T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KÜMELEME PROBLEMLERİ İÇİN GELİŞTİRİLMİŞ TEK ADAY OPTİMİZASYON ALGORİTMASI

YÜKSEK LİSANS TEZİ

CİHAT DOĞAN

TEZ DANIŞMANI

PROF. DR. UĞUR YÜZGEÇ

BİLECİK, 2024

10604065

T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KÜMELEME PROBLEMLERİ İÇİN GELİŞTİRİLMİŞ TEK ADAY OPTİMİZASYON ALGORİTMASI

YÜKSEK LİSANS TEZİ

CİHAT DOĞAN

TEZ DANIŞMANI

PROF. DR. UĞUR YÜZGEÇ

BİLECİK, 2024

10604065

BEYAN

"**Kümeleme Problemleri için Geliştirilmiş Tek Aday Optimizasyon Algoritması**" adlı yüksek lisans tezinin hazırlık ve yazımı sırasında bilimsel araştırma ve etik kurallarına uyduğumu, başkalarının eserlerinden yararlandığım bölümlerde bilimsel kurallara uygun olarak atıfta bulunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, tezin herhangi bir kısmının Bilecik Şeyh Edebali Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını, aksinin tespit edileceği muhtemel durumlarda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

<table>
<tr><td colspan="4">Bu çalışmanın,
Bilimsel Araştırma Projeleri (BAP), TÜBİTAK veya benzeri kuruluşlarca desteklenmesi durumunda; projenin ve destekleyen kurumun adı proje numarası ile birlikte, ETİK KURUL onayı alınması durumunda ise ETİK KURUL tarih karar ve sayı bilgilerinin beyan edilmesi gerekmektedir.</td></tr>
<tr><td>DESTEK ALINMIŞTIR</td><td></td><td>DESTEK ALINMAMIŞTIR</td><td>X</td></tr>
<tr><td colspan="4">Destek alındı ise;</td></tr>
<tr><td colspan="4">Destekleyen kurum;</td></tr>
<tr><td colspan="2">Desteğin Türü</td><td colspan="2">Proje Numarası</td></tr>
<tr><td colspan="2">1- BAP (Bilimsel Araştırma Projesi)</td><td colspan="2" rowspan="3"></td></tr>
<tr><td colspan="2">2- TÜBİTAK</td></tr>
<tr><td colspan="2">Diğer;……</td></tr>
<tr><td colspan="4">ETİK KURUL onayı var ise;</td></tr>
<tr><td colspan="2">ETİK KURUL karar tarih/sayı:</td><td colspan="2">……/……</td></tr>
</table>

Cihat DOĞAN

Tarih

……..

İmza

……..

ÖN SÖZ

Bu tez yeterlik çalışmasının yazılmasında, çalışmamı sahiplenerek takip eden danışmanım Sayın Prof. Dr. Uğur YÜZGEÇ'e değerli katkı ve emekleri için teşekkürlerimi ve saygılarımı sunarım.

Savunma sınavı sırasında değerli jüri üyeleri Sayın Prof. Dr. Uğur YÜZGEÇ'e, Sayın Prof. Dr. Cihan KARAKUZU'ya ve Sayın Doç. Dr. Serhat YILMAZ'a çalışmamın son haline gelmesindeki değerli katkıları adına teşekkürlerimi ve saygılarımı sunarım.

Son olarak bu günlere ulaşmamdaki emekleri adına değerli aileme teşekkür ederim.

Cihat DOĞAN

2024

ÖZET

KÜMELEME PROBLEMLERI İÇİN GELİŞTİRİLMİŞ TEK ADAY OPTİMİZASYON ALGORİTMASI

Optimizasyon problemlerinin etkili bir şekilde ele alınması için, verimli optimizasyon algoritmalarının geliştirilmesi büyük önem taşımaktadır. Bu tez çalışması kapsamında, 2022 yılında Shami ve arkadaşları tarafından (Shami, Grace, Burr, & Mitchell, 2022) literatüre kazandırılan Tek Aday Optimizasyonu (SCO: Single Candidate Optimizatin) Algoritması ele alınmaktadır.

Bu çalışmada ele alınan SCO algoritması, basit ve anlaşılır bir algoritma olup diğer popülasyon tabanlı sezgisel algoritmalardan en büyük farkı, tek bir aday çözüm ile optimizasyon probleminin çözümünü daha hızlı bulmaya çalışmasıdır. Ancak diğer sezgisellerde olan yerel minimumlara takılma gibi temel sorunlara da sahiptir. Öncelikle SCO yapısında sınır değeri aşma problemi (aday çözümün arama uzayının dışına çıkması) ele alınmış ve arama başarımını iyileştirmek için, hızlandırılmış karşıt öğrenme tabanlı bir mekanizma algoritma yapısına entegre edilmiştir. Böylece bu çalışmada, SCO algoritmasına hızlandırılmış karşıt öğrenme mekanizması eklenerek elde edilen, Hızlandırılmış Karşıt Öğrenme tabanlı Tek Aday Optimizasyonu (AccOppSCO: Accelerated Opposition Learning based Single Candidate Optimization) adı verilen yeni bir optimizasyon algoritması önerilmiştir.

Önerilen AccOppSCO algoritmasının başarımını değerlendirmek için, literatürden çeşitli optimizasyon problemleri seçilmiştir. Yapılan değerlendirme, AccOppSCO algoritmasının orijinal SCO algoritmasına kıyasla daha doğru çözümler üretebildiğini ortaya koymaktadır.

Önerilen AccOppSCO algoritmasının başarımı, ayrıca bir kümeleme problemi üzerinde değerlendirilmiştir. Kümeleme probleminde önerilen AccOppSCO algoritması, orijinal SCO algoritmasına kıyasla daha üstün yakınsama göstermektedir.

Son olarak, önerilen AccOppSCO algoritması; Genetik Algoritma (GA), Farksal Gelişim (DE) ve Parçacık Sürüsü Optimizasyonu (PSO) gibi klasik sezgisel optimizasyon algoritmalarıyla karşılaştırılmıştır. Sonuçlara göre, AccOppSCO algoritması yakınsama ve çözüm kalitesi açısından orijinal SCO algoritmasından daha iyi başarım göstermektedir.

Anahtar Kelimeler: Kıyaslama, Kümeleme, Sezgisel, Tek Aday Optimizasyon.

ABSTRACT

IMPROVED SINGLE CANDIDATE OPTIMIZATION ALGORITHM FOR CLUSTERING PROBLEMS

In order to effectively address optimization problems, the development of efficient optimization algorithms is of great importance. This thesis focuses on the Single Candidate Optimization (SCO) algorithm introduced in the literature by Shami et al. in 2022 (Shami, Grace, Burr, & Mitchell, 2022).

The SCO algorithm discussed in this study is a simple and comprehensible algorithm, and its main distinction from other population-based heuristic algorithms is its attempt to find the solution to the optimization problem more quickly using a single candidate solution. However, it also faces fundamental issues such as getting stuck in local optima, similar to other heuristics. Initially, the problem of boundary violation (the candidate solution exceeding the search space) has been addressed within the SCO structure, and an accelerated opposition learning-based mechanism has been integrated into the algorithm's structure to improve search performance. Thus, in this study, a new optimization algorithm named Accelerated Opposition Learning based Single Candidate Optimization (AccOppSCO) has been proposed by incorporating the accelerated opposition learning mechanism into the SCO algorithm.

To evaluate the performance of the proposed AccOppSCO algorithm, various optimization problems from the literature have been selected. The evaluation indicates that the AccOppSCO algorithm is capable of producing more accurate solutions compared to the original SCO algorithm.

The performance of the proposed AccOppSCO algorithm is also evaluated on a clustering problem. In the clustering problem, the proposed AccOppSCO algorithm demonstrates superior convergence compared to the original SCO algorithm.

Finally, the proposed AccOppSCO algorithm is compared with classical heuristic optimization algorithms such as Genetic Algorithm (GA), Differential Evolution (DE) and Particle Swarm Optimization (PSO). According to the results, the AccOppSCO algorithm performs better than the original SCO algorithm in terms of convergence and solution quality.

Keywords: Benchmark, Clustering, Heuristic, Single Candidate Optimization.

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

1. GİRİŞ

Optimizasyon, verilen şartlar altında en iyi sonucun elde edilmesi işlemidir. Bir fonksiyonun maksimum ya da minimum değerini, verilen şartları da sağlayarak araştırmaya optimizasyon problemi adı verilir. Optimizasyon problemlerini incelemek ve çözmek için geliştirilen yöntemlerin tümüne optimizasyon teknikleri adı verilmektedir (Kahraman, 2010). Optimizasyon problemleri bilim, mühendislik, ekonomi ve diğer disiplinler dahil olmak üzere çeşitli alanlarda büyük önem taşımaktadır. Bu tür problemlerin birincil amacı, bir dizi kısıtlamayı yerine getirirken mümkün olan en iyi sonucu elde etmek veya belirli bir hedefi optimize etmektir. Farklı yaklaşımlarla optimum çözümü bulmak için çok sayıda optimizasyon tekniği kullanılır. Ancak bazı optimizasyon problemlerinde mevcut yöntemler yetersiz kalabilir ve bu da sürekli olarak daha etkili çözümler arayışına yol açar.

Sezgisel optimizasyon algoritmaları, dünya üzerindeki gerçek karmaşık problemlere matematiksel modelleme, algoritma ve istatistik gibi bilimsel yöntemlerden faydalanarak çözümler sunmaktadır (Özçelik & Gündüz, 2019). Sezgisel algoritmalar, verimli ve uyarlanabilir yaklaşımlar sunarak karmaşık problemlerin üstesinden gelmede temel bir rol oynar. Bu algoritmalar, maliyet fonksiyonunu optimize etmek ve en uygun çözümleri aramak için doğal olaylardan veya sosyal davranışlardan ilham alır. Uygulama alanları, mühendislik, işletme, yapay zeka ve bilgisayar bilimi dahil olmak üzere geniş bir alanı kapsar. Sezgisel algoritmalara örnek olarak; Genetik Algoritma (GA) (Kumar, Husain, Upreti, & Gupta, 2010), Parçacık Sürüsü Optimizasyonu (PSO) Algoritması (Kennedy & Eberhart, 1995), Karınca Kolonisi Optimizasyonu (ACO) Algoritması (Dorigo & Di Caro, 1999), Benzetilmiş Tavlama (SA) Algoritması (Rutenbar, 1989), Farksal Gelişim (DE) Algoritması (Price, Storn, & Lampinen, 2005) ve Yapay Arı Kolonisi (ABC) Algoritması (Karaboga, Gorkemli, Ozturk, & Karaboga, 2012) verilebilir. GA ve PSO gibi popülasyon tabanlı sezgisel algoritmalar belirli sakıncalara sahip olabilirler. Bu algoritmalar çok sayıda aday çözümün sürdürülmesini ve güncellenmesini gerektirdiğinden, önemli sorunlardan biri yüksek hesaplama maliyetidir. Aynı zamanda bu algoritmaların arama başarımı, parametrelerinin uygun şekilde ayarlanmasına büyük ölçüde bağlıdır.

Tek Aday Optimizasyonu (SCO) algoritması, geleneksel sezgisel arama algoritmalarından, özellikle popülasyon tabanlı yöntemlere dayalı olanlardan farklı, yeni bir stratejidir (Shami, Grace, Burr, & Mitchell, 2022). SCO algoritması basit, düşük parametreli, düşük maliyetli ve yüksek performanslı bir sezgisel algoritmadır (Emek & Yüzgeç, 2024a). Popülasyon tabanlı sezgisel yöntemlerin aksine SCO, iyileştirilmiş çözümler elde etme hedefiyle, optimizasyon süreci boyunca yalnızca tek bir aday çözüme odaklanır. Optimizasyon süreci, her biri aday çözümün konumunu güncellemek için farklı yaklaşımlar kullanan iki aşamadan oluşur. Tek çözüme dayalı algoritmalar ve iki fazlı yaklaşımlar tarihsel olarak ayrı sezgisel optimizasyon yöntemleri olarak kabul edilirken, SCO bu kavramları birleşik ve sağlam bir algoritmada birleştirir. Bu algoritmanın dikkate değer bir yönü, aday çözümün konumunu yalnızca mevcut konumu gibi kendi bilgilerine dayalı olarak güncelleyen benzersiz bir denklem setinin uygulanmasıdır. SCO, bu unsurları entegre ederek optimizasyon problemlerini çözmek için etkili bir yaklaşım sağlar. SCO algoritması belirli avantajlar sunarken, potansiyel sınırlamalarını dikkatle değerlendirmek önemlidir. Tek bir aday çözüme dayalı yapılan arama stratejisine güvenmek, çözüm uzayını kapsamlı bir şekilde keşfetme yeteneğini engelleyebilir. Çeşitli aday çözümlerin yokluğu, yerel minimumlardan kaçmada zorluklara yol açabilir ve algoritmanın genel optimumu bulma yeteneğini sınırlayabilir. Ayrıca, SCO'nun başarımı, elde edilen nihai çözümün kalitesini etkileyebilecek olan ilk aday çözümün konumuna duyarlıdır. Diğer bir sakınca, SCO tarafından kullanılan keşif mekanizmasının, aday çözümün konumunun hızla sıfır noktasına yakınsamasına neden olarak, optimal çözümün sıfır olmayan çözüm noktalarındayken yakınsamayı potansiyel olarak geciktirebilmesidir.

Karşıt tabanlı öğrenme, aday çözümlerin yalnızca orijinal konum değerlerine göre değil aynı zamanda karşıt konum karşılıklarına göre değerlendirilmesini içeren karşıtlık kavramına dayanmaktadır (Tizhoosh, 2005). Karşıt tabanlı öğrenme mekanizması, optimizasyon algoritmalarının keşif kabiliyetini ve yakınsama hızını artırmayı amaçlamaktadır. Karşıt çözümleri göz önünde bulundurarak, arama sürecine ek çeşitlilik getirir. Bu çeşitlilik, algoritmanın arama alanının farklı bölgelerini eş zamanlı olarak keşfetmesini sağlar, bu da daha kapsamlı bir keşfe yol açabilir ve daha iyi çözümler bulma olasılığını artırabilir.

Bu tez çalışmasında, farklı sezgisel algoritma geliştirme yöntemleri (karşıt öğrenme, mutasyon operatörleri, sınır değeri güncellenmesi gibi) ile algoritmanın başarımının arttırılarak, arama yeteneğinin geliştirilmesi amaçlanmıştır. Bu kapsamda, hızlandırılmış karşıt öğrenme tabanlı bir mekanizma SCO algoritmasının yapısına entegre edilerek Hızlandırılmış Karşıt Öğrenme Tabanlı SCO (AccOppSCO) algoritması elde edilmiştir. Yeni önerilen AccOppSCO algoritmasının başarımını test etmek için, literatürden iyi bilinen 23 minimizasyon problemi alınmış ve kıyaslama için karşılaştırmalı testleri gerçekleştirilmiştir. Elde edilen sonuçlar, AccOppSCO algoritmasının orijinal SCO algoritmasına göre daha başarılı olduğunu göstermektedir.

Son olarak, geliştirilmiş SCO yapısı kümeleme problemi için uyarlanarak başarım yönünden popüler üç meta-sezgisel ile karşılaştırılıp değerlendirilmesi yapılmış olup bunlar: Genetik operatörler yardımıyla evrim mekanizmasını uygulayan araştırma algoritmaları olan gerçek kodlu Genetik Algoritma (GA), Kuş ve balık sürülerinin sosyal davranışları gözlemlenerek geliştirilen popülasyon temelli bir optimizasyon algoritması olan Parçacık Sürü Optimizasyonu (PSO) Algoritması ve çaprazlama, mutasyon ve seçim gibi genetik algoritmalarda bulunan benzer operatörleri kullanan popülasyon tabanlı bir algoritma olan Farksal Gelişim (DE) Algoritmasıdır.

SCO Algortması alanındaki literatür taramasında, aşağıdaki çalışmalara ulaşılmıştır;

Dokur, Şengör, Erdoğan ve Yüzgeç (2023) tarafından "SCO-MLP Tabanlı Gerçek Bir Dağıtım Şebekesi İçin Akıllı Sayaç Veri Odaklı Voltaj Tahmin Modeli" başlıklı yapılan çalışmada, Düşük voltajlı (LV) dağıtım şebekelerinde doğru düğüm voltajı tahmini için Tek Aday Optimizasyon Cihazı (SCO)-Çok Katmanlı Algılayıcı (MLP) adlı yeni bir makine öğrenimi modelini tanıtmışlardır. Önerilen modelin, ölçülen değerlere göre ortalama 1,296 voltluk bir sapma ile umut verici bir voltaj tahmin yeteneği sergilediğini ortaya çıkarmışlardır (Dokur, Şengör, Erdoğan, & Yüzgeç, 2023).

Emek ve Yüzgeç (2024a) tarafından "Kaotik Mutasyon Stratejisine Dayalı SCO Algoritmasının Mühendislik Tasarım Problemlerine Uygulanması" başlıklı yapılan çalışmada, kaotik mutasyon fonksiyonlarının CSCO'ya entegre edilmesi, keşif ve yerelde arama arasında bir denge sağlanmasına yardımcı olmuş ve üstün optimizasyon sonuçları elde edilmiştir. (Emek & Yüzgeç, 2024a).

Emek ve Yüzgeç (2024b) tarafından “Kaotik Mutasyon Stratejisine Dayalı Tek Aday Optimizasyon Algoritması” başlıklı yapılan çalışmada, Chaucy, Gaussian ve Levy gibi kaotik fonksiyonlara dayalı yeni bir mutasyon tekniğinin, SCO algoritmasının performansını iyileştirdiği görülmüştür (Emek & Yüzgeç, 2024b).

JP Appadurai, LS Raveendran, R. Latha, E. Gangadevi, ML Shri ve S. Gite (2024) tarafından “SCO Optimizasyon Tekniğiyle Sınıflandırma İçin Hibrit Tip Model Kullanan Siber Güvenlik” başlıklı yapılan çalışmada, Ağ Saldırısı Algılama Sistemlerini güçlendirmek için geliştirilmiş bir yaklaşım önerilmiştir. Önerilen yaklaşım, SCO'nun hiperparametre optimizasyonunu ve özellik seçimini nasıl iyileştirebileceğini, daha doğru ve siber saldırıya dayanıklı bir NIDS'ye nasıl yol açabileceğini göstermiştir (Appadurai, ve diğerleri, 2024).

RajyaLakshmi, T., Vinta, KSR (2024) tarafından “Karmaşık kelime tanımlama ve eş anlamlı kelime üretimi için etkili derin öğrenme tabanlı Idrcnn ve Bdc-Lstm modelleri” başlıklı yapılan çalışmada, Geliştirilmiş Derin Kalıntı Evrişimsel Sinir Ağı (IDRCNN) tekniğinin parametreleri ayarlanarak ve optimize edilerek sınıflandırma performansını artırmak için SCO Algoritması kullanılmıştır. Deneysel sonuçlara göre, önerilen yaklaşım alternatif yaklaşımlardan daha iyi performans göstermiştir (RajyaLakshmi & Vinta, 2024).

SEVS Pillai, R. Vallabhaneni, PK Pareek ve S. Dontu (2024) tarafından “Gelişmiş Ağ Saldırısı Algılama Sistemi için SCO Optimizasyonu ile Hibrit Sınıflandırma Modeli Kullanılarak Siber Güvenliğin Güçlendirilmesi” başlıklı yapılan çalışmada, Ağ Saldırısı Algılama Sistemlerinin (NIDS) iyileştirilmesi önerilmiştir. SCO, daha doğru ve siber saldırıya dayanıklı bir NIDS oluşturmak için hiperparametre optimizasyonunu ve özellik seçimini iyileştirmiş, önerilen model tüm metriklerde en yüksek değerlerle öne çıkma açısından olağanüstü bir performans göstermiştir (Pillai, Vallabhaneni, Pareek, & Dontu, 2024).

W. Chagra ve SB Attia (2024) tarafından “Stokastik Yöntemlerle Eğitilen RBF Sinir Ağına Dayalı Doğrusal Olmayan Model Tahmini Kontrol” başlıklı yapılan çalışmada, SCO algoritması tarafından eğitilen Doğrusal Olmayan Model Tahmini stratejisi önerilmiştir. SCO algoritmasıyla elde edilen model, daha az gizli düğüm gerektirirken üstün izleme performansı göstermiştir (Chagra & Attia, 2024).

2. KÜMELEME

Kümeleme, birbirine daha çok benzeyen verilerin gruplara ayrılabilmesi için makine öğrenimi ve istatistik alanlarında geliştirilen bir yöntemdir. Kümeleme, bir grup veri nesnesinin bir grubun üyelerinin benzerliğini maksimuma çıkaracak ve diğer yandan iki farklı gruptaki üyelerin benzerliğini en aza indirecek şekilde gruplandırılmasıdır. Kümeleme; İş ve Pazarlama, Finans, Sosyal Bilimler, Saha Robotiği, Matematiksel Kimya, İklim Bilimi, Petrol Jeolojisi, Jeokimya, World Wide Web, Bilgisayar Bilimi, Tıp, Biyoloji ve Biyoinformatik gibi birçok alanda kullanılmaktadır.

2.1. Veri

Veri, bilgisayarların algıladığı, işlediği veya depoladığı sayılar, semboller, metinler, görüntüler, sesler gibi her türlü bilgiye denilir. Bilimsel çalışmalarda veri; bir araştırma veya deneyin sonucunda elde edilen ölçümler, gözlemler, analizler veya sonuçlar olabilir. Veri, bir şeyin gerçek durumunu anlamak, öngörü yapmak veya kararlar almak için kullanılan temel bir unsurdur. Ancak, toplanan ve depolanan veri kendi başına anlam ifade etmez; onu anlamlı hale getiren, doğru bir bağlamda analiz edilmesi ve yorumlanmasıdır. Bu süreç, bilim ve teknolojide veri odaklı kararların alınmasını sağlar.

2.2. Veri Madenciliği

Günümüz bilgi çağında, her gün milyarlarca veri parçası oluşturulmaktadır. Bu büyük veri yığınlarındaki desenleri, ilişkileri ve anlamlı bilgileri çıkarmak için makine öğrenimi, istatistiksel analiz, veri tabanı yönetimi ve veri görselleştirme gibi veri madenciliği yöntemleri kullanılmaktadır.

Yapay zeka, kendini geliştirebilen sistemler üzerine kuruludur. Ancak bu gelişme, doğru ve kaliteli veriye dayanır. Veri madenciliği, makine öğrenimi ve yapay zeka gibi teknolojilerin temelini oluşturur ve bu teknolojilerin başarımını arttırmada gerekli olan büyük veri kümelerini elde etmek için kullanılır. Yapay zeka, veri madenciliği ile elde edilen verileri kullanarak, makine öğrenimi ve diğer yöntemlerle sistemleri eğitir ve otomatikleştirir. Böylece daha akıllı ve veri odaklı sistemler oluşturulabilir.

Veri madenciliğinde; veri analizi için programlama dilleri (Python, R), veritabanı yönetim sistemleri (Oracle, SQL), veri görselleştirme araçları (Tableau, Excel), makine öğrenimi kütüphaneleri (TensorFlow, Scikit-learn, Keras) ve büyük veri işleme araçları (Hadoop, Spark, Storm) gibi birçok teknoloji ve araç kullanılmaktadır.

2.2.1. Veri Madenciliğinin Aşamaları

Veri madenciliği aşamaları aşağıdaki adımlardan oluşur:

1. Veri Toplama: İlk adım, analiz için gerekli verilerin toplanmasıdır. Bu süreçte, yapılandırılmış veya yapılandırılmamış veriler farklı kaynaklardan alınarak bir veri seti oluşturulabilir.

2. Veri Ön İşleme: Veri madenciliği için kullanılan veri setleri genellikle karmaşık olabilir. Bu aşamada; veri seti temizlenir, eksik değerler doldurulur, gereksiz veriler çıkarılır ve veri normalleştirilir.

3. Modelleme: Veri setine uygun bir model seçilir ve veri madenciliği teknikleri kullanılarak modele veri uygulanır.

4. Model Değerlendirme: Elde edilen sonuçların geçerliliğini kontrol etmek için, keşfedilen modelin doğruluğu ve kullanışlılığı değerlendirilir ve anlamlı bilgiler elde edilir.

5. Sonuçların Uygulanması: Elde edilen bilgi ve desenler, karar verme süreçlerine veya yeni stratejilere entegre edilir.

2.2.2. Veri Madenciliği Yöntemleri

Veri madenciliğinde, yöntemin uygulanacağı veri setine ve uygulama yapılacak alana göre farklı yöntemler kullanılmaktadır. Bilgi keşfi sürecinde tanımlanan probleme göre, veri analiz aşamasında kullanılacak yöntemler belirlenir. Bu yöntemler aşağıda yer almaktadır:

- Sınıflandırma: Verilerin belirli kategorilere veya sınıflara atanmasını sağlar.
- Kümeleme: Veri noktalarını benzer özelliklere sahip gruplara ayırır.
- Regresyon Analizi: Değişkenler arasındaki ilişkiyi anlamak için kullanılır.
- Derin Öğrenme: Yapay sinir ağları ve büyük veri kümeleriyle karmaşık desenlerin keşfedilmesi için kullanılır.

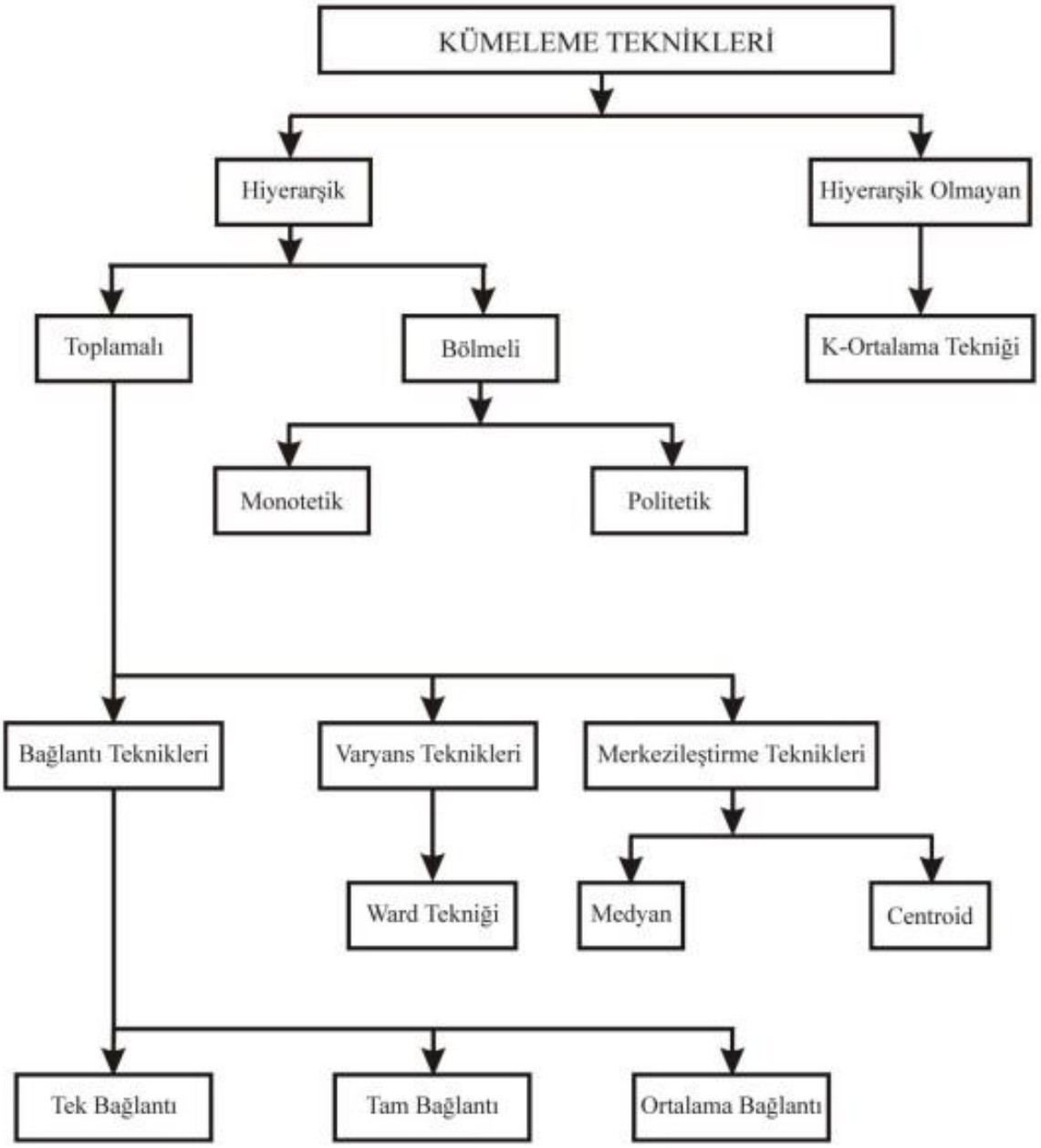
Bu çalışmada, veri madenciliği yöntemlerinden kümeleme yöntemi ele alınmaktadır.

2.3. Kümeleme Yöntemi

Kümelemede, benzerlikler denetimsiz öğrenmeyle ortaya çıkarılır. Denetimsiz öğrenme, makine öğreniminin temel adımlarından biri olup küme analizine dayanır. İnsan beynini taklit ederek verileri benzerlik durumlarına göre gruplandırır. Kümeleme ile, bir gruptaki üyelerden birbirine daha benzer olanlarını aynı kümede bir araya getirilerek farklı kümeler oluşması sağlanır. Popüler küme kavramları, küme üyeleri arasında küçük mesafelere sahip grupları, veri alanının yoğun alanlarını, aralıkları veya belirli istatistiksel dağılımları içerir. Kümeleme analizi, istenen özellikler elde edilene kadar genellikle veri ön işleme ve model parametrelerinin değiştirilmesini gerektirir.

Kümeleme, bir grup veri nesnesinin, bir grubun üyelerinin benzerliğini maksimuma çıkaracak ve diğer yandan iki farklı gruptaki üyelerin benzerliğini en aza indirecek şekilde gruplandırılmasıdır. Kümelemenin kendisi bir optimizasyon problemi olarak ifade edilebilir, bu nedenle optimizasyon algoritmaları, özellikle evrimsel algoritmalar ve metasezgisel yöntemler kullanılarak çözülebilir. Ayrıca, Otomatik Kümeleme olarak bilinen genişletilmiş bir kümeleme görevi sürümü, evrimsel yöntemler kullanılarak kolayca çözülebilir. Otomatik kümeleme problemlerinde küme sayısı bilinmez; bu nedenle bazı klasik kümeleme yaklaşımları bu görevi gerçekleştirmek için kullanılamaz. Bununla birlikte, bazı iyi hedefler tanımlandığında, otomatik kümeleme, tıpkı sıradan kümeleme gibi evrimsel yöntemlerle gerçekleştirilebilir. Kümeleme problemlerini çözmek için Lloyd'un k-Ortalamalar Algoritması, DBSCAN, OPTICS ve Bulanık Kümeleme gibi birçok yaklaşım önerilmektedir. Bu algoritmaların her birinin kendine göre avantajları ve sakıncaları bulunmaktadır. Kümeleme; Makine Öğrenimi, Veri Madenciliği ve İstatistiksel Veri Analizi gibi çeşitli alanlarda ortaya çıkmakta olup Örüntü Tanıma, Görüntü İşleme, Sinyal İşleme, Biyoinformatik ve Bilgi Erişimi gibi birçok alanda uygulamalara sahiptir (Heris, 2015).

Kümeleme tekniklerinin genel yapısı Şekil 2.1’de verilmiştir.



Şekil 2.1. Kümeleme Tekniklerinin Genel Görünüm Şeması

Kaynak: (Şen, 2014)

2.3.1. Kümeleme Algoritmaları

Kümeleme algoritmaları küme modellerine göre kategorilere ayrılabilir. Bir küme modelini diğerine tercih etmek için matematiksel bir neden olmadıkça, belirli bir problem için en uygun kümeleme algoritmasının genellikle deneysel olarak seçilmesi gerekir. Kümeleme algoritmalarından en çok kullanılanları aşağıda yer almaktadır:

2.3.1.1. Bağlantı Tabanlı Kümeleme (Hiyerarşik Kümeleme)

Bağlantı tabanlı kümeleme, verileri hiyerarşik bir yapı içinde gruplamak için kullanılan, verilerin yapısını anlamada ve benzerlikleri belirlemede etkili bir yöntemdir. Bu yöntem, verilerin benzerlik veya uzaklık ölçütlerine dayanarak, verileri bir ağaç yapısı (dendrogram) şeklinde organize eder ve kümeler oluşturmasını sağlar. Bir ağaç yapısında, y ekseni kümelerin birleştiği mesafeyi gösterirken, kümelerin karışmaması için nesneler x ekseni boyunca yerleştirilir.

2.3.1.2. Merkez Tabanlı Kümeleme

Merkez tabanlı kümeleme, verileri gruplamak için kullanılan bir yöntemdir ve genellikle K-Means algoritması ile ilişkilendirilir. Bu yöntem, her bir kümenin merkezini belirleyerek çalışır.

Merkez tabanlı kümelemede öncelikle, kaç adet küme oluşturulacağı belirlenir. Bu, kullanıcı tarafından belirlenen bir parametredir. Belirlenen küme sayısı kadar rastgele merkez seçilir. Bu merkezler, verilerin dağılımını temsil eder. Her bir veri noktası, en yakın merkeze atanır. Bu, genellikle Euclidean mesafesi gibi bir mesafe ölçütü kullanılarak yapılır. Her küme için, o kümedeki tüm veri noktalarının ortalaması alınarak yeni merkezler hesaplanır. Veri noktalarının merkeze atanması ve yeni merkezlerin hesaplanmasına dair işlemler, merkezler değişmeyene veya belirli bir durma kriterine ulaşılana kadar tekrarlanır.

Merkez tabanlı kümeleme, özellikle büyük veri setleri üzerinde etkili bir şekilde çalışabilir ve hızlı bir şekilde sonuçlar elde edilmesini sağlar. Ancak, bu yöntem bazı sınırlamalara da sahiptir. Örneğin, küme sayısının önceden belirlenmesi gereklidir ve verilerin kümeleme sonuçlarını etkileyen başlangıç merkezlerinin seçimi, sonuçların kalitesini etkileyebilir. Ayrıca, merkez tabanlı yöntemler, küme şekillerinin küresel ve benzer boyutlarda olduğu varsayımına dayanır, bu nedenle karmaşık şekilli kümeler için uygun olmayabilir.

2.3.1.3. Dağıtım Tabanlı Kümeleme

Dağıtım tabanlı kümeleme, verilerin belirli bir dağılım modeline dayalı olarak gruplandığı bir kümeleme yöntemidir. Bu yaklaşım, verilerin altında yatan dağılımı anlamak ve bu dağılıma göre kümeleri oluşturmak için kullanılır.

Dağıtım tabanlı kümelemede ilk olarak, verilerin hangi dağılım modeline (örneğin, Gauss dağılımı) uyduğuna karar verilir. Bu, verilerin özelliklerine ve analizin amacına bağlı olarak değişebilir. Seçilen dağılım modeline göre, her bir küme için gerekli parametreler (örneğin, ortalama ve varyans) tahmin edilir. Bu genellikle maksimum olasılık tahmini (MLE) veya benzeri yöntemlerle yapılır. Veriler, belirlenen dağılım modeline göre kümelere atanır. Her bir veri noktası, hangi kümenin dağılımına daha uygun olduğuna göre atanır. Atama işlemi sonrasında, her küme için dağılım parametreleri güncellenir. Bu adım, verilerin yeniden değerlendirilmesi ve daha doğru kümeleme sonuçları elde edilmesi için tekrarlanır.

Dağıtım tabanlı kümeleme, özellikle verilerin belirli bir dağılım gösterdiği durumlarda etkili olabilir. Örneğin, Gaussian Mixture Model (GMM) gibi yöntemler, verilerin birden fazla Gauss dağılımı ile temsil edildiği durumlarda kullanılır. Bu yöntem, karmaşık veri yapılarının daha iyi anlaşılmasına ve daha esnek kümeleme sonuçları elde edilmesine olanak tanır.

Dağıtım tabanlı kümeleme yöntemleri, modelin doğru bir şekilde seçilmesi ve parametrelerin doğru bir şekilde tahmin edilmesi gerektiğinden bazı zorluklar içerebilir. Ayrıca, büyük veri setlerinde hesaplama maliyetleri yüksek olabilir.

2.3.1.4. Yoğunluk Tabanlı Kümeleme

Yoğunluk tabanlı kümeleme, verilerin yoğunluklarına dayalı olarak gruplandığı bir kümeleme yöntemidir. Bu yaklaşım, veri noktalarının yoğunluk bölgeleri etrafında kümelendiği varsayımına dayanır. Kümeleri ayırmak için gerekli olan seyrek alanlardaki veri noktaları genellikle gürültü (outlier) ve sınır noktaları olarak kabul edilir.

Yoğunluk tabanlı kümeleme sürecinde ilk olarak, her bir veri noktası için komşuluk tanımlanır. Bu, belirli bir mesafe (epsilon) içinde bulunan diğer veri noktalarını içerir. Her bir veri noktasının etrafındaki komşu noktaların sayısı hesaplanır. Eğer bir noktanın komşu sayısı belirli bir eşik değerinin (minPts) üzerindeyse, bu nokta yoğun bir nokta olarak kabul edilir. Yoğun noktalardan başlayarak, bu noktaların komşuluğundaki diğer yoğun noktalarla

birleştirilir. Bu süreç, yoğunluk bağlantılı noktalar bulunana kadar devam eder. Böylece bir küme oluşturulur. Yoğunluk tabanlı kümeleme, yoğun noktalara bağlı olmayan ve belirli bir yoğunluk eşik değerinin altında kalan noktaları gürültü (outlier) olarak tanımlar.

Yoğunluk tabanlı kümeleme, küme şekillerinin karmaşık olduğu durumlarda etkili olabilir. Bu yöntem, gürültü noktalarını tanımlama yeteneğine sahiptir, bu da veri analizi için önemlidir. Ayrıca yoğunluk tabanlı yöntemler, önceden belirlenmiş bir küme sayısına ihtiyaç duymaz. Ancak, yoğunluk tabanlı kümeleme yöntemleri, parametrelerin (epsilon ve minPts) doğru bir şekilde ayarlanmasını gerektirir ve bu ayarlamalar veri setine bağlı olarak değişiklik gösterebilir. Ayrıca, yüksek boyutlu verilerde yoğunluk hesaplamaları zorlaşabilir.

2.3.1.5. Izgara Tabanlı Kümeleme

Izgara tabanlı kümeleme, verileri belirli bir ızgara yapısına bölerek kümeleme yapan bir yöntemdir. Bu yaklaşım, verilerin belirli bir alan içinde düzenlenmesini ve analiz edilmesini kolaylaştırır. Izgara tabanlı kümeleme, düşük hesaplama karmaşıklığına sahip olması nedeniyle genellikle büyük veri setlerinde, hızı ve etkili bir analiz yapmak için tercih edilen bir yöntemdir.

Bu yöntemde, veri alanı belirli bir boyutta hücrelere (ızgara hücreleri) bölünür. Her bir hücre, belirli bir alanı temsil eder. Her bir hücrede bulunan veri noktalarının sayısı hesaplanır. Bu sayım, hücrenin yoğunluğunu belirlemek için kullanılır. Belirli bir yoğunluk eşiğini aşan hücreler, yoğun hücreler olarak kabul edilir. Bu hücreler, komşu yoğun hücrelerle birleştirilerek kümeler oluşturur. Düşük yoğunluklu hücreler ise genellikle gürültü olarak değerlendirilir. Oluşturulan kümeler, analiz edilerek sonuçlar çıkarılır.

Izgara tabanlı kümeleme yönteminde verilerin ızgara yapısına bölünmesi, büyük veri setlerinde hızlı bir şekilde işlem yapılmasını sağlar. Yöntem, uygulanması ve anlaşılması kolaydır. Düşük yoğunluklu hücreler gürültü olarak tanımlanabilir, bu da veri analizi için faydalıdır.

Izgara tabanlı kümelemede, ızgara boyutunun seçimi sonuçları önemli ölçüde etkileyebilir. Yanlış bir boyut seçimi, kümelerin yanlış tanımlanmasına yol açabilir. Yüksek boyutlu verilerde, ızgara yapısının yönetimi zorlaşabilir ve hesaplama maliyetleri artabilir.

2.3.2. Kümeleme Sonuçlarının Değerlendirilmesi

Kümeleme sonuçlarının değerlendirilmesi, kümelemenin kendisi kadar zor olup elde edilen kümelerin kalitesini ve anlamlılığını belirlemek için kritik bir adımdır. Bu değerlendirme, çeşitli yöntemler ve metrikler kullanılarak yapılabilir. Popüler yaklaşımlar, kümelemenin tek bir kalite puanıyla özetlendiği " iç " değerlendirmeyi, kümelemenin mevcut bir "temel gerçeği" sınıflandırmasıyla karşılaştırıldığı "dış" değerlendirmeyi, bir insan uzman tarafından " manuel " değerlendirmeyi ve amaçlanan uygulamada kümelemenin faydasını değerlendirerek " dolaylı " değerlendirmeyi içerir (Feldman & Sanger, 2007).

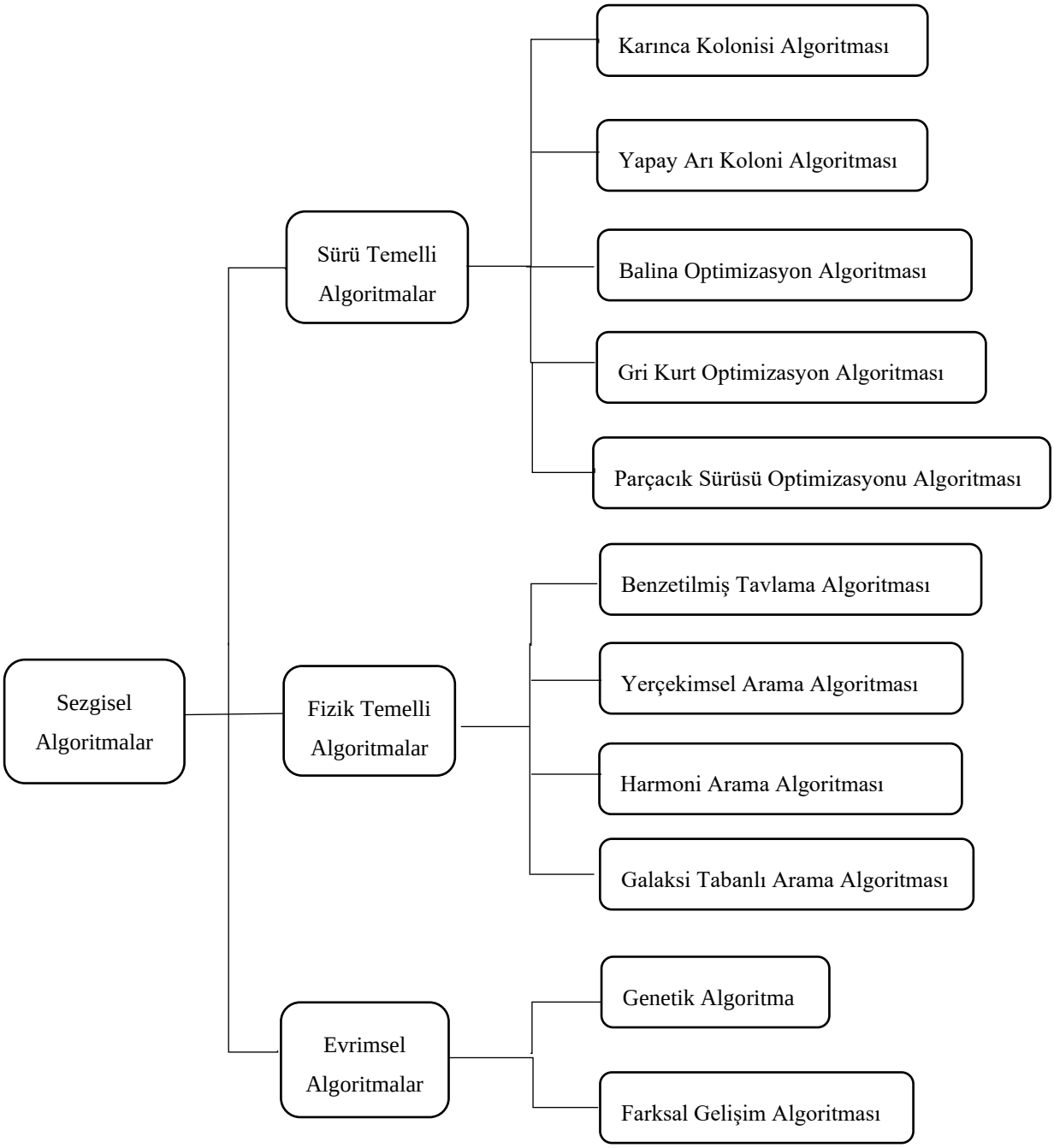
3. SEZGİSEL ALGORİTMALAR

Sezgisellik, bir problemin çözümünde nihai ve mükemmel bir sonucun elde edilmesi yerine, kabul edilebilir derecede uygunluğa sahip bir çözüme ulaşmak için kullanılan yaklaşımdır. Günlük hayatda pek çok sezgisel problem örneği ile karşılaşılır. Önceden elde olan bilgiler ışığında bu problemlere ilişkin bir çözüm üretilir. Sonrasında ise üretilen bu sonucu değerlendirip buradan edinilen tecrübeye dayanarak aynı problemle tekrar karşılaşıldığında daha iyi bir sonuç üretilebilir.

Sezgisel algoritmalar en iyi sonucu bulacaklarını garanti etmezler, fakat makul bir iterasyon sonunda genellikle en iyiye yakın olan çözüm yoluna hızlı ve kolay bir şekilde ulaşırlar. Sezgisel algoritmalarda amaç fonksiyona göre, kısıtlara göre gibi farklı güncelleme mekanizmaları ile aday çözümlerin pozisyonları iterasyonlar boyunca değiştirilerek global çözüme yakın bir çözüm bulunmaya çalışılır.

Sezgisel algoritmalarda sürü temelli, fizik temelli ve evrimsel algoritmalar olarak sınıflandırabilecek birçok yöntem kullanılmaktadır. Sezgisel algoritmalar, arama uzayında o algoritmaların temel stratejisine göre arama yapar. Bu algoritmaya örnek olarak; Genetik algoritmada genlerin değişim durumlarına göre (Kumar, Husain, Upreti, & Gupta, 2010), Yapay Arı Koloni algoritmasında arıların hareket yapısına göre (Karaboga, Gorkemli, Ozturk, & Karaboga, 2012), Karınca Kolonisi algoritmasında karıncaların en kısa yolu bulma yöntemine göre (Dorigo & Di Caro, 1999), Gri Kurt optimizasyonunda gri kurtların doğadaki avlanma stratejilerine göre (Mirjalili, Mirjalili, & Lewis, Grey Wolf Optimizer, 2014), Balina optimizasyonunda kambur balinaların beslenme davranışına göre (Mirjalili & Lewis, The Whale Optimization Algorithm, 2016), Farksal Gelişim algoritmasında bireylerin arasındaki farkın bir başka bireye ilave edilmesine göre (Price, Storn, & Lampinen, 2005), Çiçek Tozlaşma algoritmasında çiçekli bitkilerin üreme sürecine göre (Yang, 2012), Yerçekimsel Arama algoritmasında Newton'un hareket kanunlarından ikincisi olan ivme kanunu ve evrensel çekim kanuna göre (Rashedi, Nezamabadi-pour, & Saryazdi, 2009) aday çözümler arama uzayında en iyi çözüm noktasına ulaşmak için güncellenmektedir. Sezgisel algoritmalar, büyük boyutlu optimizasyon problemleri için kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalardır.

Sezgisel algoritmaların sınıflandırılması Şekil 3.1'de verilmiştir.



Şekil 3.1. Sezgisel Algoritmaların Sınıflandırılma Şeması

3.1. Sezgisel Algoritma Yöntemleri

3.1.1. Sürü Temelli Sezgisel Algoritmalar

Sürü, birbirleriyle etkileşen dağınık yapılı bireyler yığını anlamında kullanılır. Bireyler problemde aday çözüm olarak ifade edilebilir. Sürülerde N adet temsilci bir amaca yönelik davranışı gerçekleştirmek ve hedefe ulaşmak için birlikte çalışmaktadır. Kolaylıkla gözlenebilen bu "kollektif zekâ" temsilciler arasında sık tekrarlanan davranışlardan doğmaktadır. Temsilciler faaliyetlerini idare etmek için basit bireysel kurallar kullanmakta ve grubun kalan kısmıyla etkileşim yolu ile sürü amaçlarına ulaşmaktadır. Grup faaliyetlerinin toplamından bir çeşit kendini örgütleme doğmaktadır (Akyol & Alataş, 2012).

Sürü temelli sezgisel algoritmalarda, birçok hayvan topluluğundaki avlanma ile tehlikelerden kaçınma yöntemleri model olarak belirlenmiştir. Sürü zekâsı tabanlı optimizasyon algoritmaları kuş, balık, kedi ve arı gibi canlı sürülerinin hareketlerinin incelenmesiyle geliştirilmiştir (Akyol & Alataş, 2012). Başlıca bilinen sürü temelli sezgisel algoritmalar; karınca kolonisi algoritması, yapay arı kolonisi algoritması, balina optimizasyonu algoritması, gri kurt optimizasyon algoritması ve parçacık sürüsü optimizasyonu algoritmasıdır.

3.1.1.1. Karınca Kolonisi Algoritması

Karınca Kolonisi Algoritması (Ant Colony Optimization, ACO), 1999 yılında Marco Dorigo ve Gianni A. Di Caro tarafından geliştirilmiştir. Bu algoritmada, doğal karınca kolonilerinin yiyecek bulma davranışından esinlenilmiştir. ACO, özellikle sıralı optimizasyon problemlerinde belirli bir kısıtlama altında en iyi çözümü bulma amacıyla kullanılan sezgisel bir tekniktir. Karınca kolonisi algoritması, bir grup karıncanın çözüm alanında dolaşarak farklı çözümleri keşfetmesini ve bu çözümler üzerinden feromon adı verilen bir madde bırakarak en iyi çözümleri zamanla daha fazla ziyaret etmelerini teşvik etmesini esas alır.

3.1.1.2. Yapay Arı Koloni Algoritması

Yapay arı koloni algoritması (ABC), 2005 yılında Derviş Karaboğa tarafından ortaya atılan sürü zekasına dayalı bir optimizasyon tekniğidir. Az sayıda parametre içermesi, kolay uygulanabilir ve sade bir çalışma prensibine sahiptir. Doğada arıların besin aramadaki davranışlarından ilham alınarak geliştirilmiştir. Yapay Arı Kolonisinde üç çeşit arı türü görev yapmaktadır: Görevli Arılar, Kaşif Arıları ve Gözcü Arılar. ABC algoritmasında arıların

bütün davranışları birebir modellenmiştir, ancak bazı varsayımlarda bulunulmuştur. Bunlardan birisi, her bir nektarın çıkarılmasında sadece bir görevli arının olmasıdır. Bu yüzden algoritmada kullanılacak besin sayısı ile görevli arı sayısı birbirine eşit olacaktır. Diğer bir varsayım ise işçi arı sayısı ile gözcü arı sayısının birbirine eşit olmasıdır. Bu varsayımda bulunulmasına rağmen aslında bir nektara gidip gelen işçi arının var olan besin kaynağının tükenmesi halinde işçi arının kaşif arı olması da söz konusudur. Bir kaynağın uygunluk değeri besinin kalitesiyle doğru orantılıdır. Dolayısıyla ABC ile optimum değere ulaşılmaya çalışılır.

3.1.1.3. Balina Optimizasyon Algoritması

Balina Optimizasyon Algoritması (Balina Optimization Algorithm, WOA), 2016 yılında Seyedali Mirjalili ve çalışma arkadaşları tarafından geliştirilmiştir. Bu algoritma, doğada bulunan balinaların avlanma ve iletişim davranışlarını taklit ederek güçlü bir optimizasyon yöntemi sunmaktadır. Algoritma kambur balinaların avlanırken kullandıkları kabarcık avlanma stratejilerinden esinlenerek geliştirilmiştir. WOA, özellikle sürekli ve kesirli optimizasyon problemlerinde etkin bir şekilde kullanılmaktadır.

3.1.1.4. Gri Kurt Optimizasyon Algoritması

Gri Kurt Optimizasyon Algoritması (Grey Wolf Optimizer- GWO), 2014 yılında Seyedali Mirjalili ve grubu tarafından geliştirilmiştir. Gri Kurt optimizasyon algoritması, gri kurtların avcılık davranışını ve toplumsal davranışlarını taklit ederek önerilmiştir. Toplumsal hiyerarşi ile ilgili olarak, gri kurtlar alfa, beta, delta ve omega olarak sınıflandırılır (Karakaş & Yüzgeç, 2019). Algoritmanın temel adımları, gri kurtların pozisyonları, avlarını bulma stratejileri ve grup içindeki hiyerarşinin belirlenmesi üzerine kuruludur. GWO, bu davranışları matematiksel modellere dönüştürerek karmaşık optimizasyon problemlerini çözmek için kullanılır. Gri Kurt Optimizasyonu, çeşitli uygulama alanlarında, özellikle mühendislik, makine öğrenimi ve matematiksel modelleme gibi alanlarda yaygın olarak kullanılır.

3.1.1.5. Parçacık Sürüsü Optimizasyonu

Parçacık Sürüsü Optimizasyonu (PSO), bir optimizasyon algoritmasıdır ve özellikle karmaşık problemlerin çözümünde kullanılır. PSO, 1995 yılında James Kennedy ve Russ Eberhart tarafından geliştirilmiştir. Bu yöntem, doğadaki kuş sürülerinin veya balık gruplarının hareket biçimlerinden esinlenmiştir.

PSO, çözüm uzayında parçacık adı verilen bireyleri aracılığıyla optimum konumu arar (Karakuzu, 2017). PSO algoritmasında, çözüm uzayında dolaşan parçacıklar bir sürü oluşturur. Her parçacığın bir konumu ve hız vektörü bulunur. Her parçacık kendi en iyi konumunu ve tüm parçacıklar içindeki en iyi konumu takip eder. Bu bilgiler, parçacıkların güncellenmiş hız ve konumlarını hesaplamak için kullanılır. Çoğu optimizasyon algoritmasına göre hızlı çözümler üretebilir. Bazı durumlarda yerel minimumlara takılabilir. PSO, mühendislik optimizasyonu, makine öğrenimi, yapay zeka, veri madenciliği ve birçok bilimsel alanda yaygın olarak uygulanmaktadır.

3.1.2. Fizik Temelli Algoritmalar

Fizik tabanlı sezgisel algoritmalar, karmaşık optimizasyon problemlerini çözmek için fizik prensiplerinden ilham alan bir optimizasyon yöntemi sınıfıdır. Bu algoritmalar, optimizasyon sürecinde çözümü bulmak için yerçekimi kuvvetleri, elektromanyetizma ve parçacık çarpışmaları gibi doğal olaylarını taklit eder.

3.1.2.1. Benzetilmiş Tavlama Algoritması

Benzetilmiş Tavlama Algoritmasında, algoritmanın çalışması demir tavlama işlemine benzer. Demir tavlanırken, demir parça önce ısıtılır sonra soğumaya bırakılır. Bu şekilde herhangi bir sayısal ölçüme de benzer bir yaklaşım uygulanabilir. Bu yöntem bölgesel en iyi çözümlere (local optima) takılmamak için iyidir. Soğutma işlemi bu algoritmada daha iyi sonuçların bulunmasını sağlayacak yeni komşu çözümlerin üretilmesini sağlayan üstel bir ifadedir.

3.1.2.2. Yerçekimsel Arama Algoritması

Yerçekimsel Arama Algoritması (Gravitational Search Algorithm-GSA), 2009 yılında Rashedi ve arkadaşları tarafından Newton'un hareket ve yerçekimi kanunlarından esinlenerek geliştirilmiş sezgisel optimizasyon algoritmasıdır. Arama uzayındaki her bir parçacık, Yerçekimsel Arama algoritmasında bir kütle olarak kabul edilir, bu sebeple Yerçekimsel Arama algoritmasını bir yapay kütle sistemi olarak tanımlamak mümkündür (Rashedi, Nezamabadi-pour, & Saryazdi, 2009). Arama uzayındaki tüm kütleler Newton'un evrensel çekim kanununa göre birbirlerini çekerler ve yerçekimi kuvveti ile birbirlerine kuvvet uygularlar. Yerçekimi kuvvetine maruz kalan kütleler arama uzayı içerisinde hareket ederek optimum sonuca ulaşmaya çalışırlar.

3.1.2.3. Harmoni Arama Algoritması

Harmoni Arama Algoritması (Harmony Search Algorithm), bir optimizasyon algoritmasıdır ve müzik teorisinden esinlenerek geliştirilmiştir. Bu algoritma, bir grup müzisyenin uyumlu bir melodi oluşturmak için çaba göstermelerine benzetilmiştir. Temel olarak, çözüm uzayında arama yaparak en iyi çözümleri bulmayı amaçlar.

3.1.2.4. Galaksi Tabanlı Arama Algoritması

Galaksi Tabanlı Arama Algoritması (GSA- Galaxy-based Search Algorithm), doğal bir adaptasyon sürecini taklit eden bir optimizasyon algoritmasıdır. Bu algoritma, uzay tabanlı bir arama stratejisi kullanarak çözümler arasında keşif ve yerelde arama yapılmasını sağlar. Farklı galaksilerdeki yıldızlar, çözümler arası etkileşimleri temsil eder ve bu yıldızların konumları, çözüm uzayındaki potansiyel çözümleri ifade eder.

3.1.3. Evrimsel Algoritmalar

Evrimsel Sezgisel Algoritmalar, doğal seçilim ve evrimsel süreçlerden esinlenen bir optimizasyon teknikleri sınıfıdır. Bu algoritmalar, biyolojik organizmaların üreme ve doğal seçilim yoluyla gelişmesine benzer şekilde, aday çözümlerin yinelemeli olarak üretilmesine ve değerlendirilmesine dayanır. Mutasyon, çaprazlama ve en uygun olanın hayatta kalması ilkelerini taklit ederek, evrimsel sezgisel yöntemler, geleneksel analitik yöntemler kullanılarak çözülmesi zor olan sorunlara optimuma yakın çözümler bulmak için karmaşık arama uzaylarını keşfedebilir.

3.1.3.1. Genetik Algoritmalar

Evrimsel Sezgisel Algoritmaların en iyi örneği olan Genetik algoritmalar, doğadaki biyolojik evrim sürecini taklit ederek, çözümleri genetik çaprazlama ve mutasyon yoluyla geliştirirler. Genetik algoritmanın temel ilkesi, bir popülasyon içinde bir sorunu çözmek için uygun çözümleri rastgele üretmek ve bu çözümleri zaman içinde iyileştirmektir (Balcı, Yüzgeç, & Dokur, 2024). Genetik algoritmalar, özellikle çok sayıda yerel minimuma sahip, doğrusal olmayan ve karmaşık problemlerde etkili olmak için tasarlanmıştır.

Genetik Algoritmaların temel ilkeleri, 1975 yılında John Holland tarafından ortaya atılmıştır. Genetik algoritmalarda, belirli bir çözüm uzayında rastgele bireyler oluşturulur. Tüm bireylerin uygunluk değerleri hesaplanır. Uygunluk değerlerine göre bireyler seçilir. Seçilen bireyler üzerinde çaprazlama ve mutasyon işlemleri uygulanır ve yeni bireyler

oluşturulur. Yeni bireyler, mevcut bireylerle yer değiştirir ve farklı bir popülasyon oluşturur. Belirli bir durdurma kriterine ulaşıncaya kadar (örneğin maksimum nesil sayısı veya uygunluk değeri) adımlar tekrarlanır. Genetik algoritmalar popülasyon temelli bir yaklaşım olup, farklı türde problemler için uyarlanabilirler. Paralel çözüm bulma yeteneği sunarlar ve sadece birkaç parametreye ihtiyaç duyarlar. Genetik algoritmalar; mühendislik optimizasyonu, yapay zeka, makine öğrenimi, otomasyon, finans, veri madenciliği, grafiksel görüntü işleme ve daha birçok alanda yaygın olarak kullanılmaktadır.

3.1.3.2. Farksal Gelişim Algoritması

Evrimsel Algoritmalara diğer bir örnek, popülasyon içindeki bireylerin farklarını kullanarak çözümleri geliştiren ve popülasyonun en iyi üyeleri arasında bilgi paylaşımı sağlayan Farksal Gelişim algoritmasıdır. 2005 yılında Price, Storn ve Lampinen tarafından geliştirilmiştir ve karmaşık, doğrusal olmayan, çok sayıda yerel minimuma sahip problemler için etkili bir çözüm sunar.

Farksal Gelişim algoritmasında, belirli bir çözüm uzayında rastgele bireyler oluşturulur. Çiftler oluşturularak yeni bireyler türetilir. İki rastgele birey arasındaki fark kullanılarak yeni bireyler oluşturulur. Yeni bireyler çarpazlama işleminden geçirilir ve daha iyi çözümler seçilir. Belirli bir durdurma kriteri (örneğin belirli bir nesil sayısı veya hedef uygunluk) karşılanana kadar adımlar tekrarlanır. Ölçek faktörü ve çaprazlama oranı olmak üzere az sayıda parametreye ihtiyaç duyar. Karmaşık sorunlarda iyi bir keşif kapasitesine sahiptir. Çeşitli çok amaçlı optimizasyon problemleri için uyarlanabilir. Farksal Gelişim algoritması, mühendislik optimizasyonu, veri madenciliği, yapay zeka, makine öğrenimi ve daha birçok alanda kullanılmaktadır.

4. TEK ADAY OPTİMİZASYON (SCO) ALGORİTMASI

Tek Aday Optimizasyon (SCO) algoritması, tüm optimizasyon süreci boyunca yalnızca tek bir aday çözüm kullanarak geleneksel arama algoritmalarından ayrılan bir yaklaşımı temsil eder. Birden fazla aday çözüme dayanan sürü tabanlı algoritmaların aksine, SCO daha iyi optimizasyon sonuçları elde etme birincil hedefiyle yalnızca tek bir çözüme odaklanır. Optimizasyon sürecini iki farklı aşamaya bölen SCO, hem keşif hem de yerelde arama yeteneklerine sinerjik olarak katkıda bulunan farklı stratejiler ve mekanizmalar kullanır. Tek aday yaklaşımı ile iki aşamalı stratejinin bu benzersiz kombinasyonu, SCO'nun her metodolojinin güçlü yönlerinden etkili bir şekilde yararlanmasını sağlayarak güçlü ve çok yönlü bir optimizasyon algoritması ortaya çıkarır.

SCO algoritmasında optimizasyon probleminin keşif fazında, aday çözümün konumu aşağıdaki formüle göre güncellenir:

$$x(j) = \begin{cases} S(j) + (w|S(j)|), if\ r_1 < 0.5 \\ S(j) - (w|S(j)|), diğer \end{cases} \tag{4.1}$$

$$w(i) = e^{-\left(\frac{b \cdot i}{i_{max}}\right)^b} \tag{4.2}$$

burada $x(j)$ aday çözümün konumunu gösterir, j boyutu temsil eder, w ağırlık parametresi, $S(j)$ optimizasyon işlemi sırasında genel en iyi adayı temsil eder, b sabit parametreyi (2,4), i iterasyon sayısını gösterir, i_{max}maksimum iterasyon sayısını ve r_1 0 ile 1 arasındaki rasgele sayıyı temsil eder. SCO'nun ikinci aşamasında (yerelde arama fazı), ilk aşamada elde edilen en iyi pozisyonun yakınlığının kapsamlı bir keşfiyle başlayarak kapsamlı bir arama gerçekleştirilir. Daha sonra, ikinci aşamanın son kısmında, umut vaat eden bölgelere daha yoğun bir şekilde odaklanılmasını sağlamak için arama alanı daraltılır.

Aşağıdaki Eşitlik (4.3)'de, aday çözümün ikinci aşamadaki konumunu güncelleştirdiği süreç ayrıntılı olarak açıklanmaktadır:

$$x(j) = \begin{cases} S(j) + \left(wr_2(ub(j) - lb(j))\right), if\ r_2 < 0.5 \\ S(j) - \left(wr_2\left(ub(j) - lb(j)\right)\right), diğer \end{cases} \quad (4.3)$$

burada ub ve lb arama alanının üst ve alt sınır değerlerini gösterir ve r_2 rasgele sayıyı temsil eder. SCO'nun önemli bir yönü, fonksiyon değerlendirmelerinin sayısı arttıkça parametrenin üstel olarak azalmasıdır. Bu uyarlanabilir davranış, optimizasyon süreci boyunca keşif ve yerelde aramayı dengelemede hayati bir rol oynar. Nispeten yüksek bir değerle başlayarak, SCO arama alanını etkili bir şekilde araştırırken, yavaş yavaş azaltarak daha sonraki aşamalarda yerelde aramayı teşvik eder. Ayrıca SCO, ikinci aşamada pozisyon güncellemesini değiştirerek yerel optimal noktalarında sıkışıp kalma zorluğunu da ele almaktadır.

Ardışık fonksiyon değerlendirmeleri belirli sayıda bir eşik değerinde (m) uygunluk iyileştirmesi sağlamazsa, aday çözümün konumu aşağıdaki prosedüre göre ayarlanır:

$$x(j) = \begin{cases} S(j) + \left(r_4(ub(j) - lb(j))\right), if\ r_3 < 0.5 \\ S(j) - \left(r_5\left(ub(j) - lb(j)\right)\right), diğer \end{cases} \quad (4.4)$$

burada r_3, r_4 ve r_5 0-1 arasında rasgele sayıları temsil eder. Konum güncelleme eşitliği (4.4), SCO algoritması içindeki aday çözüm için yerel aramadan keşfe geçişi kolaylaştırmada kritik bir rol oynar. Konum güncelleme mekanizmasındaki bu değişiklik, algoritmanın arama alanının alternatif bölgelerini keşfetmesini ve optimal olmayan çözümlerde takılıp kalmaktan kaçınmasını sağlar.

Bazı değişkenlerin konumlarını güncellemek bazen değerlerinin aralık veya sınırların dışına çıkmasına neden olabilir. Değişkenlerin sınırları aşmasını sınırlamak için, güncellenen konumlar, değerleri sırasıyla üst ve alt sınırları geçtiği durumlarda şu şekilde ayarlanır:

$$x(j) = \begin{cases} S(j) & if\ x(j) > ub(j) \\ S(j) & if\ x(j) < lb(j) \end{cases} \quad (4.5)$$

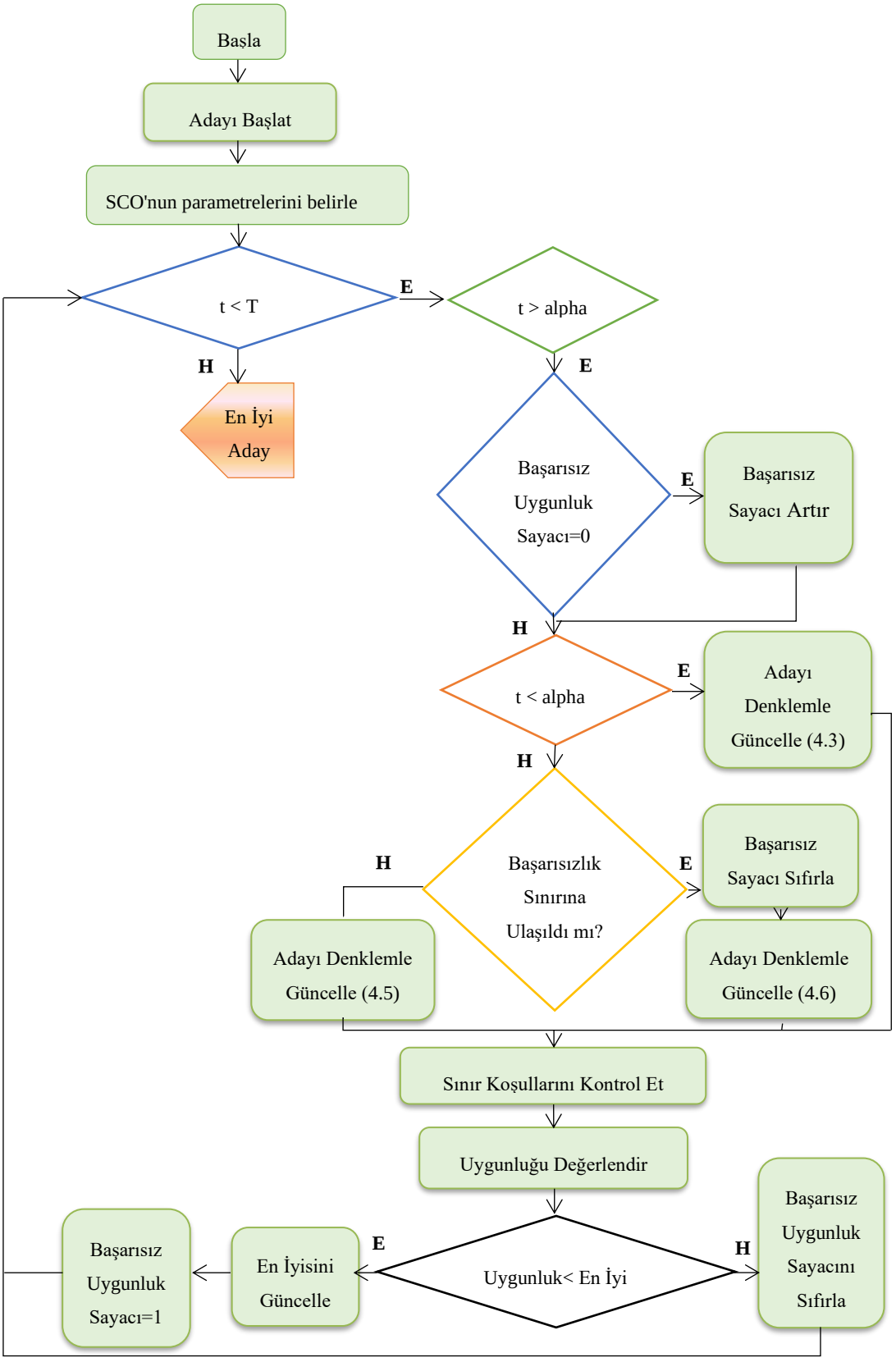
Eşitlik (4.5)'de, güncellenen bir aday çözümün güncellenen boyutuna, güncellenen konum sınırların dışına çıkarsa küresel en iyi değerin konumu atanır.

SCO'da, tek bir aday çözüm rastgele üretilir ve daha sonra daha iyi bir çözüm aramak için yinelemeli olarak güncellenir. Önerilen algoritmanın adımları aşağıdaki gibi açıklanmıştır. Süreç, arama alanında rastgele bir aday çözüm üreterek, uygunluk/maliyet değerini değerlendirerek, bu adayı küresel en iyi konum (gbest) olarak ve uygunluk değerini f(gbest) küresel en iyi maliyet değeri olarak kaydederek başlar. İlk aday çözüm, aşağıdaki Eşitlik (4.3)'deki gibi üretilir:

$$x(j) = lb(j) + r_1\big(ub(j) - lb(j)\big) \tag{4.6}$$

Burada $lb(j)$ ve $ub(j)$ arama alanının alt ve üst sınırları, r_1 [0,1] aralığında rastgele bir sayıdır.

SCO akış diyagramı, Şekil 4.1'de verilmiştir. Maksimum iterasyon sayısına (T) ulaştığında sonlanan tekrarlayan süreç, aday çözümünün konumunu güncelleyerek başlar. POO, başarısız uygunluk iyileştirmelerini saymak için ilk sayaçtır. Uygunluğu iyileştirmek için başarısız girişimlerin sayısı m=5 olarak alınmış olup, eşik değerinde (m) uygunluk iyileştirmesi sağlamazsa, aday çözümün konumu eşitlik (4.4)' e göre ayarlanır. İlk aşamadaki işlev değerlendirmelerinin sayısı alpha=1000 alınarak başarısız uygunluk iyileştirmeleri sayacı Eşitlik (4.2)'ye göre çalıştırılır. Aday çözüm (S), birinci ve ikinci aşamadaki konumunu sırasıyla Eşitlik (4.1) ve (4.3)'e göre günceller. Aday konumu güncellendikten sonra, yeni oluşturulan aday çözüm f(S)'in uygunluk değeri değerlendirilir ve f(gbest) ile karşılaştırılır. Eğer f(S), gbest uygunluk değerinden f(gbest) daha iyiyse, gbest ve f(gbest) sırasıyla S ve f(S) ile değiştirilir. Tekrarlayan süreç, maksimum fonksiyon değerlendirmesi sayısı T'ye ulaşılana kadar devam eder.



Şekil 4.1. SCO Algoritması Akış Diyagramı

5. HIZLANDIRILMIŞ KARŞIT ÖĞRENME TABANLI TEK ADAY OPTİMİZASYON (AccOppSCO) ALGORİTMASI

Bu bölümde, orijinal SCO algoritmasının arama başarımını iyileştirmek ve bazı handikaplarını (sınırlı keşif yeteneği, ilk aday çözüm seçimine duyarlılık vb.) aşmak için, algoritma yapısına entegre edilen hızlandırılmış karşıt öğrenme tabanlı mekanizma ele alınmaktadır.

Karşıt Tabanlı Öğrenme (OBL: Opposition Based Learning), karşıtlık kavramını optimizasyon algoritmalarına dahil eden, çözümleri orijinal değerlerine ve karşıt karşılıklarına göre değerlendiren bir mekanizmadır. OBL kavramı ilk olarak eski Çin felsefesindeki Yin-Yang sembolünden gelmiştir. OBL kavramına göre, bir aday nokta çözümden uzaksa, bu adayın zıt noktası çözüme o noktadan daha yakın olabilir (Yüzgeç & Karakuzu, 2018). Karşıt çözümleri göz önünde bulundurarak, ek çeşitlilik sunar ve potansiyel olarak iyileştirilmiş optimizasyon sonuçlarına yol açarak arama uzayının keşfedilmesini kolaylaştırır (Tizhoosh, 2005). Burada mevcut OBL mekanizması, optimizasyon sürecinin yakınsama hızını arttırmak için bir ivme katsayısı (C_a) ile güçlendirilmiştir (Dinkar & Deep, 2018).

Karşıtlık tabanlı öğrenmede birinci aşama karşıtlık tabanlı başlangıç popülasyonun belirlenmesi, bir sonraki aşama ise karşıtlık temelli jenerasyon atlama işlemidir (Karakaş & Yüzgeç, 2019). Karşıt tabanlı öğrenme mekanizmasını başlatma aşamasında, aday, karşıt adayı türetmek için kullanılır. Bu süreç, rasgele başlatma yoluyla elde edilene kıyasla daha avantajlı kabul edilen bir birey üretebilmektedir. Karşıt tabanlı tekniklerin dahil edilmesiyle, bu başlatma adımı, atanan aday çözümün müteakip optimizasyon prosedürleri için uygunluğunu artırabilir. Çözüm hakkında bilginin bulunmaması durumunda, adaylar stokastik optimizasyon sürecinin başlatılmasında rastgele belirlenir. Rastgele adayların belirlenmesine ek olarak, onların karşıt adayları karşıt tabanlı popülasyon başlatma ile hesaplanır (Yüzgeç & İnaç, 2016).

Prosedür aşağıda özetlenmiştir:

$$\check{x} = (ub + lb) - x \tag{5.1}$$

burada x aday çözmün konumunu, $\check{x}$ rasgele adayın karşıt konumunu temsil eder. Optimizasyon sürecindeki her yinelemeden sonra, önceden tanımlanmış bir atlama olasılığı değeri (Jr) kullanılarak bir karşıt aday belirlenir. Ardından, bir sonraki yineleme için mevcut ve karşıt adayların bir kombinasyonundan üstün aday seçilir. Karşıtlığa dayalı nesil atlamanın uygulanması, aşağıdaki ifadede gösterildiği gibi, atlama hızının (Jr) [0,1] aralığında rasgele bir sayı ile karşılaştırılmasına dayanır:

$$\check{x}(i+1) = \begin{cases} ub + lb - x(i), if\ r_4 < Jr \\ x(i), diğer \end{cases} \tag{5.2}$$

OBL stratejisinin benimsenmesi yakınsama ivmesini sağlarken, üretilen karşıt aday beklenmedik keşif davranışı sergileyebilir. Üretimin erken aşamalarında keşif artışı faydalı olsa da, algoritma üretim sayısı arttıkça kademeli olarak yerelde aramaya doğru kaymalıdır. Sonuç olarak, burada keşif ve yerelde arama arasında bir denge kurmak için OBL'yi bir ivme katsayısı (C_a) ile uyarlanabilir hale getirmeyi içeren ikinci bir strateji kullanılmıştır. Bunu başarmak için, ivme katsayısı C_a keşif ve yerelde arama arasındaki dengeyi yaklaşık olarak belirlemek için kullanılır ve yineleme sayısı arttıkça uyarlanabilir şekilde azaltılır (Dinkar & Deep, 2018). Bu parametrenin (C_a) hesaplanması aşağıda verildiği gibi elde edilebilir:

$$C_a = C_{max} - i \cdot \frac{C_{max} - C_{min}}{i_{max}} \tag{5.3}$$

burada C_{max} maksimum değerdir (1) ve C_{min} minimum değerdir (1e-5), i mevcut yinelemeyi temsil eder ve i_{max} maksimum yinelemeyi belirtir. Eşitlik (5.3) bu prosedürle aşağıdaki şekilde güncellenir:

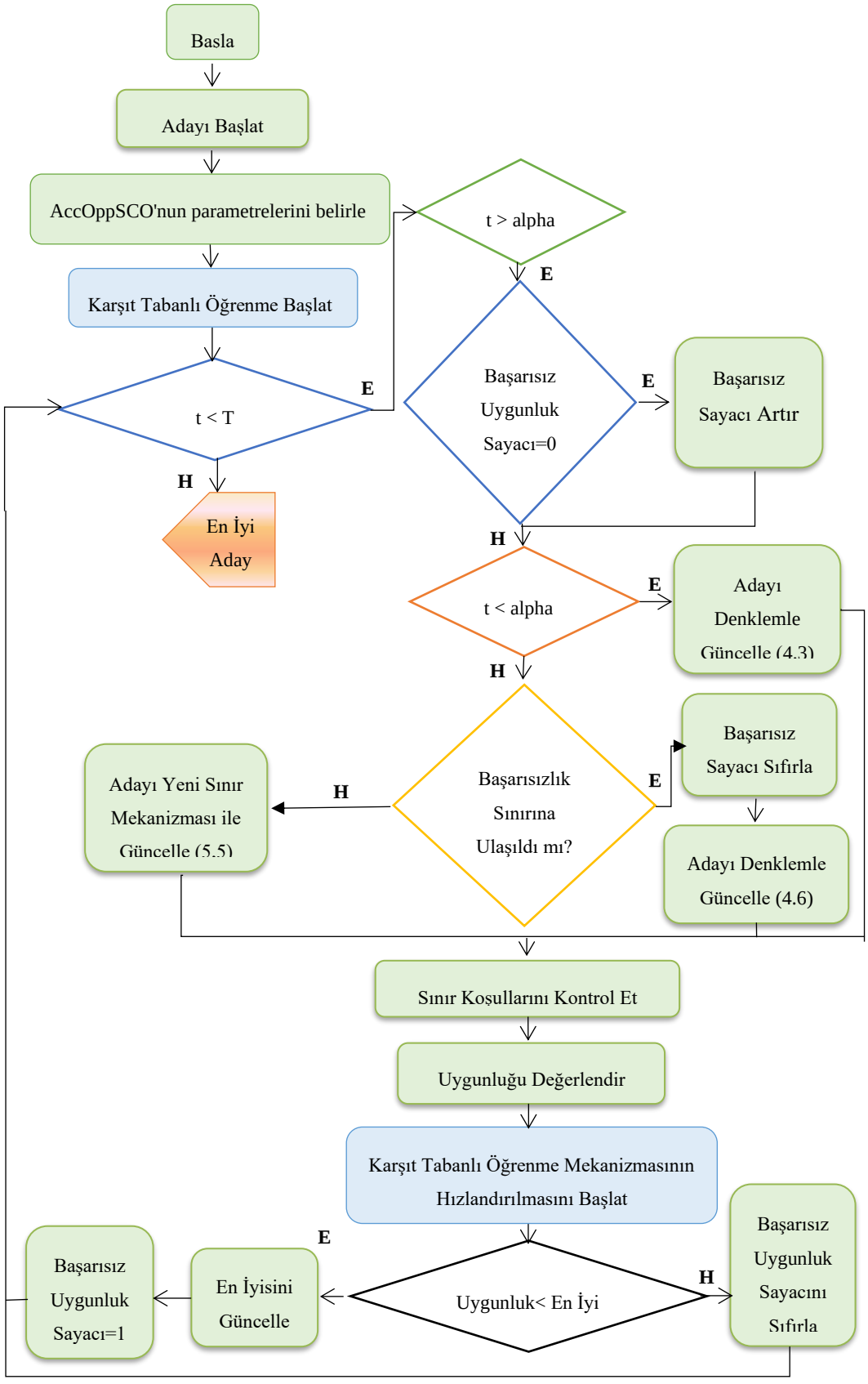
$$\check{x} = (ub + lb - x) \cdot C_a \tag{5.4}$$

Bazı değişkenlerin konumlarının güncellenmesi, değerlerinin aralık veya sınırların dışına çıkmasına neden olabilir. Bu üst ve alt sınırları belirlemek için, önerilen AccOppSCO algoritmasında yeni bir sınır değer kontrol mekanizması kullanılmıştır. Bu mekanizma Eşitlik (5.5)'de verildiği gibi çalışmaktadır.

$$x = \begin{cases} \frac{ub+S}{2} \ , & if\ x > ub \\ \frac{lb+S}{2} \ , & if\ x < lb \end{cases} \tag{5.5}$$

Burada x, güncellenen aday çözümün konumunu, S en iyi aday çözümün konumunu ub ve lb ise sırayla üst ve alt sınır değerlerini göstermektedir.

AccOppSCO algoritmasının akış diyagramı Şekil 5.1'de verilmiştir.



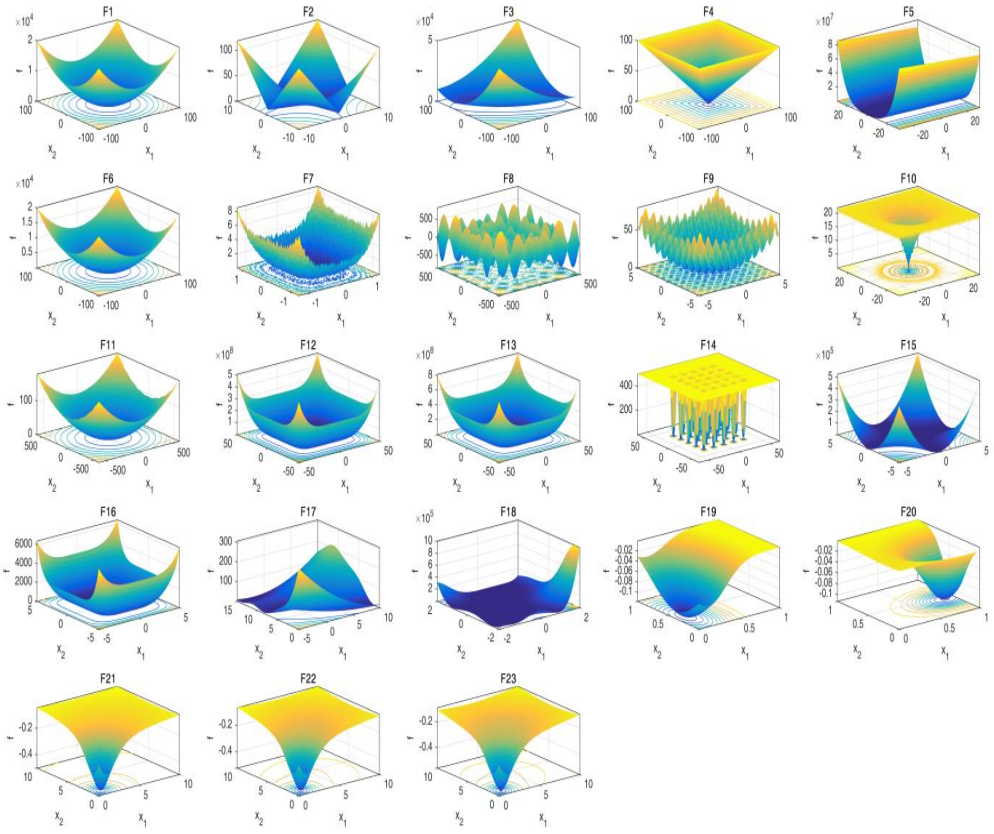
Şekil 5.1. AccOppSCO Algoritması Akış Diyagramı

6. DENEYSEL SONUÇLAR

Bu çalışmada, ele alınan Tek Aday Optimizasyonu (SCO) algoritması ile arama başarımını iyileştirmek için, hızlandırılmış karşıt öğrenme tabanlı bir mekanizmanın algoritma yapısına entegre edildiği AccOppSCO algoritması karşılaştırılmıştır.

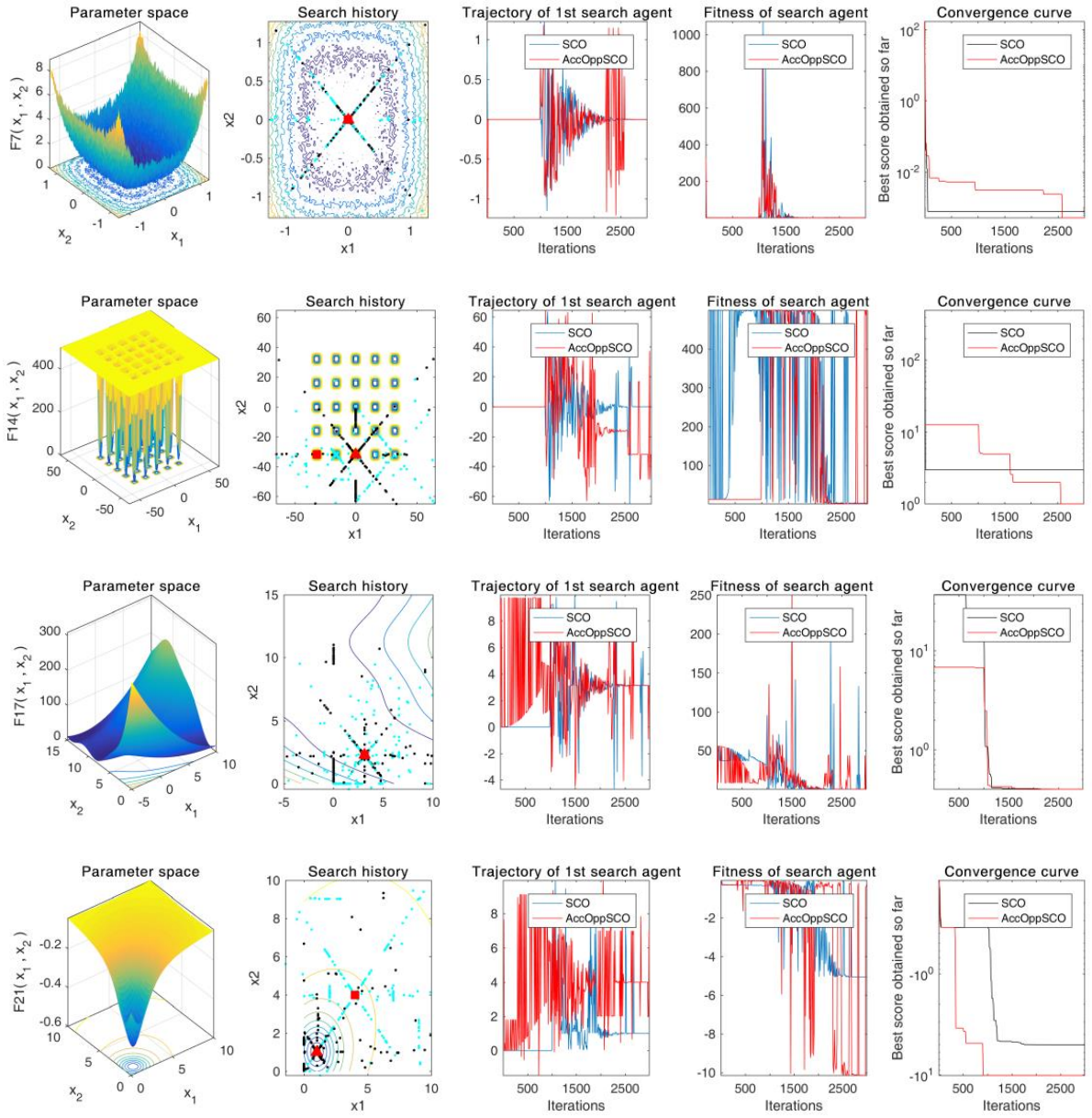
6.1. AccOppSCO ile SCO Algoritmalarının Başarımlarının Karşılaştırılması

Her bakımdan maksimum başarımı elde eden ve daha iyi olan algoritmayı bulmak için, diğer algoritmalarla kıyaslama yapılması gerekmektedir. Bu kıyaslama yapılırken de algoritmalar arasındaki farkları ve avantajları belirlemek için algoritma analizine ihtiyaç duyulmaktadır. AccOppSCO ile SCO algoritmalarını başarım yönünden karşılaştırmak için yapılan analizlerde, literatürden iyi bilinen 23 minimizasyon problemi ele alınmıştır. SCO ve AccOppSCO algoritmalarının eş zamanlı olarak fonsiyonları test edilmiş ve bu kıyaslama problemlerinin sonuçlarına ait üç boyutlu yüzeyleri iki boyut için Şekil 6.1'de grafikler halinde gösterilmiştir.



Şekil 6.1. Bu Çalışmada Kullanılan Kıyaslama Fonksiyonları

İlk olarak, her iki algoritma için bu fonksiyonlardan rastgele seçilen dört farklı minimizasyon problemi F7, F14, F17 ve F21'in iki boyutlu analizleri yapılmıştır. Çalışmada kullanılan analizler, optimizasyon süreci boyunca adayın arama uzayındaki hareketini gösteren arama geçmişi analizini, adayın konumunun ilk boyutunun değişimini gösteren yörünge analizini, optimizasyon boyunca adayın maliyet değerindeki değişim analizini ve aday çözümlerin yakınsamasını gösteren yakınsama analizini içerir. Şekil 6.2, seçilen dört kıyaslama için analiz sonuçlarını göstermektedir.



Şekil 6.2. SCO ve AccOppSCO'nun Bazı Kıyaslama Fonksiyonları İçin Grafiksel Analizleri

6.1.1. Arama Geçmişi (Search History) Analizi

Arama geçmişi analizi ile optimizasyon aşaması boyunca arama alanındaki aday çözümlerin evrimi izlenmektedir. Şekil 6.2'de ikinci sütundaki arama geçmişi grafiklerinde, SCO ve AccOppSCO algoritmasının belirlenen dört fonksiyon üzerinde çözüm konumlarının yineleme adımları verilmiştir.

Grafiklerdeki kırmızı renkli kareler SCO'nun ve kırmızı renkli üçgenler AccOppSCO'nun en son bulunan en iyi çözümü belirtmekte, açık mavi ile gösterilenler SCO ve siyah ile gösterilenler ise AccOppSCO aday çözümlerinin hangi bölgelerde olduğunu göstermektedir. Grafiklere bakıldığında aday çözümlerinin genellikle tek bir noktada yoğunlaştıkları görülmektedir. Fonksiyon grafiklerinde bulunan çukur alanlara bakıldığı zaman, AccOppSCO aday çözümlerinin SCO aday çözümlerinin konumuna göre daha doğru bir şekilde bu çukur alanlarda yoğunlaştıkları gözlemlenmektedir.

6.1.2. Yörünge (Trajectory of 1st Search Agent) Analizi

Yörünge analizi, 1. aday çözümün ilk pozisyonunun arama uzayındaki değişimini göstermektedir. Bu analizde seçilen kıyaslama fonksiyonları için, Şekil 6.2'de üçüncü sütundaki arama geçmişi grafiklerinde, belirli yinelemelerde SCO ve AccOppSCO'nun 1. aday çözümünün optimizasyon boyunca değişen ilk pozisyonlarını görülmektedir.

6.1.3. Aday Çözümün Uygunluğu (Fitness of Search Agent) Analizi

Bu analizde, optimizasyon yinelemeleri boyunca adayın uygunluk değerindeki değişimleri gözlemlenmektedir. SCO ve AccOppSCO algoritmaları için arama uzayında aday çözümlerin uygunluğu grafikler halinde Şekil 6.2'de gösterilmiştir. Dördüncü sütunda görüldüğü gibi AccOppSCO algoritması, SCO algoritmasına göre sıfıra daha hızlı yaklaşmakta, ancak daha fazla salınım gerçekleştirmektedir.

6.1.4. Yakınsama Eğrisi (Convergence Curve) Analizi

Bu analiz, optimizasyon algoritmalarının başarımını değerlendirmek için kullanılan önemli bir araç olup aday çözümlerin optimuma doğru ilerlemesini değerlendirmektedir. Bu eğri, algoritmanın bir problem üzerindeki çözüm sürecinde maliyet fonksiyon değerinin nasıl değiştiğini göstermektedir. Şekil 6.2'de son sütundaki yakınsama eğrilerinde, SCO ve AccOppSCO'nun hata değerleri karşılaştırılmıştır. Dört fonksiyonun her biri için aynı

başlangıç parametreleri kullanılmıştır. Grafiklere bakıldığı zaman AccOppSCO algoritması, dört fonksiyon için de SCO algoritmasından daha başarılı sonuçlar vermiştir.

7. ve 14. fonksiyonun yakınsama eğrileri analiz edildiğinde, SCO algoritmasının iç sınırlamalarının üstesinden gelerek keşif aşamasında ve daha sonrasında ilerleme kaydedemedikleri, algoritma yerel optimumda sıkıştığı için öyle kaldıkları ve küresel çözüme yakınsayamadıkları görülmüştür. Buna karşılık, AccOppSCO algoritması SCO'nun bu sınırlamaları nedeniyle keşif aşamasının ilk 1000 yinelemesinde ilerleme kaydemeyerek ilk 1000 yinelemeden sonra, algoritmanın yerel optimumda sıkışması nedenleri ile 7. fonksiyon için 2500. yinelemeye kadar, 14. fonksiyon için ise önce 1500. yinelemeye sonra da 2500. yinelemeye kadar öyle kalmış ve bu noktadan sonra küresel çözüme doğru ilerleyebilmiştir. AccOppSCO algoritması SCO algoritmasına göre yerel optimumlardan kaçınmaya çalışarak ve küresel çözüme yakınsayarak daha verimli bir başarım göstermiştir. SCO algoritmasının 17. fonksiyon için sonuç üretme hızında AccOppSCO algoritması onu geride bırakmıştır. SCO algoritmasının 21. fonksiyon için (0,0) çözümleri olan problemler için hızlı sonuçlar ürettiği bilinmektedir. Ancak bu durumda bile AccOppSCO algoritması onu geride bırakmıştır.

Analiz, önerilen AccOppSCO algoritmasının orijinal SCO algoritmasına göre özellikle keşif aşamasında ve yerel minimumlardan kaçınmada önemli bir avantaja sahip olduğunu göstermektedir. Bu sonuçlara göre, optimizasyon sürecinde önerilen AccOppSCO yapısının, SCO yapısına göre daha iyi bir yakınsamaya sahip olduğu açıkça görülmektedir.

6.2. Optimizasyon Test Fonksiyonu Sonuçları

Tablo 6.1'de, SCO ve AccOppSCO algoritmalarının 30 çalıştırma için, istatistiksel sonuçları özetlenmiştir. SCO ve AccOppSCO algoritmalarının 23 test fonksiyonu üzerinde 30 çalıştırılmasının karşılaştırma sonuçlarına göre, AccOppSCO'nun SCO algoritmasından daha iyi başarım gösterdiği açıktır.

AccOppSCO, aşağıdaki metriklere dayalı olarak üstün başarım sergilemiştir: en iyi ve en kötü metrik değerleri açısından AccOppSCO algoritması, tüm test işlevlerinde 19/23 başarı oranıyla SCO algoritmasını %82,61 oranında geride bırakır. Ayrıca sonuçlardan AccOppSCO, 23 fonksiyonun 16'sında Medyan ve Ortalama metriklerde SCO'dan daha başarılı (%69,57). Ayrıca AccOppSCO, SCO'ya kıyasla daha düşük bir standart sapmayla

(%86,96) test işlevlerinde daha tutarlı ve istikrarlı bir başarım göstermiştir. Bu bulgulara dayanarak, AccOppSCO'nun test fonksiyonlarını optimize etmede SCO'dan daha iyi başarım gösterdiği sonucuna varılabilir.

Tablo 6.1. 30 Çalıştırma İçin SCO ve AccOppSCO İstatistiksel Sonuçları

	SCO					AccOppSCO				
No	**En İyi**	**En Kötü**	**Medyan**	**Ortalama**	**Std. Sapma**	**En İyi**	**En Kötü**	**Medyan**	**Ortalama**	**Std. Sapma**
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	1.89E-304	2.02E-247	9.74E-272	1.27E-247	0.00E+00	1.39E-293	4.97E-260	1.03E-270	3.42E-261	0.00E+00
3	0.00E+00	3.53E-239	0.00E+00	1.18E-239	0.00E+00	0.00E+00	5.32E-166	0.00E+00	1.77E-167	0.00E+00
4	5.53E-181	1.45E-32	9.37E-133	4.85E-33	2.65E-32	1.33E-191	1.71E-10	4.07E-126	5.70E-12	3.12E-11
5	2.85E+01	2.90E+01	2.89E+01	2.88E+04	1.14E-01	2.87E+01	2.89E+01	2.89E+01	2.89E+01	5.70E-02
6	1.46E-01	6.50E-01	2.10E-01	2.50E-01	1.13E-01	1.46E-01	4.32E-01	3.10E-01	3.08E-01	6.97E-02
7	1.98E-02	2.95E-03	3.13E-04	4.57E-04	5.38E-04	7.36E-06	5.16E-04	1.82E-04	2.18E-04	1.46E-04
8	9.98E+03	-7.30E+03	-8.23E+03	-8.28E+03	6.97E+02	-1.01E+04	-7.14E+03	-8.37E+03	-8.29E+03	7.36E+02
9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
10	8.88E-13	8.88E-13	8.88E-13	8.88E-13	4.01E-27	8.88E-16	8.88E-16	8.88E-16	8.88E-16	4.01E-31
11	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	1.96E-02	2.58E-01	5.43E-02	7.27E-02	5.14E-02	1.18E-02	2.11E-01	8.80E-02	8.44E-02	4.82E-02
13	3.33E-01	2.99E+03	1.51E+03	1.52E+04	7.05E-01	7.29E-01	2.45E+00	1.84E+00	1.69E+00	4.50E-01
14	9.98E-01	1.99E+03	9.98E-01	1.20E+04	4.04E-01	9.98E-01	1.99E+00	9.98E-01	1.06E+00	2.52E-01
15	3.52E-04	1.23E-01	1.35E-03	7.05E-03	2.27E-02	3.26E-04	2.70E-02	8.30E-04	3.49E-03	5.72E-03
16	-1.03E+03	-1.03E+03	-1.03E+03	-1.03E+03	0.00E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	0.00E+00
17	3.98E-01	3.98E-01	3.98E-01	3.98E-01	0.00E+00	3.98E-01	3.98E-01	3.98E-01	3.98E-01	0.00E+00
18	3.00E+00	3.00E+00	3.00E+00	3.00E+00	0.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	0.00E+00
19	-3.86E+03	-3.86E+03	-3.86E+03	-3.86E+03	0.00E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	0.00E+00
20	-3.32E+03	-3.13E+03	-3.32E+03	-3.27E+04	7.67E-02	-3.32E+00	-3.10E+00	-3.20E+00	-3.23E+00	9.22E-02
21	-1.02E+01	-2.63E+00	-1.02E+01	-8.64E+04	2.61E+04	-1.02E+01	-5.10E+00	-1.02E+01	-9.81E+00	1.28E+00
22	-1.04E+01	-2.77E+03	-1.04E+01	-8.30E+04	2.86E+04	-1.04E+01	-1.04E+01	-1.04E+01	-1.04E+01	1.81E-15
23	-1.05E+01	-5.13E+03	-1.05E+01	-9.28E+04	2.32E+04	-1.05E+01	-5.18E+00	-1.05E+01	-9.82E+00	1.85E+00

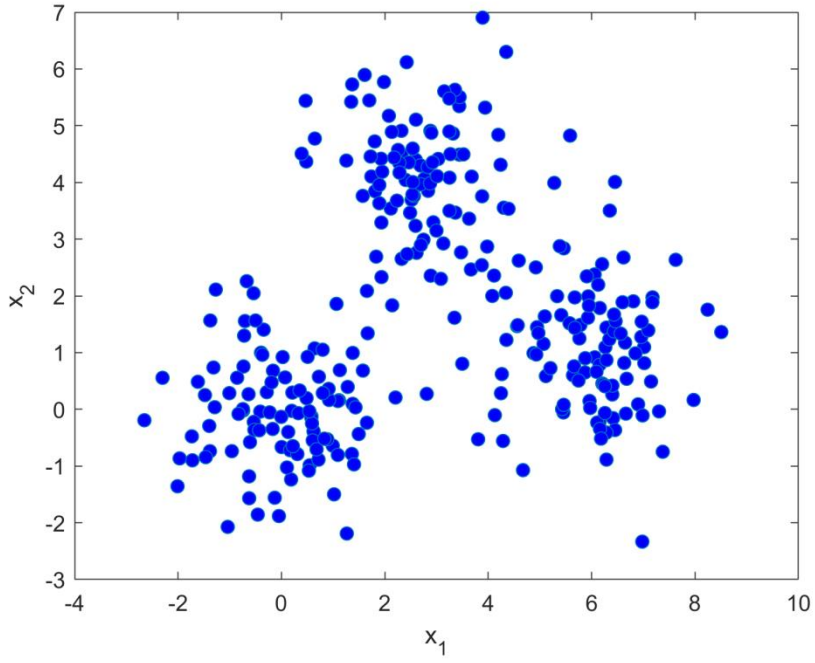
6.3. Kümeleme Problemini Kullanarak AccOppSCO Algoritmasının Başarımının Değerlendirilmesi

Bu bölümde, kümeleme problemini kullanarak önerilen AccOppSCO algoritmasının başarımı değerlendirilmiştir. Kümeleme problemi, veri noktalarını belirli bir kritere veya benzerlik ölçüsüne göre ayrı kümeler halinde gruplandırmayı amaçlayan bir makine öğrenimi görevidir. Problem çözümünde veri noktaları ile küme merkezleri arasındaki mesafeler hesaplanır ve her veri noktası en yakın kümeye atanır. Daha sonra küme içi mesafelerin toplamı hesaplanır, bu da kümelerin ne kadar kompakt olduğunu gösterir. Kümeleme maliyeti, küme içi mesafelerin toplamının değeridir. Maliyet fonksiyonu formülü aşağıda verilmiştir:

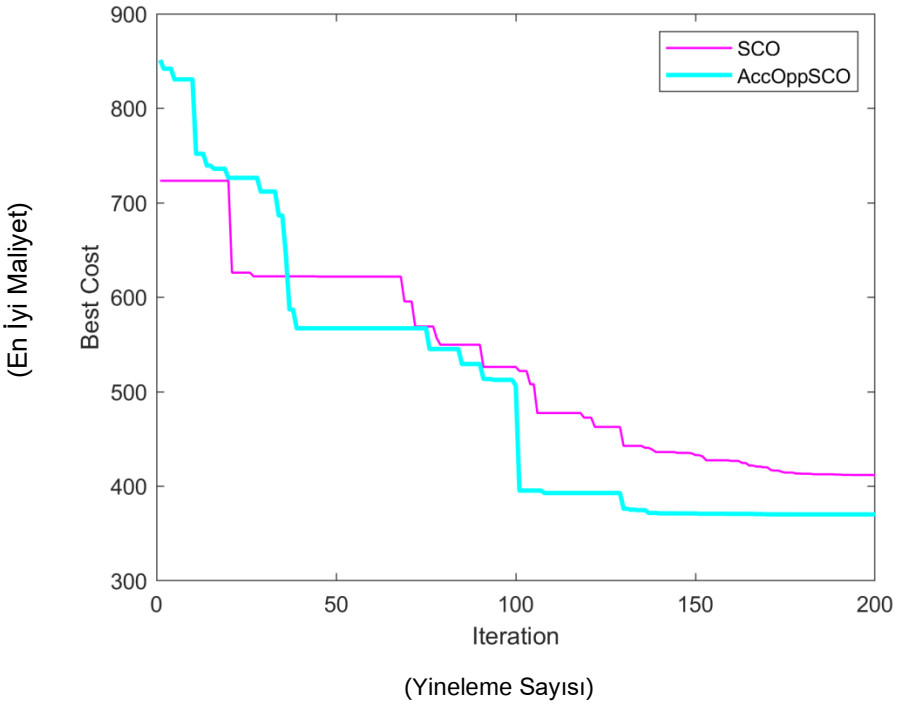
$$J = \sum_{i=1}^{N} min_j \left\| x_i - c_j \right\|^2 \tag{6.1}$$

Burada J değeri maliyeti, x_i $i.$ durumu gösterir, c_j J kümesinin merkez noktasını temsil eder, N ise durum sayısıdır. Bu çalışmada 300 adet iki boyutlu veriden oluşan bir kümeleme veriseti kümesi kullanılmıştır.

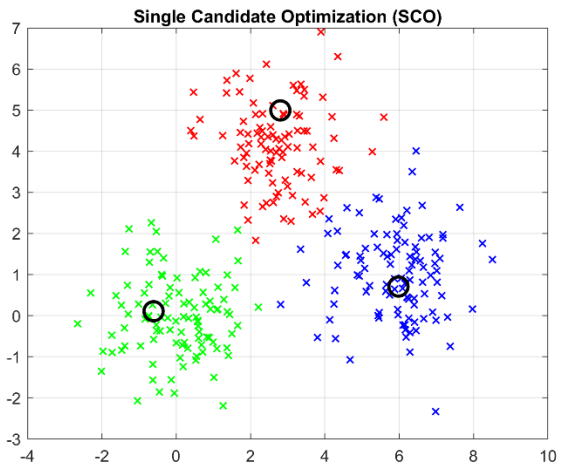
Çalışmada kullanılan veri kümesi Yarpiz web sitesinden alınmıştır (Heris, 2015) ve Şekil 6.3'de gösterilmiştir. İlk olarak, orijinal SCO tarafından elde edilen sonuçlar ve önerilen AccOppSCO algoritmaları üç küme için karşılaştırılmıştır. Şekil 6.4, orijinal SCO'nun ve önerilen AccOppSCO algoritmalarının yakınsama eğrilerini birlikte göstermektedir. Bu şekilden de anlaşılacağı üzere AccOppSCO algoritması SCO algoritmasından daha iyi bir yakınsama başarımına sahiptir. Hızlandırılmış karşıt öğrenme tabanlı mekanizması sayesinde, önerilen AccOppSCO algoritması kümeleme problemini çözmede daha iyi bir yakınsama göstermektedir. Şekil 6.5'de, her iki algoritmanın kümeleme sonuçları gösterilmiştir. Üç küme için elde edilen sonuçlara bakıldığında, AccOppSCO ile elde edilen merkez noktalarının SCO ile elde edilenlerden daha başarılı olduğu açıktır.

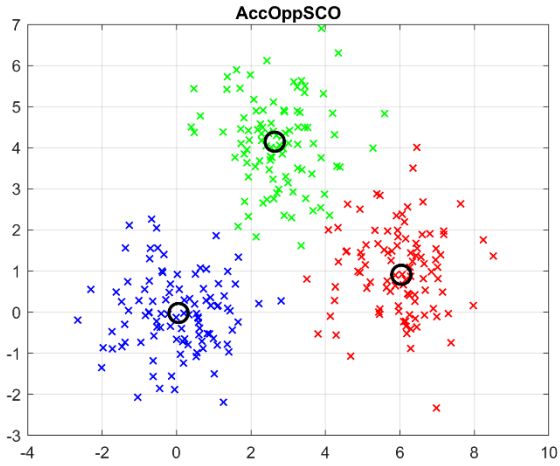


Şekil 6.3. Bu Çalışmada Kullanılan Kümeleme Verileri



Şekil 6.4. SCO ve AccOppSCO'nun Yakınsama Eğrileri





Şekil 6.5a. SCO Sonuçları

Şekil 6.5b. AccOppSCO Sonuçları

Diğer testte, AccOppSCO algoritması temel sezgisel algoritmalarla karşılaştırılmıştır. Bu algoritmalar Genetik Algoritma (GA), Farksal Gelişim (DE) ve Parçacık Sürüsü Optimizasyon (PSO) Algoritmasıdır. GA, PSO ve DE algoritmalarının kodları ve parametreleri Yarpiz web sitesinden alınmıştır (Heris, 2015). Karşılaştırma sınaması için, küme sayısı beş olarak ayarlanmıştır. Beş yinelemeli olarak optimizasyon koşturulmuş ve elde edilen istatistiksel sonuçları Tablo 6.2'de özetlenmiştir. Bu tabloda, maksimum, min, ortalama ve standart sapma (STD) olmak üzere dört metrik verilmektedir. Sonuçlar, AccOppSCO algoritmasının minimum metrikte üçüncü en iyi değere sahip olduğunu ve en iyi sonucun GA'ya ait olduğunu göstermektedir. Maksimum ölçüm sonuçları, AccOppSCO'nun SCO algoritmasından daha iyi başarıma sahip olduğunu göstermektedir. Ortalama metrik değerlerde GA algoritması ilk sırada yer almakta ve AccOppSCO, SCO algoritmasından daha iyi başarım sergilemektedir. Standart sapma sonuçlarında orijinal SCO algoritmasının en yüksek sapma değerine sahip olduğu anlaşılmaktadır.

Tablo 6.2. AccOppSCO ve Diğer Sezgisel Yöntemlerin İstatistiksel Sonuçları

	GA	**PSO**	**DE**	**SCO**	**AccOppSCO**
MIN	311.6968	316.0045	317.2864	323.7386	311.9062
MAX	312.6416	316.0064	320.6813	348.1509	342.1745
ORT	311.9643	316.0056	319.2246	336.5650	320.4333
STD	0.3887	0.0010	1.3591	10.1035	12.6023

Son olarak, tekrarlanan çalıştırmalar için algoritmaların çalışma süreleri analiz edilmiştir. Tablo 6.3, çalışma sürelerinin istatistiksel sonuçlarını göstermektedir. Sonuçlardan AccOppSCO ve SCO mekanizmasında tek aday çözüm yaklaşımının zaman maliyeti açısından büyük avantajı açıkça görülmektedir.

Tablo 6.3. AccOppSCO ve Diğer Sezgisel Yöntemlerin Çalışma Zamanı Sonuçları

	GA	**PSO**	**DE**	**SCO**	**AccOppSCO**
MIN	4.2917	5.0511	5.1029	0. 4976	0.5916
MAX	5.4334	5.7265	5.7248	1.2126	1.3003
ORT	5.0801	5.2809	5.3448	0.7311	0.9650
STD	0.4648	0.2599	0.2459	0.32	0.3435

7. SONUÇLAR

Bu tez çalışmasında, Tek Aday Optimizasyonu (SCO) algoritması yapısı ile SCO'nun arama başarımını iyileştirmek için hızlandırılmış karşıt öğrenmeye dayalı bir mekanizmanın entegre edilmesi ile elde edilen AccOppSCO algoritmasının başarımları karşılaştırılmıştır. Bu karşılaştırma için literatürden 23 kıyaslama problemi ele alınmıştır. 23 test fonksiyonu üzerinde 30 çalıştırılmasının Tablo 6.1'deki karşılaştırma sonuçlarına göre AccOppSCO, SCO algoritmasına göre daha iyi başarım göstermiştir. AccOppSCO algoritması En İyi ve En Kötü metrik değerleri açısından tüm test işlevlerinde 19/23 başarı oranıyla SCO algoritmasını %82,61 oranında geride bırakmış, 23 fonksiyonun 16'sında Medyan ve Ortalama metriklerde SCO'dan daha başarılı (%69,57) olmuş ve ayrıca daha düşük bir Standart Sapmayla (%86,96) test işlevlerinde daha tutarlı ve istikrarlı bir başarım göstermiştir.

AccOppSCO ile SCO algoritmalarını başarım yönünden karşılaştırmak için yapılan analizlerde; ele alınan 23 minimizasyon probleminden test problemleri olarak 7, 14, 17 ve 21 olmak üzere dört farklı minimizasyon problemi seçilmiş ve iki boyutlu analizleri yapılmıştır. Çalışmada kullanılan analizlerden; optimizasyon süreci boyunca adayın arama uzayındaki hareketini gösteren arama geçmişi analizinde, adayın konumunun ilk boyutunun değişimini gösteren yörünge analizinde, optimizasyon boyunca adayın maliyet değerindeki değişim analizinde ve aday çözümlerin yakınsamasını gösteren yakınsama analizinde elde edilen sonuçlara göre, optimizasyon sürecinde önerilen AccOppSCO yapısının SCO yapısına göre daha iyi bir yakınsamaya sahip olduğu açıkça görülmüştür.

Ayrıca kümeleme probleminin çözümü için önerilen AccOppSCO algoritmasının başarımı değerlendirilmiştir. Değerlendirme için Yarpiz web sitesinden elde edilen 300 adet iki boyutlu veri noktasından oluşan bir kümeleme veri kümesi kullanılmıştır. AccOppSCO'yu SCO ile karşılaştırmanın yanı sıra, AccOppSCO algoritmasını popüler sezgisel algoritmalarla, yani Genetik Algoritma (GA), Farksal Gelişim (DE) ve Parçacık Sürüsü Optimizasyonu (PSO) ile karşılaştırmak için testler yapılmıştır. AccOppSCO, karşıtlığa dayalı öğrenme ve uyarlanabilir hızlanma katsayısının birleştirilmesi sayesinde, keşif ve yerelde arama arasında daha iyi bir denge sağlayarak, iyileştirilmiş optimizasyon sonuçlarına yol açmıştır. Elde edilen sonuçlar, AccOppSCO'nun En İyi, Medyan, Ortalama ve Standart Sapma gibi çeşitli metrikler açısından sürekli olarak SCO'dan daha iyi başarım gösterdiğini göstermektedir. Ayrıca AccOppSCO, test fonksiyonlarını çözmede gelişmiş kararlılık ve tutarlılık

sergileyerek karmaşık optimizasyon zorluklarının üstesinden gelme yeteneğini sergilemektedir. AccOppSCO, erken yakınsamadan kaçınırken arama alanını verimli bir şekilde keşfetme becerisiyle, çok çeşitli gerçek dünya uygulamaları ve karmaşık optimizasyon sorunları için büyük bir potansiyele sahiptir. Daha fazla araştırma ve iyileştirmeler yapıldıkça, AccOppSCO'nun çeşitli alanlarda değerli ve güvenilir bir optimizasyon algoritması olarak ortaya çıkması beklenmektedir.

Kümeleme problemi için, AccOppSCO'daki hızlandırılmış karşıt öğrenme tabanlı mekanizma, gelişmiş yakınsama ve daha başarılı kümeleme sonuçlarına katkıda bulunmuş ve böylece AccOppSCO algoritması orijinal SCO algoritmasına göre üstün başarım göstermiştir. AccOppSCO algoritmasını popüler sezgisel algoritmalarla, yani Genetik Algoritma (GA), Parçacık Sürüsü Optimizasyonu (PSO) ve Farksal Gelişim (DE) ile karşılaştırmak için yapılan testlerdeki beş yineleme optimizasyonunun istatistiksel sonuçları, AccOppSCO algoritmasının minimum metrikte ikinci en iyi değeri elde ettiğini ve GA'nın en iyi sonucu verdiğini göstermektedir. Maksimum metrik açısından AccOppSCO, SCO algoritmasından daha iyi başarım sergilemektedir. Ortalama metrik değerleri göz önüne alındığında, GA algoritması ilk sırayı almış ve AccOppSCO, SCO'dan daha iyi başarım göstermiştir. Standart sapma sonuçları, AccOppSCO algoritmasının en yüksek sapma değerine sahip olduğunu ortaya koymuştur.

Ayrıca, tekrarlanan çalıştırmalar için algoritmaların çalışma süreleri ve istatistiksel sonuçları analiz edilmiştir. Sonuçlar, AccOppSCO ve SCO mekanizmalarında kullanılan tek adaylı çözüm yaklaşımının önemli avantajını vurgulamaktadır.

Sonuç olarak, AccOppSCO algoritmasının geleneksel SCO algoritmasıyla kapsamlı karşılaştırma sonuçlarına dayalı olarak, oldukça etkili ve umut verici bir optimizasyon tekniği olduğu kanıtlanmıştır. Kümeleme problemi için AccOppSCO algoritması orijinal SCO algoritmasına göre üstün başarım göstermiştir. AccOppSCO'daki hızlandırılmış karşıt öğrenme tabanlı mekanizma, gelişmiş yakınsama ve daha başarılı kümeleme sonuçlarına katkıda bulunmuştur. AccOppSCO algoritması, verimli çalışma süreleri ve dikkat çekici sonuçlarıyla, kümeleme sorunlarına umut verici bir yaklaşım olarak ortaya çıkmaktadır. Ayrıca AccOppSCO, çeşitli metriklerde GA, PSO ve DE popüler sezgisel algoritmalarına karşı rekabetçi bir başarım göstermiştir.

KAYNAKÇA

Akyol, S., & Alataş, B. (2012). Güncel Sürü Zekası Optimizasyon Algoritmaları. *Nevşehir Üniversitesi Fen Bilimleri Enstitü Dergisi*(1), 36-50.

Appadurai, J. P., Raveendran, L. S., Latha, R., Gangadevi, E., Shri, M., & Gite, S. (2024). Cybersecurity Using Hybrid Type Model for Classification Through SCO Optimization Technique. *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)* (s. 1390-1396). Greater Noida, India: IEEE.

Balcı, M., Yüzgeç, U., & Dokur, E. (2024). Adaptive Neuro Fuzzy Hybrid Models Based on Metaheuristic Algorithms for Wind Speed Forecasting. *7. International Ankara Multidisciplinary Studies Congress*, (s. 553-562). Ankara.

Chagra, W., & Attia, S. B. (2024). Nonlinear Model Predictive Control Based on RBF Neural Network Trained by Stochastic Methods. *2024 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)* (s. 1-6). Hammamet, Tunisia: IEEE.

Dinkar, S. K., & Deep, K. (2018). Accelerated Opposition-Based Antlion Optimizer with Application to Order Reduction of Linear Time-Invariant Systems. *Invariant SystemsArabian Journal for Science and Engineering*(3), 2213-2241.

Dokur, E., Şengör, İ., Erdoğan, N., & Yüzgeç, U. (2023). Smart Meter Data-Driven Voltage Forecasting Model for a Real Distribution Network Based on SCO-MLP. *2023 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, (s. 1-5). Grenoble, France.

Dorigo, M., & Di Caro, G. A. (1999). Ant colony optimization: a new meta-heuristic. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress onVolume: 2* (s. 1470–1477). Washington, DC, USA: IEEE Xplore.

Emek, H. İ., & Yüzgeç, U. (2024a). Application of Chaotic Mutation Strategy Based SCO Algorithm to Engineering Design Problems. *7. International Ankara Multidisciplinary Studies Congress*, (s. 571-581). Ankara, Turkey.

Emek, H. İ., & Yüzgeç, U. (2024b). Chaotic Mutation Strategy Based Single Candidate Optimizer Algorithm. *7. International Ankara Multidisciplinary Studies Congress*, (s. 582-589). Ankara, Turkey.

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* New York: Cambridge University Press.

Heris, M. K. (2015). *Evolutionary Data Clustering in MATLAB.* [Erişim: 09.11.2024, https://yarpiz.com/64/ypml101-evolutionary-clustering]

Holland, J. H. (1975). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* London: University of Michigan Press, Ann Arbor.

Kahraman, Ü. A. (2010). Klasik ve Zeki Optimizasyon Teknikleri ile Problemlerin Çözümlenmesi ve Karşılaştırılması. (Yayınlanmamış Yüksek Lisans Tezi). *Sakarya Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi*, 1. Sakarya: Sakarya Üniversitesi.

Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization.* Kayseri: Erciyes Üniversitesi.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2012). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review, 42*, 21-57.

Karakaş, M., & Yüzgeç, U. (2019). Opposition based Gray Wolf Algorithm for Feature Selection in Classification Problems. *3rd International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT 2019.* Ankara.

Karakuzu, C. (2017). On the performance of newsworthy meta-heuristic algorithms based on point of view fuzzy modelling. *Turkish Journal of Electrical Engineering and Computer Sciences, 25*(6), 4706-4721.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks* (s. 1942-1948). Perth, WA, Australia: IEEE.

Kumar, M., Husain, M., Upreti, N., & Gupta, D. (2010). Genetic algorithm: Review and application. *Journal of Information & Knowledge Management*, Available at SSRN 3529843.

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software*, 51-67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, 46-61.

Özçelik, T. Ö., & Gündüz, G. (2019). Sezgisel Algoritmaları Kullanarak Raf Optimizasyonu Çalışması. *Avrupa Bilim ve Teknoloji Dergisi*(16), 977-982.

Pillai, S. E., Vallabhaneni, R., Pareek, P. K., & Dontu, S. (2024). Strengthening Cybersecurity using a Hybrid Classification Model with SCO Optimization for Enhanced Network Intrusion Detection System. *2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT)* (s. 1-9). Bengaluru, India: IEEE.

Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential Evolution: A Practical Approach to Global Optimization.* Berlin: Springer.

RajyaLakshmi, T., & Vinta, K. S. (2024, June). An effective deep learning based Idrcnn and Bdc-Lstm models for complex word identification and synonym generation. *International Journal of Information Technology*.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences, 179*(13), 2232-2248.

Rutenbar, R. (1989). Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine, 5*(1), 19-26.

Shami, T. M., Grace, D., Burr, A. G., & Mitchell, P. D. (2022). Single candidate optimizer: a novel optimization algorithm. *Evolutionary Intelligence, 17*, 1-25.

Şen, T. (2014, Haziran). Kümeleme ve Genetik Algoritma Destekli Yaklaşımlarla Kapasite Kısıtlı Araç Rotalama Probleminin Çözümü: Perakende Zincirinde Uygulanması. *Sakarya Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi*, 23. Sakarya: Sakarya Üniversitesi.

Tizhoosh, H. (2005). Opposition-Based Learning: A New Scheme for Machine Intelligence. *nternational Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). 1*, s. 695-701. Vienna, Austria: IEEE.

Yang, X. S. (2012). Flower Pollination Algorithm for Global Optimization. *Proceedings of the 11th International Conference on Unconventional Computing and Natural Computation* (s. 240-249). Orléans, France: Unconventional Computation and Natural Computation (UCNC 2012).

Yüzgeç, U., & İnaç, T. (2016). Opposition Based Spiral Optimization Algorithm. *International Conference on Computer Science and Engineering*, 124-128.

Yüzgeç, U., & Karakuzu, C. (2018). Training Multi-Layer Perceptron using Opposition based Learning Spiral Optimization Algorithm. *International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18)*, (s. 75-79). Safranbolu.