine ait noktalar arasındaki Öklid uzaklık farkının toplamını en aza indirmeyi amaçlamaktadır. N-boyutlu Öklid uzayında koordinatların verdiği x ve y noktaları için denklem 3.4'de uzaklık formülü verilmiştir.

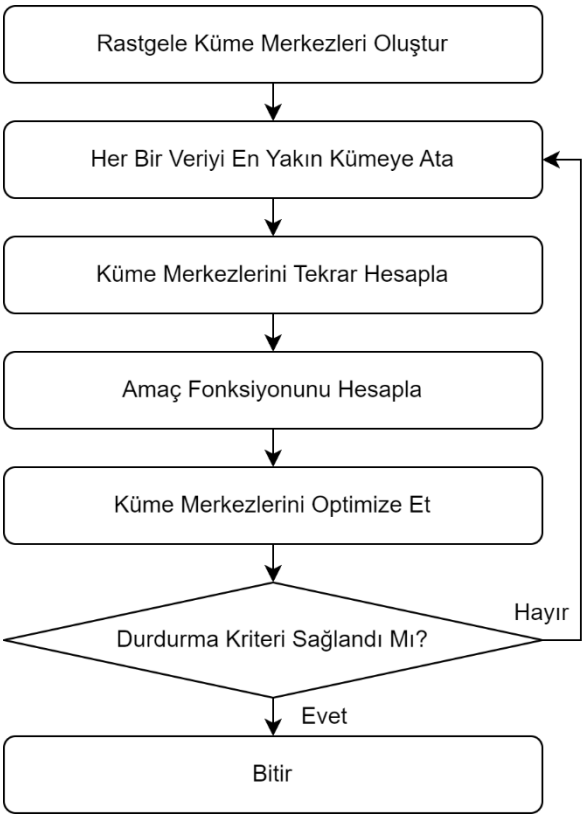
$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.4}$$

Veri setindeki her nokta bir kümeye atandığından, amaç fonksiyonu denklem 3.5'teki gibi tanımlanabilir.

$$f = \sum_{i=1}^{n} d(x_i - z_j) \tag{3.5}$$

x_i, z_j küme merkezi ile k_j kümesine atanmış bir veridir, n, k_j kümesine atanmış olan veri sayısıdır ve j, 1 ile önceden tanımlanmış küme sayısı olan K arasında tanımlanmaktadır. Kümeleme problemini çözmedeki amaç fonksiyonumuz f fonksiyonunu en aza indirmektir.

Bir amaç fonksiyonumuz olduğu için kümeleme problemini kolaylıkla meta sezgisel algoritmalar kullanarak çözüm üretebiliriz. Bu tez çalışmasında kullanılan yaklaşımda, küme merkezleri optimize edilmesi gereken değişkenlerdir. İlk olarak, K kümeleri için küme merkezleri rastgele oluşturulmaktadır. Bundan sonra, tüm veriler en yakın küme merkezi ile en yakın kümeye atanmaktadır. Böylece her verinin atanmış olduğu bir kümeye bulunabilmektedir. Her veriyi en yakın kümeye atadıktan sonra, algoritmanın işini kolaylaştırmak için kümeye atanan noktaların basitçe ortalaması alınarak yeni bir küme merkezi bulunur. Daha sonra yeni küme merkezleri ile f amaç fonksiyonu hesaplanır. Optimizasyon süreci, önceden tanımlanmış durdurma kriterleri karşılanana kadar yeni küme merkezleriyle devam eder. Sürecin akış şeması Şekil 3.7'de verilmiştir.



Şekil 3.7. Kümeleme problemi algoritmasının akış şemasıdır.

4. TEST ORTAMI

Önerilen algoritmayı literatürde çok sık kullanılan dört adet test fonksiyonu üzerinde test ettik. Bu fonksiyonlar çok modlu, tek modlu, ayrılabilir ve ayrılamaz özelliklere sahiptir. Çok modlu fonksiyonların ikiden fazla yerel ekstremum noktası vardır ve yerel ekstremum noktalarında yakalanma olasılığı nedeniyle tek modlu fonksiyonlara kıyasla çözülmesi zor fonksiyonlardır. Ayrılamaz olarak nitelendirilen bir fonksiyonda optimize edilecek olan değişken değerler birbirlerine bağımlıdır. Ayrılabilir olarak nitelendirilen bir fonksiyonda optimize edilecek değişkenler bağımsız değişkenlerdir. Bir değişkendeki değişim diğer bir değişkenin durumunda değişikliğe sebep olmaz. Ayrılamaz olarak nitelendirilen bir fonksiyonu çözmek, ayrılabilir nitelikli bir fonksiyonu çözmekten daha zordur çünkü fonksiyonun her değişkeni diğer değişkenlere bağlıdır. Algoritmayı test etmek için kullanılan test fonksiyonları denklemler 4.1, 4.2, 4.3 ve 4.4'te verilmiştir. Bu denklemler veriliş sırasına göre sırasıyla Rosenbrock, Ackley, Griewank ve Rastrigin fonksiyonları olarak adlandırılmaktadır. Tüm kıyaslama problemleri minimizasyon problemleridir; bu nedenle minimum sonuçlar daha iyidir.

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{4.1}$$

$$f(x) = -20 \times exp\left(-0.2 \times \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} \cos(2\pi x_i)\right) + 20 + \exp(1) \tag{4.2}$$

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{4.3}$$

$$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)] \tag{4.4}$$

Bu test fonksiyonları denklemlerinde d değişkeni problemin boyutunu x değişkeni ise optimize edilecek olan değişkeni ifade etmektedir. Rastrigin, Ackley ve Griewank denklemleri çok modlu, Rosenbrock denklemi ise tek modlu bir özelliğe sahiptir. Ayrıca Ackley, Griewank ve Rosenbrock denklemleri ayrılamaz, Rastrigin denklemi ayrılabilir özelliğe sahiptir. Tablo 4.1 üzerinde bu test fonksiyonlarına ait tanım aralıkları ve küresel minimum değerleri verilmiştir.

Tablo 4.1. Test fonksiyonlarının tanım aralıklarını ve özelliklerini gösterir tablodur.

Test Fonksiyonu	Tanım Aralığı	Küresel Minimum Noktası
Rosenbrock	[-30, 30]	$f(x^*) = 0\ x^* = (1, \dots, 1)$
Ackley	[-32.768, 32.768]	$f(x^*) = 0\ x^* = (0, \dots, 0)$
Griewank	[-600, 600]	$f(x^*) = 0\ x^* = (0, \dots, 0)$
Rastrigin	[-5.12, 5.12]	$f(x^*) = 0\ x^* = (0, \dots, 0)$

Bu tez çalışmasında önerilen algoritmayı sırasıyla 10, 20 ve 30 boyut değerleri ile bu dört test fonksiyonu üzerinde PSO, GA, DAA, GİA, DYA, KOA, HŞO algoritmaları ile karşılaştırılmıştır. Karşılaştırılan algoritmaların kontrol parametreleri tablo 4.2'de verilmiştir. Tüm test durumlarında popülasyon sınırı 100 olarak ayarlanmıştır. Ayrıca, adil değerlendirme için iterasyon sınırı 10, 20 ve 30 boyutları için sırasıyla 500, 1000 ve 1500 olarak ayarlanmıştır.

Tablo 4.2. Algoritmaların kontrol parametreleridir.

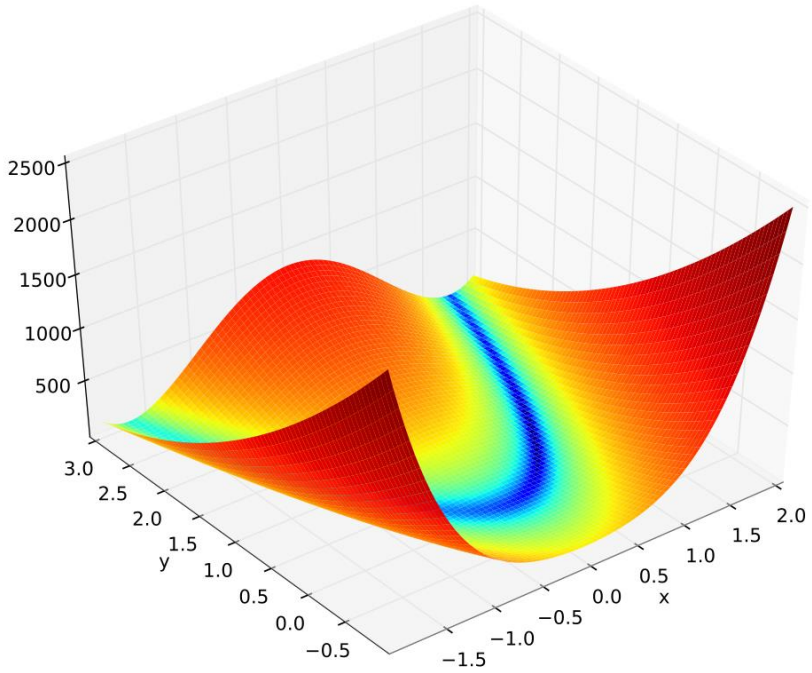
Algoritma	Kontrol Parametreleri
EAA	D=1,49 (Odunun difüzyon katsayısı), Başarısızlık Sınırı=15
PSO	C_1=1,49, C_2=1,49
GA	Çaprazlama Oranı=0,75, Mutasyon Oranı=0,1

Tablo 4.2. (Devamı) Algoritmaların kontrol parametreleridir.

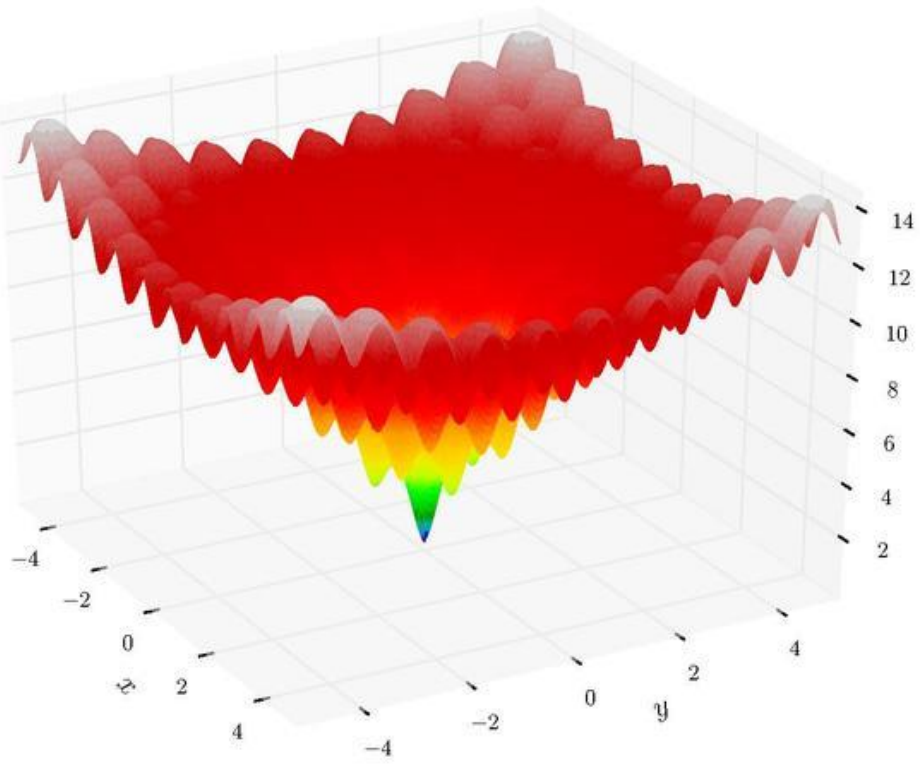
Algoritma	Kontrol Parametreleri
DAA	Metot=Örten DAA
GİA	Bütün parametreler varsayılan değerlerde kullanılmıştır.
DYA	Bütün parametreler varsayılan değerlerde kullanılmıştır.
KOA	Bütün parametreler varsayılan değerlerde kullanılmıştır.
HŞO	Bütün parametreler varsayılan değerlerde kullanılmıştır.

Tablo 4.2'de verilen kontrol parametreleri değerleri her bir algoritma için literatürde yapılan çalışmalardan faydalanılarak en iyi sonuç elde edilen parametre değerleri olacak şekilde seçilmiştir.

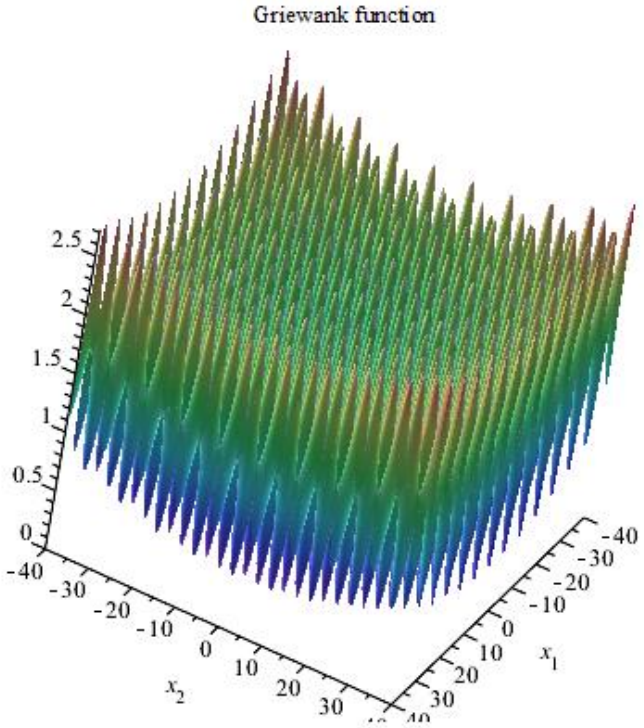
Rosenbrock, Ackley, Griewank ve Rastrigin fonksiyonlarının 2 boyut için çizilen temsil grafikleri sırasıyla şekiller 4.1, 4.2, 4.3 ve 4.4'te verilmiştir [45-48].



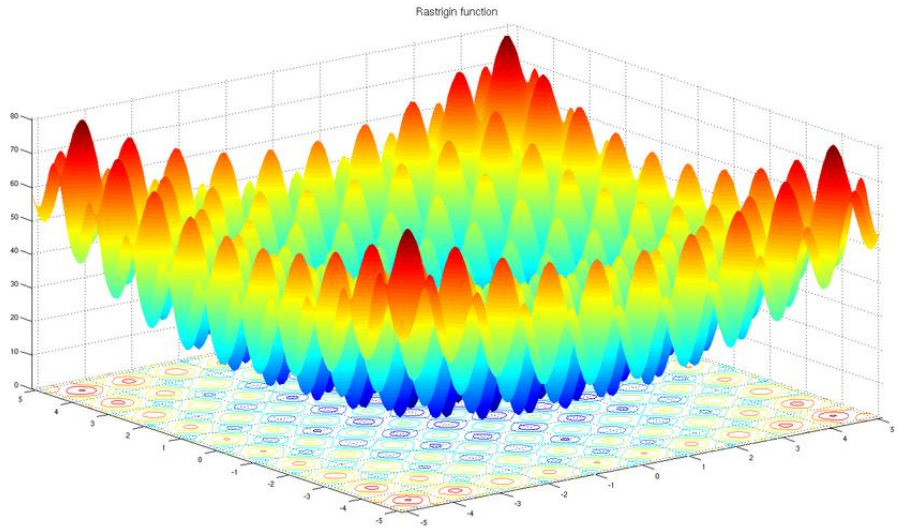
Şekil 4.1. Rosenbrock fonksiyonunun 2 boyutlu temsilidir.



Şekil 4.2. Ackley fonksiyonunun 2 boyutlu temsilidir.



Şekil 4.3. Griewank fonksiyonunun 2 boyutlu temsilidir.



Şekil 4.4. Rastrigin fonksiyonunun 2 boyutlu temsilidir.

Bu tez çalışmasında önerilen algoritma ve popüler kümeleme algoritması olan X-Ortalamalar algoritması kümeleme problemini çözme performansı açısından karşılaştırılmıştır. Karşılaştırma için, UCI Makine Öğrenimi Deposu tarafından sağlanan ve gerçek hayattaki verilerden elde edilmiş veri kümeleri olan Iris [49], Wine [50], Seeds [51] ve HCV [52] veri kümeleri kullanılmıştır. Hem önerilen algoritma hem de X-Ortalamalar algoritması için maksimum iterasyon 150 ve en yüksek küme sayısı 10 olarak belirlenmiştir. Ayrıca önerilen algoritmanın ajan sayısı 200, hata limiti 5 olarak belirlenmiştir. Bu veri setlerinin özellikleri tablo 4.3'te verilmiştir. Davies-Bouldin indeksi (DB-Index) [53], kümeleme problemi performansını değerlendirmek için kullanılmıştır. DB-Index, kümelerin kompaktlığını ve ayrıklığını ölçen bir değerlendirme indeksidir. DB-Index değeri ne kadar küçükse bulunan küme merkezleri için kümeleme performansının daha iyi olduğunu göstermektedir.

Tablo 4.3. Kümelemede kullanılan veri setlerinin özellikleridir.

Veri Seti	Veri Sayısı	Özellik Sayısı
Iris	150	4
Wine	178	13

Tablo 4.3. (Devamı) Kümelemede kullanılan veri setlerinin özellikleridir.

Veri Seti	Veri Sayısı	Özellik Sayısı
Seeds	210	7
HCV	615	14

Bunun yanı sıra bu tez çalışmasında önerilen algoritma IEEE CEC 2019'un "100 Basamak Yarışması" üzerinde test edilmiştir. IEEE CEC 2019'un "100 Basamak Yarışması" hakkında daha detaylı bilgiler [54] numaralı referansta bulunabilir. Yarışma on test fonksiyonundan oluşmaktadır; yedi tanesi eksen olarak kaydırılmış ve döndürülmüştür. Diğer üç problem ise yeni ve zorlu bir yapıdadır. Yarışmadaki amaç, fonksiyonun doğruluğunu zaman sınırı olmadan 10 haneye kadar hesaplamaktır. IEEE CEC 2019 fonksiyonlarının sınırları ve boyut değerleri tablo 4.4'te verilmiştir. Önerilen algoritmanın kontrol parametreleri popülasyon=30, Difüzyon Katsayısı=1.49, İterasyon Sayısı=1E+06 olarak ayarlanmıştır.

Yarışma temelinde Ezop'un kaplumbağa ve tavşan hikayesi yer almaktadır. Hikayeye göre tavşan ile kaplumbağa yarışmış ancak tavşan avantajlarının rehavetine kapılarak bitiş çizgisine yakın bir yerde uyuya kalmıştır. Bütün dezavantajlarına rağmen kaplumbağa ısrarla yarışmaya devam etmiş ve bitiş çizgisini tavşandan önce geçmiştir. Bu hikayeden esinlenen yarışmada amaç algoritmaların hızlarının mı tam doğru sonuca olanak verdiği yoksa ısrarla uzun soluklarla arama yapan algoritmaların mı eninde sonunda tam doğru sonucu bulduğu test edilmektedir.

Yarışmanın içeriğinde her algoritma 100 kez çalıştırılmakta ve en iyi 50 çalıştırmadaki bulunan doğru basamak sayısına göre bir puan elde edilmektedir. Bütün problemlerin optimum değeri 1 olarak ayarlanmıştır ve tam doğru sonuç 1.000000000 olarak belirlenmiştir. Her bir fonksiyonda soldan başlayarak doğru bilinen her bir basamak o çalıştırma için en son puanı belirleyecek kriteri oluşturur. Her bir fonksiyon için en fazla 10 puan alınabilmektedir. Bütün fonksiyonlardan elde edilen puanlar toplanılarak 100 üzerinden bir puan elde edilmekte ve puanlara göre kıyaslama yapılmaktadır.

Tablo 4.4. IEEE CEC 2019 100 Basamak Yarışması fonksiyonlarının boyut ve tanımlı olduğu aralık değerleridir.

No	Fonksiyonlar	Boyut	Arama Uzayı
1	Storn's Chebyshev polinom uydurma problemi	9	(−8192,8192)
2	Ters Hilbert matris problemi	16	(−16.384,16.384)
3	Lennard-Jones minimum enerji kümesi	18	(−4,4)
4	Rastrigin fonksiyonu	10	(−100,100)
5	Griewank fonksiyonu	10	(−100,100)
6	Weierstrass fonksiyonu	10	(−100,100)
7	Değiştirilmiş Schwefel fonksiyonu	10	(−100,100)
8	Genişletilmiş Schaffer F6 fonksiyonu	10	(−100,100)
9	Happy Cat fonksiyonu	10	(−100,100)
10	Ackley fonksiyonu	10	(−100,100)

4.1. IEEE CEC 2019 "100 Basamak Yarışması" Fonksiyonları Tanımları ve Grafikleri

Tüm test fonksiyonları, aşağıdaki denklem 4.5'teki gibi tanımlanmış minimizasyon problemleridir:

$$\text{Min } f(x), \qquad x = [x_1, x_2, \dots, x_D]^T \tag{4.5}$$

Burada D boyutları ifade etmektedir. Aşağıda verilmiş olan tanımlar ise bu on test fonksiyonunun değiştirilmiş hallerinde olan tanımlar içerisinde kullanılmıştır. Kaydırma matrisi o_{i1}, D adet elemandan oluşmuş $[-80,80]^D$ aralığında rastgele dağıtılan

kaydırılmış global optimum için kullanılmıştır her bir problem için ayrıca tanımlanmıştır.

Tüm test fonksiyonları ölçeklenebilirdir. 4 ile 10 arasındaki fonksiyonlar herhangi bir D>1 durumu için geçerlidir, 1 ile 3 arasındaki fonksiyonlar sırasıyla D = 2n, n^2 ve 3n için geçerlidir, burada n = 1, 2... şeklinde tanımlanmış bir tam sayıdır. Ayrıca, fonksiyonlar 4–10 o matrisine göre kaydırılmıştır ve M matrisine göre döndürülmüş, ancak 1–3 fonksiyonları kaydırma döndürme işlemlerine tabi tutulmamıştır.

Kolaylık sağlamak için, aynı arama uzayları 4 ile 10 arasındaki fonksiyonlar için tanımlanmıştır. 1 ile 3 arasındaki fonksiyonlar için arama aralıkları tablo 4.4'tedir.

Dönme matrisi olan M_i, her bir fonksiyon için döndürme matrisi, koşul sayısı c = 1 veya 2 olan Gram-Schmidt orto-normalleştirmesi tarafından standart normal dağılımlı girişlerden üretilmiştir. Her bir fonksiyonun parametrelerinin birbirine tamamen bağlı olduğunun garanti edilmesi için 4 ile 10 arasındaki fonksiyonlara farklı döndürme matrisleri atanmıştır. 1 ile 3 arasındaki fonksiyonlar tamamen parametreye bağlıdır ve bu nedenle döndürülme yapılmamıştır.

Her bir fonksiyonun tanımları, kaydırma ve döndürme işleminden sonraki 2 boyut için grafikleri aşağıda her bir fonksiyon başlığı altında verilmiştir.

4.1.1. Storn’s chebyshev polinom uydurma problemi

Bu fonksiyon 2 boyut için tanımlı olmadığından bir grafiği yoktur. Fonksiyonun tanımı denklem 4.6’da verilmiştir ve özellikleri aşağıdaki gibidir.

- Çok modlu ve tek bir global minimuma sahiptir.
- Çok fazla şart barındırmaktadır.
- Ayrıştırılamaz; tamamen parametreler birbirlerine bağlıdır.

$$f(x) = p_1 + p_2 + p_3$$

$$p_1 = \begin{cases} u < d \quad ise \quad (u-d)^2 \\ diğer\ durumlarda \quad 0 \end{cases} \quad u = \sum_{j=1}^{D} x_j (1.2)^{D-j}$$

$$p_2 = \begin{cases} v < d \quad ise \quad (v-d)^2 \\ diğer\ durumlarda \quad 0 \end{cases} \quad v = \sum_{j=1}^{D} x_j \left(\frac{2k}{m} - 1\right)^{D-j} \tag{4.6}$$

$$p_3 = \sum_{k=0}^{m} p_k \quad , \quad k = 0,1,\dots,m, \ \ m = 32D$$

$$D = 9\ için\ d = 72.661$$

4.1.2. Ters hilbert matris problemi

Bu fonksiyon 2 boyut için tanımlı olmadığından bir grafiği yoktur. Fonksiyonun tanımı denklem 4.7'de verilmiştir ve özellikleri aşağıdaki gibidir.

- Çok modlu ve tek bir global minimuma sahiptir.
- Fazla şart barındırmaktadır.
- Ayrıştırılamaz; tamamen parametreler birbirlerine bağlıdır.

$$f(x) = \sum_{i=1}^{n} \sum_{k=1}^{n} |w_{i,k}|$$

$$\left(w_{i,k}\right) = W = HZ - I, \qquad I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.7}$$

$$H = \left(h_{i,k}\right), \qquad h_{i,k} = \frac{1}{i+k-1}, \qquad i,k = 1,2,\dots,n, \qquad n = \sqrt{D}$$

$$Z = \left(z_{i,k}\right), \qquad z_{i,k} = x_{i+n(k-1)}$$

4.1.3. Lennard-Jones minimum enerji kümesi

Bu fonksiyon 2 boyut için tanımlı olmadığından bir grafiği yoktur. Fonksiyonun tanımı denklem 4.8'de verilmiştir ve özellikleri aşağıdaki gibidir.

- Çok modlu ve tek bir global minimuma sahiptir.
- Ayrıştırılamaz; tamamen parametreler birbirlerine bağlıdır.

$$f(x) = 12.7120622568 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{d_{i,j}^2} - \frac{2}{d_{i,j}} \right)$$

$$d_{i,j} = \left(\sum_{k=0}^{2} \left(x_{3i+k-2} - x_{3j+k-2} \right)^2 \right)^3, \qquad n = D/3 \tag{4.8}$$

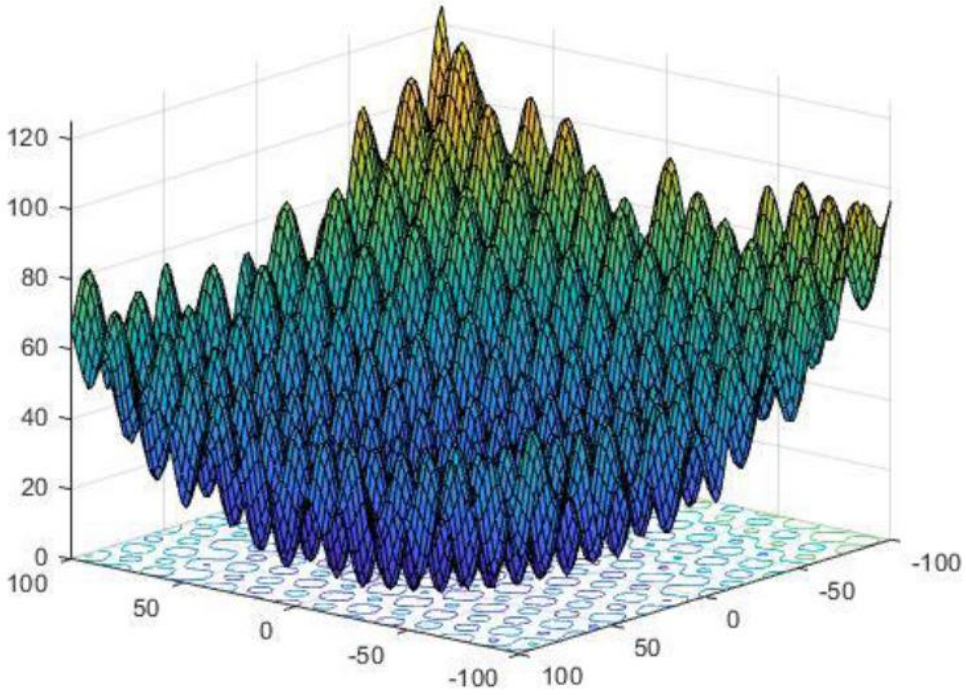
4.1.4. Rastrigin fonksiyonu

Fonksiyonun tanımı denklem 4.9'da verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.5'te verilmiştir.

- Çok modludur
- Ayrıştırılamaz yapıdadır.
- Yerel optimumların sayısı çok fazladır ve sondan bir önceki optimum, küresel optimumdan uzaktır.

$$f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$F(x) = f\left(M(x - o_4)\right) + F^* \tag{4.9}$$



Şekil 4.5. Rastrigin fonksiyonunun 2 boyut için grafiğidir.

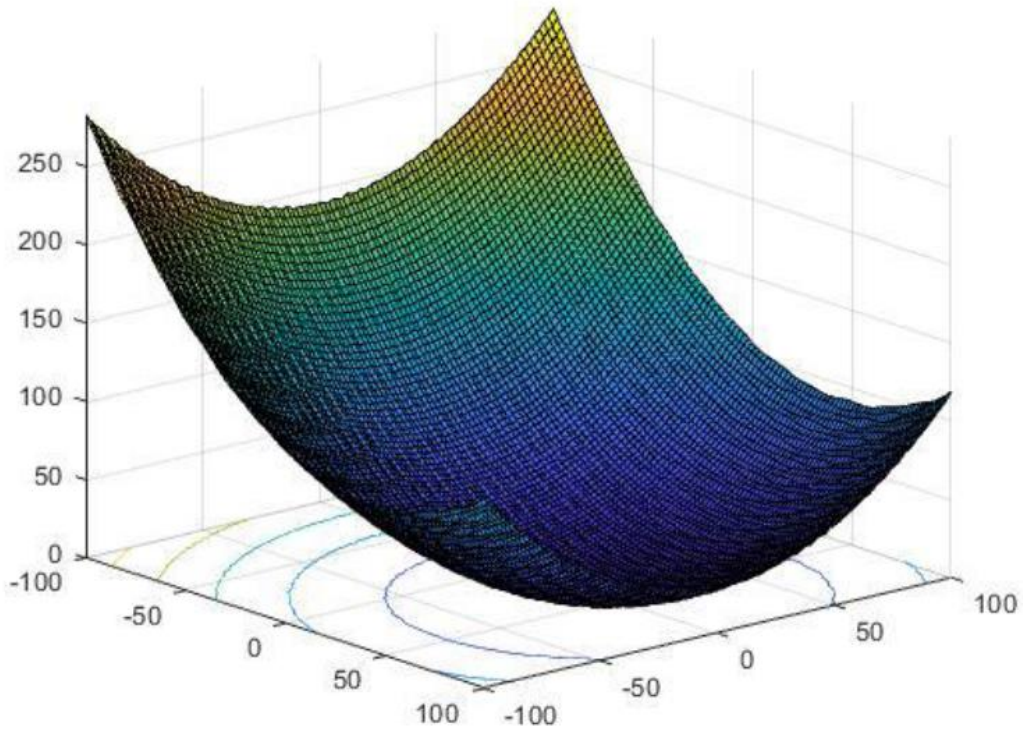
4.1.5. Griewank fonksiyonu

Fonksiyonun tanımı denklem 4.10'da verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.6'da verilmiştir.

- Çok modludur.
- Ayrıştırılamaz yapıdadır.

$$f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (4.10)$$

$$F(x) = f\big(M(x - o_5)\big) + F^*$$



Şekil 4.6. Griewank fonksiyonunun 2 boyut için grafiğidir.

4.1.6. Weierstrass fonksiyonu

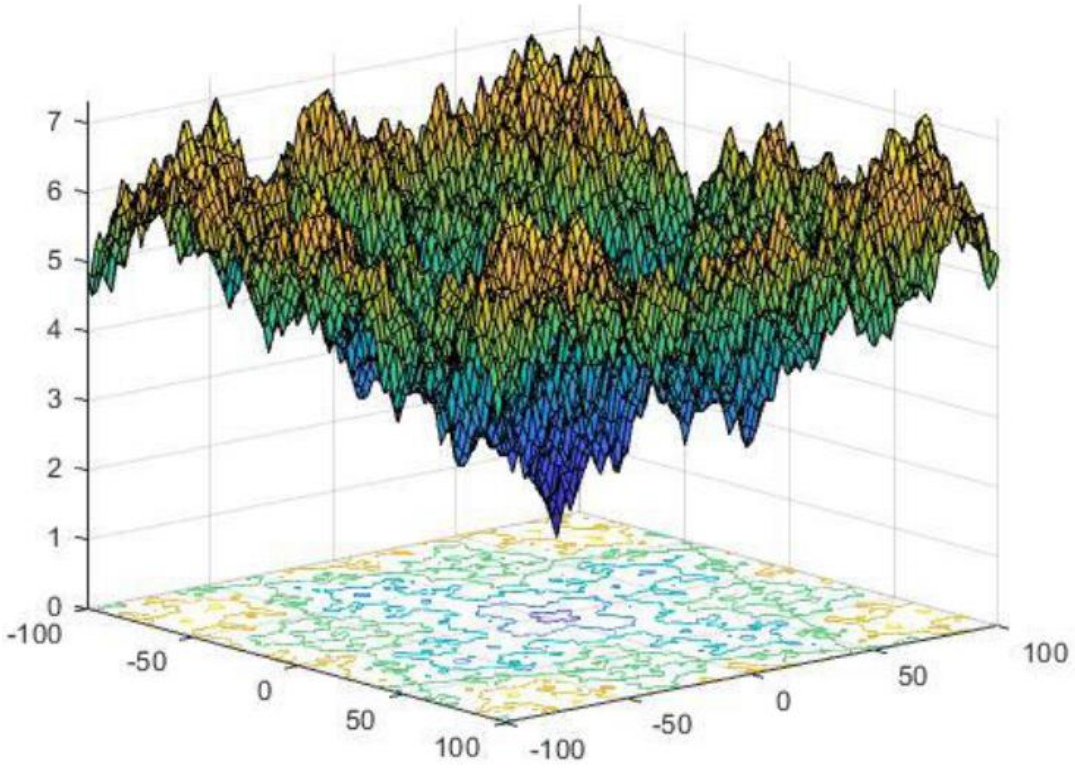
Fonksiyonun tanımı denklem 4.11'de verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.7'de verilmiştir.

- Çok modludur
- Ayrıştırılamaz yapıdadır.
- Lokal optimum sayısı çok fazladır.

$$f(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right) \right] \right) - D \sum_{k=0}^{k_{max}} a^k \cos(\pi b^k) \tag{4.11}$$

$$a = 0.5, \qquad b = 3, \qquad k_{max} = 20$$

$$F(x) = f\left(M(x - o_6)\right) + F^*$$



Şekil 4.7. Weierstrass fonksiyonunun 2 boyut için grafiğidir.

4.1.7. Değiştirilmiş Schwefel fonksiyonu

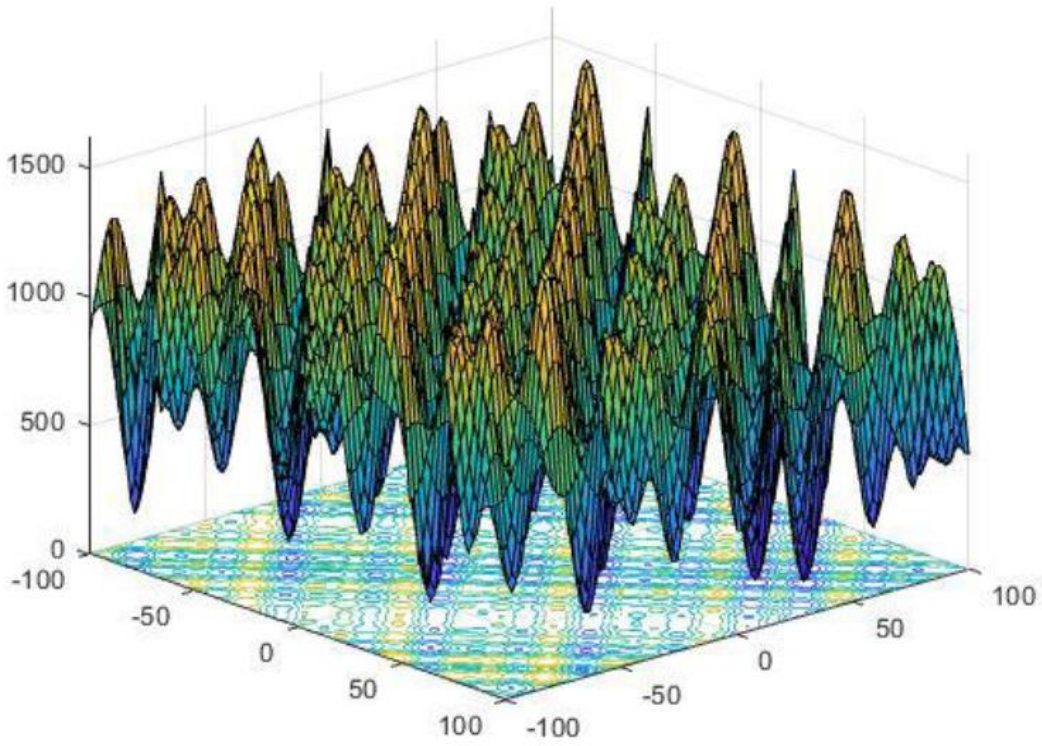
Fonksiyonun tanımı denklem 4.12'de verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.8'de verilmiştir.

- Çok modludur
- Ayrıştırılamaz yapıdadır.
- Lokal optimum sayısı çok fazladır.

$$f(x) = 418.9829D - \sum_{i=1}^{D} g(z_i), \qquad z_i = x_i + 420.9687462275036$$

$$g(z_i) = \begin{cases} |z_i| \le 500 & z_i \sin(|z_i|)^2 \\ z_i > 500 & \left(500 - mod(z_i, 500)\right)\sin\left(\sqrt{|500 - mod(z_i, 500)|}\right) - \frac{(z_i - 500)^2}{10000D} \\ z_i < -500 & \left(mod(|z_i|, 500) - 500\right)\sin\left(\sqrt{|mod(z_i, 500) - 500|}\right) - \frac{(z_i + 500)^2}{10000D} \end{cases} \quad (4.12)$$

$$F(x) = f\left(M(x - o_7)\right) + F^*$$



Şekil 4.8. Değiştirilmiş Schwefel fonksiyonunun 2 boyut için grafiğidir.

4.1.8. Genişletilmiş Schaffer F6 fonksiyonu

Fonksiyonun tanımı denklem 4.13'te verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.9'da verilmiştir.

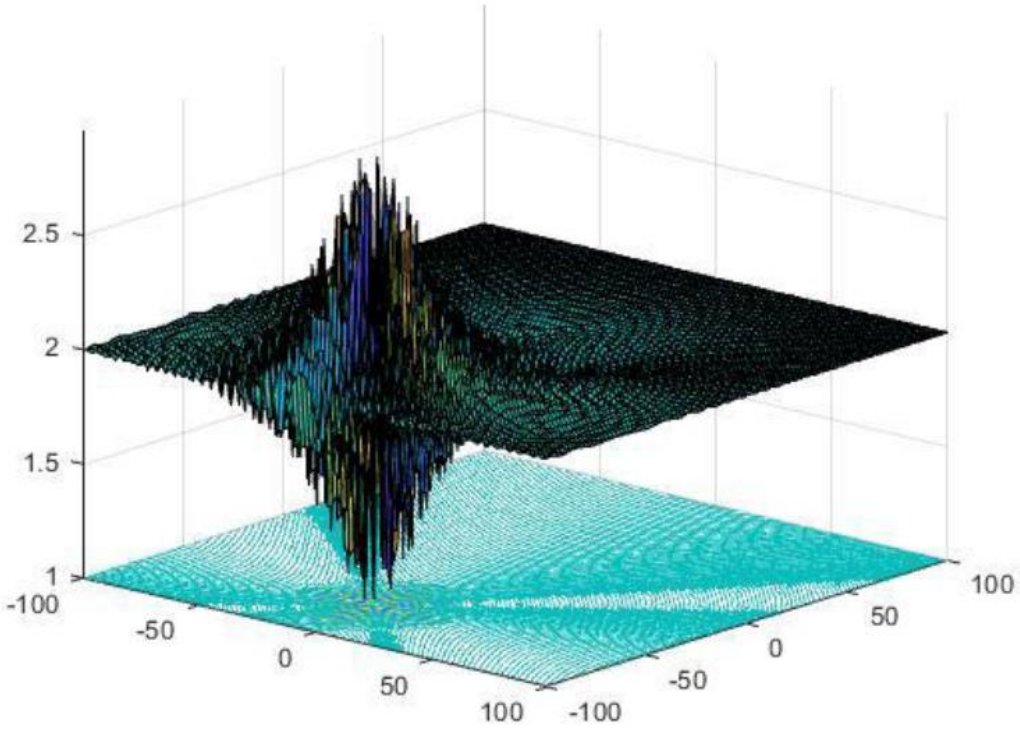
- Çok modludur

- Ayrıştırılamaz yapıdadır.
- Lokal optimum sayısı çok fazladır.

$$g(x,y) = 0.5 + \frac{\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{\left(1 + 0.001(x^2+y^2)\right)^2}$$

$$f(x) = g(x_1,x_2) + g(x_2,x_3) \ldots + g(x_{D-1},x_D) + g(x_D,x_1) \tag{4.13}$$

$$F(x) = f\left(M\left(0.005(x - o_8)\right)\right) + F^*$$



Şekil 4.9. Genişletilmiş Schaffer F6 fonksiyonunun 2 boyut için grafiğidir.

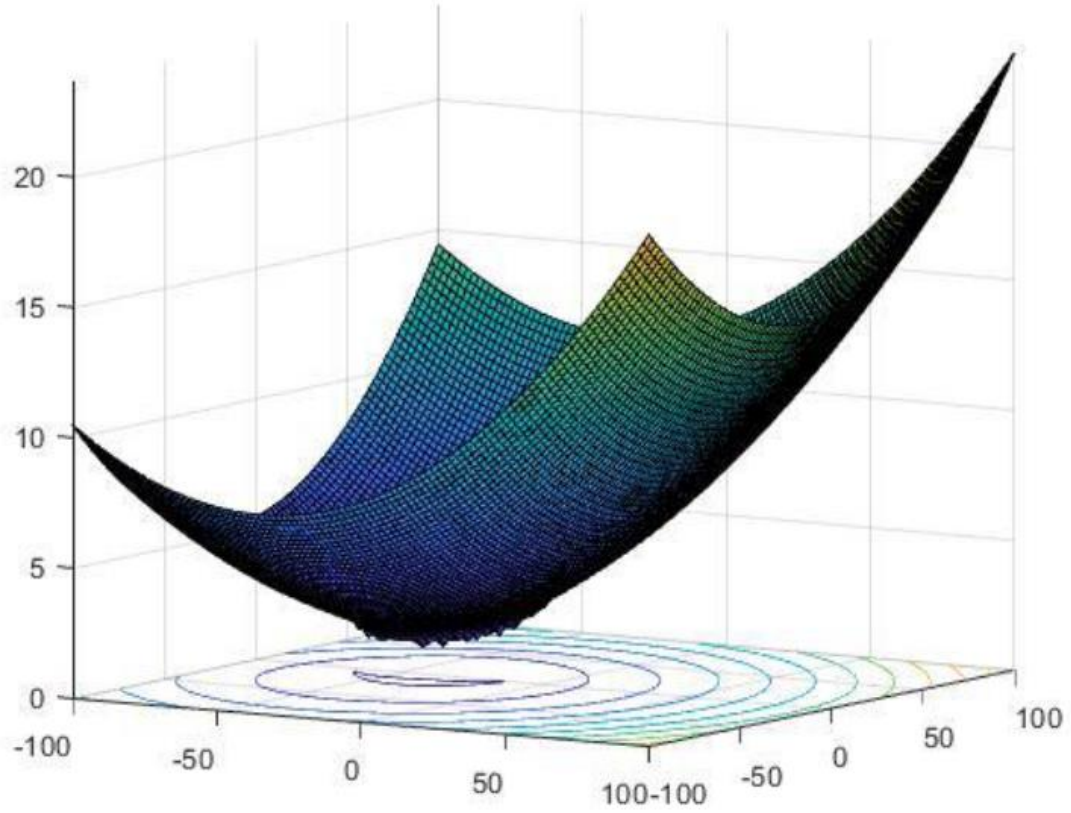
4.1.9. Happy Cat fonksiyonu

Fonksiyonun tanımı denklem 4.14'te verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.10'da verilmiştir.

- Çok modludur
- Ayrıştırılamaz yapıdadır.

$$f(x) = \left|\sum_{i=1}^{D} x_i^2 - D\right|^{1/4} + \frac{\left(0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_,\right)}{D} + 0.5 \tag{4.14}$$

$$F(x) = f\left(M(x - o_9)\right) + F^*$$



Şekil 4.10. Happy Cat fonksiyonunun 2 boyut için grafiğidir.

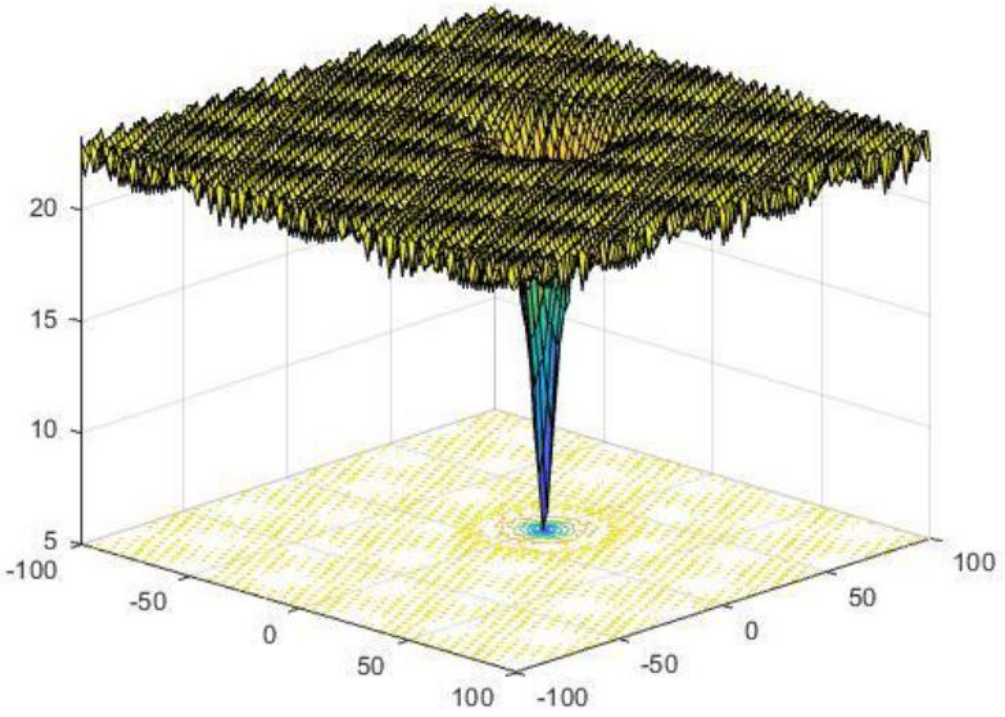
4.1.10. Ackley fonksiyonu

Fonksiyonun tanımı denklem 4.15'te verilmiştir ve özellikleri aşağıdaki gibidir. Fonksiyonun iki boyut için grafiği şekil 4.11'da verilmiştir.

- Çok modludur
- Ayrıştırılamaz yapıdadır.

$$f(x) = -20\exp\left(0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e \tag{4.15}$$

$$F(x) = f\big(M(x - o_{10})\big) + F^*$$



Şekil 4.11. Ackley fonksiyonunun 2 boyut için grafiğidir.

4.2. İstatistiksel Testler

Algoritmaların karşılaştırılabilirliği sıklıkla hipotez testi yoluyla çıkarılmıştır [55]. Bununla birlikte, sonuca varmak için sıfır hipotezi H0 ve alternatif hipotez H1'i belirlemek gerekmektedir. Sıfır hipotezi, genellikle karşılaştırılan algoritmalar arasında hiçbir fark olmadığını belirten bir ifadedir. Alternatif hipotez ise farklılıkları temsil eder. Bu tez çalışmasında H0 ve H1 hipotezleri aşağıdaki şekilde ele alınmıştır.

H0: Karşılaştırılan algoritmalar arasında fark yok.

H1: Karşılaştırılan algoritmalar arasında fark var.

İstatistiksel test olasılık değeri olan α ayrıca hipotezin çürütülmesi gerekip gerekmediğini de belirler. Testimiz için kabul edilebilir α değeri 0,05'tir.

Parametrik olmayan bir istatistiksel test olan Friedman testi, ilk olarak Friedman tarafından tanıtılmıştır [56,57]. Belirli algoritmaların nasıl davrandığına ilişkin farklılıkları belirtmek için kullanılmıştır. Karşılaştırılan algoritmalar için Friedman test sonuçları sütunlarda görüntülenir ve test durumları satırlarda gösterilir.

Test, her satırı, satırdaki sütunların değerlerine göre sıralayarak başlar ve ardından her sütun için genel sıra değerlerini hesaplar. Testin anlamlılığını hesaplamak için k-1 serbestlik dereceli (sd) X^2 (Ki-kare) dağılımı kullanılır; burada k, karşılaştırılan yöntemlerin sayısıdır.

Serbestlik derecesi için uygun X^2 değerlerini [58] numaralı referans içerisinde bulabiliriz. (sd)=7 ve α =0,05 olduğu için [58] numaralı referans içindeki tablodan X^2 değeri 14,07 olarak beklenmektedir. Gerçek X^2 değeri beklenenden yüksekse sıfır hipotezi reddedilmelidir.

İki örnek veya algoritma arasındaki farkları belirlemek için kullanılan başka bir parametrik olmayan istatistiksel test, Wilcoxon işaretli sıralama testidir [59]. Fark vektörünü oluşturmak için, test tipik olarak, iki algoritmanın N*1 boyutlarına sahip çıktıları arasındaki tutarsızlıklarla başlar; burada N, toplam test sayısıdır. Daha sonra minimum değerden başlayarak vektördeki her satıra “1” sıra değerini atar. Ardından fark vektörünün R_- ve R_+ değerleri hesaplanır. Testin T değeri, min(R_-, R_+), verilere göre hesaplanır. Sonuç olarak T değeri kullanılarak testin olasılık değeri p hesaplanır.

5. TEST SONUÇLARI

5.1. Test Fonksiyonlarının Sonuçları ve Karşılaştırılması

Tüm test senaryoları, ilgili ayarlarla on kez çalıştırılmıştır. Rosenbrock, Ackley, Griewank ve Rastrigin fonksiyonlarının 10, 20 ve 30 boyuttaki değerlendirme sonuçları tablo 5.1'de verilmiştir. Tablo 5.1 içindeki tüm değerler, karşılık gelen algoritmalarla yürütülen on çalışmanın ortalamasıdır. Tablo 5.1 üzerinde verilmiş olan her bir problem boyutu için üstteki değerler on çalıştırma sonucunda elde edilen sonuçların ortalamasını hemen altında yazan parantez içerisindeki değerler ise yine on çalıştırma sonucunda elde edilen sonuçların standart sapmalarını göstermektedir.

Önerilen algoritma EAA'nın her test senaryosunda DYA, KOA ve HŞO ile neredeyse benzer sonuçlar verdiğini tablo 5.1'de görebilmekteyiz. Önerilen algoritma, on iki senaryonun altısında yapılan on tekrarlı çalıştırmanın hepsinde kesin bir çözüm bulmuştur. Ayrıca Hem HŞO hem de KOA, on iki senaryonun altısında on tekrarlı çalıştırmanın hepsinde de kesin bir çözüm bulmuştur. DYA ise yalnızca beş kesin çözüm bulmuştur.

Şekil 5.1'de Rosenbrock fonksiyonu 10 boyut, Şekil 5.2'de Rosenbrock fonksiyonu 20 boyut, Şekil 5.3'te Rosenbrock fonksiyonu 30 boyut, şekil 5.4'de Ackley fonksiyonu 10 boyut, 5.5'de Ackley fonksiyonu 20 boyut, 5.6'de Ackley fonksiyonu 30 boyut, şekil 5.7'te Griewank fonksiyonu 10 boyut, şekil 5.8'te Griewank fonksiyonu 20 boyut, şekil 5.9'te Griewank fonksiyonu 30 boyut, şekil 5.10'da Rastrigin fonksiyonu 10 boyut, şekil 5.11'de Rastrigin fonksiyonu 20 boyut ve şekil 5.12'de Rastrigin fonksiyonu 30 boyut için yakınsama grafikleri verilmiştir.

Şekil 5.1, 5.4, 5.7 ve 5.10'da ESA'nın keskin bir düşüş modeli gerçekleştirdiğini görebiliriz. Bu düşüş modeli algoritma bünyesindeki başarısız ajanların silinmesi prosedürünün bir ürünüdür. EAA yerel bir minimum noktada sıkışıp kaldığında bu başarısız ajanlar silinmekte ve kendilerine karşılık gelen kutuplarda yepyeni ajanlar oluşturulmaktadır. Rastrigin fonksiyonu, benzer zorlukta olan Ackley ve Griewank fonksiyonlarının yanı sıra, üç test fonksiyonundan en kolay olanıdır.

EAA, tüm test senaryolarında GA, PSO, GİA ve DAA'dan daha iyi çözümler bulmuştur, ancak yalnızca 20 ve 30 boyutlu Ackley fonksiyonunda; EAA algoritması, DYA, KOA ve HŞO'dan daha iyi sonuçlar bulmuştur. Bunun dışında EAA algoritması, DYA, KO ve HŞO algoritmaları ile karşılaştırıldığında benzer sonuçlar üretmektedir.

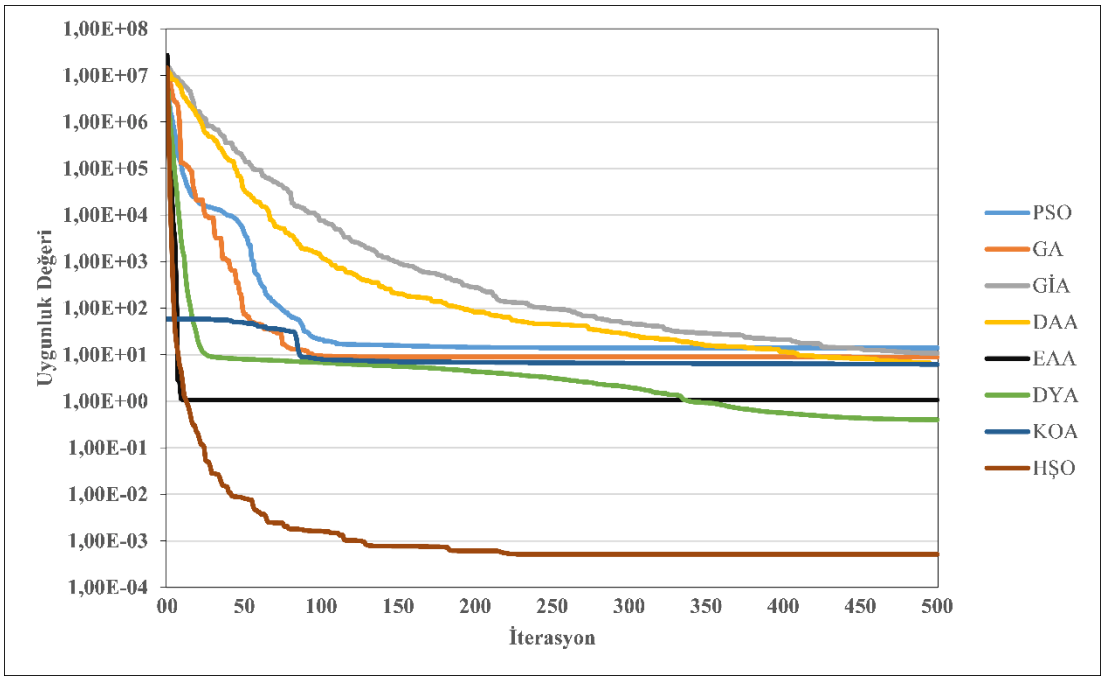
Önerilen algoritma, büyük iterasyon sayıları ve zaman sınırları dahilinde problemleri çözmek için tasarlanmıştır. Eğer iterasyonları ikiye katlarsak, önerilen algoritma DYA, KOA ve HŞO ile aynı performansı göstermekte ve her zaman tüm test durumları için kesin çözümleri bulmaktadır.

Tablo 5.1. Test fonksiyonlarının değerlendirme sonuçlarıdır.

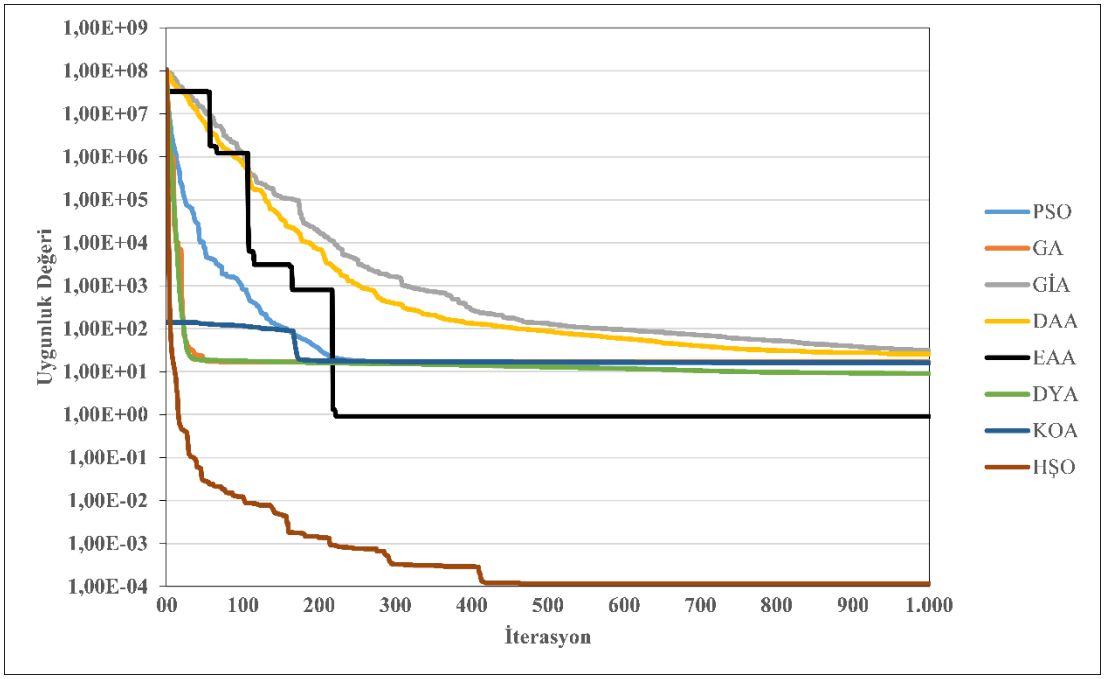
Test Fonksiyonu	Boyut	PSO	GA	GİA	DAA	DYA	KOA	HŞO	EAA
Rosenbrock	10	1,40E+01	8,92E+00	1,07E+01	6,24E+00	3,99E-01	6,23E+00	5,14E-04	1,07E+00
		(2,30E+01)	(7,72E-02)	(2,83E+00)	(1,99E+00)	(1,95E-01)	(3,01E-01)	(5,25E-04)	(1,74E+00)
	20	1,59E+01	1,70E+01	3,11E+01	2,56E+01	8,95E+00	1,62E+01	1,16E-04	9,07E-01
		(1,57E+00)	(5,65E+00)	(1,88E+01)	(2,01E+01)	(4,14E-01)	(2,86E-01)	(1,42E-04)	(1,62E+00)
	30	2,54E+01	2,60E+01	4,26E+01	6,31E+01	1,84E+01	2,64E+01	1,68E-04	1,02E-01
		(2,49E+00)	(8,65E+00)	(2,59E+01)	(2,63E+01)	(7,96E-01)	(3,57E-01)	(1,66E-04)	(1,90E-01)
Ackley	10	7,99E-01	1,14E-10	1,72E-02	1,16E-04	4,44E-15	8,88E-16	8,88E-16	1,34E-04
		(9,90E-01)	(3,19E-10)	(9,85E-03)	(6,63E-05)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(2,43E-04)
	20	2,95E-01	0,00E+00	4,49E-03	5,21E-05	3,38E-15	8,88E-16	8,88E-16	0,00E+00
		(5,65E-01)	(0,00E+00)	(5,43E-03)	(3,97E-05)	(1,63E-15)	(0,00E+00)	(0,00E+00)	(0,00E+00)
	30	1,20E-01	0,00E+00	3,07E-04	3,42E-05	4,09E-15	8,88E-16	8,88E-16	0,00E+00
		(3,46E-01)	(0,00E+00)	(1,63E-04)	(2,73E-05)	(1,07E-15)	(0,00E+00)	(0,00E+00)	(0,00E+00)

Tablo 5.1. (Devamı) Test fonksiyonlarının değerlendirme sonuçlarıdır.

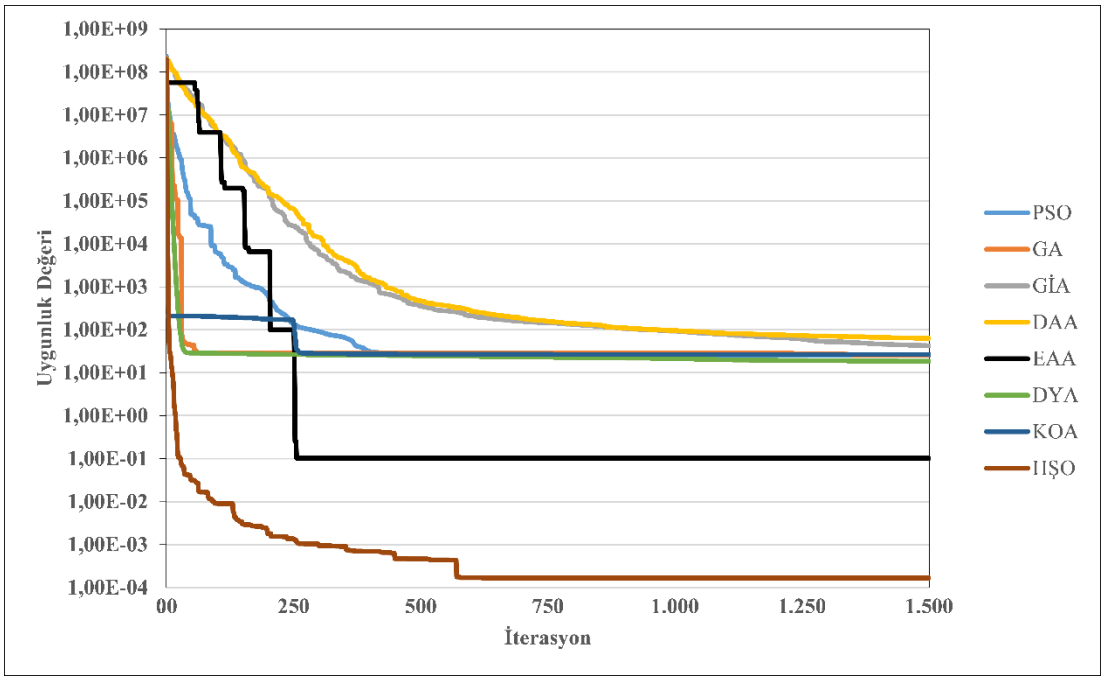
Test Fonksiyonu	Boyut	PSO	GA	GİA	DAA	DYA	KOA	HŞO	EAA
Griewank	10	2,42E-01	1,22E-05	6,17E-02	3,27E-03	2,37E-09	0,00E+00	0,00E+00	2,96E-07
		(2,18E-01)	(3,66E-05)	(2,70E-02)	(4,11E-03)	(7,10E-09)	(0,00E+00)	(0,00E+00)	(5,19E-07)
	20	3,69E-02	0,00E+00	9,67E-04	2,59E-06	0,00E+00	0,00E+00	0,00E+00	0,00E+00
		(1,51E-02)	(0,00E+00)	(1,15E-03)	(4,73E-06)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)
	30	1,22E-01	0,00E+00	7,07E-06	2,90E-08	0,00E+00	0,00E+00	0,00E+00	0,00E+00
		(2,45E-01)	(0,00E+00)	(7,18E-06)	(5,00E-08)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)
Rastrigin	10	6,48E+00	5,69E-01	2,61E+00	1,32E-03	0,00E+00	0,00E+00	0,00E+00	0,00E+00
		(1,21E+01)	(9,94E-01)	(6,71E-01)	(1,85E-03)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)
	20	2,11E+01	2,69E+00	7,92E+00	1,45E-01	0,00E+00	0,00E+00	0,00E+00	9,73E-07
		(1,17E+01)	(5,49E+00)	(1,55E+00)	(3,87E-01)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(2,38E-06)
	30	3,99E+01	0,00E+00	1,52E+01	1,48E-01	0,00E+00	0,00E+00	0,00E+00	0,00E+00
		(1,95E+01)	(0,00E+00)	(3,39E+00)	(3,58E-01)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)



Şekil 5.1. Rosenbrock fonksiyonu 10 boyut için yakınsama grafiğidir.

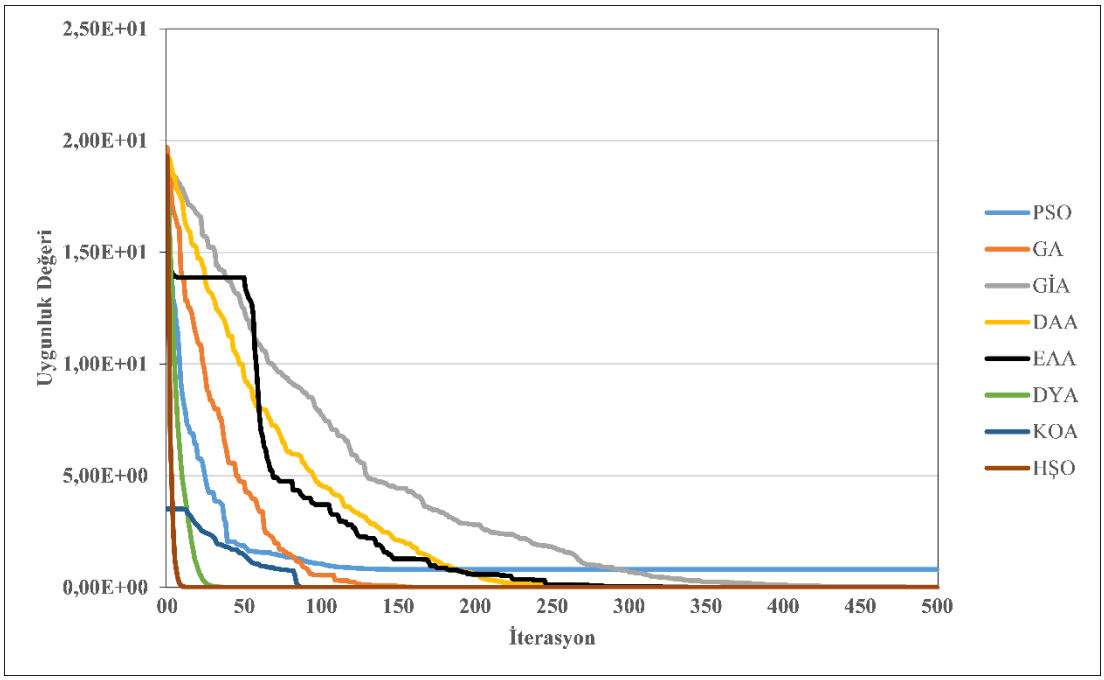


Şekil 5.2. Rosenbrock fonksiyonu 20 boyut için yakınsama grafiğidir.

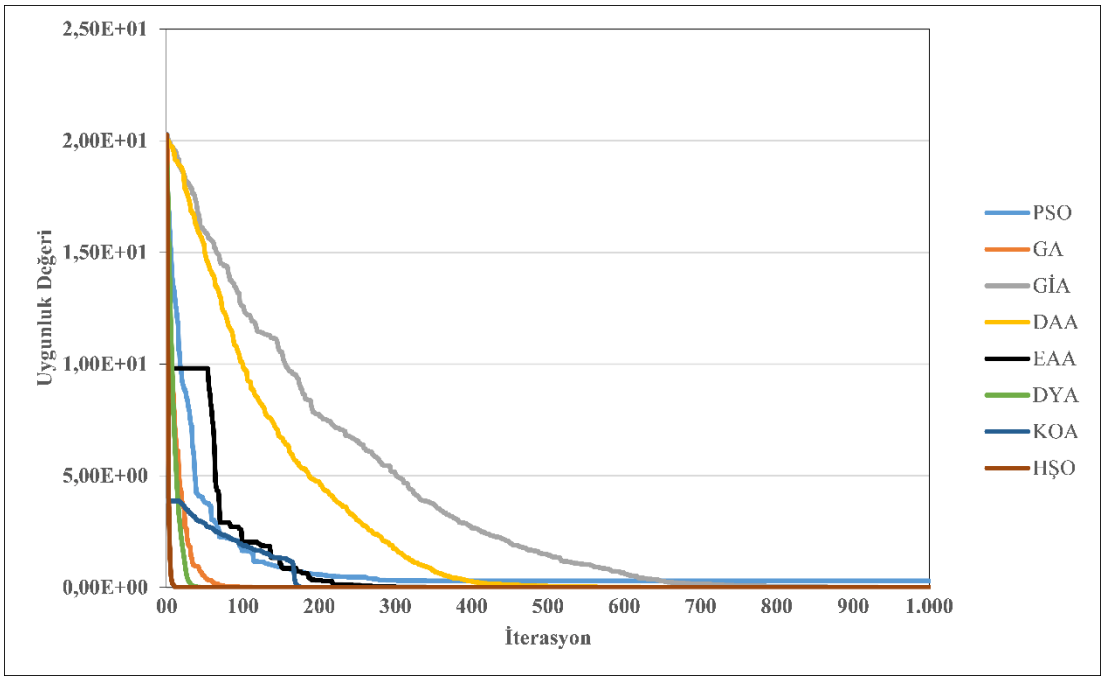


Şekil 5.3. Rosenbrock fonksiyonu 30 boyut için yakınsama grafiğidir.

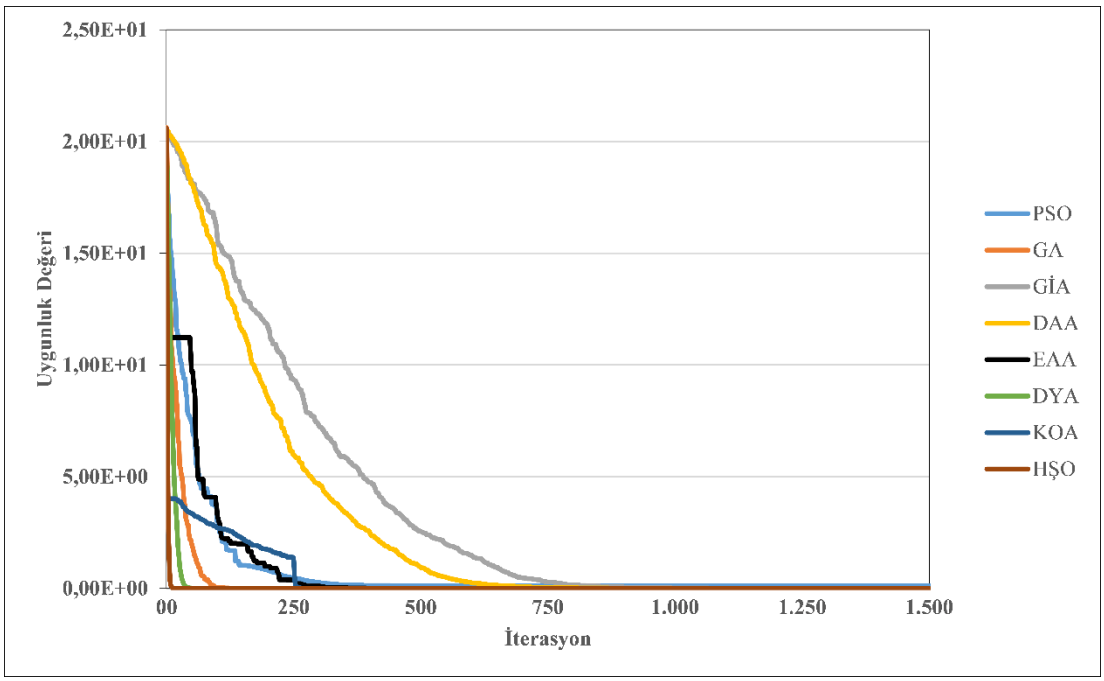
Yukarıdaki şekiller 5.1, 5.2 ve 5.3'te görüldüğü üzere önerilen algoritmamız HŞO ve DYA algoritmaları hariç diğer algoritmalardan daha iyi bir sonuç üretmiştir. Genel olarak HŞO, DYA ve EAA algoritmaları dışındaki algoritmalar tek ve çözüme uzak bir noktada takılırken bu üç algoritma minimum değer olan "0" a çok yaklaşmışlardır.



Şekil 5.4. Ackley fonksiyonu 10 boyut için yakınsama grafiğidir.



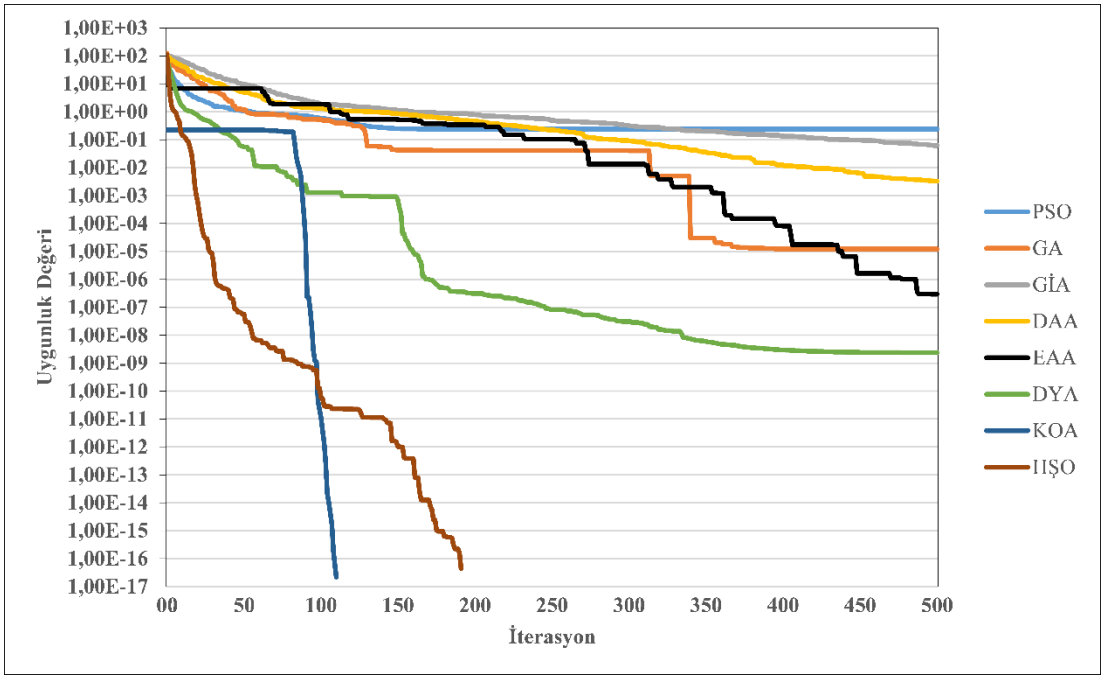
Şekil 5.5. Ackley fonksiyonu 20 boyut için yakınsama grafiğidir.



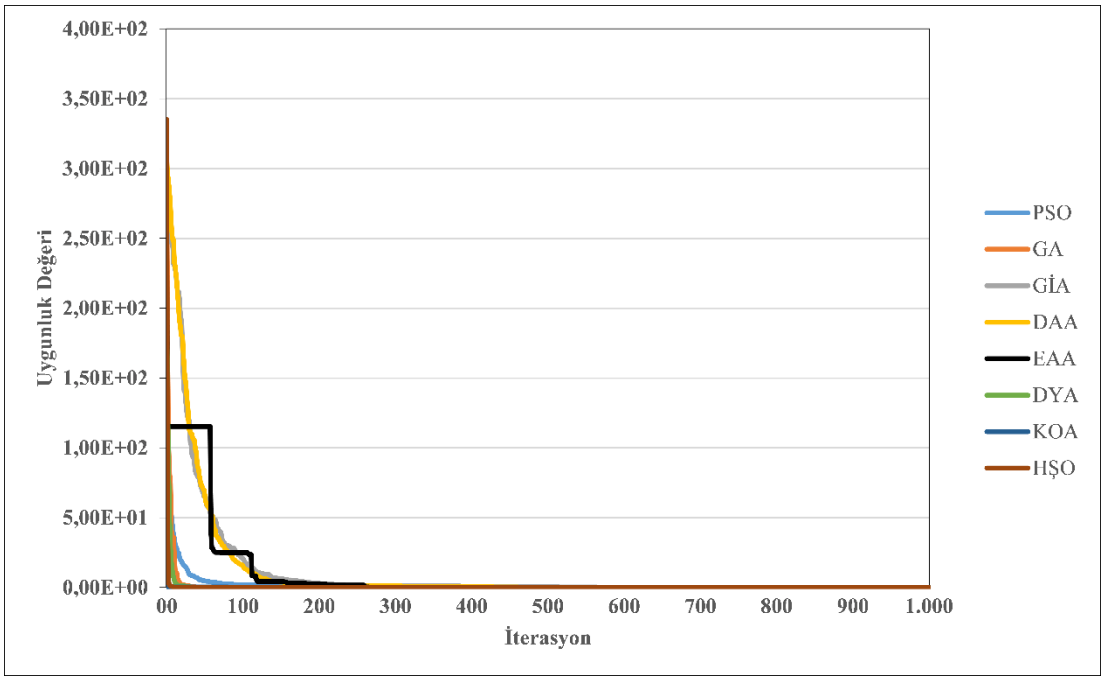
Şekil 5.6. Ackley fonksiyonu 30 boyut için yakınsama grafiğidir.

Yukarıdaki şekiller 5.4, 5.5 ve 5.6'da görüldüğü üzere tüm algoritmalar küresel minimum noktası olan "0" değerine çok yaklaşmışlardır. Şekil 5.4'ten ve tablo 5.1'deki veriler üzerinden değerlendirme yapacak olacak olursak 10 boyut için en yakın sonucu HŞO ve KOA algoritmaları vermişlerdir. Ancak PSO algoritmasının

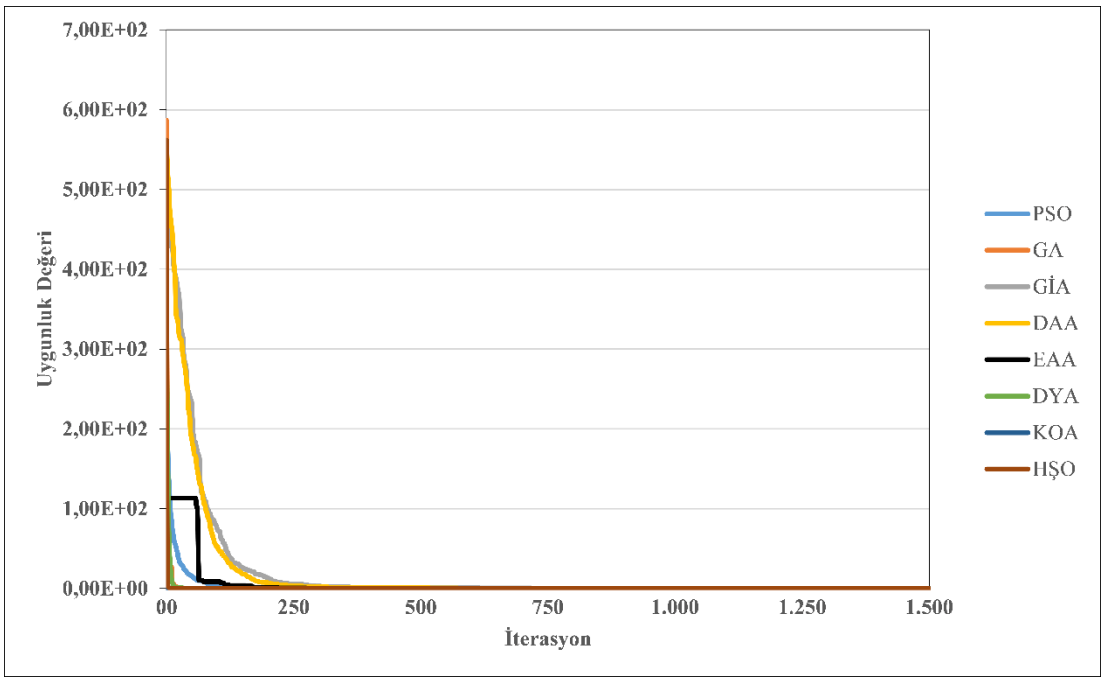
bulduğu ortalama sonuç hariç tüm algoritmalar küresel minimum noktasına çok yakın olduğundan bu algoritmalar küresel minimum noktasını bulmuştur diyebiliriz. Şekil 5.5, 5.6 ve tablo 5.1'deki verilerden yola çıkarak problem boyutu artmasına ve buna bağlı olarak problemin zorlaşmasına rağmen iterasyon limitinin artması sayesinde bütün denemelerde GA ve EAA algoritmaları tam sonucu bulmuştur.



Şekil 5.7. Griewank fonksiyonu 10 boyut için yakınsama grafiğidir.



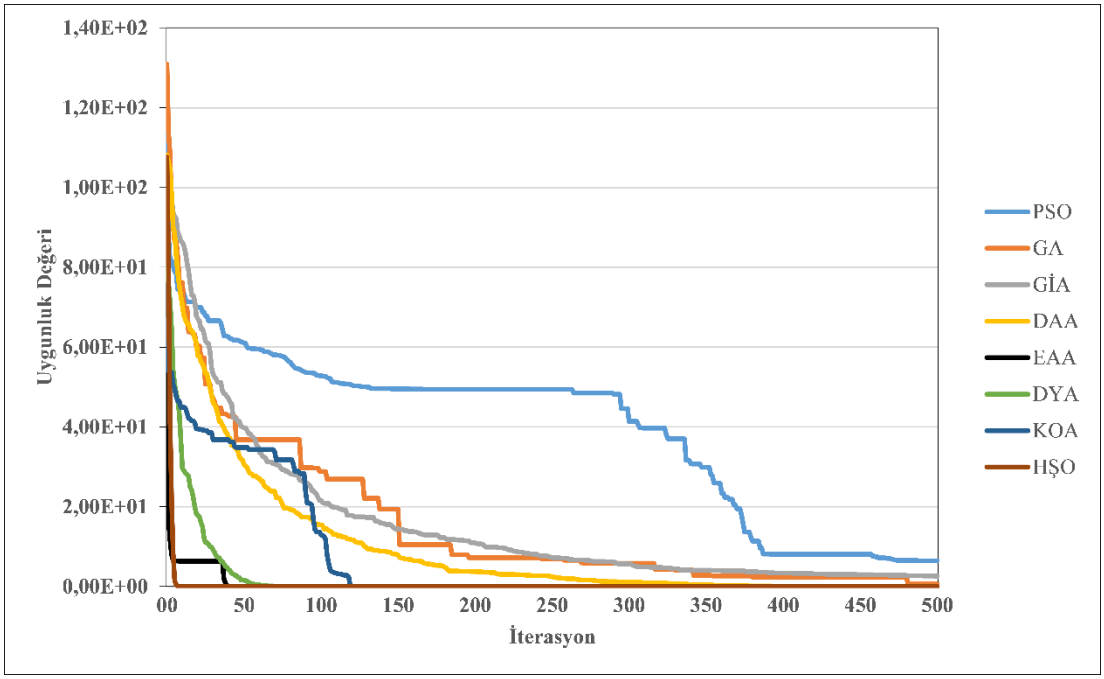
Şekil 5.8. Griewank fonksiyonu 20 boyut için yakınsama grafiğidir.



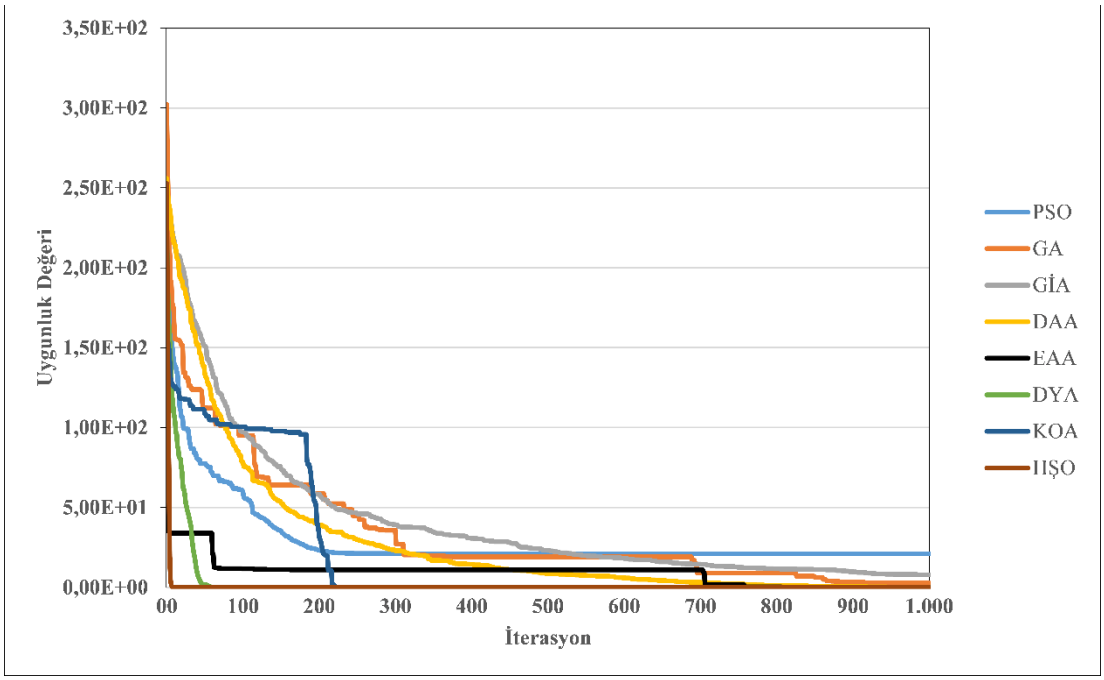
Şekil 5.9. Griewank fonksiyonu 30 boyut için yakınsama grafiğidir.

Yukarıdaki şekiller 5.7, 5.8 ve 5.9 incelendiğinde Ackley fonksiyonu test senaryoları hakkında yaptığımız bir yorumun benzeri burası içinde geçerlidir. Tablo 5.1'de gösterilen verilere göre EAA algoritması sadece 10 boyut için tam sonucu bulamamıştır. Ancak problemin boyutu ve zorluğu artmasına rağmen iterasyon limiti

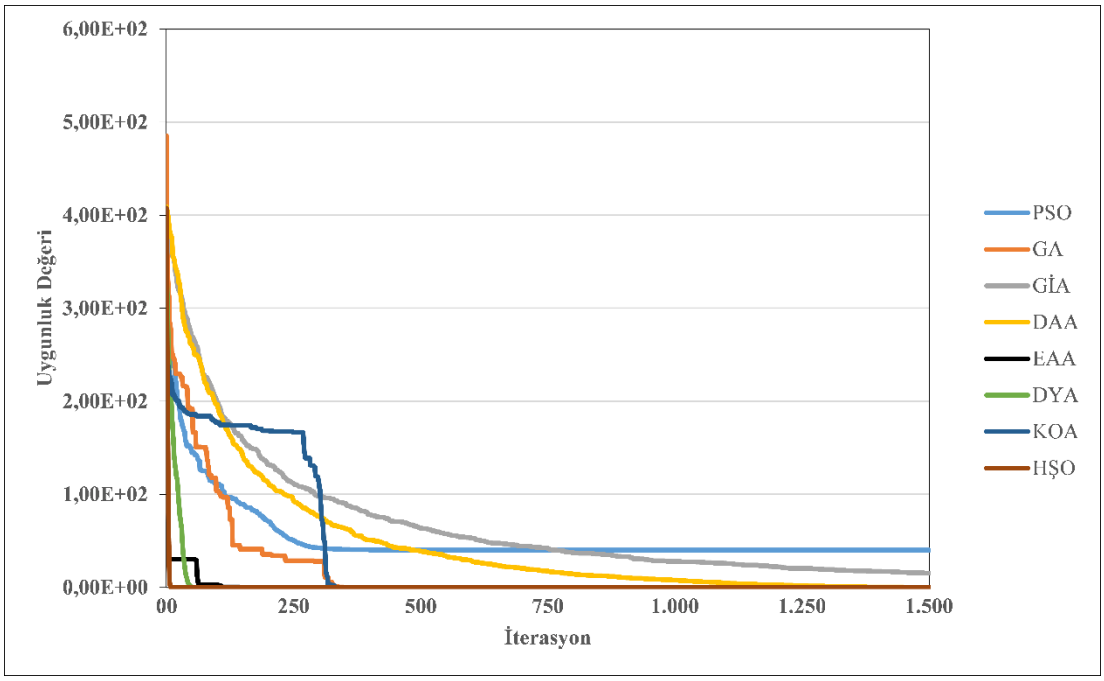
arttığı için diğer iki senaryoda EAA ve birçok algoritma tam sonucu bütün denemelerde bulmuştur.



Şekil 5.10. Rastrigin fonksiyonu 10 boyut için yakınsama grafiğidir.



Şekil 5.11. Rastrigin fonksiyonu 20 boyut için yakınsama grafiğidir.



Şekil 5.12. Rastrigin fonksiyonu 30 boyut için yakınsama grafiğidir.

Şekiller 5.10, 5.11 ve 5.12 üzerinden bakıldığında birçok algoritma Rastrigin fonksiyonunun çözümünde tam sonuca yaklaşamamıştır. Tablo 5.1 Rastrigin fonksiyonu için incelendiğinde tablo üzerinde görüleceği üzere önerilen algoritma 10 ve 30 boyut için tam sonucu bulmuştur. Ancak 20 boyut için tam çözüme çok yaklaşmıştır. Bazı algoritmalar ise bu tam sonuca neredeyse hiç yaklaşamayarak yerel minimum noktalarında takılmıştır.

Yukarıda verilmiş olan bütün yakınsama grafikleri topluca incelendiğinde önerilen algoritma EAA, problem çözme esnasında kendi bünyesinde barındırdığı birçok mekanizma sayesinde yerel minimum noktalarına takılmamıştır. Grafiklerde gözüktüğü üzere birçok noktada ani düşüşler sergilemiştir. Bu ani düşüşlerin olduğu noktalarda algoritma yerel minimum noktalarından kendini kurtarmış ve aramaya başka bir alandan devam ederek küresel minimuma doğru ilerlemiştir.

5.2. Test Fonksiyonu Sonuçlarının İstatiksel Değerlendirmeleri

5.2.1. Friedman testinin sonuçları ve değerlendirilmesi

Tablo 5.2'de hangi algoritmanın daha başarılı olduğunu görebiliriz. En düşük sıra değerine sahip olan algoritma, diğer algoritmalara göre daha iyi bir performans

sergilemektedir. Tablo 5.2'de gösterildiği gibi, EAA tüm test işlevlerinde en iyi ikinci performansı göstermiştir.

Tablo 5.2. Algoritmaların ortalama sıra değerleridir.

Algoritmalar	Ortalama Sıra Değeri
PSO	7,33
GA	4,17
GİA	7,08
DAA	5,92
DYA	3,13
KOA	3,21
HŞO	2,21
EAA	2,96

Bunun yanı sıra, olasılık değeri 1,0245E-10 olarak hesaplanmıştır bu değer α =.005 değerinden çok daha düşüktür. X^2 değerinin ise 60,842672 olduğu ve bu değerin beklenen değer olan 14.07 değerinden yüksek olduğu bulunmuştur. Bu hesaplanan değerler, H0, "Karşılaştırılan algoritmalar arasında fark yoktur" boş hipotezinin reddedilmesi gerektiği anlamına gelmektedir. Bu durum, H1 hipotezinin doğru olduğu anlamına gelmektedir: "Karşılaştırılan algoritmalar arasında fark vardır". Friedman testinin sonuçları tablo 5.3'te verilmiştir.

Tablo 5.3. Algoritmaların Friedman testi istatistikleri

Friedman testi parametresi	Değer
N	12
Ki-Kare X^2	60,842672

Tablo 5.3. (Devamı) Algoritmaların Friedman testi istatistikleri

Friedman testi parametresi	Değer
sd	7
p	1,0245E-10

5.2.2. Wilcoxon işaretli sıra testinin sonuçları ve değerlendirilmesi

Karşılaştırılan yöntemler arasındaki ana farklar, p değerleri ile tanımlanmaktadır. Sıfır hipotezinin reddedilebilmesi için bu sayıların 0,05'ten küçük olması gerekmektedir. Aksi durumda karşılaştırılan yöntemlerinin performansının eşdeğer olduğu kabul edilmelidir.

Tablo 5.4'te EAA'nın karşılaştırılan algoritmaların çoğundan farklı bir performans sergilediğini görebiliriz. Bu, önerilen algoritmanın karşılaştırılan algoritmalardan farklı davrandığı anlamına gelmektedir. Ancak DYA ve KOA algoritmalarına bakacak olursak algoritmamızın bu algoritmalara benzer sonuçlar verdiğini görmekteyiz.

Tablo 5.4. Wilcoxon işaretli sıralar testi için p değerleridir.

Algoritma	PSO	GA	GİA	DAA	DYA	KOA	HŞO	EAA
PSO	1	0,049860204	0,157939311	0,136097381	0,002217721	0,049860204	0,002217721	0,002217721
GA		1	0,002217721	0,346521712	0,020879263	0,085281059	0,020767229	0,042522478
BSA			1	0,034170473	0,002217721	0,002217721	0,002217721	0,002217721
DS				1	0,002217721	0,002217721	0,002217721	0,004741768
MPO					1	0,498962299	0,017960478	1
KO						1	0,10880943	0,400236144
HHO							1	0,035465865
ESA								1

5.3. Kümeleme Probleminin Çözümünün Sonuçları ve Karşılaştırılması

Kümeleme probleminin performans analizi için Davies-Bouldin indeksi (DB-Index) kullanılmıştır. Tüm veri kümeleri için "k" küme sayısı iki ile başlamakta ve on ile bitmektedir. Bu durumda otuz altı test senaryomuzun olduğu ortaya çıkmaktadır. EAA ve X-Ortalamalar algoritmasının her test senaryosu için Davies-Bouldin indeksini hesaplanmış ve karşılaştırılmıştır. DB-Index değerlerinin karşılaştırması Tablo 5.5'te, DB-Index değerlerinin EAA ve X-Ortalamalar algoritması için grafiksel karşılaştırması Iris veri seti için şekil 5.13'te, Seeds veri seti için şekil 5.14'te, Wine veri seti için şekil 5.15'te ve HCV veri seti için şekil 5.16'da verilmiştir.

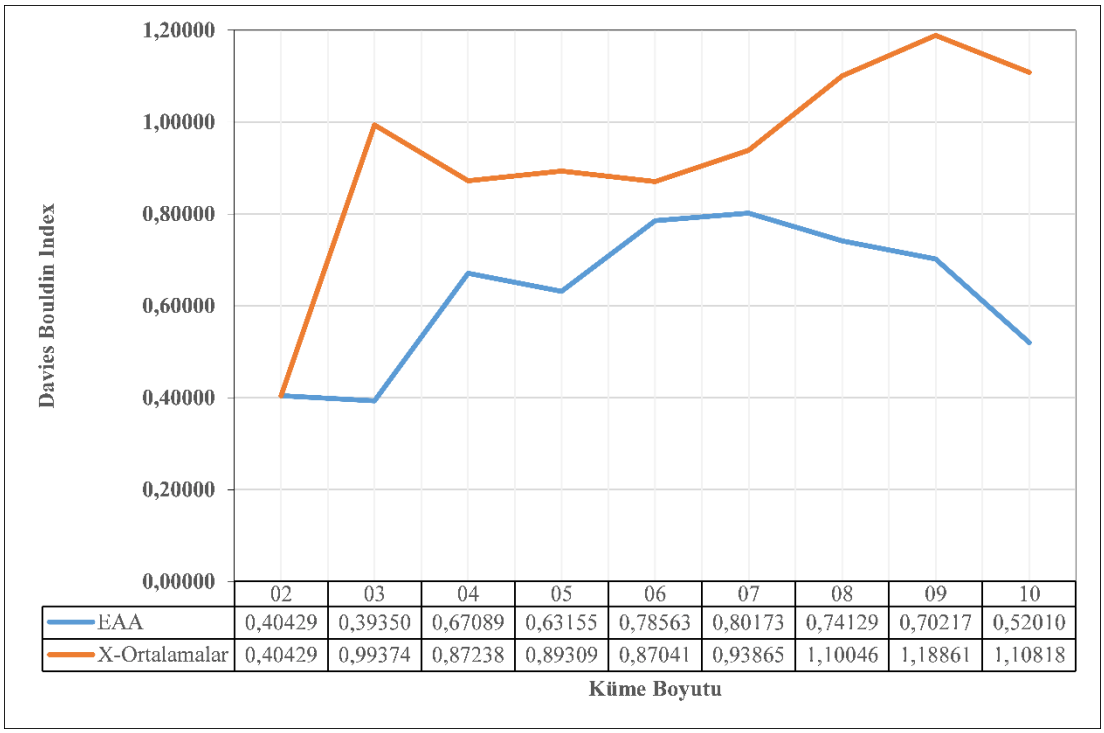
Tablo 5.5'te EAA'nın tüm test durumlarında X-Ortalamalar algoritmasından daha iyi kümeleme performansına sahip olduğunu göremekteyiz. Ayrıca veri seti bazında bakarsak, ESA her veri setinde X-Ortalamalar algoritmasından daha iyi kümeleme yapmıştır. K'nin iki olduğu test durumlarında, EAA ve X-Ortalamalar algoritmaları aynı DB-Index'e sahiptir. Bu durum, kümeleme probleminin çözümünde EAA'nın popüler kümeleme algoritması X-Ortalamalar algoritması ile aynı performansı yürüttüğünü göstermektedir. Sonuç olarak, önerilen algoritmanın kümeleme problemini çözebileceği kanıtlanmış olmaktadır.

DB-Index değerleri, optimum "k" sayısını belirlemek ve kümeleme analizi yapmak için kullanılmaktadır. Daha küçük DB-Index değerleri, oluşturulan kümelerin kompakt ve birbirinden uzak olduğunu göstermektedir.

Şekiller 5.13, 5.14, 5.15 ve 5.16'da görüldüğü üzere EAA algoritması bütün veri setlerinde her bir küme boyutu için daha düşük DB-Index değeri elde etmiştir. Bu değerin düşük olması algoritmanın ürettiği veri kümeleri kendi içerisinde daha sıkı bir yapıya sahip olduğunu ve diğer kümeler ile olan ayrıklığının ise daha yüksek olduğunu göstermektedir.

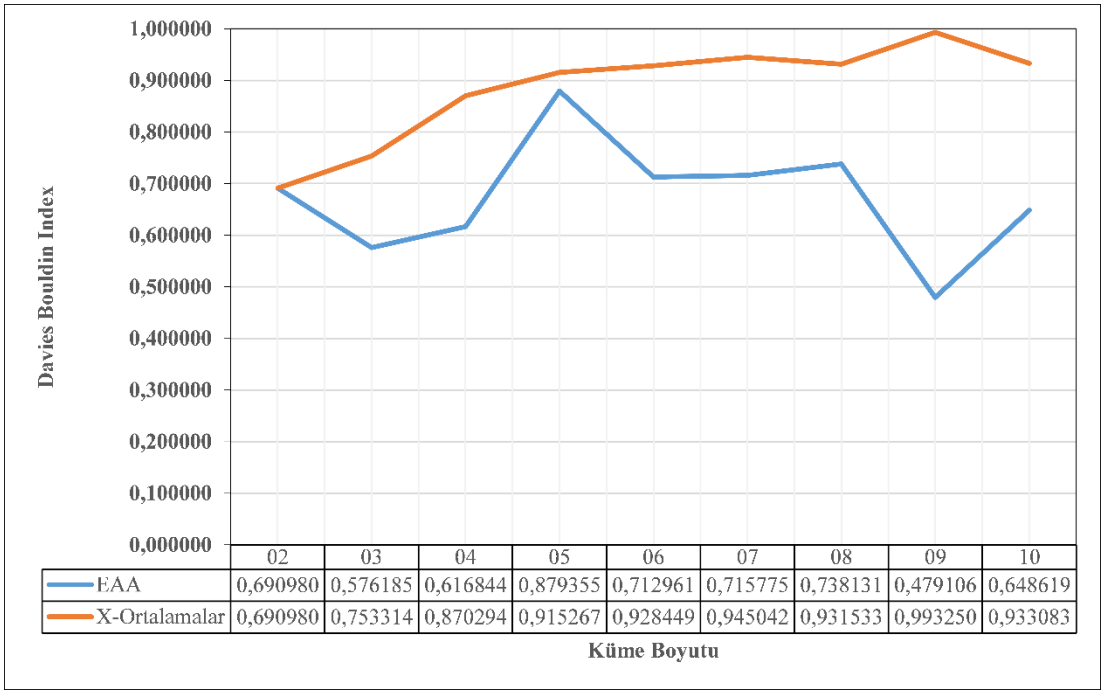
Tablo 5.5. EAA ve X-Ortalamalar algoritmaları için veri setlerinden elde edilen DB-Index değerleridir.

Veri Seti	Algoritma	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
Iris	EAA	0,40429	0,39350	0,67089	0,63155	0,78563	0,80173	0,74129	0,70217	0,52010
	X-Ortalamalar	0,40429	0,99374	0,87238	0,89309	0,87041	0,93865	1,10046	1,18861	1,10818
Wine	EAA	0,478784	0,469158	0,544345	0,396696	0,340718	0,397525	0,293482	0,441473	0,363269
	X-Ortalamalar	0,478784	0,534243	0,546019	0,572231	0,539896	0,582457	0,556637	0,590100	0,508098
Seeds	EAA	0,690980	0,576185	0,616844	0,879355	0,712961	0,715775	0,738131	0,479106	0,648619
	X-Ortalamalar	0,690980	0,753314	0,870294	0,915267	0,928449	0,945042	0,931533	0,993250	0,933083
HCV	EAA	1,01460	0,63705	0,86314	0,62969	0,61198	0,62158	0,52562	0,60620	0,52728
	X-Ortalamalar	1,01460	1,35145	1,27062	1,01429	1,23783	1,25735	1,23565	1,28411	1,25329



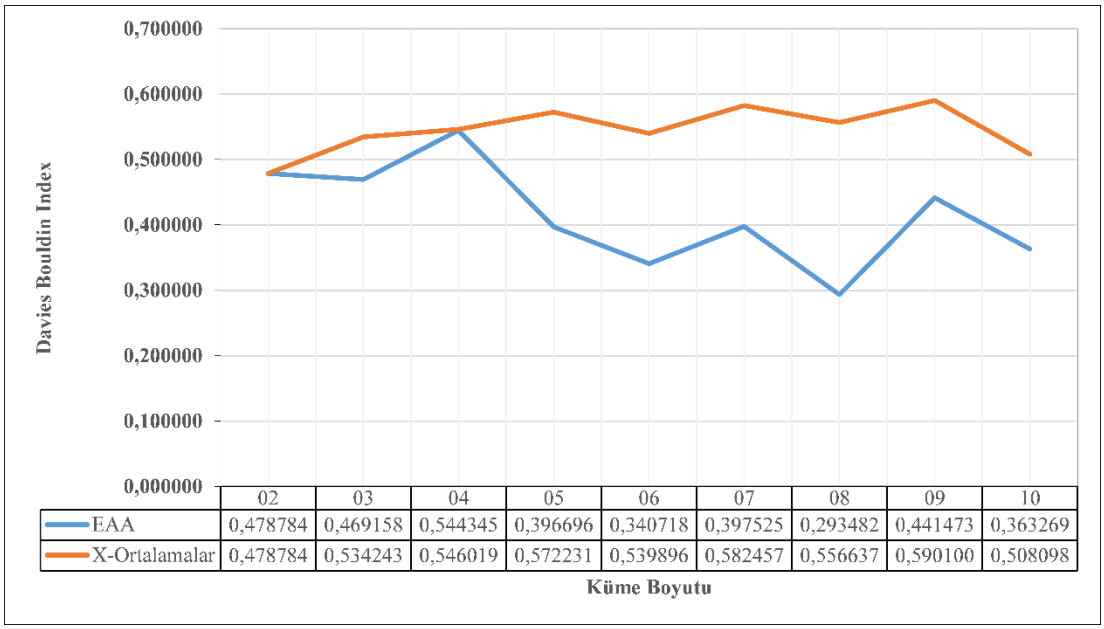
	02	03	04	05	06	07	08	09	10
EAA	0,40429	0,39350	0,67089	0,63155	0,78563	0,80173	0,74129	0,70217	0,52010
X-Ortalamalar	0,40429	0,99374	0,87238	0,89309	0,87041	0,93865	1,10046	1,18861	1,10818

Şekil 5.13. Iris veri seti üzerinde kümeleme işlemi sonucu DB-Index değerleridir.



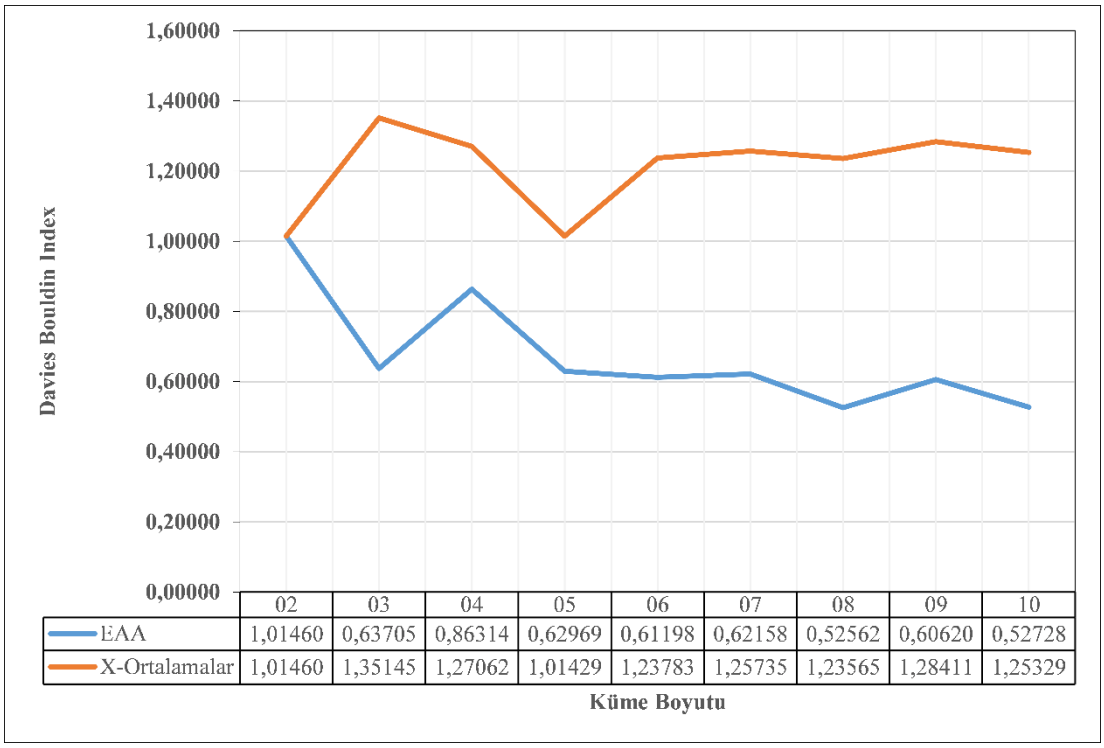
	02	03	04	05	06	07	08	09	10
EAA	0,690980	0,576185	0,616844	0,879355	0,712961	0,715775	0,738131	0,479106	0,648619
X-Ortalamalar	0,690980	0,753314	0,870294	0,915267	0,928449	0,945042	0,931533	0,993250	0,933083

Şekil 5.14. Seeds veri seti üzerinde kümeleme işlemi sonucu DB-Index değerleridir.



	02	03	04	05	06	07	08	09	10
EAA	0,478784	0,469158	0,544345	0,396696	0,340718	0,397525	0,293482	0,441473	0,363269
X-Ortalamalar	0,478784	0,534243	0,546019	0,572231	0,539896	0,582457	0,556637	0,590100	0,508098

Şekil 5.15. Wine veri seti üzerinde kümeleme işlemi sonucu DB-Index değerleridir.



	02	03	04	05	06	07	08	09	10
EAA	1,01460	0,63705	0,86314	0,62969	0,61198	0,62158	0,52562	0,60620	0,52728
X-Ortalamalar	1,01460	1,35145	1,27062	1,01429	1,23783	1,25735	1,23565	1,28411	1,25329

Şekil 5.16. HCV veri seti üzerinde kümeleme işlemi sonucu DB-Index değerleridir.

5.4. 100 Basamak Yarışmasının Sonuçları

100 Haneli Mücadele, Ezop'un Kaplumbağa ve Tavşan arasındaki efsanevi yarışından esinlenmiştir. Masalın ana fikri, başarı kriteri olarak hıza çok fazla güvenmenin her zaman için uygun olmadığıdır. Bu nedenle, yarışmanın birincil amacı, ısrarla veya agresif arama yapan algoritmaların daha iyi olup olmadığını belirlemektir.

Yarışma sonuçları ve analizi [60] numaralı referansta yayınlanmıştır. Yarışmanın sonuçları aşağıda tablo 5.6'da verilmiştir Önerilen EAA algoritmamızın 63,96 puana sahip olduğunu tablo 5.6 üzerinde görebiliriz. Donanım sınırlılıkları sebebiyle iterasyon limitimiz 10E+06 ve popülasyon büyüklüğü 30 olarak ayarlanmıştır. Ancak yarışmada yarışan diğer algoritmaların iterasyon sayılarına bakıldığında ortalama olarak iterasyon sayısı 7*10E+08 ile 8*10E+10 arasında yer almaktadır ve popülasyon büyüklüğü 200 ile 31623 arasındadır. Donanım sınırlamalarına rağmen algoritmamız başarılı sonuçlar elde etmiştir. Algoritmamız daha önce yapılan yarışmanın sonuçlarına [60] göre birincil algoritmalar tablosunda alt sıralardaki algoritmalara yakın yer alırken, ikincil algoritmalar tablosunda orta sıralardaki algoritmalara yakın yer almıştır.

Tablo 5.6. EAA algoritmasının IEEE CEC 2019 100 Basamak Yarışması sonuçlarıdır.

Fonksiyon	Doğru Hane Sayısı											Puan
	0	1	2	3	4	5	6	7	8	9	10	
1	0	0	0	0	0	0	0	0	0	0	50	10
2	0	0	1	1	24	23	0	0	0	0	0	4,32
3	0	0	0	0	0	0	0	0	0	0	50	10
4	0	0	5	11	34	0	0	0	0	0	0	3,58
5	0	0	0	0	0	1	4	34	10	1	0	7,12
6	0	0	0	9	22	4	5	10	0	0	0	4,7
7	0	0	38	2	4	4	2	0	0	0	0	2,6
8	0	0	0	0	0	0	4	2	6	37	1	8,58
9	0	0	0	0	0	0	0	0	0	0	50	50
10	0	0	4	40	5	1	0	0	0	0	0	3,06
					Toplam							63,96

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında, EAA adı verilen doğadan ilham alan yeni bir meta sezgisel optimizasyon algoritması geliştirilmiştir. Yüksek dirençli bölgelerde elektriğin hareketine dayanmaktadır. Diğer meta sezgisel algoritmalar gibi, EAA'nın da arama başlatmak için bir popülasyona ihtiyacı vardır. Bu popülasyon, bir hafızası ve bir ömrü olan arama ajanlarından oluşmaktadır. Ayrıca EAA, negatif ve pozitif kutuplar olarak adlandırılan benzersiz bir yapıya sahiptir. Bu terimleri elektriksel terimlerle eşleştirdiğimizi varsayarsak o halde popülasyonun voltaj gibi, arama ajanlarının elektronlar ve kutup yapısının negatif ve pozitif gibi elektriksel polarite yapısı olduğunu söyleyebiliriz. Bu nedenle, zıt kutuplardaki elektronlar birbirlerine doğru hareket etme eğilimindedir. Bu hareket eğilimi, EAA'nın arama mekanizmasının temelidir.

Önerilen algoritmanın etkinliğini ve güvenilirliğini belirlemek için, EAA dört test fonksiyonu ile test edilmiş ve dört farklı veri seti ile NP-zor bir problem olan kümeleme problemini çözmüştür. Tez çalışmasında kullanılan test fonksiyonları ve veri kümeleri literatürde yaygın olarak kullanılmaktadır. Tüm test fonksiyonları ile test edilen senaryolar yedi farklı algoritma üzerinde karşılaştırılmıştır: GA, PSO, GİA, DAA, DYA, KOA ve HŞO algoritmaları, önerilen EAA algoritmasına benzer arama mekanizmalarına sahip algoritmalardır. Sonuç olarak, önerilen EAA algoritması, GA, PSO, GİA ve DAA algoritmalarına göre daha iyi küresel arama, yerel arama ve yakınsama özellikleri sergilemektedir. DYA, KOA ve HŞO algoritmalarıyla kıyaslandığında ise benzer sonuçlar üretmiştir. Ayrıca, büyük bir yakınsama oranıyla EAA algoritması, GA, PSO, GİA ve DAA algoritmalarından daha iyi sonuçlar sağlamaktadır. Çoğu zaman diğer algoritmalar kesin çözümleri bulamazken, EAA birçok senaryoda kesin çözümleri bulmaktadır. Bunun yanı sıra, istatistiksel bulgular test sonuçlarını desteklemekte ve EAA'nın karşılaştırılan diğer algoritmalardan daha iyi performans gösterdiğini göstermektedir. 100 Basamak Yarışması'nın test sonuçları, algoritmamızın sınırlamalarla bile karmaşık sorunları çözebileceğini göstermektedir. Ayrıca EAA, kümeleme performansı için sorunu verimli bir şekilde

ele almış ve X-Ortalamalar algoritmasından daha iyi bir kümeleme kapasitesi sergilemiştir.

Problemleri çözmek için süre ve iterasyon limitini arttırırsak önerdiğimiz algoritmanın daha iyi sonuçlar verebileceğini görmekteyiz. Önerilen algoritmanın tasarımı nedeniyle, arama mekanizması, değerli arama alanlarını aramak için yavaş ama kalıcı bir hızda çalışmaktadır. Küresel arama için büyük adımlar kullanmak yerine, algoritmamız çoğunlukla küçük adımlar kullanır. Bu nedenle kesin çözümü bulmak için iterasyon sayısı artırılmalıdır. Başarısızlık mekanizması ve ilk başlatma şeması sayesinde, önerilen algoritma zorlu problemlerde bile yerel noktalarda asla takılmamaktadır. Sonuçlar, önerilen algoritmanın kesin çözüme yaklaşabildiğini, ancak iterasyon limitleri nedeniyle bazı durumlarda kesin çözüme asla ulaşamadığını göstermektedir. Önerilen algoritma, başlatma şeması, kutup arama mekanizması ve en iyi çözümlerin güncelleme stratejisi açısından karşılaştırıldığında diğer meta sezgisel algoritmalardan farklıdır.

Küresel ve yerel arama hareketleri için çalışmalarımızda sabit bir katsayı kullanılmıştır. Yavaş hareket etme ve geç yakınsamanın üstesinden gelmek için algoritma üzerinde dinamik bir katsayı uygulanabilir.

Deneylerimizde ahşabın difüzyon katsayısı kullanılmıştır. Gelecekteki çalışmalar için farklı difüzyon katsayıları kullanılabilir ve algoritma üzerindeki etkisi karşılaştırılabilir. Ek olarak, başlangıç popülasyonunun farklı ilk başlatma şemaları incelenebilir.

KAYNAKLAR

[1] Türk Dil Kurumu (2023, 8 Mayıs). Optimizasyon. https://sozluk.gov.tr/

[2] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2), 83-97. https://doi.org/10.1002/nav.3800020109

[3] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107. https://doi.org/10.1109/TSSC.1968.300136

[4] Dechter, R., & Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3), 505-536. https://doi.org/10.1145/3828.3830

[5] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, 1942-1948. https://doi.org/10.1109/ICNN.1995.488968

[6] Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence* (1st MIT Press ed). MIT Press.

[7] Civicioglu, P. (2012). Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers & Geosciences*, 46, 229-247. https://doi.org/10.1016/j.cageo.2011.12.011

[8] Civicioglu, P. (2013). Backtracking Search Optimization Algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219(15), 8121-8144. https://doi.org/10.1016/j.amc.2013.02.017

[9] Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Systems with Applications*, 152, 113377. https://doi.org/10.1016/j.eswa.2020.113377

[10] Minh, H.-L., Sang-To, T., Abdel Wahab, M., & Cuong-Le, T. (2022). A new metaheuristic optimization based on K-means clustering algorithm and its application to structural damage identification. *Knowledge-Based Systems*, *251*, 109189. https://doi.org/10.1016/j.knosys.2022.109189

[11] Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, 97, 849-872. https://doi.org/10.1016/j.future.2019.02.028

[12] Reddy, R., Kulkarni, A. J., Krishnasamy, G., Shastri, A. S., & Gandomi, A. H. (2023). LAB: A leader–advocate–believer-based optimization algorithm. *Soft Computing*. https://doi.org/10.1007/s00500-023-08033-y

[13] Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic algorithms: A comprehensive review. *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* içinde (ss. 185-231). Elsevier. https://doi.org/10.1016/B978-0-12-813314-9.00010-4

[14] Nanda, S. J., & Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation, 16*, 1-18. https://doi.org/10.1016/j.swevo.2013.11.003

[15] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1*, 281-298.

[16] Kaya, Y., Uyar, M., & Tekin, R. (2011). *A novel crossover operator for genetic algorithms: Ring crossover*. https://doi.org/10.48550/ARXIV.1105.0355

[17] Jalali Varnamkhasti, M., Lee, L. S., Abu Bakar, M. R., & Leong, W. J. (2012). A genetic algorithm with fuzzy crossover operator and probability. *Advances in Operations Research*, *2012*, 1-16. https://doi.org/10.1155/2012/956498

[18] Vrajitoru, D. (1998). Crossover improvement for the genetic algorithm in information retrieval. *Information Processing & Management*, *34*(4), 405-415. https://doi.org/10.1016/S0306-4573(98)00015-6

[19] Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, 24*(4), 656-667. https://doi.org/10.1109/21.286385

[20] Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and Its Applications*, *391*(6), 2193-2196. https://doi.org/10.1016/j.physa.2011.12.004

[21] I. Abuiziah & N. Shakarneh. (2014). *A review of genetic algorithm optimization: Operations and applications to water pipeline systems*. https://doi.org/10.5281/ZENODO.1090464

[22] Sivaraj, R., & Ravichandran, T. (2011). A review of selection methods in genetic algorithm. *International Journal of Engineering Science & Technology*, *3*(5), 3792-3797.

[23] Razali, N. M., & Geraghty, J. (2011). Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. *Proceedings of the World Congress on Engineering 2011*, *2*, 1134-1139.

[24] Abdoun, O., & Abouchabaka, J. (2011). A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. *International Journal of Computer Applications*, *31*(11), 49-57.

[25] Bezdek, J. C., Boggavarapu, S., Hall, L. O., & Bensaid, A. (1994). Genetic algorithm guided clustering. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, 34-39. https://doi.org/10.1109/ICEC.1994.350046

[26] Sarkar, M., Yegnanarayana, B., & Khemani, D. (1997). A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters*, *18*(10), 975-986. https://doi.org/10.1016/S0167-8655(97)00122-0

[27] Kuncheva, L., & Bezdek, J. C. (1997). Ection Of Cluster Prototypes From Data By A Genetic Algorithm. *5th European Congress on Intelligent Techniques and Soft Computing*, 1683-1688.

[28] Murthy, C. A., & Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, *17*(8), 825-832. https://doi.org/10.1016/0167-8655(96)00043-8

[29] Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition*, *33*(9), 1455-1465. https://doi.org/10.1016/S0031-3203(99)00137-5

[30] Krishna, K., & Narasimha Murty, M. (1999). Genetic K-means algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *29*(3), 433-439. https://doi.org/10.1109/3477.764879

[31] Weiguo Sheng & Xiaohui Liu. (2004). A hybrid algorithm for k-medoid clustering of large data sets. *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, 77-82. https://doi.org/10.1109/CEC.2004.1330840

[32] Lu, Y., Lu, S., Fotouhi, F., Deng, Y., & Brown, S. J. (2004). Fgka: A fast genetic k-means clustering algorithm. *Proceedings of the 2004 ACM Symposium on Applied Computing*, 622-623. https://doi.org/10.1145/967900.968029

[33] Lu, Y., Lu, S., Fotouhi, F., Deng, Y., & Brown, S. J. (2004). Incremental genetic k-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics*, *5*(1), 172. https://doi.org/10.1186/1471-2105-5-172

[34] Cowgill, M. C., Harvey, R. J., & Watson, L. T. (1999). A genetic algorithm approach to cluster analysis. *Computers & Mathematics with Applications*, *37*(7), 99-108. https://doi.org/10.1016/S0898-1221(99)00090-5

[35] Tseng, L. Y., & Bien Yang, S. (2001). A genetic approach to the automatic clustering problem. *Pattern Recognition*, *34*(2), 415-424. https://doi.org/10.1016/S0031-3203(00)00005-4

[36] Bandyopadhyay, S., & Maulik, U. (2001). Nonparametric genetic clustering: Comparison of validity indices. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *31*(1), 120-125. https://doi.org/10.1109/5326.923275

[37] Bandyopadhyay, S., & Maulik, U. (2002). Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, *35*(6), 1197-1208. https://doi.org/10.1016/S0031-3203(01)00108-X

[38] Van Der Merwe, D. W., & Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, 215-220. https://doi.org/10.1109/CEC.2003.1299577

[39] Cohen, S. C. M., & De Castro, L. N. (2006). Data clustering with particle swarms. *2006 IEEE International Conference on Evolutionary Computation*, 1792-1798. https://doi.org/10.1109/CEC.2006.1688524

[40] Chuang, L.-Y., Hsiao, C.-J., & Yang, C.-H. (2011). Chaotic particle swarm optimization for data clustering. *Expert Systems with Applications*, *38*(12), 14555-14563. https://doi.org/10.1016/j.eswa.2011.05.027

[41] Yang, F., Sun, T., & Zhang, C. (2009). An efficient hybrid data clustering method based on K-harmonic means and Particle Swarm Optimization. *Expert Systems with Applications*, *36*(6), 9847-9852. https://doi.org/10.1016/j.eswa.2009.02.003

[42] Huang, K. Y. (2011). A hybrid particle swarm optimization approach for clustering and classification of datasets. *Knowledge-Based Systems*, *24*(3), 420-426. https://doi.org/10.1016/j.knosys.2010.12.003

[43] Xu, R., Xu, J., & Wunsch, D. C. (2010). Clustering with differential evolution particle swarm optimization. *IEEE Congress on Evolutionary Computation*, 1-8. https://doi.org/10.1109/CEC.2010.5586257

[44] Einstein, A., & Brown, R. (1967). *Investigations on the theory of the Brownian movement* (Repr. of Methuen ed. 1926). Dover Publ.

[45] Gorn, M. the. (2009). *English: Rosenbrock function over*. Own work. https://commons.wikimedia.org/wiki/File:Rosenbrock_function.svg#

[46] Gaortizg. (2012). *English: This function is used as a test function in order to evaluate the performance of optimization algorithms.* https://commons.wikimedia.org/wiki/File:Ackley%27s_function.pdf

[47] Gisling. (2015). *English: 2nd order Griewank function 3D plot*. Own work. https://commons.wikimedia.org/wiki/File:Griewank_function_3D_plot.png

[48] Diegotorquemada. (2010). *English: Rastrigin function*. Own work. https://commons.wikimedia.org/wiki/File:Rastrigin_function.png

[49] R.A. Fisher. (1936). *Iris* [Veri Seti]. UCI Machine Learning Repository. https://doi.org/10.24432/C56C76

[50] M. Forina. (1992). *Wine* [Veri Seti]. UCI Machine Learning Repository. https://doi.org/10.24432/C5PC7J

[51] Magorzata Charytanowicz, J. N. (2010). *Seeds* [Veri Seti]. UCI Machine Learning Repository. https://doi.org/10.24432/C5H30K

[52] Lichtinghagen, R., Klawonn, F., & Hoffmann, G. (2020). *HCV data* [Veri Seti]. UCI Machine Learning Repository. https://doi.org/10.24432/C5D612

[53] Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224-227. https://doi.org/10.1109/TPAMI.1979.4766909

[54] Price, K. V., Awad, N. H., Ali, M. Z., & Suganthan, P. N. (2018). *Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization* [Teknik Rapor]. Nanyang Technological University.

[55] Conover, W. J. (1999). *Practical nonparametric statistics* (3rd ed). Wiley.

[56] Friedman, M. (1937). 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. https://doi.org/10.1080/01621459.1937.10503522

[57] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, *11*(1), 86-92. https://doi.org/10.1214/aoms/1177731944

[58] Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures: Third edition* (3rd ed). CRC Press.

[59] Wilcoxon, F. (1946). Individual comparisons of grouped data by ranking methods. *Journal of Economic Entomology*, *39*(2), 269-270. https://doi.org/10.1093/jee/39.2.269

[60] Price, K. V., Awad, N. H., Ali, M. Z., & Suganthan, P. N. (2019). *The 2019 100-Digit Challenge on Real-Parameter,Single Objective Optimization: Analysis of Results* [Teknik Rapor]. Nanyang Technological University.

ÖZGEÇMİŞ

Ad-Soyad : Hüseyin DEMİRCİ

ÖĞRENİM DURUMU:

- **Lisans :** 2012, Selçuk Üniversitesi, Teknik Eğitim Fakültesi, Bilgisayar Sistemleri Öğretmenliği
- **Yükseklisans :** 2015, Sakarya Üniversitesi, Bilgisayar ve Bilişim Bilimleri Mühendisliği, Bilgisayar Mühendisliği Tezli Yüksek Lisans

MESLEKİ DENEYİM:

- 2013 yılından beri Sakarya Üniversitesi'nde Araştırma Görevlisi olarak çalıştı.

TEZDEN TÜRETİLEN ESERLER:

- Demirci, H., Yurtay, N., Yurtay, Y., & Zaimoğlu, E. A. (2022). Electrical search algorithm: A new metaheuristic algorithm for clustering problem. Arab*ian Journal for Science and Engineering*. https://doi.org/10.1007/s13369-022-07545-3