TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK BOYUTLU VERİLERDE ÖZNİTELİK SEÇİMİ İÇİN İKİLİ YAPAY ARI KOLONİSİ YAKLAŞIMI

Zeynep Banu ÖZGER

DOKTORA TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Danışman
Prof. Dr. Banu DİRİ

Eş-Danışman
Dr. Öğretim Üyesi Bülent BOLAT

Mayıs, 2019

TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK BOYUTLU VERİLERDE ÖZNİTELİK SEÇİMİ İÇİN İKİLİ YAPAY ARI KOLONİSİ YAKLAŞIMI

Zeynep Banu ÖZGER tarafından hazırlanan tez çalışması 22.05.2019 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programı **DOKTORA TEZİ** olarak kabul edilmiştir.

Prof. Dr. Banu DİRİ
Yıldız Teknik Üniversitesi
Danışman

Dr. Öğretim Üyesi Bülent BOLAT
Yıldız Teknik Üniversitesi
Eş-Danışman

Jüri Üyeleri

Prof. Dr. Banu DİRİ, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Zeynep ORMAN, Üye
İstanbul Üniversitesi- Cerrahpaşa _____

Doç. Dr. Arzucan ÖZGÜR, Üye
Boğaziçi Üniversitesi _____

Dr. Öğretim Üyesi Oğuz ALTUN, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğretim Üyesi Zeynep KURT, Üye
Yıldız Teknik Üniversitesi _____

Danışmanım Prof. Dr. Banu DİRİ sorumluluğunda tarafımca hazırlanan Büyük Boyutlu Verilerde Öznitelik Seçimi İçin İkili Yapay Arı Kolonisi Yaklaşımı başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Zeynep Banu ÖZGER

İmza

TEŞEKKÜR

Tezin hazırlandığı süre boyunca, bilgi ve deneyimini paylaşan, motive eden ve önerileriyle bana her zaman destek olan değerli tez danışmanım Sayın Prof. Dr. Banu DİRİ ve eş danışmanım Sayın Dr. Öğretim Üyesi Bülent BOLAT'a teşekkürlerimi sunarım. Tez izleme dönemleri boyunca komitemde yer alan, önerileriyle tezime katkıda bulunan Doç Dr. Arzucan ÖZGÜR, Doç. Dr. Zeynep Orman, Dr. Öğretim Üyesi Oğuz ALTUN'a ve çalışmamı yürüttüğüm süre boyunca yardımcı olan Dr. Pınar CİHAN'a teşekkür ederim.

Zeynep Banu ÖZGER

İÇİNDEKİLER

SİMGE LİSTESİ

p(y)	Y değişkeni için marjinal olasılık yoğunluk fonksiyonu
p(y|x)	x biliniyorken y'nin koşullu olasılığı
$\overline{r_{cf}}$	Ortalama öznitelik-sınıf korelasyonu
$\overline{r_{ff}}$	Ortalama öznitelik-öznitelik korelasyonu
$diff(A_{ij}, H_{ij})$	A_{ij} ile H_{ij} arasındaki mesafe
ϕ	Rastgele üretilen uniform dağılımlı bir değer.
S()	Sigmoid fonksiyonu
rand(0,1)	0-1 aralığında rastgele üretilmiş bir değer.
U(0,1)	0-1 aralığında rastgele üretilmiş uniform dağılımlı bir değer.

KISALTMA LİSTESİ

AMABC	Açı Modülasyonlu İkili Yapay Arı Kolonisi
BBA	Bulanık Bileşen Analizi
BCO	İkili Bakteri Algoritması
BinPSO	İkili Parçacık Sürüsü Optimizasyonu
BinABC	İkili Yapay Arı Kolonisi
BitABC	Bitişlem Operatörleri Tabanlı İkili Yapay Arı Kolonisi Algoritması
BK	Bilgi Kazancı
bS	Başarısızlık sayısı
CR	Çaprazlama oranı
DABC	Ayrık Yapay Arı Kolonisi
DDA	Doğrusal Diskriminant Analizi
DE	Diferansiyel Evrim
DisABC	Ayrık İkili Yapay Arı Kolonisi
DNA	Deoksiribo Nükleik Asit
DVM	Destek Vektör Makineleri
exBitABC	Lokal Arama Tabanlı İkili Yapay Arı Kolonisi Algoritması
FN	Sahte negatif
FP	Sahte pozitif
GA	Genetik Algoritma
GBABC	Genetik Operatörler Tabanlı Yapay Arı Kolonisi
GC	Guanine Sitosin
GcRMA	Guanin Sitosin RMA
KKO	Karınca Kolonisi Optimizasyonu

KKTF	Korelasyon Katsayısı Tabanlı Filtreleme
K-NN	K En Yakın Komşu
MAS5	Mikrodizi Paketi 5.0
MM	Uyumsuzluk
MRABC	Modifikasyon Oranına Dayalı Yapay Arı Kolonisi
mRMR	Minimum Artıklık ve Maksimum Alaka
NBPSO	Yeni Hız Tabanlı İkili Parçacık Sürü Optimizasyonu
OYSA	Olasılıksal Yapay Sinir Ağı
PM	Mükemmel uyum
PrBABC	Olasılıksal İkili Yapay Arı Kolonisi
PSO	Parçacık Sürü Optimizasyonu
QBPSO	Quantum Tabanlı İkili Parçacık Sürü Optimizasyonu
RMA	Çoklu Dizi Ortalama
TN	Gerçek negatif
TP	Gerçek pozitif
VK	Varyasyon katsayısı
YAK	Yapay Arı Kolonisi

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖZET

Büyük Boyutlu Verilerde Öznitelik Seçimi İçin İkili Yapay Arı Kolonisi Yaklaşımı

Zeynep Banu ÖZGER

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Prof. Dr. Banu DİRİ
Eş-Danışman: Dr. Öğretim Üyesi Bülent BOLAT

Bilgisayar alanındaki hızlı ilerlemeler neticesinde veri tabanlarında çok fazla öznitelik içeren büyük miktarlarda bilgi depolanmaktadır. Ancak mevcut özniteliklerin hepsi verinin yorumlanabilmesine katkı sağlamayabilir. Bu ilgisiz öznitelikler büyük arama uzayı oluşturduğundan sınıflama/kümeleme başarısını olumsuz etkilemektedir. Bu nedenle benzer veya daha iyi performans elde edebilmek için veriyi doğru temsil eden öznitelik alt gruplarının belirlenmesi önem kazanmaktadır.

Yapay Arı Kolonisi (YAK) algoritması doğadan esinlemeli bir sürü zekası optimizasyon algoritmasıdır. Algoritma, bal arılarının doğadaki besin arama davranışlarını modellemektedir. Sürekli uzay problemleri için geliştirilmiş olan algoritma, hızlı ve efektif çözümler sunmaktadır. Ancak ayrık uzay problemlerine uygulamak için modifiye edilmesi gerekmektedir.

Tez kapsamında; sınıflandırma ve kümeleme alanlarında büyük boyutlu veriler söz konusu olduğunda genel olarak karşılaşılan işlem maliyeti, hesaplama zamanı ve düşük sınıflandırma/kümeleme başarısı problemlerinin çözümü için YAK tabanlı bir yaklaşım geliştirmek amaçlanmıştır.

Öznitelik seçimi, bir optimizasyon algoritması ile çözümlenmek istendiğinde, ikili arama uzayına ihtiyaç duyduğu için tez kapsamında, YAK algoritmasının ikili uzaya taşımak amaçlanmıştır. Bu kapsamda ilk olarak, literatürde mevcut ikili YAK algoritmaları, öznitelik seçimi problemine uygulanmış ve 15 algoritma karşılaştırılarak

güçlü ve zayıf yönleri belirlenmiştir. Öznitelik seçimi için efektif olduğu görülen Bitişlem Operatörleri Tabanlı İkili Yapay Arı Kolonisi Algoritması (BitABC), altı farklı sınıflandırıcı ile farklı veri kümelerine uygulanarak, sınıflandırıcı performansları karşılaştırılmıştır. BitABC algoritmasının efektif ancak lokal arama kapasitesi yetersiz görüldüğünden ilk olarak sürünün en iyi bireyi etrafında yapılmak üzere bir lokal arama fonksiyonu eklenmiştir. Sonuçlar lokal arama fonksiyonunun başarıyı artırdığını göstermektedir. Sonraki adımda lokal aramanın kapsamı genişletilerek, algoritmanın sezgiselliğini etkilemeyecek şekilde işçi ve gözcü arı aşamalarına da eklenmiştir. Geliştirilen yöntem çeşitli büyüklüklerde 13 veri kümesinde test edilmiş ve sonuçlar evrimsel algoritmalar ile karşılaştırılmıştır.

Gen ekspresyon seviyelerini ölçmek için kullanılan bir teknoloji olan mikrodiziler, binlerce boyuttan oluşan veri kümeleridir ve her bir boyut bir geni temsil etmektedir. Hastalık ile doğrudan ilişkili genlerin tespiti için boyut indirgeme yapılması gerekmektedir. Gen seçimi işlemini YAK algoritması ile efektif bir şekilde çözebilmek için öğrenme stratejisini kendinden uyarlamalı bir yöntem ile belirleyen Olasılıksal İkili Yapay Arı Kolonisi (PrBABC) algoritması geliştirilmiş ve dokuz veri kümesinde performansı test edilmiştir. Evrimsel algoritmalar ile sonuçlar karşılaştırıldığında önerilen yöntemin gen seçiminde başarılı olduğu görülmüştür.

Anahtar Kelimeler: Öznitelik seçimi, mikrodizi gen seçimi, ikili yapay arı kolonisi, sezgisel algoritmalar, optimizasyon, makine öğrenmesi.

ABSTRACT

Binary Artificial Bee Colony Approach for Feature Selection in Large Size Data

Zeynep Banu ÖZGER

Department of Computer Engineering
Doctor of Philosophy Thesis

Advisor: Prof. Dr. Banu DIRI
Co-advisor: Asst. Prof. Dr. Bulent BOLAT

As a result of rapid advances in the computer space, large amounts of information are stored in databases that contain many attributes. However, not all existing attributes may contribute to the interpretation of the data. These unrelated attributes create a large search space, this situation negatively affects the classification / clustering performance. Therefore, in order to achieve similar or better performance, it is important to identify the subset of attributes that represent the data correctly.

Artificial Bee Colony algorithm is a nature-inspired, swarm intelligence optimization algorithm. Algorithm models the food search behavior of honey bees in the nature. The algorithm developed for continuous space problems and provides fast and effective solutions. But it has to be modified to apply to discrete space problems.

Large-scale data need more computational cost and can cause low classification/clustering performance. Within the scope of the thesis;it is aimed to develop a binary Artificial Bee Colony algorithm that can find related feature subsets.

Since the feature selection needs a binary search space, a binary version of the Artificial Bee Colony algorithm is proposed within the scope of the thesis. In this context, firstly, the binary Artificial Bee Colony algorithms available in the literature were applied to the problem of feature selection and 15 algorithms were compared and their strengths and weaknesses were examined. The bitwise operations based binary artificial bee

colony algorithm (BitABC), which is seen to be effective for feature selection, is applied to different size datasets with six different classifiers, and their classifier performances are compared. BitABC is an effective algorithm for feature selection but it is seen that its local search capacity is inadequate. Therefore, firstly, a function that make local search around the global best has been added. The results show that the local search function increases sthe performance of the algorithm. In the next step, the scope of the local search was expanded and added to the employed and onlooker bee phases in a way that would not affect the heuristic capability of the algorithm. The developed method was tested in 13 datasets and the results were compared with well-known evolutionary algorithms. According to results, the proposed method could find a better feature subset than the other methods.

Microarray is a technology used to measure gene expression levels. They are datasets with thousands of dimensions, and each dimension represents a gene. Dimension reduction is required to identify genes that are directly related to the disease. The probabilistic binary Artificial Bee Colony (PrBABC) algorithm, which determines the learning strategy by a self-adaptive method, has been developed for gene selection problem and its performance is tesed with nine microarray datasets. When comparing the results with evolutionary algorithms, the proposed method could identify related genes effectively.

Keywords: Feature selection, microarray gene selection, binary artificial bee colony, heuristic algorithms, optimization, machine learning.

1
Giriş

Bir veri kümesinin çok fazla öznitelik içermesi, o veride sınıflandırma veya kümeleme işleminin hesapsal maliyetini artırdığı gibi başarının düşmesine de neden olabilir. Veri kümesinden anlamlı olan özniteliklerin ayırt edilmesine öznitelik alt küme seçimi denir. Öznitelik seçimi, veri analizinde kullanılan önemli bir ön işleme adımıdır.

Özellik seçiminin amacı, doğruluktan ödün vermeden, belirli bir veri kümesinden, ayırt ediciliği az olan ilgisiz özellikleri eleyerek, mevcut özniteliklerden bir alt küme oluşturmaktır. Böylece, hesapsal maliyet azaltılarak, daha iyi veya en azından eşdeğer performans elde edilebilir. Öznitelik seçim yöntemleri, filtreler, sarmalayıcılar ve hibrit yöntemler olarak üç temel gruba ayrılmaktadır. Filtreler, öznitelikleri t-score, p-istatistiği vb. gibi bazı istatistiksel yöntemlere göre sıralayarak içlerinden en iyi N tane özniteliği seçer. Filtre yöntemleri, öznitelikler arasındaki ilişkileri göz ardı eder ve birbirinden bağımsız olduklarını varsayar. Bu nedenle hesaplama maliyetleri daha az olmasına karşın performansları da düşüktür [1]. Sarmalayıcılar ise, öznitelikler arasındaki ilişkileri de dikkate alarak daha iyi öznitelik altkümelerini seçen endüktif yöntemlerdir. Seçilen alt kümelerin performansı bir öğrenme algoritması ile ölçülür. Sarıcıların hesaplama karmaşıklığı filtrelerden daha yüksektir ve özniteliklerin sayısına göre katlanarak artmaktadır [2]. Hibrit yaklaşımlar ise, her iki yöntemin de avantajlarından yararlanmak için iki yöntemi birleştirir. Bu yöntemlerde, öznitelik boyutunu azaltmak için önce verilere bir filtreleme yöntemi uygulanır, ardından kalan öznitelikler arasından en iyi alt küme bir sarmalayıcı yardımıyla bulunur.

Daha iyi performans gösterdikleri için sarmalayıcılar, öznitelik seçiminde yaygın olarak uygulanmaktadır. En iyi alt kümeyi belirleyebilmek için, tüm olası öznitelik alt kümelerinin test edilmesi gerekir ki özellikle yüksek boyutlu verilerde bu mümkün değildir. Doğadan esinlemeli algoritmalar çözüm uzayında, sezgisel bir arama yaparak daha makul sürelerde daha efektif sonuçların elde edilmesini sağlar. Sezgisel algoritmalar, olası tüm çözümleri test etmez, bazı global ve yerel arama stratejileri kullanarak optimal bir çözüm bulurlar. Performansları nedeniyle, Genetik Algoritma (GA) [3, 4], Karınca Kolonisi Optimizasyonu (KKO) [5, 6], Parçacık Sürü

Optimizasyonu (PSO) [7, 8], Diferansiyel Evrim (DE) Algoritması [9, 10] gibi birçok sezgisel algoritma, öznitelik seçimi problemine uygulanmıştır.

Yapay Arı Kolonisi (YAK), Karaboğa [11] tarafından, sürekli uzaydaki problemlere optimize çözüm bulabilmek için önerilmiş bir sürü zekası algoritmasıdır ve bal arılarını modellemektedir. Algoritma sürekli uzayda birçok probleme uygulanmış ve efektif sonuçlar alınmıştır.

Tez çalışmasının amacı, YAK algoritmasının ikili bir versiyonunu geliştirmek ve öznitelik seçimi problemine uyarlamaktır. Bu kapsamda, 2. Bölümde büyük boyutlu bir veri kümesi olan mikrodizilerde veri önişleme tekniklerine değinilmiştir. Üçüncü bölümde, öznitelik seçim yöntemleri ile ilgili bilgi verilmiş, dördüncü bölümde ise YAK algoritmasının temellerinden bahsedilmiştir. Beşinci bölümde, elde edilen öznitelik kümelerinin performansını ölçmek için yararlanılan model oluşturma teknikleri ve değerlendirme metriklerine değinilmiştir. Altıncı bölümde, önerilen yöntemlerin performans sonuçları karşılaştırmalı olarak verilmiş ve son olarak yedinci bölümde bulgular tartışılmıştır.

1.1 Literatür Özeti

Yapay Arı Kolonisi algoritması, temelde sürekli uzay problemlerine çözüm üretmek için geliştirilmiştir. Literatürde algoritmanın ikili uzayda çözüm üretmek üzere modifiye edilmiş versiyonlarına bu bölümde değinilmiştir.

YAK algoritması, ilklendirme aşaması, işçi arı aşaması, gözcü arı aşaması ve kaşif arı aşaması olmak üzere 4 adımdan oluşmaktadır. Algoritma, olası çözümleri vektörler ile temsil etmektedir. Vektörün her bir boyutu optimize edilecek problemin bir parametresine karşılık gelir. İlklendirme ve işçi arı aşamalarında, yeni çözümler sunabilmek için vektörel toplama ve çıkarma işlemleri kullanılmaktadır. Algoritma sürekli uzayda çözüm üretmek üzere tasarlandığı için, kaynakları temsil eden vektörler de sürekli değerlerden oluşmaktadır.

YAK algoritmasını ikili veya ayrık uzaya taşımak için literatürde temel olarak iki yaklaşım kullanılmıştır:

1. Olası çözümler ikili/ayrık vektörler ile temsil edilerek, ilklendirme ve işçi arı aşamalarındaki yeni çözüm üretme stratejileri ikili/ayrık uzayda çözüm üretebilecek şekilde modifiye edilmiştir.
2. Çözümler sürekli uzayda üretilmiş ancak sonrasında bir dönüşüm fonksiyonu ile ikili uzaya taşınmıştır.

Literatürdeki ilk ikili YAK algoritmalarından biri olan DisABC Kasha vd. [12] tarafından önerilmiştir. İşçi arı aşamasında, yeni kaynak üretmek için mevcut vektörlere belirli bir oranda benzeyen yeni vektörler üretilmiştir. Mevcut kaynaklara olan benzerliği ölçmek için ise Jackard Benzerlik Ölçümü'nden faydalanılmıştır. Önerilen yöntem kapasitesiz tesis yerleşimi (uncapacitated facility location) problemine uygulanmıştır.

Ozturk vd. [13] DisABC'yi Genetik Algotirma ile harmanlayarak ikili uzayda arama yapabilen bir yöntem önermişlerdir (IDisABC). Buna göre DisABC'de üretilen sonuçlara çaprazlama işlemi uygulanarak popülasyonda çeşitliliği artırmayı hedeflemişlerdir. Yöntem dinamik kümeleme problemine uygulanmıştır.

Hancer vd. [14] ise DisABC algoritmasını Diferansiyel Evrim algoritmasıyla birleştirmiştir (MrABC). DisABC ile önerilen çözümler DE ile çeşitlendirilmiş ve yöntem 10 verisetinin kullanıldığı öznitelik seçimi problemine uygulanmıştır.

Singhal vd. [15], yeni kaynak üretmek için DisABC algoritmasından faydalanmış ancak farklı olarak zeki bir kaşif arı fazı kullanılmıştır. Buna göre kaşif arı fazında kaynak, rastgele ilklendirilmek yerine, popülasyondaki en iyi kaynak olarak atanmıştır. Yöntem birim belirleme (unit commitment) problemine uygulanmıştır.

Cui ve Gu [16] yeni kaynakları, ikili uzayda tanımladığı vektörler üzerinde, Diferansiyel Gelişim Algoritması operatörlerini kullanarak üretmiştir.

Ozturk vd. [17] YAK algoritmasını GA ile birlikte kullanmış ve olası çözümleri ikili vektörler ile temsil etmiştir. Önerilen yöntemde (GBABC), yeni çözüm üretmek için genetik çaprazlama operatöründen faydalanılmıştır. Önerilen yöntemin başarısı dinamik resim kümeleme ve 0-1 sırt çantası problemlerinde test edilmiştir.

Yurtkuran ve Erdal [18] ikili uzayda aday çözüm üretebilmek için çaprazlama operatörünü önermişlerdir. Buna göre önerilen dört farklı çaprazlama yönteminden biri rastgele seçilerek ilgili iterasyon için aday kaynak üretilmiştir. Yazarlar önerdikleri yöntemi tek makine çizelgeleme (single machine scheduling) problemine uygulamıştır.

Jia vd. [19], vektörleri ikili uzayda ilklendirirken, vektörün her bir boyutu için [0,1] aralığında rastgele bir sayı üretmiş ve bu sayı 0,5'den küçükse, ilgili parametreye 0, büyükse 1 değerini atamıştır. İşçi arı aşamasında ise mevcut kaynaklardan yeni çözümler üretebilmek için 'xor', 'and' ve 'or' bit işlem operatörlerini kullanmışlardır (BitABC). Yöntem 13 fonksiyonun optimizasyonu için uygulanmıştır.

Kıran ve Gündüz'de [20] çalışmalarında bit işlem operatörlerinden faydalanmıştır. Ancak, yazarlar farklı olarak yeni kaynak üretmek için sadece 'xor' operatöründen

faydalanmıştır. Yöntem kapasitesiz ikili konum (uncapacitated binary location) problemine uygulanmıştır.

Ribas vd. [21] işçi arı aşamasında ekleme (insert) ve yer değiştirme (swap) operatörlerini kullanarak arama uzayında yeni çözümler üretilmişlerdir. Önerilen yöntem seri üretim (flow-shop) problemine uygulanmıştır. Tasgetiren vd. [22] işçi arı aşamasında ekleme ve yer değiştirme operatörleri ile altı kaynak üretmiş ve bunlardan en iyi olanını lokal arama ile belirlemişlerdir. Önrilen yöntem akış-zamanı minimizasyonu (flow-time minimization) probleminde kullanılmıştır. Zhang ve Gu da [23], işçi arı aşamasında ekleme ve yer değiştirme operatörlerini kullanarak sekiz kaynak üretmiş ve bunlardan en iyisini seçerek aday çözümü belirlemişlerdir. Gözcü arı fazında ise, turnuva seçimi (tournament selection) yöntemini kullanmışlar ve yöntem seri üretim planlama problemine uygulanmıştır. Pan vd. [24] işçi ve gözcü arı aşamalarında ekleme, değiştirme ve ikili değişim (pairwise exchange) işlemlerini kullanarak üretilmiş kaynaklardan en iyi olanını seçmiştir. Önerilen yöntem seri üretim planlama problemine uygulanmıştır.

Ye vd. [25] işçi arı aşamasında mevcut kaynağın, popülasyondan rastgele seçilmiş bir kaynak ve popülasyonun en iyi kaynağı ile olan mesafesine göre bir hız vektörü elde ederek yeni kaynağı bu hız vektörüne bağlı olarak üretmişlerdir. Kaynaklar arasındaki mesafeyi ölçmek için Hamming Mesafe ölçümü kullanılmıştır. Önerilen yöntem asgari nitelik azaltma (minimum attribute reduction) problemine uygulanmıştır.

Zhang vd. [26] belirli bir Hamming mesafesine göre mevcut kaynakta bit değişimi yaparak komşu kaynak üretmişlerdir. Yazarlar önerdikleri yöntemi, büyük ölçekli entegre devrelerin tasarımında kullanılan engellerden kaçınan doğrusal Steiner minimal ağaç (obstacle-avoiding rectilinear Steiner minimal tree) problemine uygulamıştır.

Chen ve Kanoh [27]'de Hamming mesafe ölçümü tabanlı bir yöntem önermiştir. Hesaplanan benzerlik oranına göre kaynak vektörde bit değişimi yapılarak aday çözüm üretilmiştir. Önerilen yöntem graf boyama probleminde kullanılmıştır.

Literatürde, YAK algoritmasının sürekli uzayda ürettiği sonuçları çeşitli dönüşüm fonksiyonları kullanarak ikili uzaya taşıyan yöntemler de mevcuttur.

Wei ve Chen [28] YAK algoritmasıyla sürekli uzayda bulduğu çözümleri yuvarlama fonksiyonunu (round) kullanarak ikili uzaya taşımıştır. Yöntem dört fonksiyonun optimizasyonunda kullanılmıştır.

Kıran [29] yuvarlama fonksiyonunu mod alma operatörü ile birleştirerek sürekli

uzaydaki sonuçları ikili uzaya taşımıştır. Yazar önerdiği yöntemi kapasitesiz tesis yerleşimi problemine uygulamıştır.

Mandala ve Gupta [30], önerdikleri yöntemde dönüşüm foksiyonu olarak tanjant fonksiyonunu kullanırken, Tran vd. [31] ise sigmoid fonksiyonunu tercih etmiştir.

Pampara vd. [32] açı modülasyon (angle modulation) tekniğini kullanarak YAK algoritmasını ikili uzaya taşımıştır. Yöntemin yetkinliği 0-1 sırt çantası problemi ve n-Kral yöntemi ile denenmiştir.

Tez çalışması kapsamında büyük boyutlu veride öznitelik seçimi yapmak amaçlandığından, literatürde mikrodizi verilerinde optimizasyon algoritmaları ile öznitelik seçimi yapan çalışmalar da incelenmiştir. DNA mikrodizileri, hücrelerde ve dokularda gen ekspresyon profillerinde global değişiklikleri incelemek için kullanılan bir teknolojidir. Bu veriler sayısallaştırıldıkları zaman binlerce boyutu ve az sayıda örneği olan veri kümeleri oluştururlar. Mikrodizi verileri yüksek boyutlu olduğu için genellikle filtre ve sarıcı metodlar ile birlikte uygulanmıştır. Bolon vd. [33]'de, dağıtılmış bir ortamda Karşılıklı Bilgi Maksimizasyonu ve Minimum Artıklık ve Maksimum Alaka (mRMR) algoritmalarını kullanırken, [34], Bulanık Geriye Doğru Öznitelik Eleme (Fuzzy backward feature elimination) yöntemini Bağımsız Bileşen Analizi (BBA) ile birleştirmiştir. Kalaviani ve Kumar [35] önerdikleri hibrid modelde, Gaussian çekirdek yaklaşımını kullanmış ve ayırt edici genleri seçmek için bulanık kaba bir küme modeli (fuzzy rough set model) oluşturmuştur. Sun vd. [36], Destek Vektör Makineleri (DVM) sınıflandırıcısı ile Lagrange Çarpanı tabanlı bir öznitelik seçimi yöntemi önermiş ve sonuçlarını geleneksel filtre yöntemleriyle karşılaştırmıştır. Mortazavi ve Moattar [37] mikrodizi verileri için, nitel karşılıklı bilgileri (Qualitative Mutual Information) kullanan bir filtre yöntemi getirmiştir.

Guo vd. [38] Doğrusal Diskriminant Analizini (DDA), Lojistik Regresyon ile birlikte kullanmıştır. Çalışmada Lojistik Regresyon, bir optimizasyon algoritması olarak kullanılmış ve uygunluk fonksiyonu olarak sınıf ayrılabilirliği ölçüsü kullanılmıştır. Yazarlar bir sonraki çalışmalarında [39], Lojistik Regresyon sonrası öznitelik çıkarımı için Kısmi En Küçük Kareler yöntemini uygulamışlardır.

Wang vd. [40] mikrodizi veri kümelerinde İkili Bakteri Algoritması (BCO) ile öznitelik seçimi yapmıştır. Bireylerin oluşturabilecği alt kümelerdeki öznitelik sayıları, belirli bir değer ile sınırlandırıldığı için, herhangi bir filtreleme yöntemi kullanılmamıştır. Sonuçlar bazı geleneksel yöntemler ve üç evrimsel algoritma ile karşılaştırılmıştır.

Yang vd. [41], çalışmasında Bilgi Kazancı (BK) ve Korelasyon Katsayısı Tabanlı Filtreleme (KKTF) ile veri kümelerini filtrelemiş ve ardından öznitelik

alt kümelerini Geliştirilmiş İkili Parçacık Sürüsü Optimizasyonu (Binary Particle Swarm Optimization-BinPSO) algoritmasıyla optimize etmiştir. Öznitelik alt kümesi doğrulukları K-En Yakın Komşu (K Nearest Neighbour- K-NN) ve Destek Vektör Makineleri sınıflandırıcıları ile değerlendirilmiştir. Abdi vd. [42], Parçacık Sürü Optimizasyonunu hem DVM'de çekirdek parametrelerini optimize etmek hem de Minimum Artıklık ve Maksimum Alaka Düzeyi filtre metodu ile seçilen ilk n genin olduğu altkümeleri ağırlıklandırmak için kullanmıştır. Dara vd. ise [43], BinPSO algoritmasını kaba küme teorisi (Rough Set Theory) ile hibritleyerek gen seçim probleminde kullanmışlardır.

Lv vd. [44] çalışmalarında Dağılımın Tahmini (estimation of distribution) tabanlı MOEDA algoritmasını önermişlerdir. Buna göre tek değişkenli marjinal dağılım algoritması mRMR filtresi ile birleştirilerek uygulanmıştır.

Genetik algoritma gen seçimi için uygulanan popüler yöntemlerden biridir. El akadi vd. [45] genetik algoritmayı, mRMR filtresinin ardından sarıcı yöntem olarak uygularken, Yang vd. [46] filtreleme metodu olarak Bilgi Kazancı'nı tercih etmiştir. Hasnat ve Molla, [47] mikrodizi verilerini min-maks normalizasyon yöntemi ile normalize ettikten sonra, korelasyon katsayısı tabanlı filtreleme yöntemi ile gen sayısını azaltmışlardır ve ardından genetik algoritma uygulayarak gen alt kümelerini optimize etmişlerdir.

Tabakhi vd. [48] etiketlenmemiş veriler için filtre yöntemi olarak Karınca Kolonisi Optimizasyonu kullanmıştır. Bir öğrenme algoritması kullanılmamış olup, alt kümedeki genlerin mRMR değerlerinin toplamı normalize edilerek uygunluk fonksiyonu olarak kullanılmıştır.

Alshamlan vd. [49] Yapay Arı Kolonisi algoritmasını mRMR filtre metodu ile birleştirerek mikrodizilere uygulamıştır. Ardından yazarlar [50] Yapay Arı Kolonisini genetik algoritma ile hibritleştiren bir ikili YAK algoritması önermiş ve gen seçimi problemine uygulamıştır.

Mohamed vd. [51] YAK, Parçacık Sürü Optimizasyonu ve Guguklu Arama (Cuckoo search) algoritmalarını üç farklı uygunluk fonksiyonu kullanarak gen seçimi problemine uygulamıştır. Apolloni vd. [52] Diferansiyel Evrim algoritmasını Bilgi Kazancı filtresi ile uygulamıştır. Mohapatra vd. ise [53], Kedi Sürüsü Optimizasyon algoritmasını geliştirerek mikrodizi veri kümelerinde gen seçimi yapmışlardır.

Bu çalışma kapsamında, ilk olarak öznitelik seçiminde kullanılmak üzere ikili bir Yapay Arı Kolonisi Algoritması önerilmiştir. Önerilen yöntem de, önceki yöntemlerden farklı olarak, Yapay Arı Kolonisi Algoritması'nda algoritmanın sezgiselliği etkilenmeyecek

şekilde güçlü bir lokal arama özelliği eklenerek bir araya geldiğinde ayırt ediciliği yüksek olan öznitelik alt kümelerinin bulunabilmesi sağlanmıştır. İkinci olarak ise mikrodizi verilerinde gen seçimi yapmak için kullanılmak üzere, ikili bir Yapay Arı Kolonisi Algoritması önerilmiştir. Önerilen yöntemde önceki çalışmalardan farklı olarak, gen seçimi işlemi çok-amaçlı bir optimizasyon problemi olarak ele alınmış ve farklı öğrenme stratejileri incelenmiştir. Kendinden uyarlamalı bir yöntem ile farklı amaçları optimize edebilecek öğrenme stratejilerini içeren bir Yapay Arı Kolonisi algoritması geliştirilmiştir.

1.2 Tezin Amacı

Sayısal dünyada depolanan verilerin artması ile birlikte büyük verilerin saklanması ve işlem kapasitesinin artması problemleri ortaya çıkmıştır. Bu kapsamda, büyük boyutlu verilerin içinden anlamsız öznitelikleri çıkararak, ilgili öznitelikleri içeren alt kümeler belirleyerek depolama ve işlem maliyeti sorunlarının çözümü önem kazanmıştır.

Büyük boyutlu veride, ayırt ediciliği yüksek bir öznitelik alt kümesi belirlemek, büyük bir arama uzayında çözüm bulmayı gerektirir. Bu nedenle, öznitelik seçiminde sarmalayıcı yöntemler kapsamında kullanılan optimizasyon algoritmalarının, sezgisel yaklaşımlarıyla, daha efektif sonuçlar elde edeceği düşünülmüştür.

Doğadan esinlemeli bir optimizasyon algoritması olan Yapay Arı Kolonisi algoritması, sürekli uzay gerektiren problemlerde efektif çözümler sunmuştur. Ayrıca algoritmanın ayarlanması gereken parametre sayısının az olması, lokal optimumlardan kaçabilmesi ve güçlü bir global arama yeteneğine sahip olması nedeniyle, öznitelik seçimi problemini efektif bir şekilde çözebilecek ikili bir versiyonu geliştirilmek istenmiştir.

Algoritmanın önerilecek ikili versiyonu için literatürde mevcut yöntemler incelenerek, eksik ve iyi yanlarının anlaşılması üzerine, ilgili eksikliklerin giderileceği bir yöntemin iyileşme sağlayacağı düşünülmüştür. Bu kapsamda, bir araya geldiğinde ayırt ediciliğin artmasını sağlayan özniteliklerin belirlenmesi için, algoritmanın global arama yeteneğini ve sezgiselliğini kaybetmeden, bir lokal arama fonksiyonu ile desteklenmesinin başarıyı artıracağı fark edilmiştir. Ayrıca, araştırmalar neticesinde, her veri kümesinde aynı öğrenme stratejisiyle başarılı olunamayabileceği de görülmüş ve kendinden uyarlamalı bir şekilde farklı öğrenme stratejilerinin bulunduğu bir yöntem geliştirilerek, alt küme seçimine etkisi araştırılmıştır.

1.3 Hipotez

Tez çalışması, amacı, bir arama uzayında optimal çözüm bulmak olan optimizasyon algoritmalarından Yapay Arı Kolonisi algoritmasının, geliştirilecek ikili bir versiyonu ile büyük boyutlu veride, efektif bir şekilde ayırt edici öznitelik alt gruplarının bulunabileceği varsayımına dayanmaktadır. Çalışma kapsamında aşağıdaki hipotezler öne sürülmüştür:

- Sarmalayıcı yöntemler, öznitelikler arası ilişkileri dikkate aldığından, öznitelik seçiminde filte yöntemlerinden daha başarılı sonuçlar elde eder.
- Sarmalayıcı yöntemler kapsamında optimizasyon algoritmalarının kullanımı büyük arama uzayları için daha iyi sonuçlar verir.
- İkili uzayda arama yapabilecek şekilde geliştirilmiş bir Yapay Arı Kolonisi algoritması ile öznitelik alt grupları bulunabilir.
- Etkileşimli öznitelik alt gruplarını belirleyebilmek için, geliştirilecek ikili YAK algoritmasının lokal arama yeteneğinin artırılması başarıyı artırabilir.
- Geliştirilecek yöntem içerisinde birden fazla öğrenme stratejisi kullanmak farklı özelliklerdeki veri kümelerinde daha iyi performans alınmasını sağlayabilir.

2
Veri Önişleme

Bu bölümde, veri kümelerinde kullanılan veri önişleme adımlarına değinilmiş ve mikrodizi veri kümelerinde kullanılan normalizasyon yöntemleri açıklanmıştır.

2.1 Veri Önişleme Adımları

Gerçek dünya verileri ile oluşturulmuş veri kümeleri genellikle eksik, tutarsuz ve gürültülü veriler içermektedir. Bir veri kümesinin kalitesi, veri kümesinden elde edilecek modeli doğrudan etkileyeceği için, model oluşturulmadan önce veri kümesine önişleme adımları uygulanır. Veri önişleme için uygulanan temel adımlar: Veri temizleme, veri bütünleştirme, veri indirgeme, veri dönüştürme ve normalizasyon işlemlerinden oluşmaktadır [54].

2.1.1 Veri Temizleme

Veri temizleme, eksik değerlerin doldurulması, gürültülü verilerin düzeltilmesi, aykırı değerlerin belirlenmesi veya kaldırılması, verideki tutarsızlıkların giderilmesi gibi adımlar içerir.

Eksik veriler yok sayılabilir veya doldurulabilir. Eksik verileri tamamlamak için literatürde bir çok yöntem bulunmaktadır.

- Tüm eksik verilere aynı değer atanır. Bu değer özniteliğin ortalaması, ortancası olabileceği gibi rastgele bir değer de olabilir.
- İstatistiksel yöntemler kullanılarak tamamlanabilir,
- Evrimsel yöntemler kullanılarak tamamlama yapılabilir.

Veri kümesindeki gürültülü ve uç veriler ise, gruplama (binning), kümeleme, regresyon gibi yöntemler kullanılarak tespit edilip veri kümesinden çıkarılabilinir [55].

2.1.2 Veri Bütünleştirme

Veri bütünleştirme, verilerin çoklu veri tabanlarından alınan verilerin bütünleştirilmesini içerir. Böylece, aynı türde verinin farklı veri kümelerinde, farklı şekillerde ifade edilmesinden kaynaklanacak problemler önlenmiş olur. Ayrıca, verinin bütünleşmiş şekilde sunulmasını da sağlar. Özellikle, verinin büyüklüğü ve paylaşma gerekliliği arttıkça bu ön işleme adımı daha önem kazanmaktadır [55].

2.1.3 Veri İndirgeme

Büyük veri kümelerinden, özniteliklerin sayısının azaltılması işlemidir. Filtre ve sarmalayıcılardan oluşan öznitelik seçim yöntemleri ile öznitelik sayıları azaltılabilir. Temel Komponent Analizi gibi, verinin farklı bir uzaya taşınması ile mevcut özniteliklerden iyi özniteliklerin elde edilmesi ile de veri indirgeme yapılabilir. Ayrıca, kümeleme gruplama gibi yöntemler ile benzer öznitelikler gruplanarak veya kümelenerek de öznitelik sayısı azaltılabilir [55].

2.1.4 Veri Ayrıklaştırma

Sayısal verilerin kategorik verilere dönüştürülmesi işlemidir. Bu işlem aynı zamanda verideki gürültülü ve uç değerlerin de azaltılmasını sağlar. Veri ayrıklaştırma yöntemi, verinin etiketli veya etiketsiz olmasına göre değişiklik gösterebilir. Etiketsiz verilerde Eşit Aralıklı Gruplama veya Eşit Frekanslı Gruplama ile veri ayrıklaştırması yapılabilir. Etiketli veri için ise histogram analizi, korelasyon analizi, entropi yaygın kullanılan yöntemlerdendir [55].

2.1.5 Veri Dönüştürme

Veri kümesi üzerinde normalizasyon, birleştirme (aggregation), genelleştirme ve öznitelik oluşturma işlemlerini kapsar.

Normalizasyon, veri madenciliğinde, model oluşturmadan önce uygulanan temel işlemlerden biridir. Değişkenlerin ortalama ve varyans değerleri arasındaki fark fazlaysa, büyük ortalama ve yaryansa sahip değişkenler diğerlerinden baskın çıkacağı için onların etkisini azaltır. Bu farkı ortadan kaldırmak için veri kümeleri normalize edilerek, tüm değişkenler belirli bir aralığa indirgenir. Literatürde, Minimum-Maksimum Normalizasyonu, Ondalık Ölçekleme Normalizasyonu, Z-Değeri Normalizasyonu, Sigmoid Normalizasyonu gibi bir çok yöntem mevcuttur.

Birleştirme verilerin özetlenmesi işlemidir. Genelleştirme; kavram hiyerarşisi oluşturma işlemidir. Öznitelik oluşturma ise, var olan öznitelikleri kullanarak yeni

öznitelikler oluşturma işlemidir [55].

2.2 Mikrodizilerde Normalizasyon Yöntemleri

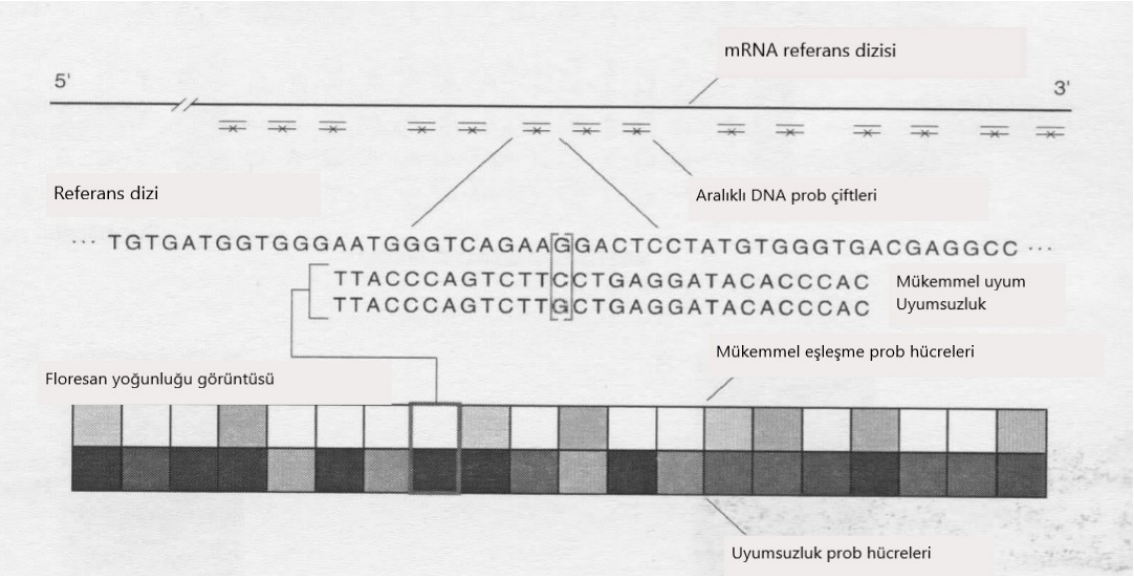
Tez kapsamında geliştirilen ikili Yapay Arı Kolonisi Algoritması ile mikrodizi veri kümelerinde gen seçimi işlemi yapılmıştır. Gen seçimi işleminden önce mikrodizi verilerinde normalizasyon işlemi yapılmıştır.

Birçok hastalık, Deoksiribo Nükleik Asit'de (DNA) belirli bir gendeki mutasyondan kaynaklanır. Ancak, DNA'lar büyük diziler olduğu için mutasyonun nerede ve ne şekilde olduğunu tespit etmek zordur. DNA mikrodizileri, genlerde mutasyon olup olmadığını belirlemek için kullanılan bir teknolojidir. DNA çipleri olarakta adlandırılan DNA mikrodizileri, birçok dizi için bir numuneyi aynı anda kontrol edebilmeyi sağlar. Cam, plastik veya silikon çip gibi katı bir yüzeye tutturulan mikroskopik DNA spotlarından oluşur. DNA'daki bilgilerin protein gibi fonksiyonel bir ürüne dönüştürülmesine gen ekspresyonu veya gen ifadesi denir. Gen ifadeleri, hücrelerin değişen çevrelerine yanıt vermelerine olanak tanır. Mikrodiziler aynı anda çok sayıda genin expresyon seviyesini ölçmeyi sağlar [56].

DNA mikrodizi teknolojisinin gelişmesi ve insan genomunun büyük oranda çözümlenmesi ile birlikte, mikrodizi teknolojisinin kullanım alanı da artmaktadır. Mikrodizilerin yaygın kullanıldığı alanlar [57]:

- Dokularda veya organizmalarda genetik tanımlanması veya genotipleme,
- Hücresel durumlar ve süreçlerin araştırılması,
- Genetik veya bulaşıcı hastalıkların tanısı,
- Belirli bir hastalığın teşhisi,
- İlaç seçimi,
- İlaç tasarımı için hedef seçimi,
- Patojen direnci,
- Protein ifadelerinde zamansal değişimlerin ölçülmesi.

Temelde iki tane mikrodizi teknolojisi bulunmaktadır; cDNA mikrodizileri ve oligonükleotid mikrodizileri. Çalışma kapsamında oligonükleotid mikrodizileri kullanılmıştır. 'Affymetrix' de denir. Oligonükleotid, adli tıp gibi genetik



Şekil 2.1 Mükemmel uyum ve uyumsuzluk probları [58]

araştırmalarda kullanılan kısa DNA veya RNA molekülleridir. Tez kapsamında kullanılan DNA mikrodiziler, Affymetrix teknolojisi ile elde edilmiştir.

Her bir DNA spotu belirli bir DNA dizisini içerir ve bunlara 'prob' denir ve oligonükleotid dizilerinde sekanslar prob kümeleri ile ifade edilir. Bir prob kümesinde 16 prob çifti bulunur ve bir prob, 20-100 ya da daha fazla nükleotid uzunluğunda olabilir. Prob çiftlerinde 1 mükemmel uyum (perfect match- PM) probu ve 1 uyumsuzluk (mismatch- MM) probu vardır. Mükemmel uyum, belirli bir gen ile tam bir eşleşme sağlar; bu nedenle, bu genin ekspresyonunu ölçer. Uyumsuzluk probları düşük seviyelerde eksprese edilen genler için doğru transkript seviyelerini bulur [59]. Mükemmel uyum ve uyumsuzluk probları Şekil 2.1'de gösterilmiştir. Bir genin ekspresyonu ise PM ve MM sinyallerinin ortalama farkı ile hesaplanır.

Tez kapsamında normalizasyon işlemi mikrodizi veri kümelerine uygulandığı için, mikrodizilere yönelik normalizasyon yöntemlerinden; Mikrodizi Paketi 5.0 (MAS5), Güçlü Çoklu Dizi Ortalama (RMA) ve Guanin Sitosin RMA (GcRMA) kullanılmıştır.

DNA mikrodizi verilerinde direk karşılaştırma yapılmasını zorlaştıran arka plan sinyalleri gibi değişkenler bulunmaktadır. Normalizasyon bu değişkenleri minimize ederek, karsılastırma yapabilmek için ortak bir baz oluşturmayı sağlayan bir işlemdir.

Mikrodizi verileri için normalizasyon, ölçülen sinyal yoğunluğu seviyelerinde biyolojik olmayan verilerin çıkarılmasını veya en aza indirilmesini amaçlar. Bu şekilde, gen ekspresyonundaki biyolojik farklılıklar uygun şekilde tespit edilebilir. Mikrodizi normalizasyon tekniklerinin üç adımı vardır: arkaplan düzeltme (background correction), normalizasyon ve özetleme.

2.2.1 MAS5 Normalizasyonu

Her prob bağımsız bir şekilde ve sırayla normalize edilir. Yani prob seviyesinde normalizasyon yapar. Prob kümesinin yoğunluk değeri, MAS5'teki bir referans dizisine dayalı doğrusal ölçeklendirme kullanılarak normalleştirilir. Algoritma, PM ve MM problarının ikisinden de faydalanır. PM ve MM arasındaki farklara bağlı normalizasyon yapar. Algoritma her bir çipi bölmelere ayırır. Her bir bölmedeki en düşük yoğunluk değerlerini seçerek bunu o bölmenin arka plan değeri kabul eder. Bölmelerin merkezlerine olan uzaklığını ölçerek bu uzaklık ile ters orantılı olacak şekilde bir arka plan sinyali çıkarır. Bu işlemi PM ve MM problarının hepsi için gerçekleştirir [60].

2.2.2 RMA Normalizasyonu

Güçlü çoklu dizi ortalama, arka plan düzeltme, normalizasyon ve özetleme işlemlerini, uyumsuzluk problarını kullanmadan, prob seviyesinde yapan bir mikrodizi normalizasyon yöntemidir. Yoğunlukları mükemmel uyum problarından her zaman daha yüksek olduğu için ve yüksek varyansa neden oldukları için uyumsuzluk problarını kullanmazlar. Yani probların yarısı göz ardı edilmiş olur. Uyumsuzluk problarını yok saymak, doğruluğu azaltırken hassaslığı artırır. Algoritma, PM problarının normal dağılıma sahip bir arka plan gürültüsüne ve üstel dağılıma sahip bir sinyal bileşeni olduğunu kabul ederek işlem yapar.

RMA, tüm çiplerin aynı dağılıma sahip olduğunu varsayar ve çoklu çip modelini kullanır. Yani çip sayısı ne kadar çoksa o kadar iyidir. Her probun değeri, çoklu dizilerde kuantil normalizasyon kullanılarak normalleştirilir. Kuantil normalizasyon, iki dağılımın aynı istatistiksel özelliklerde olması için kullanılan bir yöntemdir. Mikrodizi verisinde tüm çipleri aynı dağılıma uydurmak için kullanılır.

RMA normalizasyon yaparken tüm veri kümesini göz önüne alır. Yani, tüm veri kümesi yeni bir veri eklendiğinde baştan normalize edilmelidir. MAS5 ise her bir örneği birbirinden bağımsız olarak ölçek normalizasyonu ile normalize eder. Bu nedenle karşılaştırılabilirliği RMA kadar iyi olmayabilir [61].

2.2.3 GcRMA Normalizasyonu

Guanin Sitosin RMA ve RMA arasındaki fark, RMA'nın arka plan düzeltmesi için evrişimli bir model kullanmasıdır, ancak GcRMA, probların Guanine Sitosin (GC) içeriğini kullanır ve prob sırasına göre yapılır. Bu şekilde, MM probu seviyelerindeki varyansı azaltmak amaçlanmıştır. Pozisyona bağlı baz etkilerine göre, prob afinitesi hesaplanır. MM'ler prob afinitesine göre ayarlandıktan sonra, MM değerlerinin

kaybolmaması için PM'den çıkarılırlar. GcRMA, prob seviyesi bilgisini tutmaz. Normalizasyon ve özetleme kısımları ise RMA'de olduğu gibidir [62].

Tez kapsamında, MAS5, RMA ve GcRMA normalizasyon yöntemlerini uygulamak için R'da 'Bioconducter' deposu (repository) kullanılmıştır.

3
Öznitelik Seçimi

Makine öğrenmesi ve veri madenciliğinde öznitelik seçimi, tüm özniteliklerin bulunduğu veri kümesinden, model oluşturmak için ayırt edici özniteliklerden oluşan bir alt kümenin seçilme işlemidir. Böylece daha az öznitelik ile zaman karmaşıklığı azaltılarak, daha iyi veya en azından eş değer sınıflandırma/ kümeleme başarısı elde etmek amaçlanır.

Öznitelik seçim yöntemleri üç gruba ayrılır: filtreler, sarmalayıcılar ve hibrid yöntemler.

3.1 Filtreler

Filtreleme yöntemlerinde, öznitelikler belirli bir kritere göre sıralanarak bunların içerisinden en iyi n tanesi seçilir. Filtreleme yöntemleri öznitelikleri tek başlarına değerlendirir, birbirleri ile olan olası ilişkisini göz önüne almaz. Hesapsal maliyetleri düşük olmasına karşın, birçok uygulama için performansları iyi değildir.

Kikare istatistiği, t istatistiği, bilgi kazancı, minimum artıklık ve maksimum alaka, reliefF, ortak bilgi, korelasyon skorlama gibi pek çok filtreleme yöntemi mevcuttur. Tez kapsamında, bilgi kazancı, korelasyon katsayısı tabanlı filtreleme ve reliefF yöntemleri kullanılmıştır.

3.1.1 Bilgi Kazancı

Bilgi Kazancı (Information Gain-IG), entropi tabanlı bir öznitelik sıralama yöntemidir ve 'karşılıklı bilgi' (mutual information) olarak da adlandırılır. Entopi, bir örnek kümesinin saflığını temsil eder. Bilgi kazancı ise bir sınıf için belirli bir özniteliğin ayırt ediciliğini ölçer ve 0-1 arasında değer alır. İlgili öznitelik, sınıfları ayırt etmede ne kadar iyi ise bilgi kazancı değeri de o kadar 1'e yakın olur [63]. Bilgi kazancı, Y özelliğini (Denklem (3.1)) tanımlamak için kullanılan bilgi ile X'i (Denklem (3.2)) gözlemledikten sonra Y özelliğini tanımlamak için kullanılan bilgi arasındaki farktır

ve Denklem (3.3)'deki gibi hesaplanır. Tez kapsamında, Bilgi Kazancı yöntemi, Matlab r2017a içerisinden Öznitelik Seçimi kütüphanesi kullanılarak uygulanmıştır.

$$H(Y) = -\sum_{y \in Y} p(y) log(p(y)) \tag{3.1}$$

$$H(Y|X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) log(p(y|x)) \tag{3.2}$$

$$IG = H(Y) - H(Y|X) \tag{3.3}$$

p (y), Y değişkeni için marjinal olasılık yoğunluk fonksiyonu ve p (y|x), x biliniyorken y'nin koşullu olasılığıdır.

3.1.2 Korelasyon Katsayısı Tabanlı Filtreleme (KKTF)

Korelasyon Katsayısı Tabanlı Filtreleme yöntemi, iyi bir alt kümede bulunan özniteliklerin, sınıf bilgisi ile korelasyonu yüksek ancak birbirleri ile korelasyonunun düşük olması gerektiği hipotezine dayalı olarak çalışır. KKTF'de, alt kümelerin bilgi değerlerini ölçen bir fonksiyon ile bir arama algoritması kullanılır. 'Korelasyon' özniteliklerin benzerliklerinin ölçülmesi anlamına gelmektedir ve [-1,1] aralığında değer alır. Korelasyon, -1 ise öznitelikler arasında tam bir negatif doğrusal ilişki vardır. Korelasyon 1 ise, öznitelikler arasında tam bir pozitif doğrusal ilişki vardır. Değerin 0 olması öznitelikler arasında lineer bir ilişki olmadığı anlamına gelmektedir. [64]. Korelasyon Denklem (3.4)'deki gibi hesaplanır. Tez kapsamında KKTF, Matlab r2017a içerisindeki Öznitelik Seçimi kütüphanesi kullanılarak uygulanmıştır.

$$M_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{3.4}$$

k, öznitelik alt küme sayısıdır. $\overline{r_{cf}}$ ortalama öznitelik-sınıf korelasyonu ve $\overline{r_{ff}}$ ortalama öznitelik-öznitelik korelasyonudur.

3.1.3 ReliefF Filtreleme

ReliefF istatistik temelli bir filtreleme yöntemidir. Bir örneğin, veri kümesinde aynı sınıfta olan örneklere olan yakınlığını ve farklı sınıflardaki örneklere olan uzaklığını temel alarak bir model oluşturur. Yani yöntemin temel amacı sınıf içi varyansı minimize etmek ve sınıflar arası varyansı maksimize etmektir. Algoritma ilklendirildiğinde, bütün özniteliklerin ağırlığı 0'dır. İterasyonlar boyunca, seçtiği

rastgele bir örneğin ağırlığını Denklem (3.5)'e göre günceller [65]. Algoritmada, mesafe ölçümü olarak Öklid mesafesi kullanılmış ve tez çalışmasında Matlab r2017a'da makine öğrenmesi ve istatistik aracı ile uygulanmıştır.

$$S_i = \frac{\sum_{j=1}^{m} -diff(A_{ij}, H_{ij}) + diff(A_{ij}, C_{ij})}{m} \tag{3.5}$$

Burada i, ağırlığı hesaplanacak öznitelik ve m, veri kümesindeki örnek sayısıdır. $diff(A_{ij}, H_{ij})$, i. özniteliğin j. örneği ile, aynı sınıftan kendisine en yakın örnek arasındaki mesafedir. $diff(A_{ij}, C_{ij})$ ise, i. özniteliğin j. örneği ile farklı sınıftan en yakın örnek arasındaki mesafeyi temsil etmektedir.

3.1.4 Varyasyon Katsayısı

Varyasyon katsayısı (VK), veri noktalarının ortalamanın etrafındaki bir veri serisindeki dağılımının istatistiksel bir ölçüsüdür. Varyasyon katsayısı, standart sapmanın ortalamaya oranını temsil eder ve araçlar birbirlerinden çok farklı olsalar bile, bir veri serisinden diğerine değişim derecesini karşılaştırmak için kullanılır. Varyasyon katsayısı, popülasyon ortalaması ile ilgili olarak numunedeki verilerin değişkenlik derecesini gösterir ve Denklem (3.6) ile hesaplanır.

$$VK = \frac{\sigma}{\mu} \tag{3.6}$$

Burada σ örneklemin standart sapmasını, μ ise örnekleme ortalamasını temsil etmektedir.

3.2 Sarmalayıcılar

Sarmalayıcı yöntemlerde, filtre metodlarının aksine öznitelikler tek başlarına değil alt gruplar halinde değerlendirilir. Alt gruplar, belirli bir kurala göre veya rastgele olarak oluşturulur. Grubun kalitesini değerlendirmek için bir sınıflandırma veya kümeleme algoritması kullanılır. Filtrelerden daha iyi sonuçlar vermesine karşın, hesapsal maliyetleri daha yüksektir [2]. Tüm olası öznitelik alt kümelerinin, özellikle büyük veri kümelerinde başarısının değerlendirilmesi pek mümkün olmadığından, sezgisel yöntemler ile alt küme seçimi sarmalayıcı yöntem olarak sıklıkla kullanılmaktadır. Tez kapsamında genetik algoritma, ikili parçacık sürü optimizasyonu ve ikili diferansiyel evrim algoritmaları sarmalayıcı yöntem olarak kullanılmıştır.

3.2.1 Genetik Algoritma

Genetik Algoritma, gelecek nesillerin çocuklarını üretmek için iyi bireylerin seçildiği, doğal seleksiyon sürecini modelleyen sezgisel bir yöntemdir. Nesilden nesile, popülasyon arttıkça, kötü çözümler yok olur, iyi çözümler ise daha iyi çözümleri üretebilmek için hayatta kalır [66].

GA, birçok optimizasyon problemine uygun çözümler sunduğu için, yaygın kullanılan bir yöntemdir. Algoritmanın bazı temel kabulleri vardır:

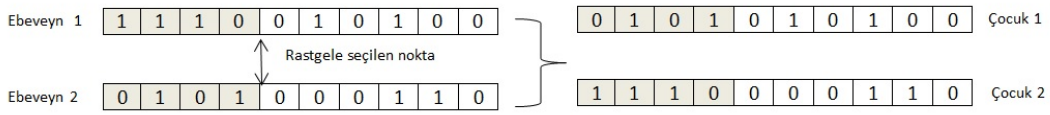
- Optimize edilecek problem vektörler ile temsil edilir ve bu vektörler 'kromozom' veya 'birey' olarak adlandırılır.
- Vektörün boyutu, optimize edilecek problemin parametre sayısı kadardır ve her bir parametre 'gen' olarak adlandırılır.
- Tüm kromozomlar 'popülasyon'u oluşturur ve 'nesil' olarak da adlandırılır.
- İteratif bir algoritmadır. Belirli bir durma kriteri sağlanana kadar işlemler tekrarlanır. Durma kriteri ise, belirli bir sayıda nesil üretilmiş olması veya optimize edilmek istenen problemde belirli bir başarıya ulaşılması olarak belirlenir.

GA'nın temel adımları aşağıdaki gibi sıralanır. Durma kriteri sağlanana kadar, 3-6 arasındaki adımlar tekrarlanır.

1. İlk popülasyon rastgele değerler ile oluşturulur.
2. İlk popülasyonun uygunluk değerleri hesaplanır.
3. Seçilim işlemi uygulanır.
4. Çapraz geçerleme işlemi uygulanır.
5. Mutasyon işlemi uygulanır.
6. Yeni bireylerin uygunluk değerleri hesaplanır.

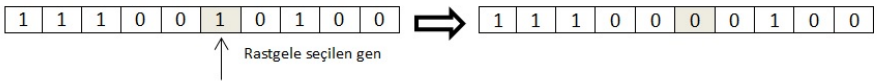
Seçilim: Bir sonraki nesili oluşturmak için çaprazlama ve mutasyon uygulanacak ebeveynlerin seçilmesi işlemidir. Bireyler, uygunluk fonksiyonlarına göre değerlendirilerek seçilir. Böylece iyi bireylerin sonraki nesile aktarılma ihtimalleri daha yüksek olur.

Çaprazlama: Genetik Algoritmanın en önemli fonksiyonlarından biri olup, seçilen her bir ebeveyn çiftine çaprazlama uygulanır. Çaprazlama işlemi Şekil 3.1'de gösterilmiştir. Buna göre ebeveyn kromozomlar rastgele belirlenen bir noktadan iki parçaya bölünür ve bu parçalar karşılıklı olarak yer değiştirilir. Böylece uygunluk değeri yüksek olan ata kromozomlar kullanılarak daha iyi bireylerin elde edilmesi amaçlanır.



Şekil 3.1 Çaprazlama operatörü

Mutasyon: Mutasyon, bir bireyin, rastgele seçilen bir geninin tersinin alınması veya rastgele seçilen iki genin yer değiştirilmesi ile yapılır. Burada amaç, lokal optimuma takılmayı önlemek ve popülasyondaki çeşitliliği artırmaktır. Mutasyon işlemi Şekil 3.2'de verilmiştir.



Şekil 3.2 Mutasyon operatörü

Çaprazlama ve mutasyon işlemleri için, algoritma ilklendirilirken bir oran tanımlanır. Çaprazlama oranı, bireylerin eşleştiklerinde çaprazlama işlemine tabi tutulma olasılığıdır. Mutasyon oranı ise bir bireyin mutasyona uğrama olasılığını verir. Tez kapsamında algoritma Matlab r2017a ile uygulanmıştır.

3.2.2 İkili Parçacık Sürü Optimizasyonu (BinPSO)

Parçacık Sürü Optimizasyonu, grup halinde hareket eden, kuş ya da balık sürüsü gibi grupların doğadaki besin arayışlarından esinlenerek Kennedy ve Eberhart [67] tarafından geliştirilmiş bir sürü zekası algoritmasıdır. Algoritmanın bir takım kabulleri vardır:

- Optimize edilecek problem, vektörler ile temsil edilir ve bu vektörler 'parçacık' olarak adlandırılır.
- Vektörlerin boyutu optimize edilecek problemin parametre sayısı kadardır.
- Bir iterasyondaki parçacıkların hepsi bir popülasyonu temsil eder.
- Bir popülasyonun en iyi parçacığı 'pbest' olarak isimlendirilir.

- O ana kadar, geçmiş tüm iterasyonlar boyunca elde edilen en iyi parçacık ise 'global best (gbest)' olarak isimlendirilir.
- Bir parçacığın bulunduğu konumdan hangi yönde ve ne kadar hareket edeceği 'hız vektörü' ile belirlenir.

PSO iteratif bir algoritma olup, durma kriteri sağlanana kadar 3-6 arasındaki adımlar tekrarlanır.

1. Parçacıklar rastgele konumlar ile ilklendirilir.
2. Parçacıkların uygunluk değerleri hesaplanır. Uygunluk değeri en yüksek parçacık 'pbest' ve 'gbest' olarak atanır. İlk popülasyon için 'gbest' ile 'pbest' aynıdır.
3. Parçacıklar için hız vektörüleri hesaplanır.
4. Hız vektörlerine göre parçacıkların yeni konumları belirlenir.
5. Yeni elde edilen popülasyonun en iyi parçacığı yeni 'pbest' olarak atanır.
6. Mevcut 'gbest'den daha iyi bir çözüm bulunduysa 'gbest' güncellenir.

V_i^k k. iterasyonun i. parçacığı için hız vektörü, $pbest^k$ k. iterasyonun 'pbest' değeri, x_i^k k. iterasyon için mevcut kaynak olmak üzere, i. parçacık için yeni hız vektörü Denklem (3.7)'deki gibi hesaplanır.

$$v_i^{k+1} = V_i^k + \phi(pbest_i^k - x_i^k) + \phi(gbest_i - x_i^k) \tag{3.7}$$

Algoritma sürekli uzayda çözüm üretmek üzere önerilmiş ancak, sonrasında ikili verisyonu geliştirilmiştir [68]. Buna göre parçacıklar ikili uzayda ilklendirilir. Hız vektörü, sürekli PSO için parçacığın hangi yönde ne kadar ilerleyeceğini belirlerken, ikili PSO için bir bitin değişme olasılığını verecektir. Elde edilen değer Sigmoid fonksiyonundan geçirilerek ilgili bitin [0-1] aralığında rastgele bir değerden büyük veya küçük olmasına göre 1 veya 0 değerini alması sağlanır. x parçacığı için i. parametrenin değeri Denklem (3.8) ile belirlenir.

$$x_i^{k+1} = \begin{cases} 1 & \text{eğer } rand(0,1) < S(v_i^{k+1}) \\ 0 & \text{eğer } rand(0,1) \geq threshold \end{cases} \tag{3.8}$$

3.2.3 İkili Diferansiyel Evrim Algoritması

Diferansiyel Evrim (DE) algoritması, popülasyon tabanlı evrimsel bir algoritmadır ve sürekli uzayda çözüm üretmek için geliştirilmiştir [69]. Algoritma, genetik operatörlerdeki gibi seçilim, mutasyon ve çaprazlama işlemlerini içerir. Ancak, farklı olarak popülasyondan rastgele seçilen üç birey kullanılarak yeni bir birey üretilir.

Engelbrecht ve Pampara [70] algoritmayı ikili uzaya taşımak için BinPSO'da olduğu gibi Sigmoid fonksiyonu ile bitlerin 0 veya 1 olma olasılıklarını belirlemiştir. Algoritma dört adımdan oluşur ve tez çalışmasında Matlab r2017a ile uygulanmıştır.

1. Başlangıç popülasyonu uzayda rastgele oluşturulur.
2. **Mutasyon:** Popülasyondan, mevcut kaynaktan farklı 3 birey seçilir ve t. nesilde j.gen için mutant birey (V_i) Denklem (3.9) ile oluşturulur. x_{i1}, x_{i2}, x_{i3} popülasyondan rastgele seçilmiş bireyler iken F'de mutasyon faktörüdür. Seçilen üç bireyden ikisinin farkının hangi oranda üçüncü bireyin genlerine etki edeceğini belirler.
$$V_{ij}^t = x_{i_3 j}^t + F(x_{i_1 j}^t - x_{i_2}^t) \tag{3.9}$$
3. **Çaprazlama:** Aday birey (U_i) oluşturulur. $f(V_{ij}^t)$, V_{ij}^t değerinin Sigmoid fonksiyonundan geçirilmesi ile elde edilir.
$$U_{ij}^t = \begin{cases} 0 & \text{eğer } rand(0,1) < f(V_{ij}^t) \\ 1 & \text{aksi halde} \end{cases} \tag{3.10}$$
4. **Seçilim:** U_i^t ve x_i^t'den iyi olanı seçilir.

4
Yapay Arı Kolonisi Algoritması

Bu bölümde sürekli uzay problemleri için geliştirilen orjinal Yapay Arı Kolonisi algortiması ile literatürde algoritmayı ikili uzaya taşımak için önerilen yöntemlerden bahsedilmiştir.

4.1 Sürekli Uzayda Yapay Arı Kolonisi Algoritması

2005 yılında Karaboğa'nın [11], bal arılarının yiyecek arama davranışlarından esinlenerek, çok boyutlu optimizasyon problemlerini çözmek için geliştirdiği bir sürü zekası algoritmasıdır. Algoritmanın avantajı, görevli ve gözcü arı fazıyla güçlü bir yakınsama sağlaması, kâşif arı fazıyla da lokal optimumları atlayabilmesi ve çözüm çeşitliliği sağlayabilmesidir. Algoritmada temelde 2 çeşit arı vardır;

1. Görevli arılar (İşçi Arılar); Kaynağı optimize etmeye çalışır
2. Görevsiz arılar
    - Gözcü arı; Görevli arıların getirdiği bilgilere göre o anki en optimize kaynağa gider.
    - Kâşif arı; Rastgele kaynak arar.

Algoritmanın bir takım temel kabulleri vardır;

1. Her bir kaynak optimizasyon probleminin olası bir çözümü, kaynaktaki nektar miktarı ise çözümün kalitesidir.
2. Her bir yiyecek kaynağı (olası bir çözüm) sadece bir görevli arı tarafından alınır. Yani görevli arı sayısı yiyecek kaynağı sayısına ve gözcü arı sayısına eşittir.
3. Her bir kaynak geliştirilememe katsayısına sahiptir. Bu değer 0 olarak ilklendirilir.

4. Gözcü ve görevli arılar keşfedilen kaynaklardan faydalanma (exploitation) işleminde, kâşif arılar ise keşif (exploration) sürecinde görev alır.

Yapay Arı Kolonisi (YAK) Algoritmasının akış diyagramı Şekil 4.1'de verilmiştir. Algoritma 4 temel adımdan oluşur ve durma kriteri sağlanana kadar 2-4 arası adımlar tekrarlanır.

1. **İlklendirme Aşaması:** Her bir yiyecek kaynağı $x_i = x_{i1}, x_{i2}, \ldots, x_{iD}$ olmak üzere, başlangıç çözümleri Denklem 4.1 ile rastgele belirlenir.

$$x_{ij} = x_j^{min} + U(0,1)(x_j^{max} - x_j^{min}) \tag{4.1}$$

   i=1,2,...,SN; SN=Yiyecek kaynağı sayısı,

   j=1,2,...,D; D= Arama uzayının boyutu,

   U(0,1)=rastgele üretilen uniform dağılıma sahip, [0-1] aralığında değer,

   x_j^{min} ve x_j^{max}, j parametresinin önceden tanımlanmış minimum ve maksimum değerleridir.

2. **İşçi Arı Aşaması:** Her bir görevli arı bir yiyecek kaynağına yönelir, Denklem 4.2 ile mevcut kaynağın komşuluğunda başka bir kaynak bulur ve bu iki kaynağın uygunluk değeri ($fitness_i$) hesaplanır. Komşu kaynağın uygunluk değeri daha iyi ise, işçi arı mevcut kaynağı unutur ve komşu kaynağı hafızasına alır ve kaynağın geliştirilememe katsayısı 0'lanır. Aksi halde, mevcut kaynağın geliştirilememe katsayısı 1 artırılır.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{4.2}$$

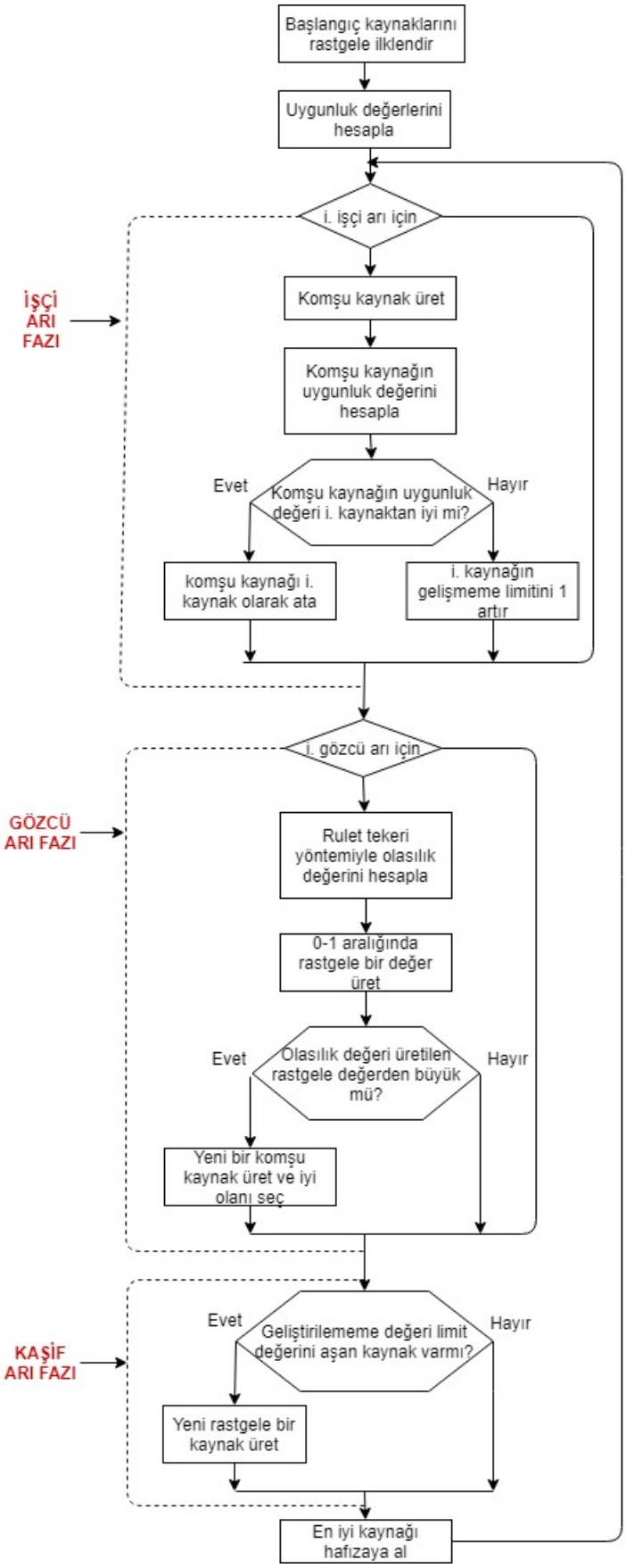
   i; mevcut yiyecek kaynağı,

   k; i^{th} kaynağın komşuluğunda rastgele seçilen bir yiyecek kaynağı.

   j; x_i ve x_k nın modifiye edilebilmesi için rastgele seçilen parametre.

   ϕ_{ij} =[-1,1] aralığında rastgele üretilen uniform dağılımlı bir değer.

3. **Gözcü Arı Aşaması:** İşçi arılar bir çevrimde araştırmalarını tamamladıktan sonra elde ettikleri o anki en optimize kaynaklara yönelik bilgileri gözcü arılara aktarır. Gözcü arılar ise Denklem (4.3)'deki rulet tekerleği yöntemi ile ilgili kaynağın seleksiyonu için bir olasılık değeri hesaplar. Bu değer [0,1] aralığında rastgele üretilen bir değerden küçükse, Denklem (4.2) kullanılarak yeni bir komşu kaynak üretilir. Mevcut kaynak ile komşu kaynak arasında aç gözlü seçim



Şekil 4.1 Yapay arı kolonisi akış diyagramı

yapılır ve seçilen kaynağın yeni işçi arısı olur.

$$p_i = \frac{fitness_i}{\sum_{j=1}^{n} fitness_j} \tag{4.3}$$

$fitness_i$; x_i'nin uygunluk değeri

4. **Kâşif Arı Aşaması:** Bir kaynağın geliştirilememe katsayısı önceden belirlenen bir eşik değerine ulaştıysa, o kaynağın bulunduğu lokal alanda en iyi çözümün, ilgili çözüm olduğu fikrinden yola çıkılarak, kâşif arı Denklem (4.1) ile yeniden rastgele bir kaynak üreterek algoritmanın lokal optimuma takılmasını önler. Her iterasyonda sadece 1 kâşif arı olabilir.

4.2 İkili Yapay Arı Kolonisi

YAK algoritmasını ikili uzaya taşımak için literatürde temelde iki yaklaşım bulunmaktadır.

1. Sürekli vektörlerde aritmetik işlemler ile yeni kaynak üretmeyi sağlayan Denklem (4.1) ve Denklem (4.2) revize edilerek ikili vektörler ile işlem yapılır.
2. Sonuçlar Denklem (4.1) ve Denklem (4.2) gibi sürekli uzayda üretilir ancak sonrasında bir dönüşüm işleminden geçirilerek ikili uzaya taşınır.

Ayrık Yapay Arı Kolonisi (DisABC) [12], ikili YAK algoritmasının ilk geliştirilen yöntemlerindendir. Kaynaklar ikili uzayda ilklendirilir. Yeni kaynak (V_i) üretmek için mevcut kaynak (x_i) ile popülasyondan rastgele seçilen kaynağın (x_k) farklılığı Denklem (4.4) ile Jackard benzerlik ölçümü kullanılarak hesaplanır.

$$\text{Farklılık}(V_i, x_i) = \phi * Benzerlik(x_i, x_k) \tag{4.4}$$

İki vektörün Jackard benzerliği ise Denklem (4.5) ile hesaplanır.

$$Benzerlik(x_i, x_k) = \frac{M_{11}}{M_{10} + M_{01} + M11} \tag{4.5}$$

M_{11}, değeri V_i ve x_i'de karşılıklı 1 olan bitlerin sayısı,

M_{10}, değeri V_i'de 1 ve x_i'de 0 olan bitlerin sayısı,

M_{01}, değeri V_i'de 0 ve x_i'de 1 olan bitlerin sayısı

ϕ değeri ise Denklem (4.6) ile belirlenir.

$$\phi = \phi_{max} - \frac{\phi_{max} - \phi_{min}}{MCN} \tag{4.6}$$

ϕ_{max} ve ϕ_{min}, ϕ vektörünün alt ve üst sınırlarıdır. MCN ise maksimum iterasyon sayısını temsil etmektedir.

DisABC'de temel mantık, Farklılık(V_i, x_i) mümkün olduğu kadar $\phi * Farklilik(x_i, x_k)$ ifadesine yakın olmasını sağlamaktır.

$$min\{1 - \frac{M_{11}}{M_{11} + M_{10} + M_{01}} - \phi * Benzerlik(x_i, x_k)\} \tag{4.7}$$

$$M_{11} + M_{01} = m1 \tag{4.8}$$

$$M_{10} \leq m_0 \tag{4.9}$$

$$M11, M10, M01 \geq 0 \tag{4.10}$$

m_1 ve m_0 sırasıyla x_i'deki toplam 1 ve 0 sayısı olmak üzere Denklem (4.8), (4.9) ve (4.10)'daki şartları sağlayan optimal M_{11}, $M10$ ve M_{01} değerleri belirlenir. Ardından V_i vektörü bir 0 vektörü olarak ilklendirilerek,

- x_i'den rastgele seçilen M_{11} tane bitin değeri V_i'de aynı konumdaki bitlere atanır.
- x_i'de değeri 0 olan bitlerden rastgele seçilen M_{10} tane bitin V_i'deki değeri 1 yapılır.

Böylece sonuçta V_i'de toplam $M_{11} + M_{10}$ tane bitin değeri 1 yapılarak aday kaynak V_i üretilmiş olur.

Öztürk vd. [13] DisABC'yi modifiye ederek Geliştirilmiş DisABC (IDisABC) yöntemini önermişlerdir. Yazarlar, DisABC ile 3 tane aday kaynak (V_{i1}, V_{i2}, V_{i3}) üretmişler ve ardından mevcut kaynak (x_i), aday kaynaklar ve sürünün en iyi çözümü (x_{global}) arasında tüm çiftler için yer değiştirme (swap) ve iki noktalı çaprazlama (two-point crossover) uygulamışlardır. Elde edilen vektörlerden uygunluk fonksiyonu bakımından en iyisi yeni aday kaynak olarak belirlenmiştir.

Hançer vd. [14] DisABC'yi Diferansiyel Evrim algoritması ile birleştirmiştir. DisABC'de popülasyondan 1 rastgele kaynak seçmek yerine 3 rastgele kaynak seçilir (x_{r1}, x_{r2} ve x_{r3}). Buna göre birinci komşu kaynak (x_{r1}) ve mutant kaynağın (Ω) farklılığını için gerekli optimal M değerleri Denklem (4.11) ile DisABC'den yararlanılarak bulunur.

$$\text{Farklılık}(\Omega, x_{r1}) = 1 - Benzerlik(x_{r2}, x_{r3}) \tag{4.11}$$

DisABC'de olduğu gibi mutant kaynak (Ω), 0 vektör olarak ilklendirilerek, DisABC'de V_i'nin elde edildiği şekilde, M değerlerine göre $M_{11}+M_{10}$ tane biti 1 yapılır. Ardından, x_i ve Ω'ya Denklem (4.12)'deki gibi rekombinasyon uygulanarak, aday kaynak (V) vektörü elde edilir. CR, çaprazlama oranını temsil etmektedir.

$$V_{ij} = \begin{cases} \Omega_{ij} & \text{eğer } rand(0,1) \leq \text{CR} \\ x_{ij} & \text{aksi halde} \end{cases} \tag{4.12}$$

Singhal vd. [15] çözüm uzayında DisABC'yi kullanarak yeni kaynak üretmiş ancak kaşif arı fazında, yazarlar, limit değeri aşılmış bir kaynak olduğunda, rastgele bir kaynak üretmek yerine, popülasyondaki en iyi kaynağı (gBest) ilgili kaynağa atamışlardır.

Cui ve Gu [16] Diferansiyel Evrim algoritmasındaki mutasyon, çaprazlama ve seçim operatörleri işçi arı aşamasında kullanılmıştır. Buna göre;

1. Mutasyon oranı (MR) ve ekleme sayısı (IT) parametreleri belirlenir. Bu parametreler sırasıyla [0,2-1] ve [3-5] aralığında olacak şekilde belirlenmiştir.

2. Mutant birey (Ω), popülasyonun en iyi kaynağı (x_{best}) ve popülasyondan rastgele seçilen bir kaynak (x_k) ile Denklem (4.13) ile üretilir. Rastgele üretilen bir değer MR değerinden küçükse, mutant bireyin IT tane biti x_{best}'den aksi halde x_k'dan alınır.

$$\Omega_i = \begin{cases} x_{best}, ITdefa & \text{eğer } rand(0,1) < \text{MR} \\ x_k, ITdefa & \text{aksi halde} \end{cases} \tag{4.13}$$

3. Çaprazlama aşamasında, rastgele üretilen bir sayı, önceden belirlenmiş çaprazlama oranı (CR) değerinden küçükse mutant birey ile mevcut kaynak (x_i) arasında çaprazlama işlemi uygulanarak 2 yeni birey üretilir. Aksi halde, bir sonraki adıma sadece mutant birey geçer. CR parametresi [0,2-1] aralığında olacak şekilde belirlenmiştir.

4. Seçim aşamasında, mutant birey ve çaprazlama aşamasında üretilen bireylerden iyi olanı aday kaynak olarak belirlenir.

Ozturk vd. [17], genetik operatörleri kullanarak işçi arı aşamasında yeni kaynak

üretmişler ve önerdikleri yöntemi Genetik İkili Yapay Arı Kolonisi (GB-ABC) olarak isimlendirmişlerdir. Buna göre aday kaynak üretmek için aşağıdaki adımlar uygulanır.

1. kaynaklar Denklem (4.14) ile rastgele ilklendirilir. Buna göre her bitin 1 veya 0 olma olasılığı %50'dir.

$$x_{ij} = \begin{cases} 0 & \text{eğer } rand(0,1) \leq 0{,}5 \\ 1 & \text{aksi halde} \end{cases} \tag{4.14}$$

2. Popülasyondan rastgele iki kaynak seçilir (x_k, x_j).

3. D, kaynaklardaki boyut sayısı olmak üzere, tamamı 0'lardan oluşan sıfır vektörü üretilir.

4. x_i mevcut kaynak ve x_{best} poülasyondaki en iyi kaynak olmak üzere x_i, x_k, x_j, x_{best} ve sıfır vektörleri ebeveyn olarak belirlenir birbirleri ile rastgele olarak birer defa eşleştirilir. Bu işlem sonucunda 5 ebeveyn çifti oluşacaktır.

5. Eşleşen vektör çiftlerine iki noktalı çaprazlama işlemi uygulanarak çocuklar üretilir. Çaprazlamanın uygulanacağı noktalar da rastgele belirlenir. Bu işlem sonucunda her ebeveyn çiftinden 2'şer tane olmak üzere 10 tane çocuk üretilmiş olur.

6. Her bir çocuk için rastgele iki bit seçilerek yer değiştirme (swap) operatörü ile 10 tane torun (grandchild) üretilir.

7. 10 çocuk ve 10 torun dan oluşan popülasyondan uygunluk değeri en yüksek olanı aday kaynak (V_i) olarak atanır.

Yurtkuran ve Erdal [18] işçi ve gözcü arı aşamalarında aday kaynak üretmek için 4 yöntem tanımlamışlardır. Ebeveyn 1, mevcut kaynak (x_i), ebeveyn 2 popülasyondan rastgele seçilen kaynak (x_k), ebeveyn tüm iterasyonlar boyunca elde edilmiş en iyi kaynak (Gbest) veya ilgili iterasyonun en iyi kaynağı (Local best-Lbest) olmak üzere bu yöntemler şöyle sıralanır.

1. Pozisyon tabanlı çaprazlama: Ebeveyn 1'den rastgele konumlarda seçilen bitler alınır, kalanlar ise ebeveyn 2 'den alınarak aday kaynak üretilir.

2. Ebeveyn 1 ve 2 arasında tek noktalı çaprazlama yapılarak aday kaynak üretilir.

3. Ebeveyn 1, 2 ve 3'den sırasıyla birer bit alarak aday kaynak oluşturulur.

4. Ebeveyn 1, 2 ve 3 için toplamları 1 olacak şekilde birer olasılık değeri tanımlanır (p_1, p_2 ve p_3). Aday kaynağın bitleri p_1 olasılıkla ebeveyn 1'den, p_2 olasılıkla ebeveyn 2'den ve p_3 olasılıkla ebeveyn 3'den alınarak kaynak elde edilir.

Yazarlar, bu 4 töntemden birini rastgele seçerek çözüm uzayında aday kaynak üretmişlerdir.

Jia vd. [19], kaynakları ikili uzayda ilklendirmek için, her bir bitin %50 olasılıkla 1 veya 0 olmasını sağlayan Denklem (4.14)'ü kullanmışlardır. İşçi arı aşamasında ise Denklem (4.2)'de kullanılan aritmetik işlemler yerine bit işlem operatörlerini kullanarak yeni kaynağı (V), Denklem (4.15) ile üretmişlerdir.

$$V_i = x_i XOR(\phi AND(x_i OR x_k)) \tag{4.15}$$

x_i; mevcut kaynak, x_k popülasyondan rastgele seçilmiş bir komşu kaynaktır. ϕ vektörü ise ilklendirme aşamasında kullanılan yöntem ile Denklem (4.14) kullanılarak oluşturulmuştur.

Kıran ve Gündüz'de [20] bit işlem operatörü tabanlı bir ikili YAK algoritması önermiştir. Buna göre yazarlar, ilk kaynakları Denklem (4.14) ile tanımlarken, işçi arı aşamasında yeni kaynak üretmek için bit işlem tabanlı 'XOR' operatörünü Denklem (4.16)'deki şekliyle kullanmışlardır.

$$V_{ij} = x_{ij} XOR(\phi(x_{ij} XOR x_{kj})) \tag{4.16}$$

Denklem (4.16) de yazarların kullandığı ϕ sembolü, %50 olasılıkla işlem yapan lojik 'değil' işlemidir. Buna göre ($x_{ij} XOR x_{kj}$) ile elde edilen sonucun, %50 olasılıkla tersi alınır. (1 ise 0, 0 ise 1 yapılır)

Ye vd. [25] kaynaklar arasındaki Hamming mesafesini ölçerek mevcut kaynağın hızını belirlemişlerdir. Mevcut kaynağın hızı Denklem (4.17) ile belirlenir.

$$hiz = \phi_1 diff(x_i, x_k) + \phi_2 diff(x_g, x_i) \tag{4.17}$$

Burada x_k popülasyondan rastgele seçilmiş komşu kaynağı ve x_g o ana kadar ki tüm iterasyonlar boyunca elde edilmiş en iyi kaynağı (global best) temsil etmektedir. 'diff' işlemi ise verilen kaynaklar arasındaki Hamming mesafesidir. ϕ_1 [-1,1] aralığında, ϕ_2 ise [0,1] aralığında rastgele üretilmiş değerlerdir.

V aday vektörünü elde etmek için mevcut kaynağın bitleri sırasıyla x_g ile karşılaştırılmıştır.

- Eğer x_{ij}=x_{gj} ise x_i'den rastgele seçilen bir bite mutasyon uygulanır.
- eğer $x_{ij} \neq x_{gj}$ ise, m boyut sayısı olmak üzere, D değeri Denklem (4.18) ile hesaplanır.

$$D_i = \begin{cases} 1 & \text{eğer } v_i<0 \\ \frac{m}{3}+1 & \text{eğer } v_i \geq \frac{m}{3} \\ v_i+1 & \text{aksi halde} \end{cases} \tag{4.18}$$

  D değeri belirlendikten sonra, h_i, i. iterasyon için x_i ve x_g arasındaki Hamming mesafesi olmak üzere, aday kaynağı (V) elde etmek için 2 durum vardır:

  - $D_i \leq h_i$ ise x_i ve x_g'de farklı olan bitlerden rastgele seçilen D_i tane bit mutasyona uğratılarak yeni kaynak üretilir.
  - $D_i > h_i$ ise x_i ve x_g'de aynı olan bitlerden rastgele seçilen $D_i - h_i$ tane bit mutasyona uğratılarak yeni kaynak üretilir.

Gözcü arı aşamasında ise, gözcü arı bir x kaynağını seçmeye karar verdiyse, o kaynak için, işçi arı aşamasındaki yöntem ile yeni kaynak üretilir.

Zhang vd. [26] benzer şekilde yeni kaynak (V) üretmek için Hamming mesafe ölçümünden faydalanarak aşağıdaki adımları izlemişlerdir.

1. x_i mevcut kaynak ve x_k popülasyondan rastgele seçilmiş bir kaynak olmak üzere, İki kaynak arasındaki Hamming mesafesi (hm) ölçülür.
2. V_i=x_i olarak ilklendirilir.
3. x_i'de, x_k'dan farklı olan bitlerin konumları belirlenerek bu bitler için, 0 veya 1 değerlerinden biri rastgele üretilir ve V_i de aynı konumdaki bitlere bu değerler atanır.
4. x_i'de, x_k ile aynı olan bitlerin konumları belirlenerek bunlardan hm tanesi rastgele seçilir. V_i'de aynı konuma karşılık gelen bitlere, 0 veya 1 değerlerinden biri rastgele seçilerek atanır.

Yukardaki adımlar izlendikten sonra yeni kaynak (V) mevcut kaynaktan en fazla 2*hm tane bit değiştirilerek üretilmiş olur.

Chen ve Kanoh [27] yeni kaynak üretme aşamasında popülasyondan rastgele kaynak seçmek yerine, mevcut kaynağa yakın olan bir kaynak seçmeyi tercih etmişlerdir. Buna

göre komşu kaynak (x_k) ile mevcut kaynak arasındaki Hamming mesafesi Denklem (4.19) ile ölçülmüş ve benzerlik değeri Denklem (4.20) ile hesaplanmıştır.

$$hd = H(x_i, x_k) \tag{4.19}$$

$$Benzerlik = 1 - (rn * \frac{hd}{n}) \tag{4.20}$$

Burada rn [0,1] aralığında rastgele üretilmiş bir sayı ve n ise kaynak vektördeki boyut sayısıdır.

Algoritmada yeni kaynak üretmek için aşağıdaki adımlar izlenir.

0-1 aralığında rastgele bir sayı üretilir (r).

- V_i=x_i olarak ilklendirilir.
- Eğer r<benzerlik ise komşu kaynaktan y tane rastgele bit seçilir ve V_i de karşılık gelen bitlere, x_k'daki ilgili değerler atanır.

Buraya kadar bahsedilen yöntemlerde, YAK algoritmasını ikilileştirmek için orjinal YAK algoritmasında, Denklem (4.2) ikili vektörler ile işlem yapabilecek şekilde revize edilmiştir. Literatürde bazı yöntemlerde ise, orjinal YAK algoritmasında olduğu gibi kaynaklar sürekli uzayda ilklendirilir. Sürekli değer içeren kaynaklar Denklem (4.2) ile vektörel aritmetik işlemler kullanılarak yeni kaynak üretilir. Ve son olarak aday kaynak (V_i) bir dönüşüm fonksiyonu yardımıyla ikili uzaya taşınır.

Wei ve Chen [28] aday kaynağı (V_i) YAK algoritması ile ürettikten sonra yuvarlama (round) fonksiyonu ile Denklem (4.21)'yi kullanarak ikili uzaya taşınmıştır. Arama uzayının minimum değeri 0 iken maksimum değeri 1 olarak belirlenmiştir.

$$V_{ij} = Yuvarla(V_{ij}) \tag{4.21}$$

Yazarlar önerdikleri yöntemi İkili Yapay Arı Kolonisi (BABC) olarak isimlendirmiştir.

Kıran [29] YAK algoritması ile sürekli uzayda aday kaynak üreterek mod alma operatörü ve yuvarlama operatörünü kullanmış ve Denklem (4.22) ile sonuçlar ikili uzaya taşınmıştır.

$$V_{ij} = Yuvarla((V_i mod 2) mod 2) \tag{4.22}$$

Mandala ve Gupta [30], aday kaynak (V_i), Denklem (4.2) ile belirlenir. Ardından aday kaynağa Denklem (4.23) ile tanjant fonksiyonu uygulanarak f(V_i) elde edilir.

Son olarak ise Denklem (4.24) ile aday kaynak ikili uzaya dönüştürülür.

$$f(V_{ij}) = tanh(|V_{ij}|) = \frac{exp(2 * |V_{ij}| - 1)}{2 * |V_{ij}| + 1)} \tag{4.23}$$

$$V_{ij} = \begin{cases} 1 & \text{eğer } f(v_{ij}) > 0 \\ 0 & \text{aksi halde} \end{cases} \tag{4.24}$$

Pampara vd. [32], açı modülasyon tekniği ile ikili dönüşüm sağlamıştır. Açı modülasyon tekniği kullanan optimizasyon algoritmalarının amacı, sürekli uzayda bir dizisi oluşturan fonksiyonları bularak bu fonksiyonun katsayılarını optimize etmektir. Buna göre, Denklem (4.25) ile ilgili bit için olasılık değeri hesaplanır.

$$g(x_i) = Sin(2\pi(x - a) * b * cos(2\pi * (x - a) * c)) + d \tag{4.25}$$

a=Fonksiyonu üretmek için yatayda gerekli kaydırma (shift) sayısı,

b=Sinüs fonksiyonunun maksimum frekansı,

c=Cosinüs fonksiyonunun frekansı,

d=Fonksiyonun dikey kaydırma sayısı,

x=Optimize edilecek problemin boyutu.

Denklem (4.25) ile elde edilen g(x), aday kaynaktaki bir bitin 1 olma olasılığını verir. Yani g(x)>0 ise ilgili bitin değeri 1 aksi halde 0 olarak belirlenir.

5
Model Oluşturma ve Değerlendirme

Bu bölümde, tez kapsamında veri kümesinden model oluşturmak için kullanılan sınıflandırma ve kümeleme algoritmalarından ve modellerin kalitesini değerlendirmek için kullanılan metriklerden bahsedilmiştir. Tez kapsamında, sınıflandırma algoritmalarının sonuçları için 'doğruluk', kümeleme algoritmalarının sonuçları için ise 'saflık' ve 'silüet kaysayısı' metrikleri kullanılmıştır. Ayrıca elde edilen sonuçların istatistiksel anlamlılığını ölçmek için yararlanılan istatistik testleri de bu bölümde anlatılmıştır.

5.1 Sınıflandırma Algoritmaları

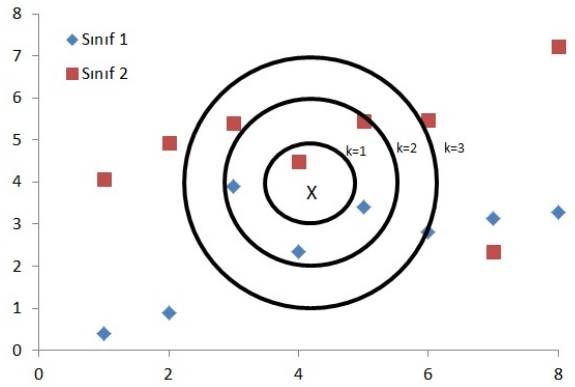
Veri kümesinden sınıf bilgisi bilinen örnekler ile bir model elde etmek için sınıflandırma algoritmaları kullanılır. Elde edilen model ise, sınıf bilgisi bilinmeyen yeni örneklerin sınıflarını tahmin etmek için kullanılır. Sınıflandırma algoitmaları, verilerin çeşitli özelliklerinden dağılımlarından vs. yararlanarak model oluşturur.

Bu tez çalışması kapsamında K En Yakın Komşu, Naive Bayes, Rastgele Orman, Olasılıksal Sinir Ağları, Destek Vektör Makineleri ve Karar Ağacı sınıflandırma algoritmaları kullanılmıştır.

5.1.1 K-En Yakın Komşu Algoritması

K-En Yakın Komşu (K-NN), sınıflandırma ve regresyon için kullanılan, uzaklığa dayalı parametrik olmayan bir algoritmadır. Bir yöntem parametrik değilse, verinin dağılımı hakkında herhangi bir varsayımda bulunamaz demektir. Algoritma bir eğitim modeli oluşturmaz. Bunun yerine, hangi sınıfa ait olduğu bilinmeyen bir örneğin sınıfı, mevcut örnekler içerisinden kendisine en yakın olan k tane örneğin sınıf bilgisine bakarak karar verir. Yakınlık ise Öklid, Mahalonobis, Manhattan, Hamming gibi mesafe ölçüm yöntemleri ile hesaplanır [71]. Tez kapsamında uzaklık hesabında Öklid mesafesi kullanılmıştır. k değeri genellikle 1, 3, 5 gibi tek sayılardan seçilmekle

birlikte, bu değer algoritmanın performansını etkileyecek önemli bir parametredir. Algoritmanın, basit ve gürültülü veriye karşı dirençli olması bir avantaj iken, k değerine bağlı olarak yapılan hesaplamada maliyetinin artmasına neden olduğu için bir dezavantajıdır. Şekil 5.1'de algoritmanın temel mantığı verilmiştir. Buna göre hangi sınıfa ait olduğu bilinmeyen 'X' örneğinin sınıfına, kendisine en yakın k adet örneğin sınıfı göz önüne alınarak karar verilir.



Şekil 5.1 K-en yakın komşu yöntemi modeli

5.1.2 Naive Bayes Algoritması

Naive Bayes (NB), Bayes Teoremine dayalı olasılıksal bir sınıflandırma algoritmasıdır. Algoritma, örneklerin birbirinden bağımsız olduğunu varsayar. Sınıf bilgisi bilinmeyen 'x' örneğinin her bir sınıf için, o sınıfa ait olma olasılığı koşullu olasılık yardımıyla Denklem (5.1) ve (5.2) ile hesaplanır [54].

$$P(c_j|x) = \frac{P(x|c_j)P(c_j)}{P(x)} \tag{5.1}$$

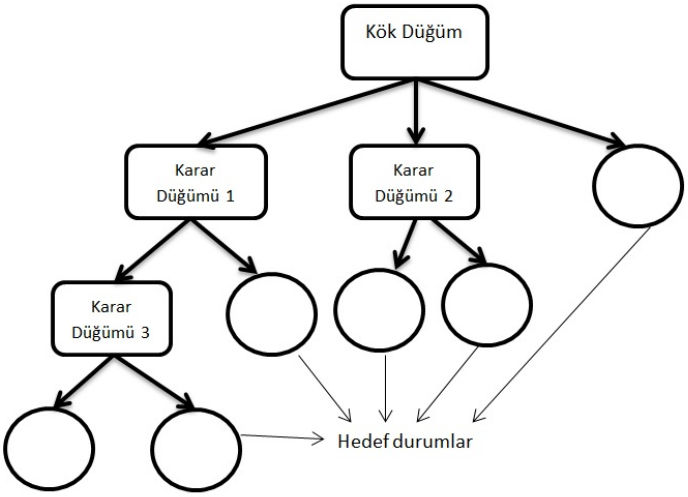
$$P(x|c_j) = \prod_{i=1}^{n} P(x_i|c_j) \tag{5.2}$$

Buna göre, sınıf bilgisi aranan 'x' örneği için:, j={1, 2, .. k} iken k veri kümesindeki sınıf sayısıdır. i={1, 2, ... n} iken n=j. sınıfa ait örnek sayısıdır.

Basit yapısına karşı genellikle iyi sonuçlar veren bir algoritma olduğu için, birçok uygulamada sıklıkla tercih edilen bir sınıflandırma algoritmasıdır.

5.1.3 Karar Ağacı Algoritması

Karar Ağacı (KA) algoritmasında amaç, eğitim verisini kullanarak belli derinliğe sahip kök ve dalların olduğu bir ağaç yapısı oluşturmaktır. Ara düğümler, sınıflandırma yapabilmek için kullanılan kuraları tanımlar ve iki veya daha fazla dal içerirler. Yaprak düğümlerde ise örneğin ait olduğu sınıf bilgileri bulunur. Düğümlere öznitelikler, sınıfları ayırt edebilirlik derecelerine göre yerleştirilir. Kök düğüm ağacın en üst seviyesindeki düğümdür ve verilerin sınıflandırılması içn ayırt ediciliği en yüksek özniteliği içerir [72]. Karar Ağacının genel yapısı Şekil 5.2'de verilmiştir. Karar Ağaçları, kolay yorumlanabilmesi ve hem nümerik hem kategorik veriye uyarlanabilmesi bakımından sıklıkla tercih edilen bir sınıflandırıcıdır.



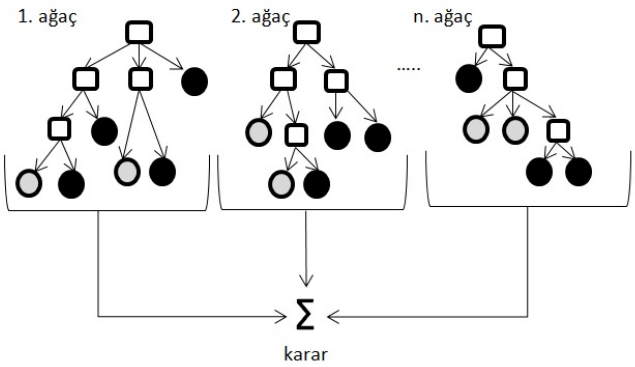
Şekil 5.2 Karar ağacı yöntemi modeli

5.1.4 Rastgele Orman Algoritması

Rastgele Orman (RO) algoritması, birden fazla karar ağacından oluşan bir topluluk öğrenme (ensemble learning) yöntemidir. Veri kümesinden rastgele seçilen öznitelik ve örnekler ile bir karar ağacı oluşturulur. Topluluktaki her bir karar ağacı için rastgele öznitelik seçme işlemi tekrarlanır. Bu şekilde oluşturulan birbirinden farklı ağaçların verdiği kararlar oylama ile birleştirilerek sınıf tahmini yapılır [73]. Algoritma hem sınıflandırma hem de regresyon için kullanılabilmektedir. Algoritmanın genel modeli Şekil 5.3'de gösterilmiştir.

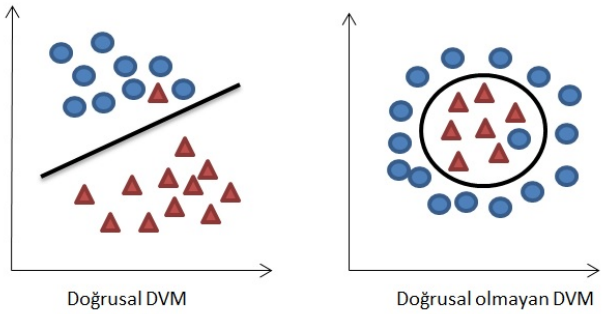
5.1.5 Destek Vektör Makineleri

Destek Vektör Makineleri (DVM), iki sınıfın verilerini birbirinden ayırmak için en uygun hiper düzlemi belirlemeyi hedefler. Hiper düzlemler, mümkün olduğunca sınıfları geniş bir aralıkla ayırmalıdır. Destek Vektör Makineleri, Doğrusal ve Doğrusal



Şekil 5.3 Rastgele orman yöntemi modeli

olmayan Destek Vektör makineleri olmak üzere ikiye ayırılır. Doğrusal DVM'de sınıflar birbirinden karar doğrusu ile ayrılır. Sınıfları ayırmak için birden fazla karar doğrusu kullanılabilir ancak, doğru sayısı arttıkça algoritmanın genelleyebilme kapasitesi azalacaktır. Doğrusal olmayan DVM'de ise, sınıflar birbirinden bir doğru ile ayrılamaz, bunun yerine çekirdek yöntemleri kullanılarak sınıfların ayrışması sağlanır [74]. Şekil 5.4'de doğrusal ve doğrusal olmayan DVM için sınıfların nasıl ayrıştırıldığı gösterilmiştir.



Şekil 5.4 Destek vektör makineleri modeli

5.1.6 Olasılıksal Yapay Sinir Ağları

Bir Olasılıksal Yapay Sinir Ağı (OYSA), ileri beslemeli bir sinir ağıdır. Her bir sınıf için olasılık yoğunluk fonksiyonu, Parzen penceresi ve parametrik olmayan bir fonksiyon ile yaklaşık olarak hesaplanır. Yeni örneğin sınıfı, sınıfların olasılık yoğunluk fonksiyonlarına göre tahmin edilir. Ardından, Bayes kuralı ile yeni veri en yüksek sonraki olasılık (posterior probability) değerine sahip sınıfa atanır. Bu yöntemle verinin yanlış sınıflandırılması olasılığı en aza indirgenmiş olur [75]. Ağ dört katmandan oluşur:

1. Giriş Katmanı

2. Örüntü Katmanı: Giriş verisinin eğitim verisine Öklid uzaklığı hesaplanır ve sigma değerleri kullanılarak Radyal Temel Çekirdek fonksiyonu uygulanır.

3. Toplama Katmanı: Hesaplanan olasılıklar toplanır ve ağın çıkış vektörünün olasılıkları bulunur.

4. Çıkış Katmanı: En yüksek olasılığa sahip sınıfı belirler.

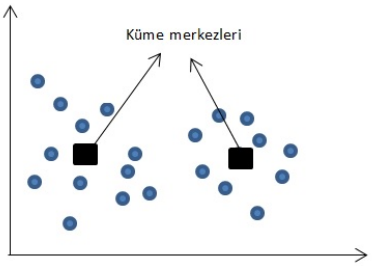
OYSA fazla hafızaya ihtiyaç duymasına karşın, sınıflandırma başarısı yüksek ve aykırı veriyi tölere edebilir bir yapıya sahiptir.

5.2 Kümeleme Algoritmaları

Kümeleme, bir veri kümesinin benzerlikerine göre gruplandıran eğiticisiz bir öğrenme yöntemidir. Amaç, birbirine yakın olan noktaların aynı kümede, uzak noktaların ise farklı kümede olmasını sağlamaktır. Tez kapsamında k-Means ve Hiyerarşik kümeleme algoritmaları, normalizasyon yöntemlerini karşılaştırmak için kullanılmıştır. Kümeleme yöntemleri için yakınlık Öklid mesafe ölçümü ile yapılmıştır.

5.2.1 K-Means Kümeleme

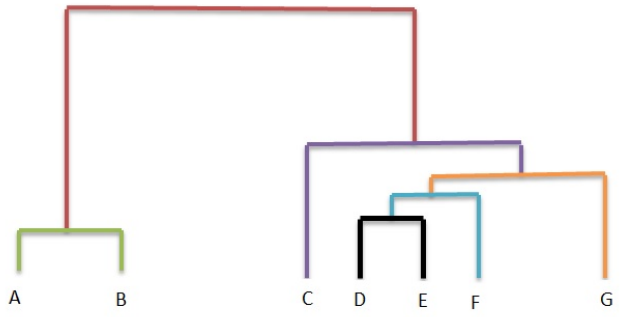
K-Means yönteminde, mevcut örneklerin içinden rastgele seçilen k tane nokta küme merkezi atanır ve veri kümesindeki n tane noktadan her biri, kendisine en yakın küme merkezinin bulunduğu kümeye atanır. Ardından her bir kümenin, içindeki noktaların ortalaması alınarak yeni küme merkezi belirlenir ve noktaların yeni küme merkezlerine olan mesafesi hesaplanır. Bu işlem, yeni hesaplanan küme merkezi bir öncekinden farklı olmayana kadar devam ettirilir. Sayısal veriler için yakınlık Öklid, Manhattan gibi mesafe ölçüm yöntemleri ile hesaplanırken, kategorik veriler için yakınlık, benzerlik (similarity) ölçümleri ile hesaplanır [76]. Algoritmanın en önemli noktası, k değerini belirlemektir. Algoritmanın kümeleme modeli Şekil 5.5'de verilmiştir. Basit ve efektif olduğu için yaygın kullanılan bir kümeleme yöntemidir.



Şekil 5.5 K-Means kümeleme modeli

5.2.2 Hiyerarşik Kümeleme

Hiyerarşik küme analizinde ki amaç, verileri her kümenin birbirinden farklı olduğu ve her kümenin içindeki verilerin birbirine büyük ölçüde benzer olduğu kümeler oluşturmaktır. Algoritma kümeleme işlemi için iki yaklaşım önerir; toplayıcı (agglomerative) veya bölücü (divisive). Toplayıcı yöntem, verilerin hepsinin ayrı kümeler olduğu varsayımı ile başlar. Ardından her bir adımda verilerin birbirine olan mesafeleri hesaplanır ve birbirine en yakın iki nokta birleştirilerek ortak bir küme oluşturulur. Bu birleştirme işlemine, küme sayısı istenilen değere gelene kadar devam edilir. Bölücü yöntemde ise, tüm noktaların aynı kümede olduğu varsayımı ile başlanır ve her bir iterasyonda kümeler bölünür. Bölücü yöntem global çözüme, toplayıcı yöntem lokal çözüme yakınsar. Ancak bölücü yöntemin, hesaplama maliyeti daha yüksek olduğu için genellikle toplayıcı yöntem tercih edilir [77]. Algoritmanın çıktısı Şekil 5.6'da görüldüğü gibi kümeler arasındaki hiyerarşik ilişkiyi gösteren bir dendogramdır.



Şekil 5.6 Hiyerarşik kümeleme modeli

5.3 Model Değerlendirme Yöntemleri

Makine Öğrenmesinde model oluşturmak için çok çeşitli yöntemler kullanılmaktadır. Bunların arasından veriye uygun olanı seçebilmek için bir takım değerlendirme kriterlerine ihtiyaç vardır. Amaç, modelin performansını ölçerek modelin uygunluğuna karar vermektir.

Model değerlendirme metriği verinin etiketli veya etiketsiz olmasına göre farklılık gösterebilmektedir. Tez çalışması kapsamında, sınıflandırma modelleri 'doğruluk' ölçütü ile kümeleme modelleri ise 'saflık' ve 'silüet katsayısı' ölçümleri ile değerlendirilmiştir.

5.3.1 Doğruluk Ölçütü

Etiketli veriler için uygulanan modellerin performanslarını değerlendirmede en yaygın kullanılan yöntem "sınıf karışıklık matrisi"dir ve verilerin gerçek sınıf bilgileri ile model tarafından öngörülen sınıflarına dair bilgilerini içerir. İki sınıfı bir veri kümesi için sınıf karışıklık matrisi Çizelge 5.1'de verilmiştir. Burada TP, gerçekte pozitif olup, sistemin de pozitif olarak etiketlediği örnek sayısıdır. TN, gerçekte negatif olup, sistemin de negatif olarak etiketlediği örnek sayısıdır. FP, gerçekte negatif olup, sistemin pozitif olarak etiketlediği örnekleri, FN ise gerçekte pozitif olup, sistemin negatif olarak etiketlediği örnek sayısını temsil eder.

Çizelge 5.1 Sınıf karışıklık matrisi

		GERÇEK	
		Pozitif	Negatif
TAHMİN	Pozitif	Gerçek Pozitif (True Positive-TP)	Sahte Pozitif (False Positive-FP)
	Negatif	Sahte Negatif (False Negative-FN)	Gerçek Negatif (True Negative-TN)

Doğruluk (accuracy), modelin doğru sınıflandırdığı örnek sayısıdır ve Denklem (5.3) ile hesaplanır.

$$\text{doğruluk} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

5.3.2 Saflık Ölçütü

Saflık (purity), kümelerin tek bir sınıfı ne ölçüde içerdiğini gösterir ve Denklem (5.4) ile hesaplanır. Yani bir kümedeki baskın sınıfa ait eleman sayısının kümedeki toplam eleman sayısına oranıdır. Burada M kümeleri, D sınıfları, N ise örnekleri temsil etmektedir. Saflık, [0-1] aralığında değer alır ve 1'e yaklaşması kümelerin saflığının arttığını gösterir [78].

$$\text{saflık} = \frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap D| \tag{5.4}$$

5.3.3 Silüet Katsayısı Ölçütü

Silüet katsayısı, her bir örneğin kendi kümesindeki öğelere olan ortalama mesafesi ile diğer kümelerdeki öğelere olan ortalama mesafesini karşılaştırır. Yani her bir örneğin kendi sınıfına ne kadar benzediğini ölçer. Silüet katsayısı [-1,1] aralığında değer alır ve değerinin yüksek olması ilgili örneğin kendi kümesindeki diğer örneklere daha çok

benzediğini göstermektedir. Veri kümesindeki her bir i örneği için silüet katsayısı (s(i)) Denklem (5.5) ile hesaplanır. Burada b(i), i örneğinin kendisine en yakın kümedeki diğer örneklerin ortalamasına olan mesafesi, a(i) ise kendisiyle aynı kümede olan diğer örneklerin ortalamasına olan mesafesidir.

$$s(i) = \frac{b(i) - a(i)}{maks(a(i), b(i))} \tag{5.5}$$

Küme içi veya kümeler arası benzerliği ölçmek için herhangi bir mesafe ölçüm yöntemi kullanılabilir. Tez kapsamında mesafe ölçümü olarak Öklid mesafesi kullanılmıştır. Optimum küme sayısını belirlemek için de kullanılabilir [79].

5.4 İstatistiksel Testler

İstatistiksel testler, belirli bir örneklem kümesinden elde edilen sonuçların, tesadüfi olup olmadığını, yani istatistiksel açıdan önemli olup olmadığını belirlemek için uygulanan testlerdir. Aynı zamanda Hipotez testi olarak da adlandırılır. Birden fazla grup arasında oluşan farkın istatistiksel açıdan önemliliğinin, bir grupta farklı şartlarda elde edilen sonuçların önemliliğinin, bir grubun dağılımının belirli bir dağılıma uygunluğunun test edilmesi gibi alanlarda kullanılır [80].

Hipotez testleri parametrik ve parametrik olmayan testler olarak ikiye ayrılır. Parametrik testler, ölçümlerin normal dağılım ile dağılmış olması, varyans değerlerinin eşit olması ve örneklem büyüklüğünün 30 veya daha fazla olması durumlarında kullanılır. Parametrik hipotez testleri şöyle sıralanır:

1. Tek örneklem t testi
2. Bağımsızlık t testi
3. Eşleştirilmiş t test
4. ANOVA testi

Verinin normal dağılıma uymadığı ve veri sayısının 30'dan az olduğu durumlarda ise parametrik olmayan testler kullanılır. Parametrik olmayan testler ise şöyle sıralanır:

1. Tek değişkenli Kolmogrov-Smirnov testi
2. Bağımsız iki değişkenli Mann-Whitney U test

3. Bağımlı iki değişkenli Wilcoxon testi
4. İkiden fazla değişkenli bağımsız Kruskal Wallis testi
5. İkiden fazla değişkenli bağımlı Friedman testi

Tez kapsamında, geliştirilen yöntemler ve karşılaştırılan yöntemlerin elde ettiği sonuçlar arasındaki farkın anlamlılığını ölçmek için Wilcoxon Rank Sum testi kullanılmıştır.

5.4.1 Wilcoxon Rank Sum Test

Wilcoxon Rank Sum Testi, iki değişkenli eşleştirilmiş t testinin parametrik olmayan versiyonu olarak kabul edilir. İki grup ortalamaları arasındaki farkın istatistiksel olarak anlamlı olup olmadığını ölçmek için kullanılır. Null hipotez, iki grup arasındaki farkın anlamlı olmadığı şeklindedir [81]. Test beş adımdan oluşur.

1. İki gruptan elde edilen veriler birleştirilerek artan sırada sıralanır.
2. Her bir grubun, sıralama neticesinde elde ettiği sıra numarası toplanır (R_1 ve R_2).
3. Eğer iki grubun eleman sayıları birbirine eşitse, bu toplamlardan küçük olanı Wilcoxon test değeridir (w).
4. Wilcoxon Rank Sum Test tablosundan, grupların eleman sayısı ve hassasiyet değerine göre kritik değer bulunur (w_{crit}).
5. Eğer w<w_{crit} ise null hipotez reddedilir ve ortalamalar arasındaki fark istatistiksel olarak anlamlıdır denir.

6
Genişletilmiş İkili Yapay Arı Kolonisi Algoritması (exBitABC)

Bu bölümde Yapay Arı Kolonisi algoritmasını ikili uzaya taşımak için önerilen exBitABC yönteminin temellerinden bahsedilmiştir. İkili Yapay Arı Kolonisi geliştirmek için öncelikle literatürdeki mevcut ikili Yapay Arı Kolonisi Algoritmaları öznitelik seçimi problemine uygulanarak güçlü ve zayıf yönleri incelenmiştir. Ardından mevcut yöntemler içerisinde başarılı olan BitABC algoritması kullanılarak en efektif sınıflandırıcının belirlenmesi için farklı sınıflandırma algoritmalarının performansları 2. bölümde karşılaştırılmıştır. BitABC algoritmasına eklenen lokal arama prosedürleri ile geliştirilen Geliştirilmiş BitABC (IBitABC) ve Genişletilmiş BitABC (exBitABC) algoritmaları, deneysel sonuçlar ile birlikte sırasıyla 3. ve 4. bölümde anlatılmıştır.

6.1 Mevcut Yöntemler İçin Deneysel Sonuçlar

Yapay Arı Kolonisi algoritması, sürekli uzay problemleri için geliştirilmiş olup, ayrık ve ikili veri içeren problemlere uygulanabilmesi için çeşitli modifikasyonlara ihtiyaç duyulmaktadır. Literatürde, YAK algoritması için önerilmiş yöntemler bulunmaktadır. Tez çalışması kapsamında ikili bir YAK algoritması geliştirilmesi amaçlandığından, literatürde mevcut ikili YAK algoritmaları, öznitelik seçimi problemine uygulanarak, mevcut yöntemlerin güçlü ve zayıf yönleri incelenmiştir.

Hançer vd. [14]'de öznitelik seçimi problemine uygulamak için, [12]'de önerilen DisABC algoritmasından esinlenilerek YAK algoritmasının Diferansiyel Evrim algoritması tabanlı, ikili bir versiyonunu geliştirmişlerdir (MDisABC). MDisABC'de, kaynaklar ikili vektörler ile temsil edilmiş ve DE'deki gibi mutant birey üretilerek aday çözüm elde edilmiştir. Mutant birey üretmek için DisABC'de olduğu gibi Jackard benzerlik ölçümü kullanılmıştır. Yazarlar geliştirdikleri yöntem ile, UCI Makine Öğrenmesi Havuzundan seçtikleri 10 veri kümesinde öznitelik seçimi yapmışlardır. Elde edilen sonuçlar, evrimsel yöntemlerden Genetik Algoritmalar [66], İkili Parçacık Sürü Optimizasyonu [68], Yeni Hız Tabanlı İkili Parçacık Sürü Optimizasyonu

(NBPSO) [82], Quantum Tabanlı İkili Parçacık Sürü Optimizasyonu (QBPSO) [83] ile karşılaştırılmıştır. Ayrıca literatürdeki ikili YAK algoritmalarından Açı Modülasyonlu İkili Yapay Arı Kolonisi (AMABC) [32], Modifikasyon Oranına Dayalı Yapay Arı Kolonisi (MRABC) [84], ayrık ikili yapay arı kolonisi (DisABC) [12] yöntemleri ile de eğitim, test kümesi sınıflandırıcı hataları, seçtikleri öznitelik sayıları ve çalışma süreleri bakımından karşılaştırılmıştır. Yazarların çalışmalarında kullandığı veri kümelerine dair bilgiler Çizelge 6.1'de verilmiştir. Yazarlar sınıflandırma algoritması olarak 5-NN kullanmışlardır.

Çizelge 6.1 Mevcut yöntemleri karşılaştırılmasında kullanılan veri kümeleri

Veriseti	Öznitelik Sayısı	Sınıf Sayısı	Örnek Sayısı
Wine	13	3	178
Vehicle	18	4	846
German	24	2	1000
WBCD	30	2	569
Ionosphere	34	2	351
Lung	56	3	32
Hill Valley	100	2	606
Musk-I	166	2	476
Madelon	500	2	2600
Isolet5	617	26	1559

Literatürde mevcut ikili YAK yöntemlerinin güçlü ve zayıf yönlerini görebilmek için, [14]'de uygulanmamış ikili YAK algoritmaları, [14]'de kullanılan parametre ve veri kümeleri ile öznitelik seçimi problemine uygulanmıştır. Bu kapsamda, [14]'de uygulanan yöntemlerden farklı olarak GBABC [17], BitABC [19], BinABC [20], DABC [26], BABC [85], BABC [28], XBABC [31], XSABC [31], ABC_{bin} [29], GbABC [30], MABC [86] yöntemleri önerildekleri çalışmalarda verilen akış diyagramları göz önüne alınarak aslına uygun olarak gerçeklenmiştir. [14]'de uygulanan yöntemler ile karşılaştırma yapabilmek adına, algoritmalar [14]'de belirlenen aşağıdaki parametreler ile uygulanmıştır.

- Populasyon sayısı:30,
- Azami iterasyon sayısı: 30,
- Parçacık sürü zekası yöntemleri için c1=1, c2=2, başlangıç ağırlığı (initial weight)=0,9, Vmax=6,
- Genetik Algoritmalar için crossover oranı 0,8, mutasyon oranı 0,2,

Hançer vd. [14]'de olduğu gibi, veri kümelerinin %70'i eğitim, %30'u test için ayrılmıştır ve eğitim ve test kümesindeki veriler rastgele seçilmiştir. Veriler 5-NN ile sınıflandırılmış ve eğitim kümesi için sonuçlar 10 kat çapraz geçerleme ile elde edilmiştir. Algoritmalar her defasında verisetlerinden rastgele örnekler seçilerek üretilen yeni bir eğitim ve test kümesi ile 30 defa çalıştırılarak ortalaması alınmıştır.

Algoritmaların seçtiği öznitelik alt kümesi için uygunluk değeri, [14]'de olduğu gibi, Denklem (6.1) ile hesaplanmaktadır. Öznitelik seçimi probleminde ki amaç fonksiyonu, sadece sınıflandırma doğruluğuna değil aynı zamanda seçilen özniteliklerin sayısına da bağlıdır. Elde edilen iki öznitelik alt kümesinden doğruluk değeri daha yüksek olan tercih edilir. Ancak, iki alt küme aynı doğruluk değerine sahipse, içlerinden öznitelik sayısı daha az olan çözüm tercih edilir. Doğruluk değeri öznitelik sayısından daha önemli olduğu için, c katsayısı 0,9995 olarak belirlenmiştir.

$$uygunluk = (c * dogruluk) + ((1 - c) * \frac{\#SecilenOznitelik}{\#ToplamOznitelik}) \tag{6.1}$$

$$dogruluk = \frac{\#DogruSiniflandirilanOrnek}{\#ToplamOrnek} \tag{6.2}$$

Her algoritmanın 30 defa çalıştırılmasından sonra, eğitim ve test kümeleri için sınıflandırma hatalarının ortalama ve standart sapma değerleri sırasıyla Çizelge 6.2 ve Çizelge 6.3'de verilmiştir. Hata fonksiyonu olarak (1-doğruluk) değeri kullanılmıştır. Çizelge 6.3'de, "O.S." olarak belirtilen satırlar, her bir algoritma için seçilen ortalama öznitelik sayısını gösterir. En iyi sonuçlar koyu olarak işaretlenmiştir. Çizelgelerde MDisABC, DisABC, AMABC, MRABC, GA, BPSO, NBPSO ve QBPSO için verilen sonuçlar ilgili çalışmadan ([14]) alınmış olup diğer yöntemler çalışma kapsamında önerildikleri çalışmalardaki akış diyagramları göz önüne alınarak uygulanmıştır.

Çizelge 6.2'de görüldüğü gibi, ABC algoritmasını ikilileştirme işlemi için genetik operatörler kullanan GbABC, eğitim kümesinde altı veri kümesi için en düşük hata değerine sahip alt kümeleri bulmuştur. BitABC ise, Wine ve Ionosphere veri kümeleri için daha iyi performans göstermiştir; BABC ve MABC birer veri kümesinde en düşük hata ile sonuca ulaşmıştır. MRABC ve MABC benzer yöntemlerdir; ikisi de bir eşik değeri belirler ve ilgili özelliği sıfır ile bir arasında rasgele bir sayıya göre alt kümeye dahil edip etmeyeceğine karar verir. Ancak eğitim kümesi performansları çok farklı çıkmıştır. İkili Parçacık Sürüsü Optimizasyonu Algoritmaları ve Genetik Algoritma, öznitelik seçimi problemleri için popüler olmasına rağmen, herhangi bir veri kümesi için minimum hata oranını elde edememiştir. Bu nedenle, GBABC algoritmasının eğitim kümesi doğruluk hatası açısından diğerlerinden daha iyi performans gösterdiği görülmektedir.

Çizelge 6.3, test kümesi için sınıflandırma hata oranlarını ve seçilen ortalama öznitelik sayısını göstermektedir. Her bir veri kümesinin toplam öznitelik sayısı, parantez içinde verilmiştir. Wine veri seti için DisABC, MDisABC, BPSO ve QBPSO birlikte %100 doğrulukla sınıflandırma yapmıştır. MDisABC, en az sayıda öznitelik kümesiyle, minimum test kümesi hatasını verdiğinden, bu veri seti için başarılı olarak kabul edilir. Vehicle veri seti için, BPSO ve MRABC en iyi sonuçları vermiştir, ancak MRABC'nin seçtiği özellik sayısı BPSO'dan daha küçüktür. Bu nedenle, MRABC Vehicle için başarılı olmuştur. Diğer veri setleri için MDisABC en iyi doğruluğu üç veriseti için elde ederken, BABC, GbABC, XSABC, XBABC ve BitABC birer kere elde etmiştir. Hill Valley ve Lung veri kümeleri için test kümesi hatasının tüm yöntemlerde yüksek çıkması, ilgili veri kümelerinin zor modellenebilen veri kümeleri olduğunu göstermektedir.

Çizelge 6.2 Mevcut yöntemlerin eğitim kümesi hatalarına göre karşılaştırılması

Yöntem	Wine	Vehicle	German	WBCD	Ionosp.	Lung	H.Valley	Musk	Madelon	Isolet
GBABC [17]	0,03 ±0,01	0,16 ±0,01	0,17 ±0	**0,03 ±0**	0,06 ±0	**0,03 ±0,03**	**0,22 ±0,01**	**0,02 ±0,01**	**0,1 ±0,01**	**0,06 ±0,01**
BitABC [19]	**0,02 ± 0**	0,16 ± 0,003	0,17 ± 0,004	0,03 ± 0,001	**0,05 ± 0,003**	0,08 ± 0,03	0,24 ± 0,004	0,05 ± 0,005	0,09 ± 0,08	0,08 ±0,002
BinABC [20]	0,04 ± 0,01	0,17 ± 0,01	0,19 ± 0,005	0,05 ± 0,01	0,09 ± 0,01	0,18 ± 0,04	0,24 ± 0,01	0,07 ± 0,01	0,15 ± 0,01	0,1 ± 0,004
DABC [26]	0,02 ± 0,002	0,17 ± 0,004	0,17 ± 0,01	0,03 ± 0,001	0,06 ± 0,004	0,09 ± 0,02	0,24 ± 0,003	0,06 ± 0,004	0,14 ± 0,01	0,1 ± 0,004
BABC [85]	0,02 ± 0,004	**0,15 ± 0,003**	0,17 ± 0,005	0,04 ± 0,001	0,05 ± 0,01	0,09 ± 0,04	0,23 ± 0,01	0,06 ± 0,01	0,15 ± 0,01	0,1 ± 0,07
BABC [28]	0,04 ± 0,08	0,18 ± 0,01	0,19 ± 0,005	0,05 ± 0,08	0,1 ± 0,01	0,22 ± 0,03	0,26 ± 0,005	0,07 ± 0,01	0,16 ± 0,01	0,1 ± 0,004
XBABC [31]	0,03 ± 0,01	0,16 ± 0,01	0,18 ± 0,004	0,04 ± 0,001	0,09 ± 0,01	0,05 ± 0,02	0,25 ± 0,004	0,04 ± 0,005	0,13 ± 0,01	0,08 ± 0,004
XSABC [31]	0,04 ± 0,01	0,17 ± 0,005	0,18 ± 0,004	0,03 ± 0,003	0,08 ± 0,01	0,05 ± 0,02	0,25 ± 0,004	0,06 ± 0,004	0,14 ± 0,002	0,1 ± 0,001
ABC_{bin} [29]	0,02 ± 0,01	0,18 ± 0,003	0,17 ± 0,003	0,03 ± 0,002	0,07 ± 0,004	0,09 ± 0,01	0,26 ± 0,005	0,06 ± 0,003	0,14 ± 0,004	0,09 ± 0,003
GbABC [30]	0,02 ± 0,001	0,16 ± 0,01	0,18 ± 0,01	0,04 ± 0,01	0,09 ± 0,01	0,08 ± 0,03	0,24 ± 0,004	0,05 ± 0,004	0,14 ± 0,01	0,09 ± 0,003
MABC [86]	0,03 ± 0,003	0,15 ± 0,01	**0,16 ± 0,002**	0,03 ± 0,002	0,06 ± 0,005	0,07 ± 0,02	0,25 ± 0,003	0,05 ± 0,005	0,14 ± 0,003	0,09 ± 0,003
MDisABC [14]	0,04 ± 0,001	0,29 ± 0,001	0,24 ± 0,004	0,03 ±0,003	0,06 ± 0,004	0,19 ± 0,03	0,40 ± 0,008	0,08 ± 0,006	0,19 ± 0,01	0,13 ± 0,006
DisABC [12]	0,04 ± 0,002	0,29 ± 0,005	0,24 ± 0,007	0,04 ± 0,005	0,06 ± 0,004	0,19 ± 0,04	0,40 ± 0,001	0,08 ± 0,01	0,2 ± 0,02	0,14 ± 0,01
AMABC [32]	0,03 ± 0,02	0,16 ± 0,01	0,17 ± 0,004	0,03 ± 0,002	0,07 ± 0,004	0,15 ± 0,03	0,26 ± 0,003	0,04 ± 0,004	0,14 ± 0,005	0,1 ± 0,002
MRABC [84]	0,04 ± 0,002	0,29 ± 0,005	0,24 ± 0,006	0,04 ± 0,004	0,09 ± 0,01	0,23 ± 0,04	0,42 ± 0,01	0,09 ± 0,008	0,20 ± 0,007	0,13 ± 0,006
GA [66]	0,04 ± 0,005	0,3 ± 0,007	0,25 ± 0,01	0,05 ± 0,004	0,09 ± 0,01	0,25 ± 0,05	0,41 ± 0,01	0,07 ± 0,01	0,18 ± 0,01	0,12 ± 0,07
BPSO [68]	0,04 ± 0,001	0,29 ± 0,003	0,25 ± 0,005	0,04 ± 0,004	0,09 ± 0,008	0,24 ± 0,04	0,43 ± 0,01	0,1 ± 0,005	0,22 ± 0,005	0,14 ± 0,005
NBPSO [82]	0,04 ± 0,001	0,29 ± 0,006	0,24 ± 0,008	0,04 ± 0,006	0,07 ± 0,01	0,2 ± 0,05	0,40 ± 0,01	0,07 ± 0,009	0,19 ± 0,01	0,12 ± 0,007
QBPSO [83]	0,04 ± 0,002	0,3 ± 0,01	0,24 ± 0,005	0,04 ± 0,004	0,07 ± 0,009	0,19 ± 0,04	0,40 ± 0,008	0,08 ± 0,006	0,19 ± 0,005	0,12 ± 0,005

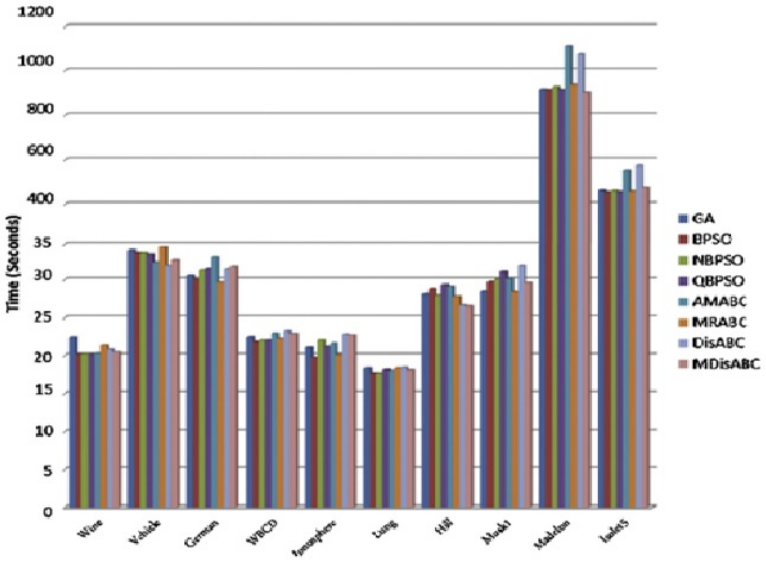
Öznitelik altkümelerinin boyutu göz önüne alındığında, BitABC altı veri kümesinde en iyi performansı göstermiştir. Özellikle Hill Valley ve Musk veri kümeleri için en iyi

Çizelge 6.3 Mevcut yöntemlerin test kümesi hatası ve seçilen öznitelik sayısına göre karşılaştırılması

Yöntem	Wine (13)	Vehicle (18)	German (24)	WBCD (30)	Ionosp. (34)	Lung (56)	H.Valley (100)	Musk (166)	Madelon (500)	Isolet (617)
GBABC [17]	0,04 ±0,04	0,33 ±0,02	0,31 ±0,02	0,08 ±0,01	0,09 ±0,02	0,55 ±0,08	0,49 ±0,02	0,14 ±0,02	0,2 ±0,02	0,16 ±0,02
O.S.	4,53	8,77	8,7	3,8	**3,93**	**9,23**	30,3	61,4	147,3	223
BitABC [19]	0,06 ± 0,006	0,35 ± 0,027	0,3 ± 0,02	0,1 ± 0,01	0,11 ± 0,02	0,46 ± 0,15	0,48 ± 0,02	0,14 ± 0,02	**0,17 ± 0,02**	0,21 ±0,02
O.S.	**4,1**	8,43	9,57	**3,57**	4,2	9,43	**11,73**	**37,7**	**52,7**	**113**
BinABC [20]	0,08 ± 0,02	0,32 ± 0,02	0,28 ± 0,02	0,05 ± 0,02	0,19 ± 0,02	0,55 ± 0,12	0,51 ± 0,02	0,15 ± 0,02	0,27 ± 0,02	0,21 ± 0,01
O.S.	5,7	10	13,03	13,93	13,6	26,77	48,7	82,47	253	306,8
DABC [26]	0,11 ± 0,01	0,31 ± 0,02	0,32 ± 0,02	0,09 ± 0,01	0,12 ± 0,03	0,57 ± 0,11	0,45 ± 0,02	0,16 ± 0,02	0,24 ± 0,02	0,19 ± 0,02
O.S.	4,63	**7,73**	8,6	6,7	5,5	14,2	23,43	47	158	307,4
BABC [85]	0,04 ± 0,02	0,31 ± 0,01	0,29 ± 0,01	**0,05 ± 0,003**	0,14 ± 0,03	0,49 ± 0,13	0,48 ± 0,02	0,18 ± 0,02	0,28 ± 0,03	0,21 ± 0,02
O.S.	6,7	10,13	11,17	11,7	5,17	20,1	23,77	57,9	217	233,9
BABC [28]	0,08 ± 0,02	0,34 ± 0,03	0,31 ± 0,02	0,07 ± 0,02	0,17 ± 0,02	0,46 ± 0,13	0,47 ± 0,03	0,18 ± 0,03	0,27 ± 0,02	0,2 ± 0,01
O.S.	6,47	9,4	12,2	14,1	14	27,8	49,7	83	253	313
XBABC [31]	0,09 ± 0,03	0,29 ± 0,02	0,29 ± 0,01	0,07 ± 0,01	0,14 ± 0,02	0,65 ± 0,11	0,53 ± 0,01	**0,11 ± 0,01**	0,27 ± 0,02	0,2 ± 0,01
O.S.	5,5	11,83	14,63	13,1	12,37	28,9	56,4	100	339,9	379,1
XSABC [31]	0,06 ± 0,02	0,34 ± 0,02	0,29 ± 0,02	0,08 ± 0,002	0,13 ± 0,04	0,48 ± 0,11	**0,44 ± 0,02**	0,14 ± 0,02	0,27 ± 0,01	0,18 ± 0,002
O.S.	6	11,13	13,87	12,4	5,63	29,7	24,83	97,7	290,7	365,7
ABC_{bin} [29]	0,04 ± 0,01	0,3 ± 0,03	0,29 ± 0,01	0,07 ± 0,01	0,14 ± 0,02	0,57 ± 0,13	0,45 ± 0,02	0,18 ± 0,03	0,24 ± 0,02	0,2 ± 0,01
O.S.	5,23	10	12,3	12,1	13,07	26,2	48,37	80,83	252,4	305,3
GbABC [30]	0,66 ± 0,001	0,84 ± 0,001	**0,29 ± 0,001**	0,61 ± 0,001	0,49 ± 0	0,7 ± 0,001	0,46 ± 0,001	0,64 ± 0,001	0,47 ± 0,001	0,45 ± 0,001
O.S.	6,93	11,67	15,5	14,8	16,97	33,9	62,97	117	342	423,4
MABC [86]	0,008 ± 0,02	0,32 ± 0,01	0,32 ± 0,001	0,06 ± 0,004	0,11 ± 0,02	0,49 ± 0,16	0,46 ± 0,02	0,13 ± 0,02	0,25 ± 0,01	0,2 ± 0,02
O.S.	5,01	11,83	16,57	10,5	9,67	22,6	37,67	103	305,9	352,4
MDisABC [14]	**0 ± 0**	0,212 ± 0,021	0,3 ± 0,02	0,07 ± 0,011	**0,06 ± 0,02**	**0,33 ± 0,08**	0,45 ± 0,02	0,15 ± 0,02	0,21 ± 0,02	**0,14 ± 0,012**
O.S.	**5,76**	9,3	8,03	11,86	5,76	24,4	30,53	75,76	195,96	300,1
DisABC [12]	**0 ± 0**	0,209 ± 0,019	0,29 ± 0,03	0,07 ± 0,011	0,07 ± 0,02	0,4 ± 0,12	0,45 ± 0,02	0,16 ± 0,03	0,23 ± 0,04	0,16 ± 0,01
O.S.	5,83	9,73	8,73	12,23	6,03	16,66	24,6	86,16	223,86	378,33
AMABC [32]	0,07 ± 0,02	0,31 ± 0,022	0,31 ± 0,023	0,08 ± 0,003	0,13 ± 0,022	0,59 ± 0,13	0,46 ± 0,019	0,16 ± 0,024	0,26 ± 0,02	0,19 ± 0,01
O.S.	5	8,63	12,23	11,2	11,4	23,5	43,8	79,7	243,4	300,2
MRABC [84]	0,001 ± 0,005	**0,207 ± 0,017**	0,3 ± 0,02	0,08 ± 0,01	0,09 ± 0,015	0,43 ± 0,14	0,46 ± 0,02	0,16 ± 0,03	0,24 ± 0,01	0,15 ± 0,01
O.S.	5,76	9,73	10,4	14,5	10,13	26,5	44,73	82,26	248,43	303,96
GA [66]	0,09 ± 0,02	0,208 ± 0,017	0,29 ± 0,02	0,07 ± 0,014	0,08 ± 0,02	0,4 ± 0,13	0,46 ± 0,02	0,17 ± 0,02	0,24 ± 0,013	0,15 ± 0,012
O.S.	6,1	10	10,7	14,96	10,93	28	45,73	81,83	250,03	306,06
BPSO [68]	**0 ± 0**	**0,207 ± 0,17**	0,29 ± 0,02	0,07 ± 0,01	0,08 ± 0,014	0,34 ± 0,13	0,46 ± 0,02	0,17 ± 0,03	0,24 ± 0,01	0,15 ± 0,01
O.S.	5,8	9,93	11,43	13,36	10,1	27,3	46,16	81,2	248,1	306,6
NBPSO [82]	0,004 ± 0,01	0,21 ± 0,019	0,29 ± 0,02	0,07 ± 0,01	0,08 ± 0,02	0,39 ± 0,16	0,46 ± 0,015	0,16 ± 0,02	0,24 ± 0,02	0,15 ± 0,01
O.S.	6,13	9,76	11,3	13,96	9,63	27,3	44,86	82,23	246,56	303,93
QBPSO [83]	**0 ± 0**	0,22 ± 0,02	0,29 ± 0,02	0,07 ± 0,01	0,08 ± 0,02	0,35 ± 0,12	0,45 ± 0,02	0,15 ± 0,02	0,22 ± 0,015	0,145 ± 0,011
O.S.	5,83	9,9	10,83	14,3	8,23	26,7	44,7	81,4	240,56	305,73

yöntem ile BitABC arasındaki fark sırasıyla yüzde 0,03 ve 0,04'tür. Ancak Hill Valley için, BitABC'nin seçtiği özniteliklerin sayısı, bu veri kümesi için en iyi doğruluğu veren XSABC'nin yarısından azdır. Benzer şekilde BitABC, Musk veri kümesini 0,16 hata oranı ve ortalama 37,7 öznitelik ile sınıflandırmıştır. Ancak, XBABC aynı veri kümesini 0,13 hata ve 100 öznitelik ile sınıflandırmıştır. Test kümesi hataları arasındaki farklar küçük olmakla birlikte, öznitelik boyutları arasındaki fark çok yüksektir.

GA, BPSO, NBPSO, QBPSO, MRABC, DisABC ve MDisABC için sonuçlar [14]'den alınmıştır. Bu algoritmaları uygulamak için, yazarlar Intel Core i7-4700HQ 2.40 GHz CPU, 8 GB RAM'e sahip bir bilgisayar ve Matlab 2013a kullanmışlardır. GBABC, BitABC, BinABC, DABC, BABC, XBABC, XSABC, BABC, ABCbin, GbABC ve MABC yöntemleri ve algoritmaları, ilgili çalışmalarda belirtilen önerilere göre, Matlab 2013a ortamında kodlanmış ve 16GB RAM, Intel Core i7-3630QM 2.4 GHz CPU içeren bir bilgisayar kullanılarak çalıştırılmıştır. Hesaplama zamanları, donanımın eşitsizliği nedeniyle ayrı ayrı analiz edilmiştir.



Şekil 6.1 Ortalama çalışma süresi [14]

Şekil 6.1, [14]'de rapor edilen algoritmaların çalışma sürelerini saniye cinsinden göstermektedir. Şekil 6.1'de görüldüğü gibi, algoritmaların hesaplama karmaşıklıkları birbirine yakındır. Madelon ve Isolet için, AMABC ve DisABC algoritmalarının karmaşıklığı diğerlerinden daha yüksektir. MDisABC algoritması özellikle veri kümesi büyük olduğunda daha iyidir, ancak küçük olanlar için ortalama bir performansı vardır. Bu çalışmada uygulanan algoritmaların hesapsal maliyetleri Çizelge 6.4'de gösterilmektedir. [14]'e benzer olarak, GBABC dışındaki algoritmaların hesaplama zamanları birbirine yakındır. Çizelge 6.2'ye göre, GBABC eğitim kümelerinde iyi performans elde etmiş ancak genellikle en yüksek sürelerde sonuçlanmıştır. Daha büyük iki veri kümesi Madelon ve Isolet için, sonuçlar arasındaki fark daha büyüktür.

BitABC hesaplama süresi bakımından, özellikle büyük veri kümelerinde, genel olarak iyi sonuçlar elde etmiştir. Ek olarak, BABC ve DABC'de iyi performans göstermiştir.

Çizelge 6.4 Uygulanan yöntemler için çalışma süresi (saniye)

Yöntem	Wine	Vehicle	German	WBCD	Ionosp.	Lung	H.Valley	Musk	Madelon	Isolet
GBABC [17]	4197,9 ±62,08	4549,4 ±90,49	4638,3 ±104	4288,2 ±61,82	4347,8 ±152,7	4013,6 ±69,07	4298,5 ±91,16	4450,7 ±234,86	30265 ±7562,9	19002 ±2269,8
BitABC [19]	390,82 ± 5,98	432,7 ± 6,61	439,5 ± 5,22	408,5 ± 7,37	396,8 ± 5,95	380,9 ± 3,89	**397,9 ± 5,05**	410,6 ± 7,1	**2124,9 ± 157,76**	**1362,4 ±50,99**
BinABC [20]	388,4 ± 8,32	422,01 ± 4,89	428,9 ± 6,17	384,99 ± 4,59	371,8 ± 5,53	359,3 ± 5,69	418,99 ± 7,31	416,5 ± 5,18	5165,6 ± 73,39	2598,9 ± 49,39
DABC [26]	397,5 ± 25,47	418,7 ± 7,86	434,4 ± 7,34	401,12 ± 5,52	386,17 ± 9,6	**353,26 ± 4,93**	398,64 ± 6,15	**395,21 ± 7,48**	3296,3 ± 897,61	1881,3 ± 74,86
BABC [85]	427,7 ± 24,75	462,2 ± 11,49	469,61 ± 10,85	432,48 ± 13,98	410,97 ± 20,49	424,44 ± 32,77	442,18 ± 13,94	437,43 ± 23,56	4204,5 ± 389,36	2162,2 ± 207,65
BABC [28]	388,44 ± 8,32	422,01 ± 4,89	**428,92 ± 6,17**	**384,99 ± 4,59**	371,85 ± 5,53	359,26 ± 5,69	418,99 ± 7,31	416,55 ± 5,18	5165,6 ± 73,39	2598,9 ± 49,39
XBABC [31]	415,88 ± 8,27	422,8 ± 8,36	443,25 ± 6,85	390,1 ± 6,17	385,58 ± 4,52	0369,02 ± 4,41	438,29 ± 7,51	451,12 ± 10,99	7988,6 ± 804,28	3319 ± 344,01
XSABC [31]	398,01 ± 7,41	437,64 ± 4,1	449,95 ± 6,28	419,55 ± 28,66	393,45 ± 4,21	376,57 ± 3,99	445,33 ± 20,39	486,67 ± 9,18	5254,2 ± 24,97	2666,6 ± 22,85
ABC_{bin} [29]	400,81 ± 4,57	442,44 ± 6,43	450,13 ± 4,96	409,71 ± 5,35	390,03 ± 3,35	384,2 ± 18,16	437,57 ± 5,35	439,57 ± 5,28	5375,3 ± 59,63	2732,6 ± 50,1
GbABC [30]	389,58 ± 6,05	**418,68 ± 6,76**	441,18 ± 10,47	413,51 ± 22,88	385,62 ± 4,5	376,37 ± 4,22	455,05 ± 21,91	463,32 ± 10,19	7554,7 ± 500,18	3673,7 ± 227,04
MABC [86]	401,59 ± 5,01	422,65 ± 7,57	442,45 ± 6,76	402,71 ± 19,6	404,16 ± 4,29	380,37 ± 6,56	426,49 ± 5,53	435,78 ± 5,37	6559,2 ± 178,14	3049,2 ± 107,25

Literatürdeki İkili Yapay Arı Kolonisi algoritmalarının öznitelik seçimi problemine ilişkin performanslarını karşılaştırmak amacıyla, 15 adet İkili Yapay Arı Kolonisi algoritması ve 4 Evrimsel Algoritma karşılaştırılmıştır. Genel olarak evrimsel metodlar GA, BPSO, NBPSO ve QBPSO, YAK yöntemlerinden daha başarılı sonuç üretememiştir. Genetik operatör tabanlı GBABC eğitim kümesinde başarılı sonuçlar almasına rağmen test küme başarısı düşük çıkmıştır. Ayrıca algoritmanın hesapsal maliyeti de yüksektir. Bu sonuçlardan, GBABC'nin genelleme yeteneğinin sınırlı olduğu görülmektedir. Test doğruluğunu karşılaştırırken hiçbir yöntem kayda değer bir üstünlük sağlayamamıştır. Ancak, MDisABC üç veri kümesi için daha iyi sonuçlar vermiştir. Öznitelik azaltma performansları karşılaştırıldığında, BitABC veri kümelerinin yarısı için çok iyi bir performans sergilemiştir [1].

6.2 Sınıflandırıcıların Etkisinin Araştırılması

Öznitelik seçimi probleminde sınıflandırıcıların etkisini araştırmak ve ileriki çalışmalarımızda, optimizasyon algoritmasının amaç fonksiyonu kapsamında kullanılacak sınıflandırma algoritmasını seçmek için, BitABC algoritması altı sınıflandırıcı ile sekiz veri kümesine uygulanarak alınan sonuçlar test kümesi hatası,

[1] Bu çalışma INISTA 2016 sempozyumunda bildiri olarak sunulmuş ve ardından bir kitap bölümü olarak yayınlanmıştır

seçilen öznitelik sayısı ve çalışma süreleri bakımından karşılaştırılmıştır. Sınıflandırma algoritması olarak, Bayes teoremi tabanlı Naive Bayes, entropi tabanlı Karar Ağacı, uzaklık tabanlı K-NN, çekirdek tabanlı Destek Vektör Makineleri, olasılık ve yapay sinir ağı tabanlı Olasılıksal Yapay Sinir Ağları ve topluluk tabanlı Rastgele Orman algoritmaları seçilmiştir. Karşılaştırma için kullanılan veri kümeleri UCI Makine Öğrenmesi Havuzundan alınmıştır. Veri kümelerine ilişkin bilgiler Çizelge 6.5'de verilmiştir.

Çizelge 6.5 Sınıflandırma yöntemlerini karşılaştırmak için kullanılan veri kümeleri

Verieti	Öznitelik Sayısı	Sınıf Sayısı	Örnek Sayısı
Vehicle	18	4	846
German	24	2	1000
WBCD	30	2	569
Hill Valey	100	2	606
Urban Land Cover	148	9	168
Musk	166	2	476
LSVT	309	2	126
Madelon	500	2	2600

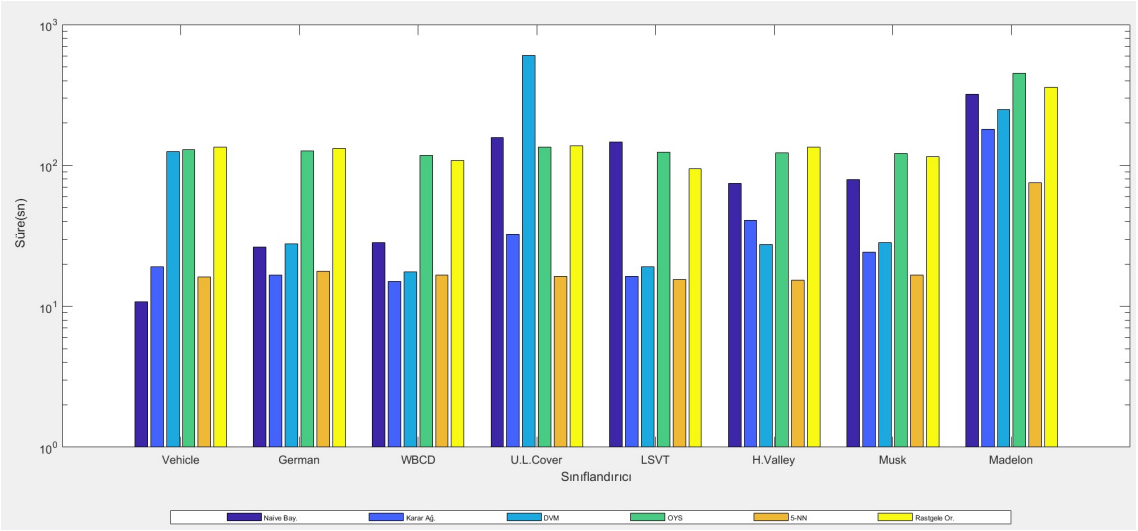
BitABC, [19] önerilen akış diyagramına göre Matlab R2017a ortamında kodlanmıştır. Kolonide 15 işçi arı, 15 gözcü arı olmak üzere 30 arı bulunmaktadır. Algoritmanın maksimum iterasyon sayısı 30 ve geliştirilememe limiti 50 olarak seçilmiştir. Veri kümeleri, rastgele seçilen örneklerden oluşacak şekilde 3 parçaya bölünmüştür. Buna göre verilerin %60'ı eğitim, %20'si geçerlilik ve kalan %20'si de test için kullanılmıştır. Uygunluk fonksiyonu Denklem 6.1, sınıflandırıcı doğruluğu ise Denklem 6.2 ile hesaplanmıştır. Sınıflandırma algoritmaları, Matlab r2017a makine öğrenmesi aracındaki hazır fonksiyonlar ve varsayılan parametreler ile çalıştırılmıştır. K-NN algoritması için k değeri 5, Rastgele Orman algoritması için ise topluluktaki ağaç sayısı 10 olarak seçilmiştir. BitABC her bir sınıflandırıcı ile 20'şer defa çalıştırılarak elde edilen sonuçların ortalaması alınmıştır.

Çizelge 6.6'da görüldüğü gibi, Naive Bayes, Karar Ağacı ve Rastgele Orman algoritmaları birer veri kümesinde en az hata ile sınıflandırma yapan öznitelikleri seçerken, 5-NN kalan beş veri kümesi için test hatası bakımından en iyi sonucu vermiştir. DVM ve OYSA ise özellikle daha küçük veri kümelerinde başarılı olamazken 5-NN'in başarısı, veri kümesi büyüdükçe daha belirgin hale gelmiştir. 10 tane karar ağacının sonuçlarını kullanan Rastgele Orman algoritması, tekil Karar Ağacına göre bazı veri kümeleri için daha iyi sonuçlar verse de iyi bir başarı elde edememiştir. Test kümesi hatası için standart sapma değerlerinin düşük olması, sınıflandırıcıların ürettikleri performanslarda kararlı davrandıklarını göstermektedir.

Çizelge 6.6 Sınıflandırıcıların doğruluk değerleri ve öznitelik sayılarına göre karşılaştırılması

Veri K.		Naive Bay.	Karar Ağ.	DVM	OYSA	5-NN	Rast. Orm.
Vehicle	Test H.	0,469±0,04	0,324±0,04	0,545±0,001	0,506±0,001	0,312±0,03	**0,29±0,03**
(18)	#Öz.	6,7±1,66	7,75±1,65	**6,4±1,57**	6,75±2	6,75±1,52	7,95±1,76
German	Test H.	0,278±0,03	0,3±0,03	0,3±0,001	0,3±0,001	**0,275±0,02**	0,306±0,03
(24)	#Öz.	7,43±2,16	**6,55±1,76**	9,3±2,05	8,85±2,6	8,15±1,69	7,3±1,52
WBCD	Test H.	0,052±0,02	0,084±0,04	0,23±0,01	0,372±0,001	**0,045±0,02**	0,064±0,02
(30)	#Öz.	8,1 ±2,58	**7,45±2,68**	13,6±2,14	11,75±2,65	7,75±1,74	8,8±2,42
U.L.Cov.	Test H.	**0,171±0,03**	0,222±0,04	0,554 ±0,03	0,293±0,04	0,174±0,03	0,211±0,03
(148)	#Öz.	28,9 ± 6,65	34,45±6,46	51,1±6,56	55,25 ±7,93	**24,35±5,78**	38,8± 7,56
LSVT	Test H.	0,43±0,09	0,228±0,06	0,23 ±0,06	0,32±0,001	**0,226±0,07**	0,232±0,082
(309)	#Öz.	81,95 ± 29,65	78±16,27	100,05±16,1	126,1 ±8,86	**66,85±12,86**	92,5± 16,84
Hill V.	Test H.	0,5±0,03	**0,426±0,05**	0,507 ±0,02	0,5±0,01	0,472±0,04	0,478±0,05
(100)	#Öz.	37,9 ± 5,54	19,5±5,13	34,05±11,02	37,35 ±6,43	**18,3±5,62**	24,75± 6,28
Musk	Test H.	0,228±0,03	0,22±0,06	0,308 ±0,03	0,431 ±0	**0,168±0,04**	0,203±0,04
(166)	#Öz.	**34,37 ± 7,24**	35,85±7,21	59,6±8,42	65,55 ±6,87	36,2±5,81	42,7± 71,83
Madelon	Test H.	0,46±0,05	0,238±0,02	0,389 ±0,02	0,4 ±0,03	**0,189±0,02**	0,354±0,02
(500)	#Öz.	147,37 ± 62,57	96,7±15,11	89,1±20,69	76,05 ±40,39	**68,45±12,03**	91,6± 22,97

Veri kümelerinin toplam öznitelik sayıları parantez içerisinde verilmiştir. Sınıflandırıcılar seçilen ortalama öznitelik sayısına göre karşılaştırıldığında, Naive Bayes ve DVM bir, Karar Ağacı ise iki veri kümesi için minimum öznitelik ile sınıflandırma yapmıştır. Sekiz veri kümesinin dört tanesi için 5-NN sınıflandırıcısı başarılı olmuştur. Sonuçlardan görülmektedir ki, 5-NN'in başarılı olamadığı dört veri kümesi için seçilen öznitelik sayıları arasında küçük farklar vardır. Ancak, 5-NN'in test kümesi hatası bakımından optimal sonucu elde ettiği veri kümelerinde diğer yöntemlerle arasındaki performans farkı daha yüksek çıkmıştır.



Şekil 6.2 Sınıflandırıcılar için ortalama çalışma süresi

Algoritmalar, Intel(R) Core (TM) i7-6700 HQ 2.60 GHz işlemci ve 16 GB RAM'e sahip bir bilgisayarda çalıştırılmıştır. Şekil 6.2'de, sınıflandırma algoritmalarının BitABC ile birlikte ortalama çalışma süreleri saniye cinsinden karşılaştırmalı olarak verilmiştir.

Buna göre, 5-NN ve Karar Ağacı için çalışma süreleri birbirine yakın olmakla birlikte German ve WBCD dışındaki tüm veri kümelerinde 5-NN asgari çalışma süresine sahiptir. Naive Bayes için çalışma süresi küçük veri kümelerinde 5-NN'in çalışma süresine yakın seyrederken, veri kümesi büyüdükçe çalışma süresindeki artış fazla olmuştur. Sekiz veri kümesinden altı tanesi iki sınıflı, Vehicle ve Urban Land Cover ise çok sınıflıdır. İki sınıflı veri kümeleri için ikili DVM uygulanırken çok sınıflı olan veri kümeleri için DVM 'bire karşı hepsi' stratejisi ile uygulanmıştır. Buna bağlı olarak iki sınıflı veri kümelerinde DVM'nin çalışma süresi özellikle küçük veri kümelerinde az iken, çoklu sınıf içeren veri kümelerinde çalışma maliyeti yüksek çıkmıştır. OYSA ve Rastgele Orman algoritmaları diğer yöntemlere göre yaklaşık 4-5 kat daha yavaş çalışmaktadır. Rastgele Orman algoritmasının 10 tane karar ağacı içerdiği göz önüne alındığında çalışma süresinin uzun olması normaldir ancak test kümesi hatası ve öznitelik sayısı bakımından bir üstünlük elde edemediği için diğer yöntemlere göre yavaş çalışması göz ardı edilememektedir.

Bu çalışmada, sınıflandırma algoritmaları test kümesi hatası, seçilen öznitelik sayısı ve çalışma süresi bakımından karşılaştırıldığında, tüm kriterler bakımından 5-NN en başarılı ve en hızlı sınıflandırıcı olarak çıkmaktadır [2].

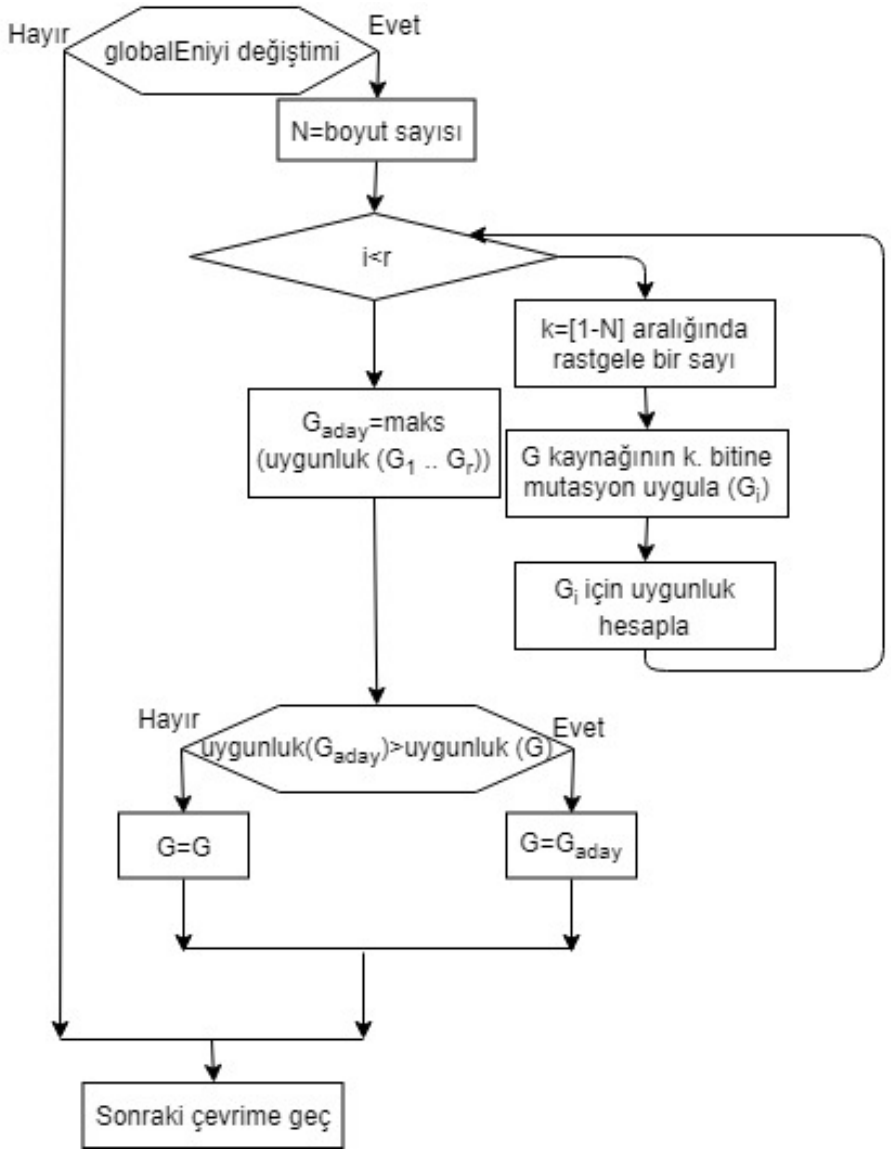
6.3 Koloninin En İyi Çözümüne Lokal Arama Eklenerek Çözümün Optimize Edilmesi

BitABC [19] algoritması, yeni kaynak üretme stratejisi ile iyi bir global arama yapabilmektedir ancak, lokal arama kapasitesi düşüktür. Bu nedenle algoritmanın lokal arama performansını iyileştirebilmek için sezgiselliğini azaltmadan bir lokal arama prosedürü yazılarak ekleme yapılmıştır.

Sürü zekası algoritmalarında, popülasyondaki bireylerden her biri arama uzayındaki olası bir çözümü temsil etmektedir. Bu çözümler içerisinde uygunluk değerine göre en iyi olan çözüm 'globalEniyi' değişkeninde saklanmaktadır. Her bir çevrimin sonunda ise popülasyondaki bireylerde, mevcut globalEniyi den daha iyi bir çözüm olup olmadığı kontrol edilir. Eğer var ise, globalEniyi güncellenir aksi halde bir sonraki çevrime geçer. globalEniyi'nin yakın çevresinde daha iyi bir kaynak bulunabileceği düşüncesinden hareketle, bir lokal arama stratejisi eklenmiştir. Buna göre her çevrimin sonunda popülasyonda mevcut globalEniyi'den daha iyi bir kaynak olup olmadığı kontrol edilir, eğer varsa globalEniyi güncellenir ve yeni globalEniyi'nin etrafında kısıtlı bir lokal arama yapılır. G= [0 0 1 0 1 0], 6 boyutlu uzayda, ilgili çevrimin globalEniyi vektörü olmak üzere lokal arama iki şekilde yapılır.

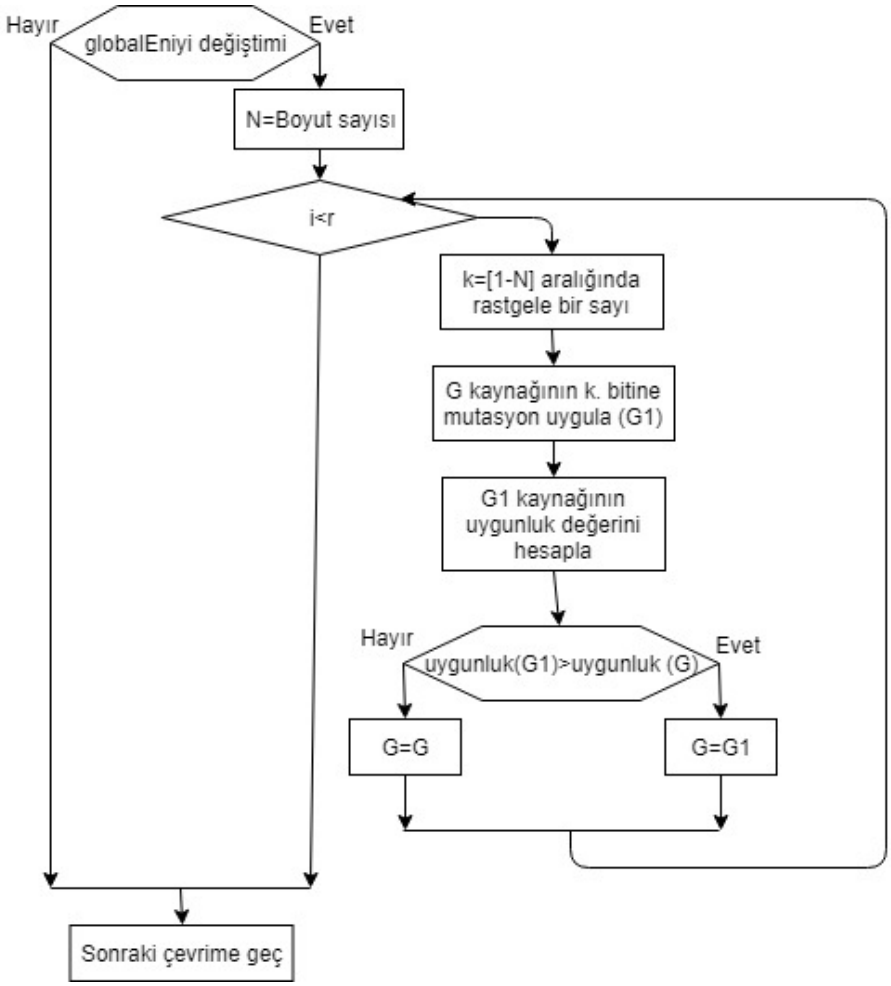
[2] Bu çalışma ISADET 2019 sempozyumunda bildiri olarak kabul edilmiştir

1. **Lokal V1:** G vektörünün rastgele seçilen bir bitine mutasyon uygulanarak r tane aday vektör oluşturulur. Bu vektörlerin uygunluk değeri hesaplanır ve uygunluk değeri en iyi olan kaynak seçilir (G_{max}). $uygunluk(G_{max}) > uygunluk(globalEniyi)$ ise globalEniyi güncellenir. Uygulanan lokal arama işleminin akış diyagramı Şekil 6.3'de gösterilmiştir.



Şekil 6.3 Lokal V1 için akış diyagramı

2. **Lokal V2:** G vektörünün rastgele seçilen bir bitine mutasyon uygulanarak G_1 elde edilir ve uygunluk değeri hesaplanır. Eğer mevcut globalEniyi'den başarılı ise, G_1'in rastgele bir biti mutasyona uğratılarak G_2 elde edilir ve G_1 ile uygunluk değerleri karşılaştırılır. Bu işlem r kez tekrarlanarak her adımda bir önceki kaynağın bir biti değiştirilerek, bir önceki kaynak ile uygunluk değeri bakımından karşılaştırılır. Uygulanan lokal arama işleminin akış diyagramı Şekil 6.4'de gösterilmiştir.



Şekil 6.4 Lokal V2 için akış diyagramı

İki yöntem arasındaki fark; birinci yöntem ile lokal arama süreci tamamlandığında globalEniyi'nin en fazla bir biti değiştirilmiş olur. İkinci yöntemde ise değiştirilen bit sayısı [1-r] arasında olacaktır. Önerilen yöntem Geliştirilmiş BitABC (IBitABC) olarak adlandırılmıştır.

Önerilen yöntemin başarısı, UCI'dan alınan, çeşitli büyüklüklerdeki 10 veri kümesi üzerinden ölçülmüştür. Kullanılan veri kümelerine dair bilgiler Çizelge 6.7'de verilmiştir.

Veri kümeleri, %70 eğitim %30 test olmak üzere iki kümeye bölünmüştür. Kümelerdeki örnekler rastgele seçilmiştir. Sınıflandırma algoritması olarak Çizelge 6.6'daki sonuçlar göz önüne alınarak, 5-NN seçilmiştir. Algoritmanın amaç fonksiyonu çözümlerin uygunluk değeri, sınıflandırıcı doğruluğu ve seçilen öznitelik sayısı göz önüne alınarak Denklem 6.1 ile hesaplanmıştır.

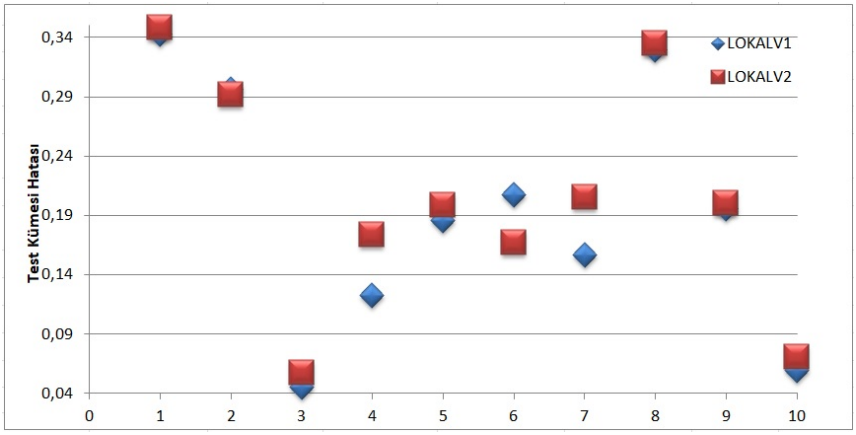
Sürüde 15 işçi arı, 15 gözcü arı olmak üzere 30 arı bulunmaktadır. Algoritmanın maksimum iterasyon sayısı ve geliştirilememe katsayısı 50 olarak belirlenmiştir. Lokal

Çizelge 6.7 Lokal arama için kullanılan veri kümeleri

Veri kümesi	Öznitelik Sayısı	Sınıf Sayısı	Örnek Sayısı
1-Vehicle	18	4	846
2-German	24	2	1000
3-WBCD	30	2	569
4-Ionosphere	34	2	351
5-Musk	166	2	476
6-Urban Land Cover	148	9	168
7-LSVT	309	2	126
8-Arrythmia	279	3	452
9-Madelon	500	2	2600
10-Secom	591	2	1567

arama süreçlerinde kullanılan r değeri 10'dur.

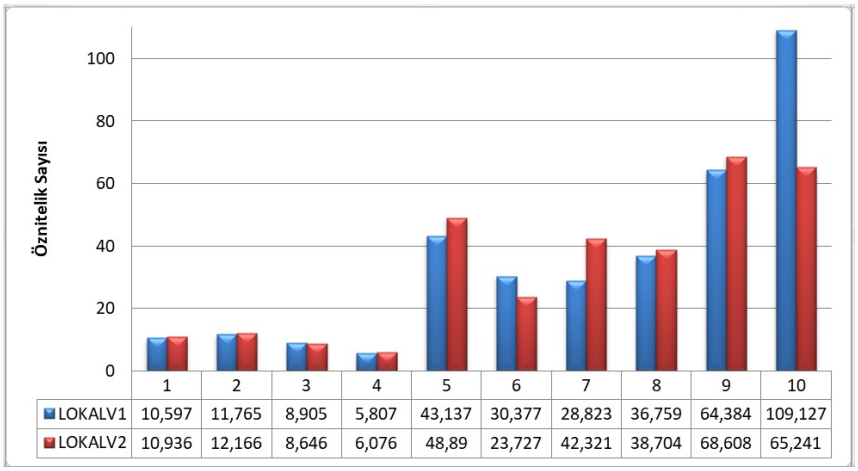
Şekil 6.5 ve Şekil 6.6'de, lokal arama yöntemleri, sırasıyla test kümesi hataları ve seçtikleri öznitelik sayılarına göre karşılaştırılmıştır.



Şekil 6.5 Lokal arama yöntemlerinin test kümesi hatasına göre karşılaştırılması

Lokal arama yöntemleri test kümesi hataları bakımından karşılaştırıldığında Vehicle, German, Arrythmia ve Madelon veri kümeleri için sonuçlar birbirine yakın çıkmıştır. Ancak, 10 veri kümesinden sekiz tanesi için Lokal V1 daha başarılı olmuştur. Lokal V2 ise German ve Urban Land Cover veri kümeleri için Lokal V1'den daha iyi sonuç üretmiş ancak German veri kümesi için iki yöntemin hata değerleri arasındaki fark çok az çıkmıştır.

Yöntemlerin seçtiği öznitelik sayılarına bakıldığında özellikle öznitelik sayısı bakımından küçük olan ilk dört veri kümelesinde sonuçlar çok yakın çıkmış, veri kümelerindeki öznitelik sayısı arttıkça fark daha belirgin hale gelmiştir. Secom veri kümesi için Lokal V2 daha az öznitelik seçmiş olmasına rağmen Şekil 6.5'ye



Şekil 6.6 Lokal arama yöntemlerinin öznitelik sayılarına göre karşılaştırılması

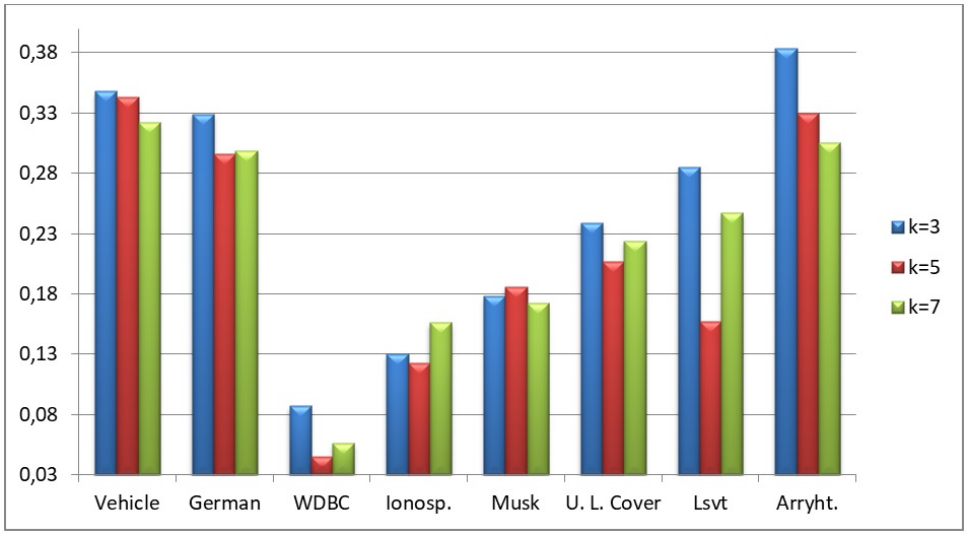
bakıldığında test kümesi hatasınında daha yüksek olduğu görülmektedir. Lokal V1 ise, öznitelik sayısı bakımından Lokal V2'ye göre başarılı olduğu tüm veri kümelerinde test kümesi hatası bakımından da daha başarılı olmuştur. Bu nedenle V1 yönteminin V2 yöntemine göre daha başarılı olduğu sonucuna varılmıştır.

Çizelge 6.8 BitABC vs IBitABC

	Test Kümesi Hatası		Öznitelik Sayısı		CPU Süresi (sn)	
Veri K.	BitABC	IBitABC	BitABC	IBitABC	BitABC	IBitABC
Vehicle	0,324±0,026	**0,277±0,019**	**7,333±1,028**	10,033±1,732	**526,86±2,037**	543,5±6,845
German	**0,261±0,014**	0,309±0,034	**9,667±1,768**	8,5±2,097	561,16±20,417	**522,77±27,29**
WDBC	0,041±0,013	**0,036±0,009**	7,8±1,669	**6,967±1,938**	**402,55±4,826**	490,69±6,45
Ionosphere	0,113±0,023	**0,086±0,037**	**4,333±0,661**	4,767±1,04	475,23±3,813	**463,52±3,88**
Musk	**0,151±0,03**	0,163±0,023	34,533±8,629	**35,767±7,37**	471,41±9,717	**458,99±6,76**
U. L. Cover	0,201±0,021	**0,182±0,025**	25,6±5,829	**23,53±6,847**	**501,07±5,424**	509,18±6,6
LSVT	0,213±0,068	**0,123±0,034**	31,367±10,414	**21,267±27,556**	454,67±34,15	**433,58±4,464**
Arryhtmia	0,302±0,031	**0,303±0,027**	31,767±7,271	**29,867±6,892**	504,97±14,496	**483,93±23,894**
Madelon	0,188±0,022	**0,181±0,015**	51,8±14,25	**51,967±12,417**	**2061,5±126,99**	2505,3±177,87
Secom	0,057±0,005	**0,055±0,004**	**77,333±16,253**	86,167±22,96	1411,3±85,881	**1401,4±90,857**

BitABC algoritmasına eklenen lokal arama stratejisinin, BitABC'nin performansına etkisini gözlemlemek için, orjinal BitABC ile Lokal V1 fonksiyonu eklenmiş IBitABC yöntemleri Çizelge 6.8'de test kümesi hataları, seçtikleri öznitelik sayıları ve çalışma süreleri bakımından karşılaştırılmıştır. Buna göre globalEniyi'nin güncellenmesine bağlı olarak işlem yapan lokal arama stratejisi sekiz veri kümesi için daha az hata ile sınıflandırma yapabilmiştir. Bu sekiz veri kümesinden beş tanesi için IBitABC, BitABC'ye göre daha az öznitelik seçmiştir. Yani önerilen yöntem doğru öznitelikleri seçmekte BitABC'den daha iyi olduğunu göstermiştir. Ayrıca, IBitABC'nin hesapsal maliyetinin BitABC'den çoğunlukla daha yüksek olmadığı da yine Çizelge 6.8'de görülmektedir.

K-NN sınıflandırma algoritması için 'k' kritik bir parametredir ve her bir örnek için

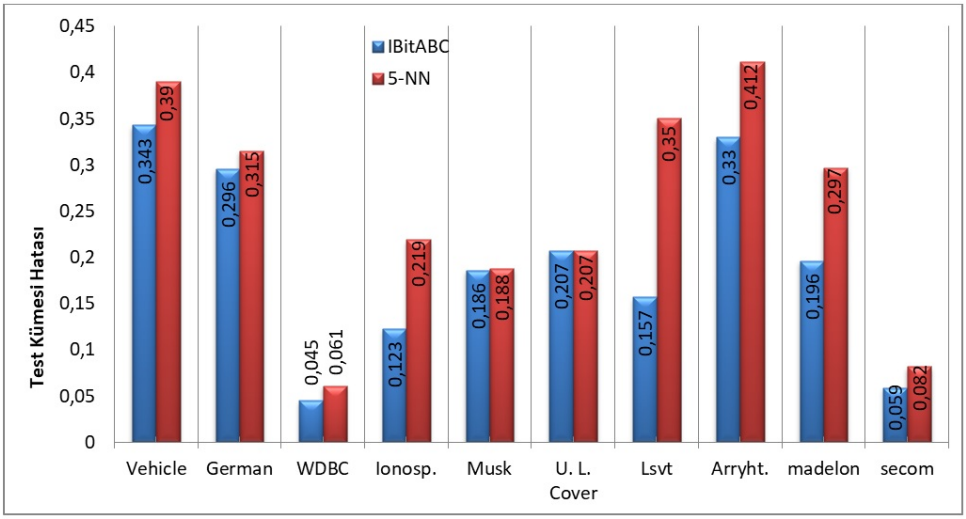


Şekil 6.7 K-NN k parametresi için test kümesi hatası

kendisine en yakın kaç örneği göz önüne alacağını belirler. Bu değerin gereğinden yüksek olması algoritmanın hesapsal maliyetini artıracağı gibi aykırı değerlere açık hale gelmesine de neden olacaktır. k değerinin gerekenden küçük olması ise, algoritmanın doğru karar verme performansını etkileyecektir. Önerilen yöntem ve veri kümeleri için doğru k değerini belirleyebilmek adına K-NN, IBitABC'ye farklı k değerleri ile uygulanmış ve sonuçlar Şekil 6.7'da test kümesi hatalarına göre sekiz veri kümesi ile çalıştırılarak karşılaştırılmıştır. k'nın değerinin 3 olduğu durumda algoritmanın doğru sınıflandırma performansı düşmüştür. k'nın değerinin 7 olduğu durumda ise az sayıda veri kümesinde başarı elde edilebilmiş ancak, k'nın değeri 5 olarak belirlendiğinde başarılı olan veri kümesi sayısı artmıştır. Buna göre sınıflandırıcı meta parametresinin en uygun değerinin 5 olduğu görülmüştür.

Bir öznitelik seçim algoritmasından beklenen, veri kümesi içinden ayırt edici öznitelikleri seçerek, sınıflandırma performansından taviz vermeden öznitelik sayısını azaltmasıdır. Önerilen yöntemin amaca uygunluğunu araştırmak için 5-NN sınıflandırıcısı herhangi bir öznitelik seçimi işlemine tabi olmadan veri kümelerine uygulanarak, elde edilen test kümesi hataları karşılaştırılmış ve sonuçlar Şekil 6.8'de verilmiştir. Buna göre IBitABC'nin sınıflandırma performansı, hiçbir öznitelik seçimi yapılmadan uygulanan 5-NN algoritmasından, 10 veri kümesinde dokuzu için daha iyi olup, bir tanesi için de eş değer düzeydedir. Öznitelik seçimi işleminin daha az öznitelik ile en azından eş değer başarı elde etmek olduğu göz önüne alındığında boyut azaltma işleminin gerekliliği ve IBitABC'nin ayırt edici öznitelikleri başarılı bir şekilde seçebildiği görülmektedir.

Sonuç olarak öznitelik seçiminde başarılı olduğu Bölüm 6.1'de görülmüş olan BitABC algoritmasına, bir lokal arama prosedürü eklenerek mevcut yöntem geliştirilmiş ve



Şekil 6.8 Öznitelik seçimi işleminin sınıflandırıcı doğruluğuna etkisi

IBitABC olarak isimlendirilmiştir. IBitABC'nin yetkinliği, 10 veri kümesi ve, Bölüm 6.2'de başarılı olduğu ispatlanan K-NN sınıflandırma algoritması ile ölçülmüş ve BitABC'den daha iyi sonuçlar verdiği görülmüştür. K-NN için uygun k değerini belirlemek için, önerilen yöntem farklı k değerleri için çalıştırılmış ve en efektif sonuçlar k'nın 5 olduğu durumda elde edilmiştir. Ayrıca, IBitABC'nin ayırt edici öznitelikleri seçme kabiliyetini ölçmek için önerilen yöntem, öznitelik seçimi yapılmadan elde edilen sınıflandırıcı hatalarına göre karşılaştırılmış ve alınan sonuçlar doğru öznitelik kümelerini seçebildiğini göstermiştir [3].

6.4 Lokal Arama Tabanlı İkili Yapay Arı Kolonisi Algoritması

Yapay Arı Kolonisi Algoritmasının ikili bir versiyonunu geliştirmek için algoritmanın komşu kaynak üretme stratejisi revize edilmiştir. Buna göre kaynaklar ikili vektörler şeklinde ilklendirilerek mevcut aday kaynak üretme stratejisine, Hamming mesafe ölçümü tabanlı bir yöntemle lokal arama eklenmiş böylece ikili vektörler ile işlem yapabilecek şekilde değiştirilmiştir. Popülasyonun yeniden üretilmesini sağlayan bir strateji ile de sürünün çeşitliliği artırılarak erken yakınsama önlenmiştir. Önerilen yöntem Genişletilmiş BitABC (exBitABC) olarak isimlendirilmiştir.

6.4.1 Önerilen Yaklaşım

YAK Algoritması ilklendirme aşamasında, optimize edilecek parametrelerin minimum ve maksimum değerlerini baz alarak, vektörler üzerinde aritmetik işlemler ile ilk kaynakları belirlemektedir (Denklem 4.1). Önerilen yöntemde, vektörler Denklem

[3] Bu çalışma UBMK 2017 sempozyumunda bildiri olarak sunulmuştur

6.3 ile ilklendirilerek ikili vektörler elde edilmiştir.

$$x_{ij} = \begin{cases} 1 & \text{eğer } rand(0,1) \geq esik \\ 0 & \text{eğer } rand(0,1) < esik \end{cases} \tag{6.3}$$

YAK algoritmasında, i. iterasyonda arama uzayında aday çözüm üretmek için, i. kaynak olan x_i, popülasyondan rastgele seçilmiş bir komşu kaynak (x_k) ve [-1,1] aralığında rastgele bir değer olan ϕ_i kullanılarak üretilmektedir. Böylece yeni üretilen kaynak, arama uzayında x_i ile x_k arasında bir konumda olmaktadır. Yeni üretilen kaynağın x_i veya x_k'dan hangisine ne kadar yakın olacağını ise ϕ_i değeri belirlemektedir. Yeni üretilen komşu kaynağın i. kaynağın uzağındaki bir konuma denk gelmesi, ilgili kaynağın bulunduğu lokal bölgenin yeterince taranmadan kaybedilmesine neden olacaktır. Nektar bulunma potansiyeli olan bir bölgeyi kaybetmek ise algoritmanın lokal arama yeteneğini azaltacaktır. Arama uzayının küçük olduğu durumlarda bu bir sorun teşkil etmeyecektir. Ancak amacımız büyük boyutlu veride öznitelik seçimi yapmak olduğu için arama uzayı da büyük olacaktır. Algoritmanın lokal arama kapasitesini artırmak için yeni üretilecek kaynağın, mevcut kaynağa olabilecek mesafesi sınırlandırılmıştır. Böylece kaynağın bulunduğu bölgede daha iyi bir arama yapılması sağlanmıştır. Kaynakların konumları ikili vektörler ile temsil edildiğinden, aday kaynak ile mevcut kaynak arasındaki mesafeyi ölçmek için Hamming mesafe ölçüm yöntemi kullanılmıştır. Ayrıca, ikili vektörler arasında artimetik işlem yapmak çok anlamlı olmayacağından BitABC [19]'de olduğu gibi bit işlem operatörlerinden faydalanılmıştır. ϕ değeri de [-1,1] aralığında rastgele bir reel sayı yerine, rastgele değerlerden oluşturulmuş, kaynak vektörler ile aynı boyutta, ikili bir vektör olarak belirlenmiştir. Bu vektör her yeni kaynak üretiminde rastgele değerler ile yeniden üretilmektedir.

ϕ_1 ve ϕ_2 rastgele üretilmiş ikili vektörler olup, x_i mevcut kaynak, x_k popülasyondan rastgele seçilmiş komşu kaynak olmak üzere, aday kaynak v_i Denklem 6.4 ile üretilmektedir.

$$v_i = \phi_2 AND(\phi_1 AND(x_i OR x_k)) \tag{6.4}$$

Denklemde 'OR' bit işlem operatörü yeni üretilecek vektörde seçilecek öznitelikleri çeşitlendirmek için kullanılmıştır. 'AND' operatörü ile de seçilen öznitelik sayısını azaltmak amaçlanmıştır. Böylece x_i veya x_k'da '1' olan parametreler 'OR' operatörü ile alınacak, ardından 'AND' operatörü ile '1 lerin sayısı azaltılırken ϕ vektörleri kullanılacağından farklı öznitelik kombinasyonları elde edilmiş olacaktır. Aday

kaynağın mevcut kaynağa belirli bir mesafede olmasını sağlamak için ise ϕ_1 vektörü mevcut kaynaktan d birim mesafede ve ϕ_2 vektörüde ϕ_1 vektöründen d birim mesafede olacak şekilde rastgele üretilmektedir.

Örneğin

- Mevcut kaynak: $x_i = \{1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\}$,
- Komşu kaynak: $x_k = \{1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\}$ ve
- Mevcut ve komşu kaynak arasındaki mesafe dH(x_i , x_k) = 3 ve d=5 olmak üzere

$$y = x_i OR x_k = \{101111000110\} \tag{6.5}$$

Denklem (6.6) ile seçilen özniteliklerin çeşitliliği artırılmaktadır. Ancak, amacımız x_i yakınlarında daha seyrek bir vektör elde etmektir. Bu nedenle x_i kaynağından rastgele seçilen d sayıda bit değiştirilerek ϕ_1 vektörü elde edilir. d = 5 için rastgele seçilen bitler: {1,8,9,11,12} ise ϕ_1 ={0 0 1 1 1 1 0 1 1 0 1 1} olacaktır. Buna göre y_2 Denklem (6.6) ile hesaplanır:

$$y_2 = \phi_1 AND(x_i OR x_k) = \{001111000000\} \tag{6.6}$$

dH(x_i,y_2) = 1 < d. Böylece mevcut kaynağa 1 birim mesafede yeni bir kaynak elde etmiş olduk. Yeni kaynağın ayrıklığını artırabilmek için ϕ_1 vektöründen yine d birim mesafede bir ϕ_2 vektörü elde edilir. d = 5 ve rastgele seçilen bitler: {2,4,5,6,11} ise ϕ_2={0 1 1 0 0 0 0 1 1 0 0 1} olacaktır. Böylece Denklem (6.4) ile aday kaynak v_i= {0 0 1 0 0 0 0 0 0 0 0 0} olacak şekilde üretilmiş olur. v_i ile x_i arasındaki mesafe Gauss benzeri dağılmış [1-d] aralığında bir tamsayı olacaktır.

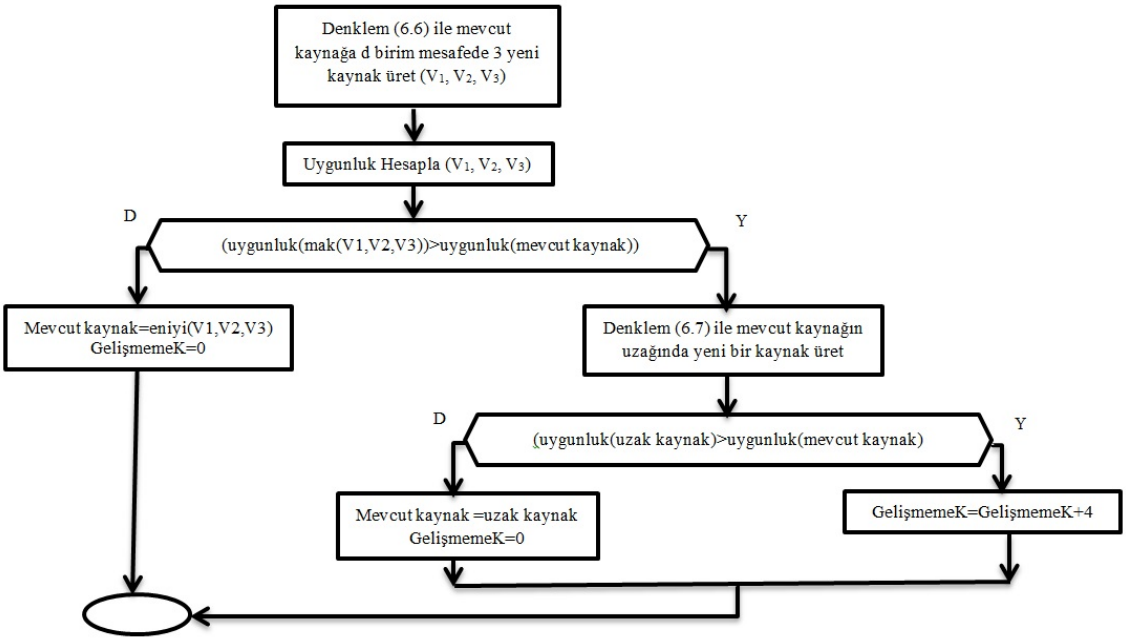
Bir öznitelik seçimi probleminde, ayırt edici öznitelik alt gruplarının belirlenmesi önemlidir. İşçi ve gözcü arı safhalarında Denklem (6.4) ile önceki ve şimdiki kaynaklar arasında küçük değişiklikler olması sağlandı. Böylece, algoritma, bir araya getirildiğinde oldukça ayırt edici olan öznitelikleri tanımlayabilmektedir.

Algoritmanın mevcut kaynağın lokal bölgesinde daha güçlü bir arama yapabilmesi için işçi arı safhasında, her bir işçi arı için Denklem (6.4) ile kendilerine d birim mesafede 3 tane yeni kaynak üretilir. Bu kaynakların uygunluk değerleri hesaplanarak işçi arının mevcut kaynaktan daha iyi bir kaynak bulup bulmadığı kontrol edilir. Eğer daha iyi bir kaynak bulunduysa mevcut kaynak güncellenir. Aksi halde mevcut kaynağın

kendi lokal bölgesi için optimal olduğu varsayılarak lokal bölgeyi değiştirmek adına BitABC'nin yeni kaynak üretme stratejisi Denklem (6.7) ile bir aday kaynak üretilir. İlgili denklem üretilen aday kaynağın mevcut kaynağa uzak bir mesafede olmasını garanti etmektedir.

$$v_i = x_i XOR(\phi_i AND(x_i OR x_k)) \tag{6.7}$$

Şekil 6.9'de exBitABC'nin yeni kaynak üretme stratejisi kısaca modellenmiştir. Denklem (6.7) ile üretilen lokal bölgenin uzağına gitmeyi sağlayacak aday kaynak da mevcut kaynağı iyileştiremediyse mevcut kaynağın geliştirilememe katsayısı dört artırılır çünkü dört defa iyileştirilmeye çalışılmıştır.



Şekil 6.9 exBitABC yeni kaynak üretme stratejisi

İlklendirme aşamasında YAK algoritması kaynakları arama uzayında rastgele konumlara yerleştirmektedir. İlklendirme sırasında kaynakların aynı olması bir şekilde önlense bile, birbirine çok yakın konumlara denk gelebilirler. Amacımız her bir arının kendi kaynağının bulunduğu bölgede lokal arama yapmasını sağlamak olduğu için, kaynakların birbirine çok yakın olarak ilklendirilmesi, arıların çakışık bölgelerde arama yapmasına neden olacaktır. Bu durumu önleyebilmek için arama uzayı bölgelere ayrılmış ve böylece her bir bölge için bir arı görevlendirilmiştir. Öznitelik seçim problemi için arama uzayının boyutunun öznitelik sayısı olduğu göz önüne alındığında herhangi iki kaynağın birbirine olan mesafesi en az Denklem (6.8)'deki kadar olabilir.

$$minMesafe = \frac{\text{ÖznitelikSayısı}}{\text{KaynakSayısı}} \tag{6.8}$$

Gözcü arı aşamasında kullanılan Rulet-Tekeri yöntemi, arıların iyi kaynaklara yönelmesini sağladığından sürüdeki çeşitliliği azaltabileceği gibi arıların çakışık bölgelerde arama yapmasına da neden olacaktır. Bu durumu önlemek için popülasyonun yeniden üretilmesini sağlayan bir strateji eklenmiştir. Buna göre her bir iterasyonun sonunda kaynaklar arasındaki Hamming mesafesi hesaplanır. Birbirine en yakın olan iki kaynağın mesafesi Denklem (6.8) ile belirlenen 'minMesafe' değerinden küçükse, bu kaynaklardan uygunluk değeri daha düşük olan yeniden ilklendirilir. Bu işlem her iterasyonun sonunda sadece bir kaynak çifti için yapılmaktadır. Bunun nedeni geliştirilme potansiyeli olan kaynakları kaybetmemektir. Bu şekilde, erken yakınsama önlenir ve çeşitlilik artırılır. Ayrıca, algoritmaya eklenen bu yenilikler ile küresel arama kapasitesi azaltılmadan daha kapsamlı yerel arama yapılması sağlanmıştır.

6.4.2 Deneysel Çalışmalar

Önerilen yöntemin geçerliliğini test etmek için UCI makine öğrenmesi havuzundan farklı büyüklüklerde 14 veri kümesi alınmıştır. Veri kümelerine dair bilgiler Çizelge 6.9'da verilmiştir. Önerilen yöntem (exBitABC) iyi bilinen sürü zekası yöntemlerinden İkili Parçacık Sürü Optimizasyonu [68] ve Genetik Algoritma [66] ile karşılaştırılmıştır. Ayrıca, sonuçlar orjinal BitABC [19] algoritmasının sonuçları ile de karşılaştırılmıştır. Önceki çalışmamızda [87] literatürde yer alan diğer birçok ikili YAK algoritmasını uygulamış olduğumuzdan dolayı bu bölümde tekrardan çalıştırma gereği duymadık. Öznitelik seçimi işleminin sınıflandırma başarısına olan etkisini kanıtlamak için, veri kümeleri herhangi bir öznitelik seçimi işlemine tabi tutulmadan 5-NN ile sınıflandırılmıştır. Deneysel çalışmalarda kullanılan veri kümelerinin öznitelik sayısı, sınıf sayısı ve örnek sayısı bilgileri Çizelge 6.9'da verilmiştir. Veri kümeleri rastgele örnekler içerecek şekilde üç parçaya bölünmüştür. Buna göre veri kümelerinin %60'ı eğitim, %20'si doğrulama ve %20'si de test olarak kullanılmıştır. İterasyonlar sırasında seçilen özniteliklerin ayırt ediciliği, doğrulama kümesi ile ölçülmüş, algoritma sonlandığında ise test kümesi ile başarısı karşılaştırılmıştır.

İlklendirme aşamasında kullanılan eşik değeri 0,85 olarak belirlenmiştir. Yani, kaynak vektörde bir bitin '0' olma olasılığı 0,85'dir. Uygulanan algoritmalar sezgisel olduğu için, her bir algoritma her bir veri kümesinde 30 defa çalıştırılmış ve ortalaması alınmıştır. exBitABC'de lokal mesafeyi belirleyen d parametresi 15 olarak belirlenmiştir. Kaşif arı aşamasında kullanılan geliştirilememe katsayısı da 50'dir.

Çizelge 6.9 exBitABC için kullanılan veri kümeleri

Veri Kümesi	#Örnek	#Sınıf	#Öznitelik
Hill-Valley	100	2	606
Urban Land Cover	148	9	168
Musk-I	166	2	476
Arrythmia	279	16	452
LSVT	309	2	126
Madelon	500	2	2600
Secom	591	2	1567
Malacious	513	2	373
Isolet5	617	26	1559
Multiple Features	649	6	2000
CNAE	857	9	1080
Micromass	1300	9	931
Gisette	5000	2	6000
Arcene	10000	2	200

BinPSO için parametreler [88]'de önerildiği gibi belirlenmiştir. Buna göre öğrenme faktörü c1 ve c2'nin değeri 2, başlangıç ağırlığı (w): 0,9, maksimum hız (V_{max}):0,6'dır. Genetik Algoritma için ise [89]'de önerildiği üzere çaprazlama ve mutasyon oranları sırasıyla 0,8 ve 0,2 olarak belirlenmiştir. BinPSO ve GA algoritmaları [88] ve [89]'de önerilen şekliyle uygulanmıştır. Algoritmalarda uygunluk fonksiyonu Denklem (6.1) ve (6.2)'ye göre hesaplanmıştır.

Popülasyon büyüklüğü, sürü zekası algoritmalarının önemli parametrelerinden biridir ve genelde problemin karmaşıklığına dayalı olarak belirlenir. İlgili parametreyi deneysel olarak belirleyebilmek için, popülasyonun 30, 40 ve 50 olduğu durumlarda tüketilen uygunluk sayısına göre test kümesi hatası Şekil 6.10'da verilmiştir. Genellikle popülasyon sayısının artmasının hatayı azaltacağı varsayılsa da sonuçlar durumun öyle olmadığını göstermektedir. Tüm veri kümelerinde, yaklaşık 1000 uygunluk değerinin tüketilmesiyle algoritma kararlı bir yapıya gelmiştir. Grafikten de görüldüğü gibi, 40 arı genellikle veri kümeleri için yeterli olmuştur. Arama uzayı büyüdükçe, optimize çözümün bulunması zorlaşacağından, popülasyonu artırma gerekliliği görülebilir ancak, uygun sayıda bireye ulaştıktan sonra popülasyonun büyütülmesi çalışma süresinin artmasına neden olacaktır. LSVT ve Secom veri kümelerinde popülasyonun 30 ve 40 olduğu durumlarda, Malacious ve Multiple Features veri kümelerinde ise popülasyonun 40 ve 50 olduğu durumlarda eğriler çakışmıştır. Genel olarak bakıldığında 30'un az, 50'nin de fazla olduğu görülmüş ve popülasyon büyüklüğü 40 olarak belirlenmiştir. YAK algoritmasında kaynaklar bir iterasyonun sonuna geldiğinde her bir kaynak hem işçi arı hem de gözcü arı aşamasında

geliştirilmeye çalışılmış olur. Bununla birlikte GA ve PSO'da, her kaynak bir döngüde sadece bir fazda geliştirilebilir. Bu nedenle exBitABC ve BitABC algoritmalarında popülasyonda 20'şer işçi ve gözcü arı, GA'da 40 birey ve BinPSO'da 40 parçacık bulunmaktadır.

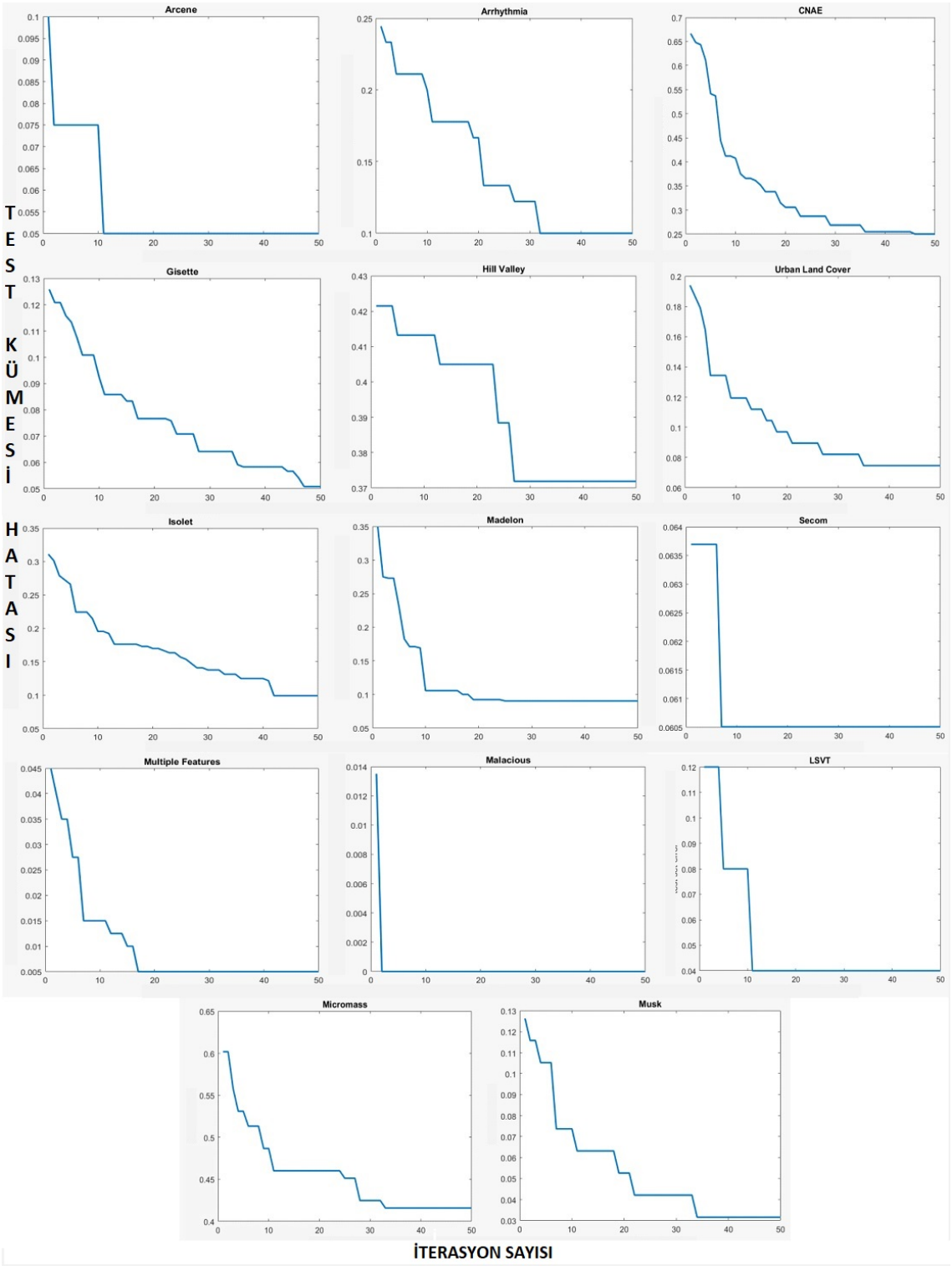
Farklı algoritmaları kıyaslarken maksimum iterasyon sayısı gibi bazı parametrelerin aynı olması gerekmektedir. Ancak, exBitABC'de her bir kaynak için birden fazla uygunluk fonksiyonu tüketilmektedir. Bir birey için tek zamanda birden fazla uygunluk fonksiyonunun tüketildiği algoritmalarda, adil bir karşılaştırma yapabilmek için durma kriteri olarak iterasyon sayısı yerine algoritmanın çalışma süresi veya tüketilen uygunluk fonksiyonu sayısının kullanılması önerilmiştir [90], [91], [92]. Arka planda çalışan uygulamalar, güncellemeler gibi algoritmanın çalışma süresine etki edecek faktörler göz önüne alındığında, durma kriteri olarak tüketilen uygunluk fonksiyonu sayısının kullanılması daha uygun görülmüştür. exBitABC algoritmasının her bir veri kümesi için, 50 iterasyon boyunca tükettiği uygunluk fonksiyonu sayısı hesaplanmış ve BitABC, GA ve BinPSO için durma kriteri olarak bu değerler kullanılmıştır.

YAK iteratif bir algortimadır ve algoritma optimum sonuca yaklaştıkça, amaç fonksiyonunun değeri sabit kalmaya başlar ve algoritma ilerlemeyi durdurur. Bu noktadan sonra iterasyonlara devam etmek sadece hesapsal maliyeti artıracaktır. exBitABC için maksimum iterasyon sayısı deneysel olarak belirlenmiştir. Algoritmanın iterasyonlar boyunca elde ettiği test kümesi hatası Şekil 6.11'de verilmiştir. Grafikten de açıkça görüldüğü üzere 50 iterasyon, algoritmanın kararlı bir yapıya ulaşması için yeterlidir.

On dört veri kümesi üzerinde exBitABC tarafından elde edilen sonuçlar BitABC, GA ve BinPSO ile test kümesi hatası ve seçtiği öznitelik sayısı bakımından Çizelge 6.10'da karşılaştırmalı olarak standart sapma değerleri ile birlikte verilmiştir. Son sütunda ise veri kümeleri, herhangi bir öznitelik seçim işlemine tabi tutulmadan 5-NN ile sınıflandırılmış ve test kümesi hatası verilmiştir. Başarılı olan yöntem koyu yazılırken, aynı sınıflandırma hatasına sahip yöntemlerde öznitelik sayısı az olan daha başarılı sayılmıştır. 'WT' satırı, algoritmaların elde ettiği sonuçların birbirlerine göre istatistiksel olarak anlamlı olup olmadığını göstermektedir. İstatistiksel anlamlılığı ölçmek için Wilcoxon Rank Sum Test [81] kullanılmıştır. "+" ve "—" işaretleri sırasıyla, exBitABC'nin sonuçlarının, ilgili algoritmadan önemli ölçüde daha iyi veya daha kötü olduğunu göstermektedir. "=" ise sonuçlar arasında istatistiksel bir fark olmadığı anlamına gelmektedir.

Çizelge 6.10'dan görüldüğü gibi, Isolet veri kümesi hariç, exBitABC diğer

Şekil 6.10 Tüketilen uygunluk fonksiyonuna göre yakınsama grafiği



Şekil 6.11 Iterasyon sayısına göre yakınsama grafiği

Çizelge 6.10 exBitABC için karşılaştırmalı performans sonuçları

		BitABC	GA	BinPSO	exBitABC	5-NN
Hill-Valley	Test H.	0,481±0,044	0,482±0,035	0,495±0,004	**0,462±0,034**	0,44±0,034
	Oz.Say	19,83±3,18	20,7±4,09	14,37±13,45	**8,9±3,45**	
(100)	WT	+	+	+		
Urban L.C.	Test H.	0,188±0,026	0,202±0,042	0,191±0,0042	**0,147±0,03**	0,192±0,025
	Oz.Say	43,23±29,91	30,07±56,62	48,93±13,8	**10,2±2,22**	
(148)	WT	+	+	+		
Musk-I	Test H.	0,191±0,042	0,179±0,047	0,173±0,042	**0,155±0,047**	0,147±0,021
	Oz.Say	21,37±4,05	38,5±7,29	57,2±12,93	**12,67±2,22**	
(166)	WT	-	-	-		
Arrythmia	Test H.	0,337±0,047	0,33±0,04	0,351±0,045	**0,296±0,04**	0,375±0,035
	Oz.Say	27,67±6,9	51,97±7,66	66,37±29,87	**10,2±2,22**	
(279)	WT	+	+	+		
LSVT	Test H.	0,247±0,09	0,26±0,09	0,24±0,08	**0,201±0,102**	0,33±0,062
	Oz.Say	30,7±6,92	49,1±8,1	83,73±44,14	**19,73±6,57**	
(309)	WT	+	+	-		
Madelon	Test H.	0,2±0,021	0,218±0,027	0,26±0,029	**0,1±0,011**	0,271±0,014
	Oz.Say	51,07±14,16	86,57±9,05	155,8±37,3	**11,3±2,13**	
(500)	WT	+	+	+		
Secom	Test H.	0,07±0,004	0,071±0,004	0,069±0,003	**0,069±0,004**	0,071±0,01
	Oz.Say	72±11,42	90,93±9,92	170,9±81,74	**41,77±21,57**	
(591)	WT	=	=	-		
Malacious	Test H.	0,025±0,017	**0,022±0,015**	0,03±0,023	0,023±0,017	0,007±0,007
	Oz.Say	70,6±9,77	75,83±7,46	125,3±77,9	**24,63±1,35**	
(513)	WT	=	=	+		
Isolet5	Test H.	**0,196±0,023**	0,221±0,025	0,242±0,024	0,211±0,022	0,277±0,018
	Oz.Say	50,6±5,9	109,23±11,59	192,73±50,95	**44,9±9,87**	
(617)	WT	-	+	+		
Mul. Feat.	Test H.	0,032±0,009	0,035±0,01	0,046±0,01	**0,03±0,01**	0,023±0,004
	Oz.Say	60,67±7,95	109,17±11,73	236,5±35,88	**44,87±14,89**	
(649)	WT	-	=	+		
Cnae	Test H.	0,31±0,041	0,309±0,045	0,285±0,047	**0,212±0,034**	0,129±0,015
	Oz.Say	67,53±10,39	157,47±8,5	350,93±25,98	**44,83±7,99**	
(857)	WT	+	+	+		
Micromass	Test H.	0,617±0,067	0,719±0,047	0,71±0,041	**0,587±0,054**	0,688±0,033
	Oz.Say	58,03±11,79	207,03±14,18	164,57±169,95	**42,9±10,02**	
(1300)	WT	+	+	+		
Gisette	Test H.	0,073±0,011	0,102±0,011	**0,69±0,007**	0,071±0,009	0,066±0,004
	Oz.Say	232,17±44,053	761,03±20,28	2104,2±166,21	**146,53±20,48**	
(5000)	WT	=	+	=		
Arcene	Test H.	0,247±0,06	0,243±0,072	0,262±0,063	**0,182±0,076**	0,229±0,053
	Oz.Say	924,5±139,33	1504,4±36,13	3233,4±904,37	**776,4±37,68**	
(10000)	WT	+	+	+		

yöntemlerden hem test kümesi hatası hem de seçtiği öznitelik sayısı bakımından daha iyi performans göstermiştir. Secom, Malacious, Multiple Features ve Gisette veri kümelerini tüm yöntemler çok küçük hatalar ile sınıflandırmıştır. Bu nedenle bu veri kümelerinde asıl başarı seçilen öznitelik sayısına göre değerlendirilmiştir ve exBitABC diğer yöntemler ile çok yakın hata değerlerini çok daha az öznitelik ile elde etmiştir. Veri kümeleri içerisinde Hill-Valley ve Micromass çalışılması zor veri kümeleridir. Çizelge 6.10'dan görüldüğü üzere, tüm öznitelikler kullanılsa dahi hata oranı yüksektir. Diğer yöntemlerin aksine exBitABC, Micromass için çok daha az öznitelik ile tüm özniteliklerin olduğu durumdan daha iyi bir performans sergilemiştir. Sonuçlar genel olarak 5-NN ile karşılaştırıldığında, öznitelik seçimi ile daha düşük veya benzer hata oranları ile sınıflandırma yapılabileceği görülmektedir. exBitABC göz ardı edildiğinde, BitABC'nin GA ve BinPSO'dan daha iyi sonuçlar elde ettiği açıkça görülmektedir. İstatistiksel analiz, sadece test kümesi hataları için uygulanmıştır. Buna göre Musk ve Secom için exBitABC istatistiksel olarak anlamlı bir fark elde edemezken, diğer veri kümelerinde genel olarak farkın anlamlı olduğu görülmektedir.

Çizelge 6.11 Algoritmaların saniye cinsinden hesapsal maliyeti

Veri kümesi	BitABC	BinPSO	GA	exBitABC
Hill-Valley	45,065 ± 0,71	**43,57 ± 0,17**	54,1 ± 0,71	48,92 ± 0,55
Urban L.C.	**46,64 ± 0,76**	48,49 ± 0,36	57,17 ± 0,75	49,91 ± 0,43
Musk-I	**42,47 ± 0,31**	45,047 ± 0,3	54,21 ± 0,85	47,32 ± 0,24
Arrythmia	**43 ± 0,62**	48,002 ± 0,28	54,68 ± 0,32	46,67 ± 0,36
LSVT	40,2 ± 0,22	**40,12 ± 0,22**	54,54 ± 4,67	45,23 ± 0,17
Madelon	101,3 ± 9,16	198,47 ± 7,44	140,33 ± 6,45	**76,61 ± 1,85**
Secom	82,21 ± 1,73	118,38 ± 5,46	97,08 ± 9,59	**80,31 ± 1,42**
Malacious	**42,64 ± 0,52**	51,21 ± 0,58	54,5 ± 0,4	48,22 ± 0,38
Isolet5	106,37 ± 25,64	174,96 ± 3,42	113,58 ± 4,38	**95,48 ± 6,27**
Mult. Feat.	147,98 ± 6,68	223,33 ± 70,36	146,07 ± 9,26	**118,04 ± 6,28**
CNAE	66,85 ± 3,38	117,13 ± 4,94	84,13 ± 3,49	**59,54 ± 0,67**
Micromass	58,86 ± 1,97	86,21 ± 2,45	65,66 ± 0,81	**58,38 ± 1,35**
Gisette	2741,3 ± 101,7	7931,7 ± 261,7	4431,4 ± 88,3	**1227,9 ± 12,3**
Arcene	100,79 ± 3,72	154,83 ± 2,65	144,62 ± 1,36	**99,06 ± 1,46**

Uygulanan algoritmaların hesapsal maliyetlerine göre karşılaştırmalı sonuçları da Çizelge 6.11'de verilmiştir. Algoritmalar 32 GB RAM ve Intel Xeon E5 2.6 GHz işlemci içeren bir server bilgisayarda Matlab 2016b ile uygulanmıştır. exBitABC sekiz veri kümesinde diğer yüntemlerden daha hızlı sonuç vermiştir. Hesaplama maliyeti bakımından veri kümesinin büyüklüğü, hücre sayısına göre değerlendirilir (#öznitelik x #örnek). Genel olarak küçük veri kümeleri için çalışma süreleri bakımından birkaç saniyelik farklar oluşurken, veri kümesi büyüdüğünde farkların daha belirgin olduğu görülmektedir. exBitABC'nin büyük veri kümelerinde daha hızlı optimizasyon

yapabildiği de açıkça görülmektedir.

Bu çalışma ile öznitelik seçimi problemi için yeni bir ikili YAK algoritması önerilmiştir. Algoritma, sezgiselliğini kaybetmeden yerel arama kapasitesini artıracak şekilde tasarlanmıştır. Önerilen yöntemin etkinliğini göstermek için, UCI Makine Öğrenmesi Havuzundan seçilen 14 veri kümesi üzerinde BinPSO, GA, BitABC ve exBitABC algoritmaları ile karşılaştırılmıştır. Veri kümelerinin çoğu için exBitABC, test kümesi hatası açısından BitABC'den daha iyi sonuçlar üretmiş ve daha kısa sürede uygun alt kümeyi belirlemiştir. Ayrıca, exBitABC'nin daha az sayıda ama ayırt ediciliği daha yüksek öznitelikleri seçebildiği de sonuçlardan görülmektedir. BinPSO ve GA ise iyi sonuçlar elde edememiştir. Bu sonuçlar, exBitABC algoritmasının yerel arama yeteneğinin artmasının test kümesi hatası, öznitelik sayısı açısından algoritmanın performansını iyileştirdiğini göstermektedir.

7
Olasılıksal İkili Yapay Arı Kolonisi Algoritması (PrBABC)

Binlerce gen içeren mikrodizi veri kümelerinden belirli bir hastalık ile ilgili ayırt edici genleri bulmak için olasılıksal bir Yapay Arı Kolonisi Algoritması önerilmiştir. mikrodizi verileri normalize edildikten sonra veri kümelerinde filtreleme yapılmış ve ardından önerilen Olasılıksal İkili Yapay Arı Kolonisi (PrBABC) yöntemi ile optimize gen alt kümesi belirlenmiştir.

7.1 Önerilen Yöntem

mikrodizi verilerinden gen seçimi için önerilen yaklaşım 3 temel adımdan oluşmaktadır.

1. Mikrodizi veri kümelerinin normalize edilmesi,
2. Normalize veri kümelerinden filtre yöntemleri ile gen sayısının azaltılması,
3. PrBABC ile gen alt kümelerinin optimize edilmesi.

7.1.1 Normalizasyon

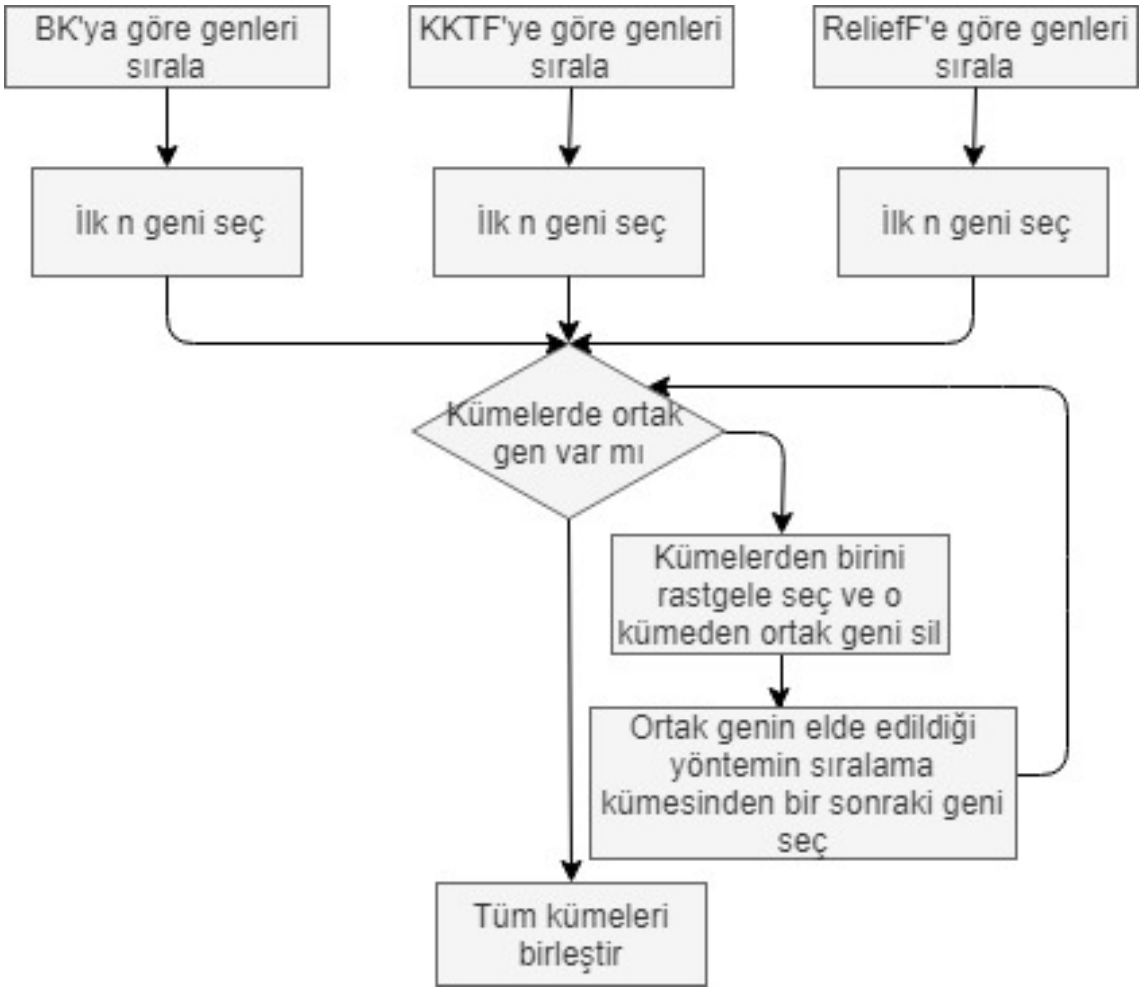
Mikrodizi veri kümelerini normalize etmek için önerilen MAS5, RMA ve GcRMA yöntemleri mikrodizi veri kümelerine uygulanarak yöntemler saflık, siluet katsayısı ve sınıflandırıcı katsayısı değerlerine göre karşılaştırılmıştır.

7.1.2 Filtreleme

Mikrodizi verilerinin en tipik karakteristik özelliklerinden biri binlerce özniteliğe ve çok az sayıda örneğe sahip olmasıdır. Veri kümesindeki öznitelik sayısını azaltarak çalışan filtreleme yöntemleri, ayırt edici öznitelikleri belirlemek için bir ön işlem sağlar. Bu amaçla, optimizasyon algoritması uygulanmadan önce veri kümelerine üç

filtre yöntemi uygulanmıştır. Entropi tabanlı Bilgi Kazancı, Korelasyon katsayısına dayalı Filtreleme, mesafeye dayalı ReliefF ve Varyasyon Katsayısı algoritmalarıdır. İlgili yöntemler farklı kriterlere göre öznitelikleri sıralamaktadır. Dört filtreleme algoritması ayrı ayrı mikrodizilere uygulanmış ve içlerinde performansı en yüksek olan 3 tanesi birlikte uygulanmak üzere seçilmiştir. Her üç yöntemin de avantajlarından faydalanabilmek için, hepsinin sonucu Şekil 7.1'de gösterildiği gibi birleştirilmiştir.

Buna göre genler, ilgili filtre yöntemlerine göre ayrı ayrı sıralanır. Her bir yöntemin sıraladığı genlerden ilk n tanesi alınarak 3 ayrı gen kümesi elde edilir. Daha sonra, kümelerde aynı gen(ler) olup olmadığı kontrol edilir. Eğer bir gen birden fazla kümede varsa, bu kümelerden rastgele seçilen birinden bu gen silinerek o kümenin elde edildiği filtreleme yönteminin sonuçlarından (n+1). gen alınarak kümeye eklenir. Bu işlem kümelerde hiçbir ortak gen kalmayana kadar devam ettirilir. Ve son olarak n tane gen içeren üç küme birleştirilerek, toplamda 3n gen içeren nihai gen kümesi elde edilir. Bu şekilde, filtreleme işlemi tamamlandıktan sonra üç farklı filtre yöntemiyle elde edilen veri kümelerinde farklı ayırt edici özelliğe sahip genler seçilmiş olur.



Şekil 7.1 Filtreleme işlemi

7.1.3 Olasılıksal İkili Yapay Arı Kolonisi

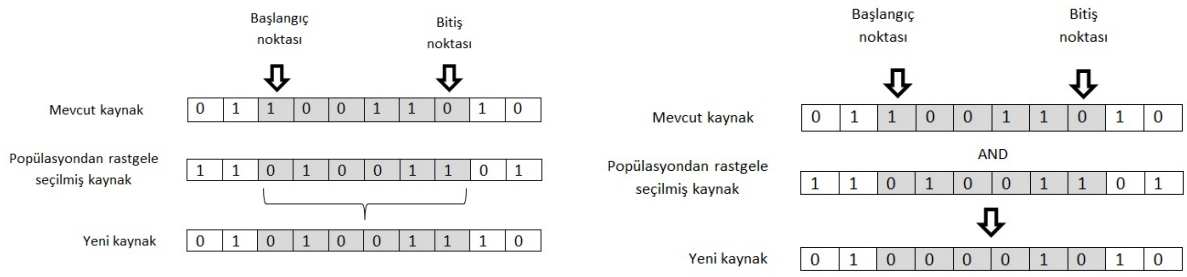
Algoritmada kaynak vektörler ikili uzayda Denklem (7.1) ile ilklendirilmiştir. Böylece kaynaklara doğrudan ikili değerler atanmıştır.

$$x_{ij} = \begin{cases} 1 & \text{eğer } rand(0,1) \geq esik \\ 0 & \text{eğer } rand(0,1) < esik \end{cases} \tag{7.1}$$

Öğrenme stratejisi, bir optimizasyon algoritmasının başarısını doğrudan etkileyecek bir faktördür ve kullanılan probleme veya verilere büyük ölçüde bağımlıdır. YAK algoritmasında öğrenme, işçi ve gözcü arı aşamalarındaki yeni kaynak üretme stratejileri ile yapılmaktadır. Öznitelik seçimi, seçilen öznitelik alt kümesinin sınıflandırıcı doğruluğunu maksimize etmeyi ve alt kümedeki öznitelik sayısını minimize etmeyi hedefleyen çok amaçlı bir optimizasyon problemi olarak tanımlanabilir. Her iki amaç için de uygun öğrenme yöntemini belirlemek için, literatürde sürekli bir uzayda çözüm üreten bir optimizasyon algoritmasını ikili uzaya taşıyabilmek için önerilen bazı öğrenme stratejileri test edilmiştir. Bunların bazılarının sınıflandırma doğruluğu açısından başarılı olduğu, bazısının da öznitelik sayısını azaltmada başarılı olduğu görülmüştür. Bu nedenle, [93]'den esinlenerek, sınıflandırıcı doğruluğunu artıran ve öznitelik sayısını azaltan iki öğrenme stratejisi olasılıksal olarak kullanılmıştır.

Deneysel çalışmalar neticesinde görülmüştür ki genetik operatörlerden çapraz geçerleme ile seçilen öznitelikler sınıflandırıcı doğruluğunu artırırken, bit işlem operatörleri ile de seçilen öznitelik sayısı azaltılabilmektedir. YAK algoritmasında, her bir arı kendi kaynağını optimize ederken bir önceki kaynak ile ilgili bilgilerden yararlanmaktadır. Bu nedenle öğrenme stratejileri bu bilgi paylaşımı göz önüne alınarak oluşturulmuştur. Sınıflandırıcı doğruluğunu artırmak için işçi arı fazında, kısmen eşleşmiş çaprazlama operatörü ile yeni kaynak üretilmiştir. Buna göre, mevcut kaynağa popülasyondan seçilen rastgele bir kaynak ile Şekil 7.2a'da görüldüğü gibi bir çaprazlama işlemi uygulanmıştır. Kısmi çaprazlama işleminde, kaynaklardan rastgele iki nokta seçilir, ardından bu noktalar arasındaki alt diziler değiştirilerek yeni bir kaynak oluşturulur. Böylece, işçi arı fazı için yeni bir kaynak üretilir. Yeni kaynağın uygunluk değeri mevcut kaynaktan daha iyi ise, yeni kaynak bir sonraki iterasyonun mevcut kaynağı olur. Bu durum mevcut kaynağın iyileştirildiği anlamına gelir. Çaprazlama işleminin avantajı, alt dizilerin bu şekilde farklı alt dizilerle birleştirilmesidir. Bu kombinasyonlar daha iyi sınıflandırma sonuçları üretebilir çünkü, optimizasyon algoritmalarında iyi çözümlerin hayatta kalma şansı daha yüksektir. Ayrıca, çaprazlama operatörü çeşitlilik sağlar ve yerel optimuma takılma riskini azaltır.

İkinci öğrenme stratejisi olarak da bit işlem 'AND' operatörü kullanılmıştır. Benzer şekilde, mevcut ve rastgele seçilen kaynaklardan, yine rastgele seçilmiş başlangıç ve bitiş noktaları arasında kalan alt dizilere, Şekil 7.2b'de gösterildiği gibi, 'AND' operatörü uygulanarak yeni bir kaynak üretilmiş olur.



(a) Çaprazlama işlemi **(b)** AND operatörü

Şekil 7.2 PrBABC için yeni kaynak üretme stratejileri

Öğrenme stratejileri, mevcut kaynağı iyileştirme durumlarına göre olasılıksal olarak seçilir. Buna göre algoritma ilklendirildiğinde, her iki stratejinin de seçilme olasılığı 0,5'dir. Her iki yöntem için de iyileştirme sayıları kaydedilmektedir. Yani bir yöntem seçildiğinde, bir kaynağın uygunluk değerini iyileştiriyorsa, bu değer bir artar. Her iterasyonun sonunda yöntemlerin seçilme olasılıkları Denklem (7.2) ile güncellenmektedir.

$$p_1 = \frac{iS_1}{\#\text{toplamİyilesme}}, p_2 = 1 - p_1 \tag{7.2}$$

p_1 ve p_2 sırasıyla, 1. ve 2. öğrenme stratejisinin seçilme olasılıklarıdır. iS_1, ilk öğrenme stratejisi tarafından üretilen başarılı yeni kaynakların sayısıdır. YAK'da, işçi arı fazındaki her kaynak ve gözcü arı fazındaki bazı kaynaklar için yeni kaynaklar üretilir. Üretilen yeni kaynak, bir öncekinden daha iyiyse yer değiştirilir. iS_1, öğrenme stratejisi olarak seçildiğinde, kaynağı iyileştirirse değeri 1 artar. Benzer şekilde ikinci öğrenme stratejisi seçilip kaynak iyileşirse, iS_2 değeri 1 artırılır. $\#toplamIyilesme$, toplam iyileştirme sayısıdır. İki stratejinin toplam olasılığı ise 1'dir. iS_1, iS_2, $\#toplamIyilesme$, p_1 ve p_2 değerleri her iterasyonun sonunda güncellenir.

Her yeni kaynak oluşturma aşamasında, 0 ile maks(p_1,p_2) arasında, düzgün dağılımlı rastgele bir sayı üretilir. Bu sayı p_1'den küçükse ilk öğrenme stratejisi, p_2'den küçükse ikinci öğrenme stratejisi tarafından ilgili iterasyon için yeni bir kaynak oluşturulur. Buna göre bir öğrenme stratejisi mevcut kaynağı üst üste birkaç kere iyileştirmesi halinde, o öğrenme yönteminin olasılık değeri ile diğeri arasındaki fark çok olacağından algoritma sürekli bir öğrenme yöntemine meyleyme eğilimi gösterecektir. Bu şekilde, olasılıklarda bu kadar belirgin farklılıklar olmasını önlemek için, rastgele üretilen sayı her iki olasılıktan da küçükse algoritmanın iki yöntemden birini rastgele seçmesi sağlanmıştır.

İşçi arı aşamasında, öğrenme yöntemlerinin kaynakları iyileştirebilme yeteneklerinden faydalanarak seçim yapılması sağlanmıştır. Benzer şekilde, gözcü arı aşamasında kaynakların başarısızlıkları göz önüne alınarak Denklem (7.3) ile seçilme olasılıkları elde edilmiştir. Buna göre, yöntemlerden biri öğrenme stratejisi olarak seçildiğinde eğer mevcut kaynağı iyileştiremediyse, o stratejinin başarısızlık sayısı (bS) değeri 1 artırılır. İşçi arı aşamasında olasılık değerinin yüksek olması, o yöntemin seçilme olasılığını artırırken burada olasılığın düşük olması seçilme ihtimalini artırmaktadır. Böylece önerilen yöntemle, algoritma kendinden uyarlamalı bir yöntemle yeni kaynak üretme stratejisini seçerken, bu stratejilerin hem iyileştirme hem de iyileştirememe yeteneklerinden faydalanmaktadır.

$$p_3 = \frac{bS_1}{\#toplamBasarisizlik}, p_4 = 1 - p_3 \tag{7.3}$$

7.2 Deneysel Çalışmalar

Çizelge 7.1'de görüldüğü gibi önerilen yöntemin başarısını ölçmek için, farklı büyüklüklerde dokuz adet mikrodizi veri kümesi kullanılmıştır.

Çizelge 7.1 PrBABC için kullanılan mikrodizi veri kümeleri

MA	#Örnek	#Sınıf	#Gen	Açıklama
Cll-Sub	111	3	12626	B hücreli kronik lenfositik lösemi alt grupları [94]
Stjude	327	7	12625	Pediatrik lenfoblastik löseminin alt grupları [95]
Dlbcl	77	2	7129	Büyük B Hücreli Lenfoma [96]
Lung Can	235	5	12625	İnsan Akciğer Karsinomları [96]
Glim.	50	2	12625	Beyin Tümörü [97]
Leuk2	72	3	12626	Löseminin alt tipleri [96]
Pros Can.	102	2	12625	Normal genler vs prostat tümörlü genler [96]
WBC	97	2	24482	Göğüs kanseri nüksetmesinde hayatta kalma [98]
CNS	60	2	7129	Merkezi sinir sistemi embriyonel tümör sonucu [96]

Mikrodizi normalizasyon yöntemleri saflık, doğruluk ve siluet katsayısı değerlerine göre karşılaştırılmıştır. Saflık değerleri K-Means ve Hiyerarşik kümeleme yöntemlerine göre ölçülmüştür. Kümeleme algoritmaları için başlangıç noktaları rastgele seçildiğinden, kümeleme işlemi 10 defa tekrarlanarak saflık değerlerinin ortalaması alınmıştır. Siluet katsayıları K-Means Algoritması tarafından üretilen kümelere göre hesaplanmıştır. Mikrodizilerde sınıf bilgisi mevcut olduğundan, küme sayıları ilgili mikrodiziin sınıf sayısı olarak kabul edilmiştir. Sınıflandırıcı doğruluğu 3-NN algoritması ile 10 kat çapraz geçerleme yapılarak elde edilmiştir. Sınıflandırma ve kümeleme algoritmaları Matlab r2017a ile uygulanmıştır. Normalizasyon yöntemlerini karşılaştırmak için elde edilen sonuçlar Çizelge 7.2'de verilmiştir.

Çizelge 7.2 Normalizasyon yöntemlerinin karşılaştırılması

		Saflık		Doğruluk	Siluet K.
Mikrodizi	Metot	K-Means	Hiyerarşik K.	3-NN	K-Means
Cll-Sub	MAS5	0,252	**0,513**	0,829	**0,491**
	RMA	0,423	0,468	**0,892**	0,217
	GcRMA	**0,577**	0,459	0,883	0,251
Lung	MAS5	0,302	0,715	0,962	0,154
	RMA	**0,506**	0,715	**0,983**	0,168
	GcRMA	0,451	**0,723**	0,974	**0,187**
CNS	MAS5	0,617	0,633	0,783	0,161
	RMA	0,617	0,633	**0,817**	**0,252**
	GcRMA	**0,683**	0,633	0,783	0,208
WBC	MAS5	0,507	0,622	**0,787**	0,172
	RMA	**0,535**	0,622	**0,78**	**0,183**
	GcRMA	0,528	**0,629**	0,773	0,111
DLBCL	MAS5	0,61	0,74	0,948	0,174
	RMA	**0,714**	0,74	**0,974**	**0,182**
	GcRMA	**0,714**	0,74	0,948	0,131
Leuk2	MAS5	0,444	0,389	0,944	**0,193**
	RMA	**0,667**	0,389	**0,986**	0,153
	GcRMA	0,639	0,389	0,972	0,109
Stjude	MAS5	0,177	0,231	0,865	**0,125**
	RMA	**0,217**	**0,24**	0,939	0,056
	GcRMA	**0,217**	**0,24**	**0,96**	0,066
Prostate	MAS5	**0,627**	0,5	0,892	0,373
	RMA	0,569	0,5	**0,931**	0,358
	GcRMA	0,598	0,52	0,912	**0,391**
Glioma	MAS5	0,6	0,54	0,88	0,12
	RMA	**0,72**	0,54	0,88	0,147
	GcRMA	**0,72**	0,54	0,88	**0,29**

Table 7.2'deki sonuçlara göre, hiyerarşik kümeleme ile elde edilen saflık değerleri genellikle birbirine benzer çıkmıştır. Bazı veri kümeleri için K-Means kümelemesi tarafından elde edilen saflık değerleri birbirine yakındır. Genel olarak, RMA ve GcRMA normalizasyon yöntemleri ile alınmış kümeleme sonuçları daha iyidir. Sınıflandırıcı doğruluğu bakımından, RMA normalizasyonu diğer yöntemlerden daha başarılı

çıkmıştır. Siluet katsayıları dikkate alındığında ise, yöntemler benzer performanslar göstermiştir. Sonuç olarak, RMA'nın bu mikrodizi verileri için diğer yöntemlere göre genellikle iyi sonuçlar verdiği görülmüştür. Bu nedenle gen seçimi için RMA yöntemiyle normalize edilmiş mikrodizi verileri kullanılmıştır.

Optimizasyon aşamasında mikrodizi verileri, 5-NN algoritması ile sınıflandırılmıştır. Veri kümeleri rastgele seçilen örneklerle 3 parçaya ayrılmıştır. Verilerin %60'ı eğitim kümesi, %20'si doğrulama kümesi ve kalan veriler de test kümesi olarak kullanılmıştır. Mikrodizi verilerinde örnek sayısının az olduğu göz önüne alınarak eğitim, test ve doğrulama kümelerinde, her sınıftan en az bir örneğin bulunması sağlanmıştır. Optimizasyon aşamasında seçilen gen alt kümeleri, eğitim kümesi ile eğitilip doğruluk kümesi ile de modelin geliştirilmesi sağlanmıştır. Tüm iterasyonlar tamamlandıktan sonra, algoritmanın seçtiği en iyi gen kümesinin performansı test kümesi ile ölçülmüştür. Doğrulama ve test kümelerindeki örnekler eğitim aşamasında kullanılmamıştır.

Önerilen yöntem, yaygın sürü zekası algoritmalarından Genetik Algoritma [89], İkili Parçacık Sürü Optimizasyonu [68] ve İkili Diferansiyel Evrim [70] algoritmaları ile karşılaştırılmıştır. Adil bir karşılaştırma yapmak için, GA, BinPSO ve BinDE, PrBABC'de olduğu gibi 3'lü filtreleme yöntemi ile gen sayısı azaltılmış mikrodizi verilerine uygulanmıştır. Elde edilen sonuçlar test kümesi sınıflandırıcı doğruluğu, seçilen gen sayısı ve çalışma sürelerine göre karşılaştırılmıştır. BinPSO, GA ve BinDE algoritmaları sırasıyla [68, 70, 89]'deki önerilere göre Matlab r2017a ortamında uygulanmıştır. Kaynak kodlar "github.com/ZBaOz/MicroarrayGeneSelection"da mevcuttur.

İlklendirme aşamasında kullanılan eşik değeri 0,8'dir. Bu değer bir genin seçilen alt kümede bulunma olasılığını temsil eder. Kaşif arı aşamasında kullanılan limit değeri 100'dür. Popülasyon büyüklüğü YAK algoritması için işçi ve gözcü arıların toplamı olup, algoritmada işçi arı sayısı gözcü arı sayısına eşit olmaktadır. Bu nedenle PrBABC için 25 işçi arı ve 25 gözcü arı olmak üzere 50 arı vardır. Aynı şekilde GA, BinDE ve BinPSO için de popülasyon büyüklüğü 50'dir. GA için [89]'de tavsiye edildiği şekilde, çapraz geçerleme oranı 0,8 ve mutasyon oranı da 0,2 olarak belirlenmiştir. BinPSO için parametreler c1 ve c2 öğrenme faktörleri 2, başlangıç ağırlığı (w) 0,9 ve maksimum hız (V_{max}) 0,6 olarak [88]'da olduğu gibi belirlenmiştir. BinDE için ise [70]'de tavsiye edildiği gibi, öğrenme faktörü (F) 1, pertürbasyon parametresi (P) 0,25 ve çaprazlama oranı (CR) 0,1 olarak belirlenmiştir. Tüm algoritmalar için maksimum iterasyon sayısı 100'dür. Algoritmalar, rastgele örneklerle oluşturulmuş farklı eğitim ve test kümeleri ile 20 defa çalıştırılmış ve istatistiksel geçerlilik için sonuçların ortalaması alınmıştır. Kaynak kod "github.com/ZBaOz/MicroarrayGeneSelection" adresinde verilmiştir.

YAK algoritmasında keşif, kaşif arı fazında yapılmaktadır ve 'limit' parametresi ile kontrol edilir. Bu nedenle bu parametre algoritmanın performansını doğrudan etkilemektedir. Karaboga ve Bastürk [99], n_e; işçi arı sayısı ve D problemin boyutu olmak üzere bu parametreyi Denklem 7.4 ile belirlemeyi önermiştir.

$$limit = n_e * D \tag{7.4}$$

Vecek vd. ise çalışmalarında [100] farklı n_e ve D değer çiftlerini denemiş, ve bu iki değişken arasında, özellikle yüksek boyutlu problemlerde, Denklem 7.4'deki gibi doğrusal bir ilişki olmadığını göstermişlerdir. Mikrodizi veri kümelerinden gen seçimi işleminde, problemin boyutu mikrodizideki gen sayısı kadardır. Mikrodiziler ise yukarda da bahsedildiği gibi, binlerce öznitelik içeren veri kümeleridir. 'limit' parametresinin Denklem 7.4 ile belirlenmesi durumunda çok yüksek bir değer alacağından algoritmanın keşif yeteneği çok azalacaktır. Bu nedenle 'limit' parametresinin probleme bağlı olduğu göz önüne alınarak, deneysel bir takım çalışmalar neticesinde 100 olarak belirlenmiştir.

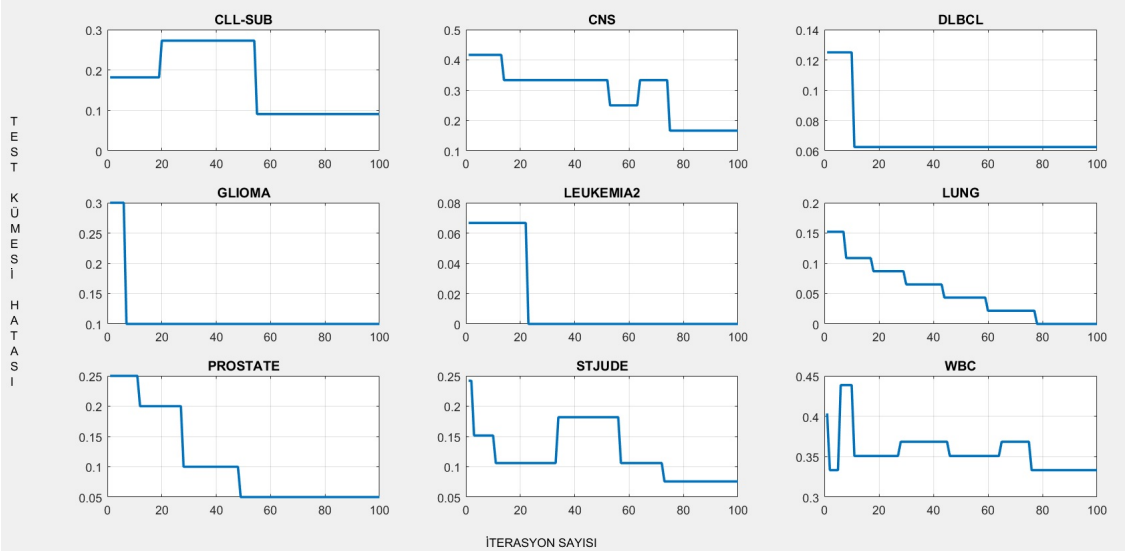
Algoritmada uygunluk değeri, sınıflandırıcı doğruluğu ve seçilen altkümedeki gen sayısına göre Denklem (7.5) ve (7.6) ile hesaplanır. Denklemde 'c' sabiti sınıflandırıcı doğruluğu ve gen sayısı değerlerinin ağırlıklarını düzenlemek için kullanılır. Bu çalışmada, sınıflandırıcı doğruluğunu artırmak gen sayısını azaltmaktan daha önemli olduğu için c 0,9995 olarak belirlenmiştir. Böylece, algoritmanın her zaman yüksek doğruluğu olan bir altkümeyi seçmesi garanti edilmiştir, ancak aynı doğruluk değerine sahip iki alt küme varsa gen sayısı az olanın seçilmesini sağlamaktadır.

$$uygunluk = (c * \text{doğruluk}) + ((1-c) * \frac{\#\text{Seçilen Öznitelik}}{\#\text{ToplamÖznitelik}}) \tag{7.5}$$

$$dogruluk = \frac{\#\text{Doğru Sınıflandırılan Örnek}}{\#\text{Toplam Örnek}} \tag{7.6}$$

Maksimum iterasyon sayısı, iteratif bir algoritmanın bir diğer önemli parametresidir ve probleme bağlıdır. Problemin zorluğu ise arama alanının boyutuyla doğrudan ilgilidir. Azami iterasyon sayısı gerekenden az bir değer seçilirse algoritma yeterince arama yapamayacağı için uygun çözüme yaklaşamaz. Gereğinden büyük alınırsa da model eğitim kümesine çok bağlı olacak ve aşırı eğitim nedeniyle eğitim kümesindeki hata azalırken test kümesindeki hata artacaktır. Algoritmanın sonuca yakınsama hızı deneysel olarak incelenmiş ve iterasyon sayısı 100 olarak belirlenmiştir. PrBABC-3F ilk 1500 genin bulunduğu veri kümesine uygulanmış ve iterasyon sayısına göre test

kümesi hatası Şekil 7.3'de gösterilmiştir. Grafiklerde 20 iterasyondan rastgele bir tanesi seçilmiştir. Azami iterasyon sayısının algoritmanın yakınsaması için yeterli olduğu Şekil 7.3'de deneysel olarak gösterilmiştir. Algoritmanın 100 iterasyona geldiğinde kararlı bir yapıya ulaştığı ve eğitim kümesine bağlı ezberleme olmadan algoritmanın sonuca varmış olduğu açıkça görülmektedir.



Şekil 7.3 PrBABC için yakınsama grafiği

Mikrodizilere BK, KKDF, ReliefF ve VK filtre yöntemleri uygulanmış ve ilk 1500, 1000, 750, 500, 300 ve 100 gen seçilmiştir. PrBABC-3F ayrı ayrı performansı değerlendirilen 4 filtreleme yönteminden en iyi 3 tanesi olan BK, KKDF ve ReliefF yöntemlerinin birlikte uygulanarak gerçekleştirilen filtreleme işlemini temsil etmektedir. Örneğin, yukarıda bahsedildiği gibi ilk 1500 gen, her filtre yöntemiyle elde edilen ilk 500 gen kümesinin birleştirilmesiyle oluşmuştur. PrBABC-BK, PrBABC-KKDF, PrBABC-RF ve PrBABC-VK PrBABC'nin sırasıyla BK, KKDF, ReliefF ve VK filtreleri ile uygulandığı anlamına gelmektedir. Filtreleme yöntemleri test kümesi hatası bakımından karşılaştırılmış ve sonuçlar standart sapma değerleri ile Çizelge 7.3'de verilmiştir. Her veri kümesi için en iyi sonuçlar koyu olarak gösterilmiştir. Sonuçlar, birçok veri kümesi için 3 filtreleme metodunun bir arada uygulanmasının tek başına uygulanmalarından daha iyi sonuçlar verdiğini göstermiştir. DLBCL ve Leukemia2 mikrodizilerinde test kümesi hataları tüm yöntemler için genel olarak 0,1'den küçüktür ve BK ve ReliefF yöntemleri, zaman zaman daha iyi sonuçlar vermiştir. Ancak, bu yönemler ile üçlü filtreleme yönteminin elde ettiği hata değerleri arasındaki fark çok küçük çıkmıştır. Üçlü filtreleme yöntemini göz ardı ettiğimizde, ReliefF algoritmasının BK, KKDF ve VK'ye göre daha iyi sonuçlar verdiği görülmektedir. ReliefF, sınıfları biribirinden ayıran marjları maksimize etmeyi amaçlamaktadır. Sonuçlardan açıkça görülmektedir ki, üç filtre metodunu bir arada uygulamak, doğruluk değeri daha yüksek gen alt kümelerini elde etmeyi sağlamıştır.

Çizelge 7.3 Filtre metodlarının test hatasına göre karşılaştırılması

	Metot	CLL-SUB	CNS	DLBCL	Glioma	Leuk2	Lung	Prostate	Stjude	WBC
1500	PrBABC-3F	**0,259 ±0,1**	**0,242 ±0,08**	**0,031 ±0,04**	**0,14 ±0,1**	**0,03 ±0,04**	**0,042 ±0,03**	**0,1 ±0,07**	**0,125 ±0,04**	**0,325 ±0,05**
	PrBABC-KKDF	0,355 ±0,12	0,367 ±0,1	0,081 ±0,06	0,255 ±0,14	0,083 ±0,09	0,076 ±0,03	0,165 ±0,07	0,174 ±0,04	0,383 ±0,06
	PrBABC-BK	0,289 ±0,09	0,383 ±0,14	0,044 ±0,05	0,205 ±0,14	0,053 ±0,05	0,053 ±0,03	0,153 ±0,05	0,131 ±0,04	0,372 ±0,05
	PrBABC-RF	0,284 ±0,1	0,308 ±0,09	0,069 ±0,06	0,16 ±0,1	0,06 ±0,06	0,053 ±0,03	0,123 ±0,07	0,139 ±0,03	0,361 ±0,04
	PrBABC-VK	0,339 ±0,1	0,417 ±0,12	0,103 ±0,07	0,295 ±0,12	0,09 ±0,07	0,059 ±0,02	0,182 ±0,09	0,17 ±0,04	0,391 ±0,04
1000	PrBABC-3F	**0,227 ±0,1**	**0,279 ±0,13**	0,031 ±0,05	**0,15 ±0,09**	**0,023 ±0,03**	**0,035 ±0,03**	**0,095 ±0,06**	**0,116 ±0,05**	**0,333 ±0,04**
	PrBABC-KKDF	0,286 ±0,1	0,375 ±0,17	0,116 ±0,08	0,285 ±0,13	0,1 ±0,06	0,08 ±0,03	0,21 ±0,07	0,22 ±0,05	0,397 ±0,06
	PrBABC-BK	0,314 ±0,09	0,3 5±0,01	**0,028 ±0,03**	0,165 ±0,11	0,033 ±0,05	0,053 ±0,03	0,143 ±0,07	0,13 ±0,04	0,368 ±0,05
	PrBABC-RF	0,275 ±0,11	0,313 ±0,11	0,059 ±0,05	0,2 ±0,11	0,067 ±0,05	0,057 ±0,03	0,122 ±0,07	0,142 ±0,04	0,348 ±0,04
	PrBABC-VK	0,323 ±0,08	0,433 ±0,12	0,125 ±0,08	0,25 ±0,11	0,05 ±0,07	0,062 ±0,02	0,177 ±0,07	0,152 ±0,05	0,404 ±0,04
750	PrBABC-3F	**0,223 ±0,11**	**0,263 ±0,09**	**0,028 ±0,04**	**0,12 ±0,08**	**0,013 ±0,03**	**0,053 ±0,03**	**0,09 ±0,06**	**0,109 ±0,05**	**0,33 ±0,04**
	PrBABC-KKDF	0,354 ±0,11	0,45 ±0,12	0,138 ±0,09	0,305 ±0,14	0,137 ±0,08	0,077 ±0,04	0,22 ±0,07	0,219 ±0,04	0,4 ±0,06
	PrBABC-BK	0,282 ±0,09	0,371 ±0,12	0,031 ±0,05	0,21 ±0,16	0,033 ±0,05	0,05 ±0,03	0,1 ±0,06	0,138 ±0,05	0,384 ±0,05
	PrBABC-RF	0,282 ±0,09	0,283 ±0,08	0,034 ±0,04	0,165 ±0,12	0,073 ±0,07	**0,053 ±0,03**	0,14 ±0,09	0,145 ±0,04	0,365 ±0,04
	PrBABC-VK	0,293 ±0,11	0,433 ±0,11	0,1 ±0,07	0,22 ±0,12	0,04 ±0,06	0,068 ±0,03	0,202 ±0,09	0,16 ±0,04	0,378 ±0,06
500	PrBABC-3F	**0,234 ±0,1**	**0,263 ±0,12**	0,047 ±0,05	**0,12 ±0,11**	**0,017 ±0,04**	**0,046 ±0,03**	**0,08 ±0,06**	**0,127 ±0,04**	**0,339 ±0,06**
	PrBABC-KKDF	0,302 ±0,12	0,383 ±0,14	0,125 ±0,06	0,33 ±0,13	0,163 ±0,11	0,112 ±0,04	0,2 ±0,08	0,23 ±0,05	0,395 ±0,06
	PrBABC-BK	0,314 ±0,12	0,375 ±0,12	0,047 ±0,05	0,17 ±0,13	0,04 ±0,06	0,054 ±0,03	0,118 ±0,06	0,137 ±0,05	0,414 ±0,05
	PrBABC-RF	0,286 ±0,1	0,271 ±0,09	**0,037 ±0,05**	0,185 ±0,09	0,057 ±0,05	0,066 ±0,03	0,135 ±0,08	0,162 ±0,03	0,341 ±0,05
	PrBABC-VK	0,293 ±0,13	0,392 ±0,12	0,097 ±0,07	0,245 ±0,14	0,053 ±0,05	0,074 ±0,04	0,21 ±0,085	0,173 ±0,04	0,411 ±0,05
300	PrBABC-3F	**0,241 ±0,09**	**0,25 ±0,14**	0,037 ±0,05	**0,155 ±0,09**	0,04 ±0,05	**0,059 ±0,03**	**0,087 ±0,07**	**0,13 ±0,05**	**0,328 ±0,05**
	PrBABC-KKDF	0,352 ±0,1	0,404 ±0,12	0,197 ±0,1	0,285 ±0,1	0,187 ±0,1	0,079 ±0,04	0,223 ±0,09	0,275 ±0,05	0,428 ±0,06
	PrBABC-BK	0,266 ±0,08	0,363 ±0,15	**0,034 ±0,05**	0,19 ±0,12	**0,037 ±0,034**	0,068 ±0,04	0,113 ±0,09	0,148 ±0,03	0,43 ±0,05
	PrBABC-RF	0,268 ± 0,1	0,267 ±0,1	0,041 ±0,04	0,16 ±0,11	0,043 ±0,05	**0,059 ±0,03**	0,113 ±0,07	0,151 ±0,03	0,351 ±0,04
	PrBABC-VK	0,284 ±0,09	0,45 ±0,15	0,156 ±0,07	0,31 ±0,12	0,063 ±0,07	0,065 ±0,03	0,173 ±0,07	0,152 ±0,04	0,409 ±0,06
100	PrBABC-3F	**0,209 ±0,06**	**0,233 ±0,08**	**0,047 ±0,05**	**0,145 ±0,14**	**0,043 ±0,05**	**0,07 ±0,04**	**0,068 ±0,08**	**0,144 ±0,04**	**0,337 ±0,04**
	PrBABC-KKDF	0,418 ±0,13	0,483 ±0,15	0,2 ±0,07	0,335 ±0,16	0,223 ±0,14	0,128 ±0,04	0,237 ±0,11	0,398 ±0,06	0,43 ±0,05
	PrBABC-BK	0,275 ±0,12	0,304 ±0,11	0,081 ±0,05	0,165 ±0,12	0,077 ±0,08	0,07 ±0,04	0,137 ±0,08	0,18 ±0,05	0,402 ±0,06
	PrBABC-RF	0,289 ±0,09	0,271 ±0,1	0,069 ±0,07	0,16 ±0,13	0,087 ±0,09	0,105 ±0,05	0,143 ±0,08	0,166 ±0,04	0,342 ±0,05
	PrBABC-VK	0,33 ±0,09	0,433 ±0,13	0,163 ±0,1	0,33 ±0,13	0,087 ±0,07	0,103 ±0,05	0,255 ±0,09	0,174 ±0,06	0,412 ±0,04

Önerilen yöntem, yaygın kullanılan 3 sürü zekası algoritması olan GA, BinPSO ve BinDE ile karşılaştırılmıştır. PrBABC'de olduğu gibi, üçlü filtreleme metodu tarafından seçilmiş en iyi 1500, 1000, 750, 300 ve 100 genin olduğu veri kümelerine uygulanarak sonuçlar test kümesi hatası bakımından Çizelge 7.4'de ve seçilen gen sayısı bakımından Çizelge 7.5'da karşılaştırılmıştır. Buna göre GA-3F, BinPSO-3F ve BinDE-3F sırasıyla GA, BinPSO ve BinDE algoritmalarının üçlü filtreleme ile veri kümelerine uygulanmasını temsil eder.

Çizelge 7.4'de her yöntem için birinci satır algoritmaların 20 defa çalıştırılması sonucu elde edilen ortalama test kümesi hatası ve standart sapma değerlerini göstermektedir. İkinci satırda ise Wilcoxon Rank Sum Test tarafından elde edilen, PrBABC'nin elde ettiği ortalama sonuçların diğer yöntemlere göre istatistiksel anlamlılığı verilmiştir. Buna göre; "+" ve "−" işaretleri sırasıyla, PrBABC'nin sonuçlarının, ilgili algoritmadan önemli ölçüde daha iyi veya daha kötü olduğunu göstermektedir. "=" ise sonuçlar arasında istatistiksel bir fark olmadığı anlamına gelmektedir. Algoritmaların seçtikleri ortalama gen sayıları ve standart sapma değerleri ise Çizelge 7.5'da verilmiştir.

Çizelge 7.4 ve Çizelge 7.5'daki sonuçlardan açıkça görülmektedir ki, PrBABC'nin diğer yöntemlere göre birçok veri kümesinde daha az test kümesi hatası elde etmiş olması, ayırt edici genleri seçmekte daha başarılı olduğunu açıkça göstermektedir. Çizelge 7.3'de olduğu gibi diğer yöntemler DLBCL, Leukemia2 ve Lung veri kümelerinde daha az hata ile sonuçlanmış olsada, elde edilmiş olan hata değerlerinden ilgili mikrodizilerin kolay modellenebilen veri kümeleri olduğu açıkça görülmektedir. Bu veri kümeleri için farklılık, seçtikleri alt kümelerdeki gen sayılarına göre belirlenir. Buna göre PrBABC'nin, tüm veri kümeleri için daha yüksek veya en azından eş değer sınıflandırıcı başarısını daha az sayıda gen ile elde ettiği görülmektedir. PrBABC'de iki öğrenme stratejisi kullanılmıştır. Öğrenme yöntemlerini seçerken ayırt edici genleri belirleyebilme yeteneklerine öncelik verilmiş ve sonuçlardan da seçim işleminde başarılı olduğu görülmüştür.

Mikrodiziler ile ilgili bilgilerin bulunduğu Çizelge 7.1'den görülmektedir ki her bir veri kümesi farklı sayıda gen ve örnek içermektedir. Veri kümelerinin bazıları kolay modellenebilir olmasına karşı, bazıları ise daha zor modellenebilmektedir. Bu farklılıklar göz önüne alınarak filtreleme metoduyla farklı büyüklüklerde veri kümeleri oluşturulmuştur. Her bir mikrodizi için filtreleme metodlarının seçtiği ilk 1500, 1000, 750, 500, 300 ve 100 geni içeren veri kümelerinde gen seçimi işlemi yapılmıştır. Seçilen gen sayısı ile test kümesi hatası arasında bir denge kurmak önemlidir. Çok sayıda gen içermesi her zaman sınıflandırıcı doğruluğunun artmasını sağlamamaktadır. Büyük veri kümesi, arama uzayının da büyümesine neden olacağından nitelikli gen gruplarının bulunabilmesi daha zor olacaktır. Ayrıca, büyük

Çizelge 7.4 Evrimsel yöntemlerin test kümesi hatasına göre karşılaştırılması

		CLL-SUB	CNS	DLBCL	Glioma	Leuk2	Lung	Prostate	Stjude	WBC
1500	Pr BABC	**0,259 ±0,1**	**0,242 ±0,08**	**0,031 ±0,04**	**0,14 ±0,1**	0,03 ±0,04	0,042 ±0,03	**0,1 ±0,07**	**0,125 ±0,04**	**0,325 ±0,05**
	GA	0,268 ±0,08	0,354 ±0,09	0,044 ±0,06	0,195 ±0,12	0,047 ±0,05	0,051 ±0,03	0,14 ±0,07	0,142 ±0,04	0,338 ±0,05
		+	+	+	+	-	=	-	+	+
	Bin PSO	0,273 ±0,09	0,383 ±0,14	0,106 ±0,1	0,355 ±0,19	0,037 ±0,05	0,041 ±0,031	0,207 ±0,17	0,141 ±0,03	0,367 ±0,054
		+	+	+	+	+	+	+	+	+
	Bin DE	0,264 ±0,11	0,308 ±0,11	0,041 ±0,04	0,17 ±0,11	**0,027 ±0,03**	**0,041 ±0,03**	0,123 ±0,06	0,14 ±0,03	0,362 ±0,04
		+	+	+	+	+	+	=	+	+
1000	Pr BABC	**0,227 ±0,1**	**0,279 ±0,13**	**0,031 ±0,05**	**0,15 ±0,09**	0,023 ±0,03	**0,035 ±0,03**	**0,095 ±0,06**	**0,116 ±0,05**	**0,333 ±0,04**
	GA	0,273 ±0,12	0,321 ±0,11	0,063 ±0,06	0,2 ±0,07	**0,02 ±0,03**	0,047 ±0,033	0,125 ±0,07	0,133 ±0,03	0,358 ±0,05
		+	+	=	+	+	+	+	+	+
	Bin PSO	0,284 ±0,11	0,367 ±0,14	0,113 ±0,13	0,42 ±0,16	0,043 ±0,1	0,052 ±0,034	0,155 ±0,1	0,153 ±0,04	0,359 ±0,04
		=	+	+	+	+	+	+	+	+
	Bin DE	0,284 ±0,11	0,338 ±0,09	0,041 ±0,04	0,195 ±0,13	0,023 ±0,04	0,043 ±0,03	0,12 ±0,06	0,131 ±0,04	0,36 ±0,03
		+	+	-	+	+	+	+	+	+
750	Pr BABC	**0,223 ±0,11**	**0,263 ±0,09**	**0,028 ±0,04**	**0,12 ±0,08**	**0,013 ±0,03**	0,053 ±0,03	**0,09 ±0,06**	**0,109 ±0,05**	**0,33 ±0,04**
	GA	0,259 ±0,08	0,337 ±0,11	0,053 ±0,06	0,19 ±0,09	0,03 ±0,04	0,053 ±0,03	0,12 ±0,06	0,13 ±0,03	0,369 ±0,06
		+	+	=	+	+	+	+	+	+
	Bin PSO	0,284 ±0,11	0,4 ±0,12	0,094 ±0,12	0,295 ±0,15	0,03 ±0,05	**0,049 ±0,02**	0,202 ±0,13	0,153 ±0,04	0,363 ±0,06
		+	+	+	+	+	+	+	+	+
	Bin DE	0,277 ±0,1	0,287 ±0,12	0,041 ±0,05	0,245 ±0,11	0,023 ±0,04	0,053 ±0,03	0,118 ±0,06	0,127 ±0,04	0,343 ±0,06
		+	+	-	+	+	+	+	+	+
500	Pr BABC	**0,234 ±0,1**	**0,263 ±0,12**	**0,047 ±0,05**	**0,12 ±0,11**	**0,017 ±0,04**	0,046 ±0,03	**0,08 ±0,06**	**0,127 ±0,04**	**0,339 ±0,06**
	GA	0,257 ±0,09	0,304 ±0,12	0,063 ±0,05	0,185 ±0,13	0,03 ±0,03	0,049 ±0,03	0,118 ±0,06	0,145 ±0,06	0,361 ±0,06
		+	+	-	+	+	+	+	+	+
	Bin PSO	0,282 ±0,12	0,333 ±0,1	0,091 ±0,13	0,37 ±0,22	0,057 ±0,13	**0,03 ±0,02**	0,133 ±0,11	0,146 ±0,03	0,36 ±0,05
		+	+	+	+	-	+	+	+	+
	Bin DE	0,268 ±0,09	0,296 ±0,12	0,053 ±0,05	0,195 ±0,12	0,017 ±0,03	0,049 ±0,03	0,113 ±0,05	0,128 ±0,03	0,366 ±0,05
		+	+	=	+	+	-	+	+	=
300	Pr BABC	**0,241 ±0,09**	**0,25 ±0,14**	**0,037 ±0,05**	**0,155 ±0,09**	0,04 ±0,05	0,059 ±0,03	**0,087 ±0,07**	**0,13 ±0,05**	**0,328 ±0,05**
	GA	0,259 ±0,08	0,3 ±0,11	0,053 ±0,05	0,175 ±0,14	0,037 ±0,05	0,055 ±0,03	0,118 ±0,1	0,144 ±0,03	0,356 ±0,05
		+	+	=	+	+	+	+	+	+
	Bin PSO	0,252 ±0,08	0,313 ±0,1	0,113 ±0,13	0,295 ±0,19	0,067 ±0,12	**0,049 ±0,03**	0,105 ±0,1	0,168 ±0,03	0,346 ±0,04
		+	+	+	+	+	+	+	+	+
	Bin DE	0,245 ±0,09	0,292 ±0,1	0,044 ±0,05	0,18 ±0,08	**0,02 ±0,03**	0,051 ±0,04	0,095 ±0,06	0,145 ±0,04	0,374 ±0,04
		+	+	-	+	+	+	+	+	+
100	Pr BABC	**0,209 ±0,06**	**0,233 ±0,08**	0,047 ±0,05	**0,145 ±0,14**	**0,063 ±0,06**	0,07 ±0,04	**0,068 ±0,08**	**0,144 ±0,04**	**0,337 ±0,04**
	GA	0,27 ±0,1	0,317 ±0,11	0,066 ±0,06	0,215 ±0,12	0,073 ±0,05	0,072 ±0,03	0,153 ±0,06	0,151 ±0,03	0,371 ±0,05
		+	+	+	+	+	-	+	+	+
	Bin PSO	0,282 ±0,09	0,317 ±0,09	0,134 ±0,11	0,35 ±0,16	0,12 ±0,24	**0,064 ±0,035**	0,133 ±0,07	0,156 ±0,04	0,369 ±0,04
		+	+	+	+	+	-	+	+	+
	Bin DE	0,28 ±0,1	0,27 ±0,1	**0,044 ±0,04**	0,18 ±0,11	0,07 ±0,04	0,071 ±0,04	0,11 ±0,07	0,156 ±0,04	0,361 ±0,05
		+	+	+	+	+	=	+	+	+

veri kümesi öznitelik seçiminde hesaplama maliyetinin artmasına da neden olacaktır. DLBCL, Leukemia2 ve Lung mikrodizileri Çizelge 7.4'den de görüldüğü üzere kolay modellenebilen veri kümeleridir ve gen seçimi sonrasında çok küçük hatalar ile sınıflandırma yapılabilmektedir. Bu nedenle bu veri kümelerinde, filtreleme metodları ile seçilen ilk 100 öznitelik, gen seçimi yapmak için yeterlidir. Benzer şekilde CLL-SUB, CNS ve Prostat kanseri veri kümelerinde de PrBABC, 100 gen içeren veri kümesinde en iyi sonucu vermiştir. WBC mikrodizini için, PrBABC ilk 1500 gen ile işlem yaptığında ortalama 58 gen seçmiş ve 0,325 test kümesi hatası ile sınıflandırmıştır. İlk 300 gen ile işlem yaptığında ise, ortalama 15 gen seçmiş ve 0,328 hata ile sınıflandırmıştır. Gen sayıları arasındaki fark ile test kümesi hataları arasındaki fark göz önüne alındığında, ilk 300 genin daha iyi sonuç aldığı görülmektedir. Benzer şekilde test kümesi hatası ve seçilen gen sayısı birlikte dikkate alındığında Stjude için, filtreleme işlemi sonucunda seçilen ilk 750 öznitelik ve Glioma için ise ilk 500 özniteliğin, PrBABC'nin gen seçimi işlemi için yeterli olduğu görülmektedir.

Çizelge 7.6'da tez kapsamında önerilen IBitABC, exBitABC ve PrBABC yöntemlerinin performansları mikrodizi verikümeleri üzerinde, test kümesi hatası, seçilen gen sayısı ve çalışma süreleri bakımından karşılaştırılmıştır. Mikrodizilerde, PrBABC-3F kapsamında yapılan filtreleme işleminin etkisini gözlemlemek için sonuçlar, test kümesi hatası ve seçilen gen sayıları bakımından PrBABC-3F ile de karşılaştırılmıştır. PrBABC-3F için verilen sonuçlar, üçlü filtreleme yöntemi ile filtrelenen ilk 1500 genden oluşan veri kümelerinden elde edilen sonuçlardır. exBitABC için azami iterasyon sayısı 200 olarak belirlenmiştir. exBitABC, her adımda birden fazla uygunluk fonksiyonu tüketen bir yöntem olduğu için, PrBABC ve IBitABC'nin durma kriteri, exBitABC'nin her bir mikrodizi veri kümesi için tükettiği uygunluk fonksiyonu sayısı olarak belirlenmiştir. PrBABC-3F'de 100 iterasyon uygulanmıştır ancak PrBABC, IBitABC ve exBitABC herhangi bir filtreleme yapılmadan, tüm genleri içeren mikrodizilerde uygulandığı için, PrBABC-3F'nin iterasyon sayısı artırılmamış ve exBitABC'de tüketilen uygunluk fonksiyonu sayısına göre bir durma kriteri sağlanmamıştır. PrBABC-3F'nin çalışma kriterleri diğer yöntemlerden farklı olduğu için, PrBABC-3F ile karşılaştırma sadece test kümesi hatası ve seçilen gen sayısı bakımından yapılmış olup çalışma süresi göz önüne alınmamıştır. Çizelge 7.6'da verilen tüm yöntemler için popülasyon büyüklüğü 50 olarak belirlenmiştir.

Çizelge 7.6'da verilen sonuçlara göre tüm mikrodizilerde hem test kümesi hatası hem de seçilen gen sayısı bakımından en iyi sonuçları elde etmiştir. PrBABC-3F'nin diğer tüm yöntemlerden daha az iterasyon yürütmesine rağmen çok daha iyi sonuç almasından anlaşılmaktadır ki filtreleme işlemi gereklidir. Çalışma süresi bakımından ise PrBABC-3F iterasyon sayısı bakımından diğerlerinden farklı olduğu için karşılaştırmaya dahil edilmemiş, diğer yöntemler içinde de exBitABC tüm

Çizelge 7.5 Evrimsel yöntemlerin seçilen gen sayılarına göre karşılaştırılması

		CLL-SUB (12626)	CNS (7129)	DLBCL (7129)	Glioma (12625)	Leuk2 (12626)	Lung (12625)	Prostate (12625)	Stjude (12625)	WBC (24482)
1500	Pr BABC	**98,5 ±2,31**	**103,3 ±19**	**118,15 ±11,6**	113,35 ±12,7	**121 ±13,8**	**110,7 ±10,5**	**114 ±16,16**	**98,85 ±2,14**	**58,35 ±2,19**
	GA	148,8 ±11,98	155,5 ±12,9	151,65 ±16,14	225,7 ±18,18	217,1 ±16,13	226,8 ±13,89	148,25 ±11,46	228,2 ±12,4	162,5 ±10,18
	Bin PSO	261,9 ±181,6	183,2 ±25,93	338,9 ±26,96	**2,9 ±4,86**	325,15 ±221,4	436,6 ±20,84	299,7 ±28,61	486,5 ±136,4	318,4 ±22,61
	Bin DE	241,6 ±13,15	236,3 ±13,49	230,7 ±14,19	229,3 ±14,36	231,9 ±14,64	235,1 ±16,45	227,6 ±9,94	247,8 ±17,86	245,5 ±15,63
1000	Pr BABC	**59,55 ±1,89**	**61,4 ±1,39**	**79,1 ±7,88**	**75,65 ±2,81**	**82,95 ±8,87**	**71,85 ±8,23**	**74,1 ±9,16**	**65,1 ±1,41**	**39,05 ±1,4**
	GA	99,15 ±9,02	154,8 ±12,91	99,35 ±10,01	102,35 ±9,9	152 ±8,84	152,25 ±10,8	103,1 ±9,54	114,7 ±12,32	106,2 ±10,28
	Bin PSO	256,4 ±14,16	88,1 ±15,31	192,25 ±167,37	40 ±12	312,45 ±15,1	302,2 ±11,5	306,95 ±152,8	333,65 ±67,5	268,7 ±11,73
	Bin DE	241,6 ±13,1	152,4 ±11,94	156,55 ±13,55	149,85 ±14,8	152,1 ±10,82	156,2 ±10,78	152 ±12,3	163,65 ±10,19	163,75 ±14,6
750	Pr BABC	**45,45 ±1,1**	**45,1 ±6,07**	**60,35 ±7,4**	56,35 ±1,07	**61,95 ±4,39**	**53,95 ±9,34**	**53,05 ±1,14**	**47,6 ±15,29**	**28,4 ±1,01**
	GA	114,65 ±9,12	114,3 ±10,2	74,45 ±8,69	111,45 ±6,21	77,9 ±8,4	114,35 ±10,17	77,4 ±6,92	84,55 ±9,25	124,9 ±10,3
	Bin PSO	181,05 ±93,1	71,4 ±10,83	134,31 ±88,9	**34,45 ±79,11**	219,85 ±84,14	187,2 ±96,01	79,05 ±10,34	245,4 ±56,77	150,2 ±10,57
	Bin DE	118,65 ±9,17	115,55 ±8,49	113,2 ±11,52	114,55 ±8,49	111,55 ±7,49	115,35 ±10,12	109,7 ±7,78	124,95 ±7,23	0126,6 ±11,01
500	Pr BABC	**33,25 ±6,7**	**31,65 ±9,6**	**38,25 ±8,04**	36,2 ±6,45	**38,3 ±6,05**	**38,95 ±7,6**	**37,25 ±8,87**	**33,45 ±6,06**	**20,85 ±7,73**
	GA	75,45 ±8,39	51,4 ±5,04	48,5 ±5,73	75,15 ±9,67	52,15 ±7,52	50,8 ±6,66	75,85 ±7,84	59,85 ±9,72	57,75 ±6,77
	Bin PSO	137,55 ±56,63	47,5 ±64,06	110,4 ±9,08	**12,15 ±40,13**	136,95 ±58,69	159,35 ±57,1	115,45 ±69,95	187,4 ±32,9	126,1 ±55,93
	Bin DE	80,3 ±10,03	77 ±6,54	77,2 ±9,93	75,13 ±7,05	78 ±7,04	74 ±7,26	77,1 ±7,12	83,6 ±8,62	85,75 ±9,89
300	Pr BABC	**23,35 ±7,6**	**24,45 ±5,7**	**23,55 ±4,01**	**21,25 ±4,35**	**24,4 ±4,31**	**25,45 ±4,37**	**27,15 ±5,51**	**28 ±4,77**	**15,55 ±5,36**
	GA	48,55 ±6,06	30,2 ±4,81	31,15 ±4,02	46,35 ±5,56	29,9 ±4,58	30,2 ±5,6	44,1 ±6,72	55,45 ±7,4	50,2 ±6,06
	Bin PSO	79,6 ±40,5	42,55 ±43,1	59,6 ±51,3	28,2 ±4,69	94,6 ±46,28	82,05 ±35,93	136,1 ±81,1	106,35 ±13,9	96,4 ±18,9
	Bin DE	47,35 ±4,89	46,3 ±8,77	45,05 ±6,76	43,2 ±5,86	47,15 ±6,22	45,2 ±7,79	45,65 ±7,61	50 ±6,5	50,15 ±5,78
100	Pr BABC	**8 ±2,49**	**9,1 ±3,14**	**9,65 ±2,85**	7,8 ±2,9	**8,4 ±2,66**	**8,95 ±2,44**	**8,35 ±3,52**	**15,95 ±2,8**	**10,8 ±2,95**
	GA	15,95 ±5,07	19,05 ±2,83	15,9 ±4,05	14,45 ±3,88	15,25 ±4,22	12,8 ±3,46	14,3 ±2,9	24,4 ±3,4	21,65 ±4,25
	Bin PSO	30,2 ±8,42	20,7 ±14,71	16,4 ±17,6	**6,75 ±11,71**	29,05 ±14,74	34,4 ±10,83	23,65 ±15,67	40 ±6,45	36,9 ±8,86
	Bin DE	17,1 ±3,27	14,75 ±3,29	15,6 ±3,31	17,3 ±4,09	16,15 ±3,69	15,95 ±3,10	15,7 ±3,54	22,75 ±2,51	18,75 ±3,86

Çizelge 7.6 Tez kapsamında önerilen ikili YAK yöntemlerinin karşılaştırılması

		CLL-SUB	CNS	DLBCL	Glioma	Leuk2	Lung	Prostate	Stjude	WBC
PrBABC-3F	TestHata	**0,259 ±0,1**	**0,242 ±0,082**	**0,031 ±0,043**	**0,14 ±0,1**	**0,03 ±0,04**	**0,042 ±0,03**	**0,1 ±0,07**	**0,125 ±0,04**	**0,325 ±0,05**
	#Gen	**98,5 ±23,11**	**103,3 ±19**	**118,15 ±11,64**	**113,35 ±12,72**	**121 ±13,8**	**110,75 ±10,5**	**114 ±16,16**	**98.85 ±21.39**	**58,35 ±21,88**
	Sure(sn)	84,49 ±1,19	92,82 ±6,34	94,59 ±4,02	99,76 ±9,08	92,9 ±4,04	94,59 ±1,31	469,26 ±3,78	118,81 ±8,83	105,96 ±3,15
PrBABC	TestHata	0,29 ±0,08	0,458 ±0	0,069 ±0,07	0,255 ± 0,09	0,083 ±0,06	0,045 ±0,02	0,21 ±0,11	0,154 ±0,04	0,378 ±0,05
	#Gen	1016,05 ±156,63	632,2 ±108	699,3 ±56,21	1193,5 ±129,9	1230,4 ±90,64	1241,35 ±85,48	1160,85 ±128,79	862,05 ±150,54	1291,5 ±430,91
	Sure(sn)	1284,31 ±10,7	929,08 ±4	923,71 ±69,43	1143,74 ±21,9	1192,08 ±19,94	1933,65 ±54,4	1277,68 ±31,34	2419,61 ±77,61	3008,43 ±126,46
exBitABC	TestHata	0,32 ±0,1	0,392 ±0,11	0,094 ±0,08	0,205 ±0,13	0,093 ±0,09	0,046 ±0,03	0,205 ±0,11	0,176 ±0,06	0,374 ±0,06
	#Gen	465,5 ±96,84	288,85 ±88,05	276,95 ±12,81	491,55 ±15,66	503,6 ±6,26	512,35 ±131,65	479,05 ±23,21	555 ±236,95	908,9 ±339,5
	Sure(sn)	**766,49 ±23,86**	**623,75 ±3,74**	**646,17 ±20,28**	**728,53 ±5,3**	**731,87 ±5,42**	**1004,9 ±17,16**	**745,57 ±3,5**	**1107,1 ±15,37**	**1503,6 ±85,94**
IBitABC	TestHata	0,341 ±0,06	0,392 ±0,11	0,066 ±0,05	0,23 ±0,08	0,047 ±0,05	0,042 ±0,03	0,138 ±0,06	0,155 ±0,03	0,36 ±0,06
	#Gen	1180,15 ±38,23	684,15 ±25,94	713,05 ±22,01	1246,15 ±26,86	1257,1 ±37,8	1250,65 ±44,4	1217,35 ±47,79	1145,35 ±48,23	1978,4 ±70,75
	Sure(sn)	925,02 ±6,13	686,81 ±10,73	683,141 ±14,98	780,69 ±3,67	829,98 ±5,99	1618,66 ±36,45	902,32 ±36,86	2337,9 ±33,6	3023,56 ±17,82

mikrodizilerde en kısa sürede sonuç veren yöntem olmuştur. exBitABC, PrBABC ve IBitABC filtre uygulanmadan kendi içinde kaşılaştırıldığında IBitABC'nin test kümesi hatası bakımından, exBitABC'nin de seçilen gen sayısı ve çalışma süresi bakımından daha iyi olduğu sonuçlardan görülmektedir.

Çizelge 7.7'de, PrBABC ile elde edilen sonuçlar literatürdeki evrimsel yöntemler ile mikrodizilerde gen seçimi yapılmış çalışmalar ile sınıflandırma doğruluğu ve seçilen gen sayısı bakımından karşılaştırılmıştır. Yöntemlerin seçmiş olduğu gen sayıları parantez içinde verilmiştir. Uygulamadaki farklılıklar nedeniyle yöntemler arasında birebir karşılaştırma yapmak mümkün değildir. Ancak, bir fikir vermesi açısından sonuçların verilmesi tercih edilmiştir. PrBABC için kullanılan bir mikrodizi ilgili çalışmada kullanılmadıysa, o hücre boş bırakılmıştır. Ayrıca, mikrodiziler arasında bir takım farklılıklar vardır. PrBABC için kullanılan mikrodiziler, Çizelge 7.1'de belirtilen kaynaklardan herhangi bir normalizasyon veya öznitelik seçimi olmadan ham veri olarak alınmıştır. Aynı isimdeki veri kümeleri farklı kaynaklarda farklı gen, sınıf veya örnek sayısı ile bulunabilmektedir. Örneğin, bir veri kümesi başka bir kaynakta, normalize edilmiş veya öznitelik seçimi yapılarak gen sayısı azaltılmış olarak bulunabilmektedir. Bu durumlar için, karşılaştırma yapılacak veri kümelerindeki gen ve örnek sayıları arasındaki farklar göz ardı edilmiş, sınıf bilgileri aynı olan mikrodiziler dikkate alınmıştır. Çizelge 7.7'de verilen mikrodizilerden sadece DLBCL, PrBABC için kullanılan veri kümesi ile aynı sayıda gen ve örnek içermekte olup,

diğerleri için bu özellikler farklılık göstermektedir. Ayrıca, kullanılan yöntemlerin parametreleri, eğitim ve test kümesi oranları, sınıflandırıcılar gibi farklılıklar da bulunduğundan birebir karşılaştırma yapmak mümkün olmayıp, sonuçlar genel bir bilgi vermesi açısından verilmiştir.

PrBABC'de mikrodiziler eğitim, validasyon ve test olmak üzere 3 kümeye ayrılmıştır. Optimizasyon aşamasında, elde edilen gen alt kümeleri eğitim kümesi ile eğitilmiş, doğrulama kümesi ile modelin geçerliliği ve test kümesi ile de başarısı ölçülmüştür. Çizelge 7.7'deki bir çok çalışmada ayrı bir test kümesi kullanılmamış olup, sınıflandırıcı doğruluğu eğitim kümesinde çapraz geçerleme yapılarak ölçülmüştür. Ayrıca, diğer tüm metodlarda da belirli bir kritere göre sıralanan genlerden ilk n tanesi alınarak o gen kümesi üzerinde öznitelik seçimi yapılmıştır. n değeri 10-100 arasında değişmektedir. Bu nedenle Çizelge 7.7'de PrBABC için verilen sınıflandırıcı doğruluğu değerleri, üçlü filtreleme metodu ile seçilen ilk 50 gen için, doğrulama kümesi ile elde edilen sonuçlardır. Genellikle ilk 10, 20, ..., 100 gen için öznitelik seçimi yapılmış ve bu kümeler ile elde edilen ortalama doğruluk değerleri verilmiştir. Bazı çalışmalarda ise belirli bir gen sayısı verilmediğinden Çizelge 7.7'de ilgili yöntemlerin seçtiği gen sayısı belirtilmemiştir.

Çizelge 7.7 Literatürdeki yöntemlerin sınıflandırıcı doğruluğu ve gen sayısına göre karşılaştırılması

Metot	Cll-Sub	DLBCL	Glioma	Pros. Can.	CNS	Stjude	Lung Can	Leuk2	WBC
PrBABC	**0,973 (6,6)**	1 (5,4)	**1 (4,65)**	0,99 (5,6)	**1 (5,3)**	**0,935 (12,7)**	**0,99 (6,95)**	1 (5,8)	**0,836 (6,65)**
BCO [40]	-	**1(3,1)**	-	**1(7)**	-	-	-	**1(3,5)**	-
RLR [38]	0,747	0,93	-	-	-	0,854	-	-	-
PLSDR5 [39]	0,821	0,928	-	-	-	-	0,941	-	-
[34]	-	-	0,793 (25)	0,886 (50)	-	-	-	-	-
MOEDA [44]	-	0,99 (4)	-	0,96 (12)	-	-	0,96 (25)	1 (4)	-
LM [36]	-	0,958	-	0,947	-	-	-	-	-
mRMR-CS [51]	-	-	-	-	0,714 (7)	-	-	-	-
[37]	-	0,929	-	0,901	-	-	-	0,921	-
mRMR-ABC [49]	-	-	-	-	-	-	-	1 (20)	-
GBC [50]	-	-	-	-	-	-	-	1(8)	-
MGSACO [48]	-	-	-	0,731 (20)	-	-	0,857 (20)	-	-
[41]	-	1(4)	-	-	-	-	-	1(11)	-
IG-GA [46]	-	1 (107)	-	0,961 (343)	-	-	0,956 (2101)	0,986 (782)	-
MCSO [53]	-	-	-	0,996 (50)	-	-	-	0,817 (10)	-

BCO [40] ikili bakteri algoritması tabanlı bir yöntem olup algoritmada popülasyon büyüklüğü 50, maksimum iterasyon sayısı ise 100'dür. Sınıflandırıcı olarak 5-NN kullanılmıştır. Yazarlar bir filtreleme yöntemini kullanmak yerine,

bakteri algoritmasının seçebileceği gen sayısını önceden tanımlı bir değer ile sınırlandırmışlardır. Bu değer veri kümesine göre farklılık gösterirken, maksimum aldığı değer 50'dir. RLR [38], lojistik regresyon tabanlı bir öznitelik seçim yöntemi olup sınıflandırıcı olarak DVM kullanılmıştır. Çizelge 7.7'de verilen sonuçlar farklı filtreleme metodları ile sıralanmış ilk 1, 2, ...,50 gen ile öznitelik seçimi yapılarak elde edilmiş ortalama doğruluk sonuçlarıdır. PLSDR5 [39], lojistik regresyon tabanlı bir diğer çalışmadır ve Kısmi En Küçük Kareler (Partial Least Squares) ile öznitelik çıkarımı yapılmıştır. Çizelge 7.7'de verilen sonuçlar, ilk 10, 20, ..., 100 geni kullanarak, BBA ile elde edilmiş ortalama doğruluk sonuçlarıdır. Aziz vd. [34], ayırt edici genleri BBA ve Bulanık Geriye Doğru Öznitelik Eleme (Fuzzy Backward Feature Elimination) ile belirlemişlerdir. Elde edilen gen alt kümeleri DVM ve Naive Bayes ile sınıflandırılmıştır. Çizelge 7.7'de, iki sınıflandırıcıdan iyi olanın sonuçları verilmiştir. MOEDA [44], çok amaçlı sezgisel bir yöntemdir. Mikrodiziler, dağılımın tahmini tabanlı sezgisel bir yöntem olan mRMR ile filtrelenmiş ilk 1..50 genin bulunduğu veri kümesine uygulanmıştır. Popülasyon büyüklüğü 100 olarak belirlenmiş ve sınıflandırıcı olarak da DVM tercih edilmiştir. Sun vd. [36], gen seçimini Lagrange çarpanı ile yapmıştır. İlk 100 gen, Naive Bayes, K-NN, Rastgele Orman, ve Sınıflandırma ve Regresyon Ağacı (CART) yöntemleri ile sınıflandırılmış olup, Çizelge 7.7'de en başarılı sınıflandırıcı sonucu verilmiştir. mRMR-CS'de [51], Guguklu Arama, PSO ve YAK algoritmaları, mRMR ile filtrelenmiş mikrodizilere uygulanarak karşılaştırılmıştır. Popülasyon büyüklüğü 50 olup, azami iterasyon sayısı ise 1000'dir. En başarılı sonuçlar Guguklu Arama ile elde edildiğinden, Çizelge 7.7'de Guguklu Arama'nın elde ettiği sonuçlar verilmiştir.

Mortazavi vd. [37], mikrodizi gen sayısını Fisher Oranı veya Karşılıklı Bilgi (Mutual Information) ile azaltmış ve ardından Kooperatif Oyun Teorisi (Cooperative Game Theory) ile genleri ağırlıklandırmıştır. mRMR-ABC'de [49], veri kümeleri mRMR ile filtrelenmiş ve Yapay Arı Kolonisi ile de öznitelik seçimi yapılmıştır. Popülasyon büyüklüğü 80 ve maksimum iterasyon sayısı ise 100'dür. Her mikrodizi için mRMR ile seçilen en iyi 50, 100, 150, ..., 400 gen DVM ile sınıflandırılmıştır. YAK, %100 sınıflandırıcı doğruluğu elde eden gen alt kümesi için öznitelik seçimi yapmıştır. GBC [50] ise, benzer şekilde mRMR ile YAK algoritmasını hibridize etmiş ve önceki çalışma ile aynı parametreleri kullanmıştır. Farklı olarak, burada, genetik operatörler ile ikilileştirilmiş bir YAK algoritması önerilmiştir. MGSACO [48] Karınca Kolonisi Optimizasyonu tabanlı bir gen seçim yöntemi olup koloni büyüklüğü 100 ve azami iterasyon sayısı 50'dir. mRMR ile seçilmiş en iyi 10, 20, 30, ..., 100 gen için öznitelik seçimi yapılmıştır. Çizelge 7.7'de, DVM, Naive Bayes ve Karar Ağacı ile sınıflandırılan 20 gen içeren gen alt kümesinden en iyi sonucu veren algoritma olarak sunulmuştur. Yang vd. [41] mikrodizileri Bilgi Kazancı ve KKDF ile filtrelemiş ve geliştirilmiş bir ikili

PSO yöntemi ile gen seçimi yapmıştır. Popülasyon büyüklüğü 30 ve azami iterasyon sayısı 100'dür. Mikrodiziler K-NN ve DVM ile sınıflandırılmıştır. Çizelge 7.7'de daha başarılı olduğu için K-NN ile elde edilmiş doğruluk değerleri verilmiştir. Yazarlar bir sonraki çalışmalarında [46] BK ile genleri ağırlıklandırmış ve BK değeri 0 olan tüm genleri elemişlerdir. Kalan genler için de GA ile öznitelik seçimi yapmışlardır. Popülasyonda 30 birey bulunmaktadır ve nesil sayısı 100'dür. Sınıflandırıcı olarak ise 1-NN kullanılmıştır. MCSO [53], Guguklu Arama algoritmasının gelişmiş bir versiyonunu önermişlerdir. 10, 20, ..., 100 gen içeren 10 öznitelik alt kümesi önerilen yöntem ile seçilmiş ve K-NN ile sınıflandırılarak en iyi olanı seçilmiştir. Çizelge 7.7'de de en iyi sonuç verilmiştir.

Çizelge 7.7'deki sonuçlara göre BCO, DLBCL, Prostat Kanseri ve Leukemia2 veri kümeleri için daha iyi sonuçlar vermiştir. DLBCL ve Leukemia2 için PrBABC, BCO ile aynı sınıflandırıcı doğruluğunu elde etmiştir, ancak seçtiği gen sayısı daha fazladır. Prostat Kanseri için PrBABC daha az gen seçmiş olmasına karşın, PrBABC'nin seçtiği gen alt kümesi %99 doğruluk ile sınıflandırma yaparken, BCO %100 doğruluk ile sınıflandıran bir gen alt kümesi seçmiştir. Ancak görüldüğü üzere doğruluk değerleri arasındaki fark çok küçüktür. Diğer veri kümeleri için, PrBABC sonuçları doğruluk ve gen set büyüklüğü değerleri bakımından diğer yöntemlerden daha iyi performans göstermiştir.

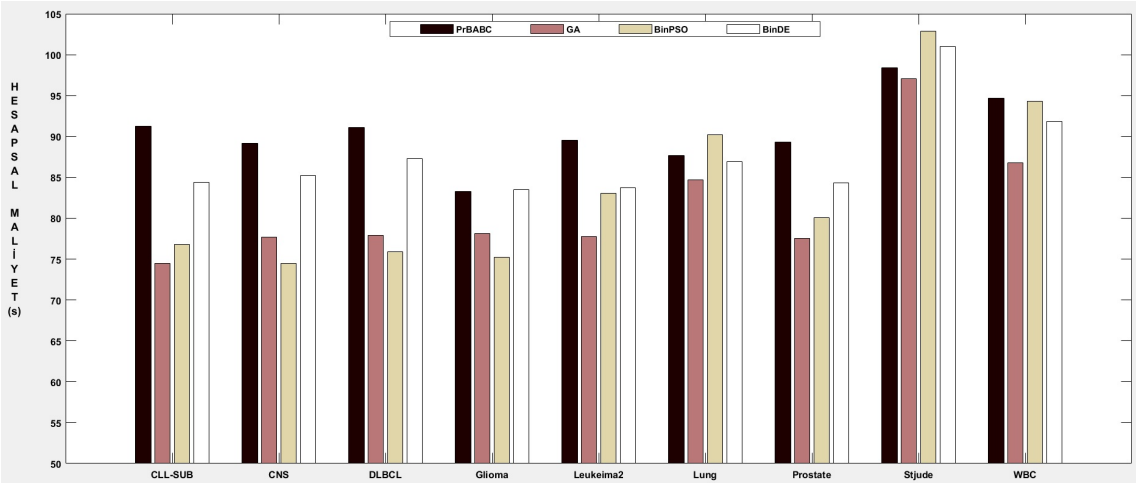
Mikrodizi veri kümelerinde, her bir öznitelik bir geni temsil etmektedir. Mikrodizilerde öznitelik seçimi yapmaktaki amaç, bir hastalık ile en çok ilgili genlerin tespit edilmesidir. Her bir hastalığı en iyi ayırt eden genleri belirleyebilmek için, algoritmalar tarafından seçilen gen alt kümeleri analiz edilmiştir. Her veri kümesi, dört yöntem kullanılarak 20'şer defa çalıştırıldığı için toplamda farklı doğruluk ve gen sayısı değerlerine sahip, 80 gen alt kümesi elde edilmiştir. Bu gen alt kümelerinde en çok bulunan 10 gen için ilgili genin sembolü veya probe kimliği Çizelge 7.8'de verilmiştir. Parantez içindeki sayılar, ilgili genlerin veri kümesindeki kaçıncı öznitelik olduğunu göstermektedir.

Algoritmalar, MATLAB R2017a kullanılarak 16 GB Ram ve bir Intel (R) Çekirdek (TM) i7-3630QM 2.4 GHz CPU'lu bir PC'de gerçekleştirilmiştir. Şekil 7.4'de, algoritmalar üçlü filtreleme ile sıralanmış ilk 100 genle başlatıldığında, çalışma süreleri karşılaştırmalı olarak verilmiştir. Bu sonuçlara göre, PrBABC'nin hesapsal maliyeti BinDE'nin, GA'nın hesapsal maliyeti ise çoğu veri kümesinde BinPSO'nun maliyetine yakındır. Genel olarak, PrBABC altkümeleri seçmek için daha fazla zamana ihtiyaç duymuştur, ancak algoritmalar arasındaki zaman farkları yaklaşık 5-15 saniye kadardır.

Çizelge 7.8 En çok seçilen genler

Mikrodizi	Gen Sembolü / Affymetrix Numarası
CLL-SUB	RABGGTA (1), ABI2(46), IL1A(85), POU6F1 (45), CYP1A2 (90), POLG (17), RELA (51), THBS4 (34), TOP1 (35), RAG1 (86)
Dlbcl	PARK7 (618), PYY (260), H1FX (659), GDF15 (16), AFFX-BioB-M_at(121), AFFX-LysX-M_at (162), LGALS9 (44), MARCKS (203), IRF4 (675), GAPDH(149)
Glioma	BMP2 (126), MAP2K1 (145), RAB40B / RAB40A / RAB40AL (89), BDNF (98), PTPRE (167), ITGB3 (195), DDB2 (265), IFIT5 (52), 1090_f_at(101)
Pros. Can.	RAB1A (83), EIF2AK2 (10), BDNF (98), CIB1 (24), RELA (51), RAB40B / RAB40A / RAB40AL (89), CSF2RB (96), EN2 (148), DHFR (197), CXCR5 (5)
CNS	C4orf8 (10), AC002486_at (67), MYH11 (85), C4orf9 (14), DNM1L (77), ARPC1B (98), MDM4 (101), OGG1 (7), SH3BP2 (11), AQP7 (42)
Stjude	TAF1 (584) IL12RB1 (74) TGFB2 (285) THRB (463) DYRK1A (554) PRKCI (679) IFIT5 (52) IGF1R (361) CYP4A11 / CYP4A22 (423) NFKBIA (499)
Lung Can.	POU6F1 (45), CYP2C8 (194), TERF1 (369), 1416_g_at (450), YWHA (257), GATA2 (81), EDN2 (103), IL2RB (394), UBE2B (253), GSTM5 (315)
Leuk2	CD44 (140), RELA (300), YWHAE (14), GAB1 (275), CASP2 (264), PSMB2 (343), IFNGR1 (43), FLT3LG (76), ISG15 (119), RARA (372)
WBC	GOT2(236), LAMC1(298), CKB(412), CANX(89), MLEC(145), PPP1CA(374), SRPRA(445), ATP6V0E1(117), LITAF(234), UBA7 / MIR5193 (6)

Bu çalışma kapsamında, mikrodizi verileri için YAK algoritması tabanlı, etkili bir gen seçim yöntemi önerilmiştir. Mikrodiziler için literatürde yaygın kullanılan normalizasyon yöntemlerinden MAS5, RMA ve GcRMA yöntemleri dokuz veri kümesine uygulanarak sonuçlar karşılaştırılmış ve RMA yöntemi ile normalize edilmiş veri kümelerinin daha başarılı sonuçlar elde ettiği görülmüştür. Gen seçimi işlemi için ise filtre ve sarmalayıcı içeren hibrid bir yöntem uygulanmıştır. Genler Bilgi Kazancı, Korelasyon Katsayısına Dayalı Filtreleme ve ReliefF filtre metodları ile ağırlıklandırılarak, her bir yöntem ile ilk n gen seçilerek toplamda 3n genden oluşan gen kümeleri elde edilmiştir. Filtreleme metotlarının tek başlarına ve birlikte uygulanması ile elde edilen sonuçlar karşılaştırıldığında, öznitelikleri farklı kriterlere göre sıralayan 3 farklı filtreleme metodunun bir arada kullanılmasının başarıyı artırdığı görülmüştür. Sarmalayıcı yöntem olarak ise, Olasılıksal bir İkili YAK algoritması önerilmiştir. Önerilen yöntemde iki farklı öğrenme stratejisi kullanılmış ve algoritmanın hangi yöntemi seçeceğine olasılıksal olarak karar vermesi sağlanmıştır. Elde edilen sonuçlar Genetik Algoritma, İkili Parçacık Sürü Optimizasyonu ve İkili Diferansiyel Evrim algoritması ile karşılaştırılmış ve öğrenme stratejisini olasılıksal olarak belirleyen yöntemin diğerlerinden daha iyi performans gösterdiği görülmüştür.



Şekil 7.4 Algoritmaların hesapsal maliyetlerine göre karşılaştırılması

Ayrıca, yöntemlerin seçtiği genler analiz edilerek, her bir mikro dizin için yöntemlerin en çok seçtiği genler de belirlenmiştir [1].

[1] Bu çalışma 'Journal of Universal Computer Science' dergisinde yayınlanmıştır

8
Sonuçlar ve Öneriler

Geliştirilecek ikili YAK algoritmasının temellerini oluşturabilmek için, literatürde mevcut İkili Yapay Arı Kolonisi algoritmalarından 15 tanesi uygulanmış ve farklı büyüklüklerdeki on veri kümesi üzerinde, öznitelik seçimi problemine uygulanmıştır. İkili YAK algoritmaları ve dört evrimsel hesaplama tabanlı yöntem birbirleri ile sınıflandırma performansı, seçilen öznitelik alt kümesi ve hesapsal maliyetleri bakımından karşılaştırılmıştır. Algoritmaların çoğu, makul sınıflandırma sonuçları ile gereksiz öznitelikleri belirleyebilmişlerdir. GA, BPSO, kuantum tabanlı bir yöntem olan QBPSO ve Parçacık Sürü Optimizasyonunun hız tabanlı bir versiyonu olan NBPSO, öznitelik seçimi problemi için popüler yöntemler olmasına rağmen, hiçbiri İkili Yapay Arı Kolonisi yöntemlerinden daha iyi sonuç vermemiştir. Bu nedenle, İkili Yapay Arı Kolonisinin öznitelik seçimi problemleri için uygun bir algoritma olduğunu iddia etmek mümkündür. İkili Yapay Arı Kolonisi metotları birbirleri ile kıyaslandığında, genetik operatör tabanlı GBABC eğitim kümeleri için daha iyi sonuçlar almış, ancak test kümesi performansının düşük çıkmasından veriyi genelleme yeteneğinin sınırlı olduğu sonucuna varılmıştır. Ayrıca, diğer yöntemlere göre hesaplama maliyetinin de daha yüksek olduğu görülmüştür. Yöntemlerin test kümesi hataları karşılaştırıldığında, hiçbir yöntemin diğerlerine kıyasla büyük bir üstünlük sağlayamadığı görülmüştür. Öznitelik azaltma performansları karşılaştırıldığında ise, bitişlem operatörleri tabanlı bir yöntem olan BitABC, veri kümelerinin yarısı için makul sürelerde çok iyi bir performans sergilemiştir. Genel olarak performanslar karşılaştırıldığında ise BitABC'nin uygulanan yöntemler arasında oldukça başarılı olduğu görülmüştür. Karşılaştırılan yöntemlerin güçlü ve zayıf yanları incelenerek geliştirilecek yöntemler için temel oluşturmuştur.

Öznitelik seçimi işleminde, belirlenen öznitelik alt kümelerinin performanslarını değerlendirmek için bir sınıflandırma/kümeleme algoritmasına ihtiyaç duyulmaktadır. Sınıflandırıcılar, verileri farklı özelliklerine göre değerlendirebilmektedir. Tez kapsamında farklı veri kümeleri ile çalışıldığından, daha çok veri kümesinde başarılı olacak snıflandırıcının seçilmesi önemlidir. Bu amaçla, BitABC algoritması sekiz

veri kümesi ile çalıştırılarak, altı farklı sınıflandırıcının performansı karşılaştırılmıştır. Verilerin farklı özelliklerinden yararlanması için, Bayes Teoremi tabanlı geliştirilen Naive Bayes, entropi bazlı karar veren Karar Ağacı, yapay sinir ağı yöntemlerinden Olasılıksal Yapay Sinir Ağı, karar ağacı ve topluluk öğrenmesi tabanlı Rastgele Orman, veriler arasındaki mesafeyi baz alan K-En Yakın Komşu ve çekirdek tabanlı bir sınıflandırıcı olan Destek Vektör Makineleri'nin performansları, test kümesi hatası ve seçilen öznitelik sayısı bakımından karşılaştırılmıştır. Sonuçlar incelendiğinde, K-NN yönteminin genel olarak daha yüksek doğruluk ve daha az öznitelik ile sınıflandırma yapabildiği gözlemlenmiştir. Ek olarak, optimizason algoritmasının iterasyonları boyunca, birçok öznitelik alt kümesinin performansının değerlendirilmesi gerekeceği göz önüne alındığında, sınıflandırma yönteminin hızlı olmasının avantaj sağlayacağı açıktır. Sınıflandırıcılar aynı optimizasyon algoritması kapsamında uygulandığından, hesapsal maliyetler arasındaki fark, sınıflandırma yöntemlerinin hızına bağlı olacaktır. Sonuçlar, çalışma süreleri bakımından da karşılaştırıldığında, K-NN'in daha hızlı sınıflandırma yapabildiği görülmektedir.

BitABC algoritmasında kullanılan bit işlem operatörlerinin, hızlı ve efektif sonuçlar verdiği görüldüğünden, ilk aşamada geliştirilecek yöntem için BitABC temel alınmıştır. Algoritmanın kullandığı bit işlem operatörlerinin, hızlı çözüm üretmeyi sağladığı görülmüştür. Olası çözümlerin ikili vektörler ile temsil edildiği göz önüne alındığında, bit işlem operatörleri vektörel işlemleri gerçekleştirebilmektedir. Birçok sürü zekası algoritmasında olduğu gibi, YAK algoritmasında da sürünün en iyi bireyi saklanmakta ve her iterasyonda kendisinden daha iyi bir birey çıkması halinde güncellenmektedir. BitABC algoritmasının ürettiği çözümler incelendiğinde, çözümlerin arama uzayında büyük adımlar ile belirlendiği görülmüş ve lokal aramanın yetersiz olduğu gözlemlenmiştir. Algoritmaya eklenecek bir lokal arama fonksiyonunun performans üzerindeki etkisini araştırmak için, ilk olarak sürünün en iyi bireyi etrafında uygulanacak şekilde bir lokal arama fonksiyonu eklenmiştir. Lokal aramanın ne şekilde yapılacağını belirlemek için, iki farklı yöntem denenmiş ve 10 veri kümesi üzerinde sonuçlar karşılaştırılmıştır. Ayrıca, önerilen yöntemin mevcut algoritmaya etkisini ölçmek için, sonuçlar BitABC ile de karşılaştırılmış ve sürünün en iyi bireyinin etrafında yapılacak bir lokal aramanın hesapsal maliyeti etkilemeden, sınıflandırıcı doğruluğunu artırdığı sonucuna varılmıştır.

Sürünün en iyi bireyinin etrafında yapılan lokal arama başarıyı artırmıştır ancak algoritmada olası çözümler işçi ve gözcü arı aşamalarında üretilmektedir. BitABC algoritması, kullandığı yeni kaynak üretme stratejisi ile yeni kaynak üretirken büyük adımlar atmakta ve bu durum bir kaynağın yakın çevresinde optimal bir çözüm varsa onun yerinin kaybedilmesine neden olmaktadır. Bu durum göz önüne alınarak, yeni kaynak üretme stratejisi, bir kaynak için yeni çözüm önerirken, ilk önce kendi

lokal bölgesinde küçük bir arama yapması, eğer iyileştiremezse büyük bir adım atarak daha uzak bir bölgeye geçmesini sağlayacak şekilde belirlenmiştir. Ayrıca, algoritmanın başında konumları rastgele belirlenen kaynakların, birbirleriyle çok yakın veya çakışık olmalarını engellemek için arama uzayı, bölgelere ayrılmış ve her bir arı bir bölge içerisinde rastgele bir konumda ilklendirilmiştir. Böylece arıların çakışık bölgelerde arama yapmasının önüne geçmenin daha iyi performans sağlayacağı düşünülmüştür. Son olarak algoritmada, gözcü arı aşamasında kullanılan Rulet Tekeri Yönteminin, erken yakınsamaya neden olabileceğinden dolayı her bir iterasyon sonunda kaynakların mesafeleri hesaplanarak, birbirlerine çok yakın olan kaynaklardan daha kötü olanı yeniden ilklendirilerek popülasyonun çeşitliliğini artırmak hedeflenmiştir. Bu şekilde geliştirilen Ikili YAK algoritmasının performansını değerlendirmek için farklı büyüklüklerde 13 veri kümesi kullanılmıştır. Önerilen yöntemin elde ettiği sonuçlar, hem BitABC algoritmasının sonuçları ile hem de yaygın kullanılan sezgisel algoritmalardan Genetik Algoritma ve İkili Parçacık Sürü Optimizasyonu algoritmasının sonuçları ile karşılaştırılmıştır. Önerilen yöntemin test kümesi doğruluğu ve seçtiği öznitelik sayısı bakımından hem BitABC'den hem de diğer sezgisel yöntemlerden daha iyi sonuçlar verdiği görülmüştür. Ancak lokal arama için eklenen özelliklerin hesapsal maliyeti artırdığı tespit edilmiştir.

Mikrodizi verileri, hastalıklar ile ilgili genlerin belirlenmesi gibi birçok sebepten kullanılmaya başlanan gen ekspresyonu seviyelerini ölçmeye yarayan büyük boyutlu verilerdir. Mikrodiziler tipik olarak az sayıda örnek ve binlerce sayıda öznitelik içerirler. Veri kümesindeki her bir öznitelik, bir geni temsil etmektedir. Yani, mikrodiziler için öznitelik seçimi aynı zamanda araştırılan konu ile ilgili olan genlerin belirlenmesi anlamına geldiğinden oldukça önemlidir. Tez kapsamında, büyük boyutlu mikrodizi verilerinde gen seçimi yapmak üzere olasılıksal bir ikili ABC yöntemi geliştirilmiş ve performansı dokuz mikrodizi veri kümesinde test edilmiştir.

YAK algoritmasında yeni kaynak üretme yöntemi öğrenme stratejisini temsil etmektedir. Geliştirilecek yöntemin öğrenme stratejisini belirlemek amacıyla yapılan araştırmalarda görülmüştür ki, bir öğrenme yöntemi farklı veri kümelerinde farklı performanslar gösterebilmektedir. Buradan yola çıkılarak, iki öğrenme stratejisi belirlenmiş ve veri kümesindeki performansına göre adapte olacak şekilde olasılıksal ikili bir YAK algoritması önerilmiştir. Öznitelik seçiminin sınıflandırıcı doğruluğunu artırmak ve seçilen öznitelik sayısını azaltmak olarak iki amacı olduğu göz önüne alınarak, öğrenme stratejilerinden biri sınıflandırıcı doğruluğunu artırmaya yönelik diğeri ise, öznitelik sayısını azaltmaya yönelik olarak belirlenmiştir.

Normalizasyon işlemi önemli veri ön işleme adımlarından biri olup, değişkenlerin ortalama ve varyansları arasındaki farkın fazla olması durumunda, sonuçlara farklı

oranlarda etki etmesini önlemek amacıyla, tüm değişkenlerin belirli bir aralığa indirgenmesi işlemidir. Normalizasyon sonucunda veri kümesinde değişiklikler olacağından, gen alt kümesi belirleme işleminin de sonuçtan etkileneceği varsayılarak, mikrodizi verilerine özgü geliştirilmiş normalizasyon yöntemlerinden MAS5, RMA ve GcRMA veri kümelerine uygulanmıştır. Normalize edilmiş veri kümeleri, sınıflandırıcı doğruluğu, kümeleme saflığı ve silüet katsayısı değerleri bakımından karşılaştırılmıştır. Elde edilen sonuçlara göre, yöntemlerden herhangi biri diğerlerine göre çok büyük bir üstünlük göstermemiş olmasına karşın, RMA ile normalize edilen verilerin diğerlerinden daha iyi sonuçlar verdiği görülmüştür.

Çok büyük boyutlu verilerde, hibrit öznitelik seçimi yöntemlerinin kullanılması, arama maliyetini azaltmak için gereklidir. Hibrit yöntemlerde, veri kümesinin öznitelik sayısı öncelikle bir filtreleme yöntemi ile azaltılır, ardından sarmalayıcı bir yöntem ile öznitelik alt kümesi belirlenir. Çok çeşitli filtreleme yöntemleri bulunmaktadır ve bunlar öznitelikleri farklı bir takım özelliklerine göre sıralamaktadır. Tek bir yöntemin filtreleme yeteneğine bağlı kalmamak için, literatürde yaygın kullanılan üç filtreleme yöntemi: Bilgi Kazancı, Korelasyon Katsayısı Tabanlı Filtreleme ve REliefF ile mikrodizi verilerindeki genler sıralanmış, her bir yöntemin belirlediği en iyi n gen alınarak toplamda 3xn genden oluşan bir gen kümesi oluşturulmuştur. Bu şekilde 3 ayrı filtreleme yöntemi ile oluşturulmuş gen kümesine ve ilgili yöntemlerin ayrı ayrı uygulanarak elde edilen gen kümesine PrBABC uygulanarak performansları karşılaştırılmıştır. Sonuçlar, üçlü filtreleme yöntemi ile elde edilen gen kümesinde öznitelik seçimi yapmanın, sınıflandırıcı doğruluğu daha yüksek gen alt kümelerinin elde edilmesini sağladığını göstermektedir. Bu durumun, farklı yöntemlerin farklı özelliklere sahip genlerden oluşan bir küme elde etmesinden kaynaklandığı sonucuna varılmıştır.

Filtreleme işleminden sonra elde edilen veri kümelerine, diğer sürü zekası yöntemlerinden, genetik algoritma, ikili parçacık sürü optimizasyonu ve ikili diferansiyel evrim algoritmaları da uygulanarak gen seçimi yapılmıştır. PrBABC'nin elde ettiği sonuçlar ile karşılaştırıldığında, PrBABC'nin, daha az gen ile daha yüksek doğrulukta sonuç elde ettiği görülmüştür. Buradan yola çıkılarak, biri sınıflandırıcı doğruluğunu artırmaya diğeri ise alt kümedeki gen sayısını azaltmaya yönelik olarak geliştirilen öğrenme stratejilerini içeren yöntemin amacına ulaştığı görülmüştür.

Referanslar

[1] N. Sánchez-Maroño, A. Alonso-Betanzos, and M. Tombilla-Sanromán, "Filter methods for feature selection–a comparative study," in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2007, pp. 178–187.

[2] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[3] S. Oreski and G. Oreski, "Genetic algorithm-based heuristic for feature selection in credit risk assessment," *Expert systems with applications*, vol. 41, no. 4, pp. 2052–2064, 2014.

[4] C.-F. Tsai, W. Eberle, and C.-Y. Chu, "Genetic algorithms in feature and instance selection," *Knowledge-Based Systems*, vol. 39, pp. 240–247, 2013.

[5] R. Forsati, A. Moayedikia, R. Jensen, M. Shamsfard, and M. R. Meybodi, "Enriched ant colony optimization and its application in feature selection," *Neurocomputing*, vol. 142, pp. 354–371, 2014.

[6] S. Tabakhi, P. Moradi, and F. Akhlaghian, "An unsupervised feature selection algorithm based on ant colony optimization," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 112–123, 2014.

[7] Y. Zhang, D. Gong, Y. Hu, and W. Zhang, "Feature selection algorithm based on bare bones particle swarm optimization," *Neurocomputing*, vol. 148, pp. 150–157, 2015.

[8] M. H. Aghdam and S. Heidari, "Feature selection using particle swarm optimization in text categorization," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 5, no. 4, pp. 231–238, 2015.

[9] A. Ghosh, A. Datta, and S. Ghosh, "Self-adaptive differential evolution for feature selection in hyperspectral image data," *Applied Soft Computing*, vol. 13, no. 4, pp. 1969–1977, 2013.

[10] S. Bhattacharyya, A. Sengupta, T. Chakraborti, A. Konar, and D. Tibarewala, "Automatic feature selection of motor imagery eeg signals using differential evolution and learning automata," *Medical & biological engineering & computing*, vol. 52, no. 2, pp. 131–139, 2014.

[11] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering, Tech. Rep., 2005.

[12] M. H. Kashan, N. Nahavandi, and A. H. Kashan, "Disabc: A new artificial bee colony algorithm for binary optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 342–352, 2012.

[13] C. Ozturk, E. Hancer, and D. Karaboga, "Dynamic clustering with improved binary artificial bee colony algorithm," *Applied Soft Computing*, vol. 28, pp. 69–80, 2015.

[14] E. Hancer, B. Xue, D. Karaboga, and M. Zhang, "A binary abc algorithm based on advanced similarity scheme for feature selection," *Applied Soft Computing*, vol. 36, pp. 334–348, 2015.

[15] P. K. Singhal, R. Naresh, and V. Sharma, "A novel strategy-based hybrid binary artificial bee colony algorithm for unit commitment problem," *Arabian Journal for Science and Engineering*, vol. 40, no. 5, pp. 1455–1469, 2015.

[16] Z. Cui and X. Gu, "An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems," *Neurocomputing*, vol. 148, pp. 248–259, 2015.

[17] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," *Information Sciences*, vol. 297, pp. 154–170, 2015.

[18] A. Yurtkuran and E. Emel, "A discrete artificial bee colony algorithm for single machine scheduling problems," *International Journal of Production Research*, vol. 54, no. 22, pp. 6860–6878, 2016.

[19] D. Jia, X. Duan, and M. K. Khan, "Binary artificial bee colony optimization using bitwise operation," *Computers & Industrial Engineering*, vol. 76, pp. 360–365, 2014.

[20] M. S. Kiran and M. Gunduz, "Xor-based artificial bee colony algorithm for binary optimization," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 21, no. 2, pp. 2307–2328, 2013.

[21] I. Ribas, R. Companys, and X. Tort-Martorell, "An efficient discrete artificial bee colony algorithm for the blocking flow shop problem with total flowtime minimization," *Expert Systems with Applications*, vol. 42, no. 15-16, pp. 6155–6167, 2015.

[22] M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and A. H. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Information sciences*, vol. 181, no. 16, pp. 3459–3475, 2011.

[23] S.-j. Zhang and X.-s. Gu, "An effective discrete artificial bee colony algorithm for flow shop scheduling problem with intermediate buffers," *Journal of Central South University*, vol. 22, no. 9, pp. 3471–3484, 2015.

[24] Q.-K. Pan, L. Wang, J.-Q. Li, and J.-H. Duan, "A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation," *Omega*, vol. 45, pp. 42–56, 2014.

[25] D. Ye and Z. Chen, "A new approach to minimum attribute reduction based on discrete artificial bee colony," *Soft Computing*, vol. 19, no. 7, pp. 1893–1903, 2015.

[26] H. Zhang and D.-Y. Ye, "Key-node-based local search discrete artificial bee colony algorithm for obstacle-avoiding rectilinear steiner tree construction," *Neural Computing and Applications*, vol. 26, no. 4, pp. 875–898, 2015.

[27] K. Chen and H. Kanoh, "A discrete artificial bee colony algorithm based on similarity for graph coloring problems," in *International Conference on Theory and Practice of Natural Computing*, Springer, 2016, pp. 73–84.

[28] L. Wei and C. Hanning, "Babc: A binary version of artificial bee colony algorithm for discrete optimization," *International Journal of Advancements in Computing Technology(IJACT)*, vol. 4, no. 14, pp. 307–314, 2012.

[29] M. S. Kiran, "The continuous artificial bee colony algorithm for binary optimization," *Applied Soft Computing*, vol. 33, pp. 15–23, 2015.

[30] M. Mandala and C. Gupta, "Binary artificial bee colony optimization for gencos' profit maximization under pool electricity market," *International Journal of Computer Applications*, vol. 90, no. 19, 2014.

[31] D. C. Tran and Z. Wu, "New approaches of binary artificial bee colony algorithm for solving 0-1 knapsack problem.," *Advances in Information Sciences & Service Sciences*, vol. 6, pp. 1–11, 2014.

[32] G. Pampará and A. P. Engelbrecht, "Binary artificial bee colony optimization," in *Swarm Intelligence (SIS), 2011 IEEE symposium on*, IEEE, 2011, pp. 1–8.

[33] V. Bolón-Canedo, K. Sechidis, N. Sánchez-Marono, A. Alonso-Betanzos, and G. Brown, "Exploring the consequences of distributed feature selection in dna microarray data," in *Proc. IJCNN'17*, IEEE, 2017, pp. 1665–1672.

[34] R. Aziz, C. K. Verma, and N. Srivastava, "A fuzzy based feature selection from independent component subspace for machine learning classification of microarray data," *Genomics Data*, vol. 8, pp. 4–15, 2016.

[35] K. K. and K. S., "An enhanced technique for ıdentifying cancer biomarkers from microarray data using hybrid feature selection technique," *International Journal of Scientific Research in Computer Science*, vol. 2, no. 3, pp. 192–198, 2017.

[36] S. Sun, Q. Peng, and X. Zhang, "Global feature selection from microarray data using lagrange multipliers," *Knowledge-Based Systems*, vol. 110, pp. 267–274, 2016.

[37] A. Mortazavi and M. H. Moattar, "Robust feature selection from microarray data based on cooperative game theory and qualitative mutual information," *Advances in bioinformatics*, vol. 2016, pp. 1–15, 2016.

[38] S. Guo, D. Guo, L. Chen, and Q. Jiang, "A centroid-based gene selection method for microarray data classification," *Journal of theoretical biology*, vol. 400, pp. 32–41, 2016.

[39] S. Guo, D. Guo, L. Chen, and Q. Jiang, "A l1-regularized feature selection method for local dimension reduction on microarray data," *Computational biology and chemistry*, vol. 67, pp. 92–101, 2017.

[40] H. Wang, X. Jing, and B. Niu, "A discrete bacterial algorithm for feature selection in classification of microarray gene expression cancer data," *Knowledge-Based Systems*, vol. 126, pp. 8–19, 2017.

[41] C.-S. Yang, L.-Y. Chuang, C.-H. Ke, and C.-H. Yang, "A hybrid feature selection method for microarray classification.," *IAENG International Journal of Computer Science*, vol. 35, no. 3, 2008.

[42] M. J. Abdi, S. M. Hosseini, and M. Rezghi, "A novel weighted support vector machine based on particle swarm optimization for gene selection and tumor classification," *Computational and mathematical methods in medicine*, vol. 2012, 2012.

[43] S. Dara, H. Banka, and C. S. R. Annavarapu, "A rough based hybrid binary pso algorithm for flat feature selection and classification in gene expression data," *Annals of Data Science*, vol. 4, no. 3, pp. 341–360, 2017.

[44] J. Lv, Q. Peng, X. Chen, and Z. Sun, "A multi-objective heuristic algorithm for gene expression microarray data classification," *Expert Systems with Applications*, vol. 59, pp. 13–19, 2016.

[45] A. El Akadi, A. Amine, A. El Ouardighi, and D. Aboutajdine, "A two-stage gene selection scheme utilizing mrmr filter and ga wrapper," *Knowledge and Information Systems*, vol. 26, no. 3, pp. 487–500, 2011.

[46] C.-H. Yang, L.-Y. Chuang, C. H. Yang, *et al.*, "Ig-ga: A hybrid filter/wrapper method for feature selection of microarray data," *Journal of Medical and Biological Engineering*, vol. 30, no. 1, pp. 23–28, 2010.

[47] A. Hasnat and A. U. Molla, "Feature selection in cancer microarray data using multi-objective genetic algorithm combined with correlation coefficient," in *ICETT 2016*, IEEE, 2016, pp. 1–6.

[48] S. Tabakhi, A. Najafi, R. Ranjbar, and P. Moradi, "Gene selection for microarray data classification using a novel ant colony optimization," *Neurocomputing*, vol. 168, no. 2015, pp. 1024–1036, 2015.

[49] H. Alshamlan, G. Badr, and A. Yousef, "Mrmr-abc: A hybrid gene selection algorithm for cancer classification using microarray gene expression profiling," *BioMed Research International*, vol. 2015, no. 2015, pp. 1–15, 2015.

[50] H. Alshamlan, G. Badr, and A. Yousef, "Genetic bee colony (gbc) algorithm: A new gene selection method for microarray cancer classification," *Computational Biology and Chemistry*, vol. 56, no. 2015, pp. 49–60, 2015.

[51] N. S. Mohamed, S. Zainudin, and Z. A. Othman, "Metaheuristic approach for an enhanced mrmr filter method for classification using drug response microarray data," *Expert Systems with Applications*, vol. 90, no. 2017, pp. 224–231, 2017.

[52] J. Apolloni, G. Leguizamón, and E. Alba, "Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments," *Applied Soft Computing*, vol. 38, no. 2016, pp. 922–932, 2016.

[53] P. Mohapatra, S. Chakravarty, and P. K. Dash, "Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system," *Swarm and Evolutionary Computation*, vol. 28, no. 2016, pp. 144–160, 2016.

[54] J. Wang, *Encyclopedia of data warehousing and mining*. iGi Global, 2005.

[55] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[56] R. Govindarajan, J. Duraiyan, K. Kaliyappan, and M. Palanisamy, "Microarray and its applications," *Journal of pharmacy & bioallied sciences*, vol. 4, no. Suppl 2, S310, 2012.

[57] A. Lesk, *Introduction to bioinformatics*. Oxford university press, 2014.

[58] S.C.Robson. (2008). Rma and gc-rma normalisation, [Online]. Available: `https://warwick.ac.uk/fac/sci/moac/people/students/2003/sam_robson/usergroups/rmavsmas5/` (visited on 04/29/2019).

[59] H. Liu, I. Bebu, and X. Li, "Microarray probes and probe sets," *Frontiers in bioscience (Elite edition)*, vol. 2, p. 325, 2010.

[60] P. D. Lauren, "Algorithm to model gene expression on affymetrix chips without the use of mm cells," *IEEE transactions on nanobioscience*, vol. 2, no. 3, pp. 163–170, 2003.

[61] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed, "Exploration, normalization, and summaries of high density oligonucleotide array probe level data," *Biostatistics*, vol. 4, no. 2, pp. 249–264, 2003.

[62] Z. Wu, R. A. Irizarry, R. Gentleman, F. Martinez-Murillo, and F. Spencer, "A model-based background adjustment for oligonucleotide expression arrays," *Journal of the American statistical Association*, vol. 99, no. 468, pp. 909–917, 2004.

[63] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[64] M. A. Hall, "Correlation-based feature selection for machine learning," 1999.

[65] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[66] J. H. Holland, "Genetic algorithms," *Scholarpedia*, vol. 7, no. 12, p. 1482, 2012.

[67] J. Kennedy and R. Eberhart, "Proceedings of ıeee international conference on neural networks," *Perth, Australia*, 1995.

[68] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, IEEE, vol. 5, 1997, pp. 4104–4108.

[69] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[70] A. P. Engelbrecht and G. Pampara, "Binary differential evolution strategies," in *2007 IEEE Congress on Evolutionary Computation*, IEEE, 2007, pp. 1942–1947.

[71] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: Consistency properties," California Univ Berkeley, Tech. Rep., 1951.

[72] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "An overview of machine learning," in *Machine learning*, Elsevier, 1983, pp. 3–23.

[73] I. Barandiaran, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 20, no. 8, 1998.

[74] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[75] D. F. Specht, "Probabilistic neural networks," *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.

[76] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[77] O. Maimon and L. Rokach, "Data mining and knowledge discovery handbook," 2005.

[78] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[79] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[80] M. G. Kendall *et al.*, "The advanced theory of statistics. vols. 1.," *The advanced theory of statistics. Vols. 1.*, vol. 1, no. Ed. 4, 1948.

[81] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[82] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "A novel binary particle swarm optimization," in *2007 Mediterranean Conference on Control & Automation*, IEEE, 2007, pp. 1–6.

[83] Y.-W. Jeong, J.-B. Park, S.-H. Jang, and K. Y. Lee, "A new quantum-inspired binary pso: Application to unit commitment problems for power systems," *IEEE Transactions on Power Systems*, vol. 25, no. 3, pp. 1486–1495, 2010.

[84] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information sciences*, vol. 192, pp. 120–142, 2012.

[85] X. Zhang and X. Zhang, "A binary artificial bee colony algorithm for constructing spanning trees in vehicular ad hoc networks," *Ad Hoc Networks*, vol. 58, pp. 198–204, 2017.

[86] M. Schiezaro and H. Pedrini, "Data feature selection based on artificial bee colony algorithm," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 47, 2013.

[87] Z. B. Ozger, B. Bolat, and B. Diri, "A comparative study on binary artificial bee colony optimization methods for feature selection," in *2016 INISTA*, IEEE, 2016.

[88] S. Mirjalili and A. Lewis, "S-shaped versus v-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

[89] S. Sivanandam and S. Deepa, *Genetic algorithm implementation using matlab, in:Introduction to Genetic Algorithms*. Berlin, Heidelberg, 2008, pp. 211–262.

[90] M. Mernik, S.-H. Liu, D. Karaboga, and M. Črepinšek, "On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation," *Information Sciences*, vol. 291, pp. 115–127, 2015.

[91] A. Draa, "On the performances of the flower pollination algorithm–qualitative and quantitative analyses," *Applied Soft Computing*, vol. 34, pp. 349–371, 2015.

[92] M. Črepinšek, S.-H. Liu, L. Mernik, and M. Mernik, "Is a comparison of results meaningful from the inexact replications of computational experiments?" *Soft Computing*, vol. 20, no. 1, pp. 223–235, 2016.

[93] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. Evolutionary Computation'2005*, IEEE, 2005.

[94] NCBI. (2005). National center of biotechnology ınformation, [Online]. Available: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE2466 (visited on 09/11/2018).

[95] St.Jude. (2018). St.jude children's research hospital, [Online]. Available: http://www.stjuderesearch.org/site/data/ALL1/all_rawdata (visited on 09/11/2018).

[96] B. Institute. (2018). Broad ınstitute cancer program legacy publication resources, [Online]. Available: http://portals.broadinstitute.org/cgi-bin/cancer/datasets.cgi (visited on 09/11/2018).

[97] NCBI. (2016). National center of biotechnology ınformation, [Online]. Available: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE82009 (visited on 09/11/2018).

[98] NCBI. (2005). National center of biotechnology ınformation (ncbı), [Online]. Available: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=gse2034 (visited on 09/11/2018).

[99] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[100] N. Veček, S. H. Liu, M. Črepinšek, and M. Mernik, "On the importance of the artificial bee colony control parameter 'limit'," *Information Technology And Control*, vol. 46, no. 4, pp. 566–604, 2017.

Tezden Üretilmiş Yayınlar

İletişim Bilgileri: zeynepozger@ksu.edu.tr

Makale

1. ÖZGER Z. B., BOLAT B., DIRI, B., A Probabilistic Multi-Objective Artificial Bee Colony Algorithm for Gene Selection, Journal of Universal Computer Science, vol.25, pp.418-443, 2019.

Konferans Bildirisi

1. ÖZGER Z. B., BOLAT B., DIRI, B., A Comparative Study on Binary Artificial Bee Colony Optimization Methods for Feature Selection, Akıllı Sistemlerde Yenilikler ve Uygulamaları Konferansı (INISTA), 2016.

2. ÖZGER Z. B., BOLAT B., DIRI B., IBitABC: Improved Binary Artificial Bee Colony Algorithm with Local Search, Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı (UBMK), 2017.

Kitap

1. ÖZGER Z. B., BOLAT B., DIRI, B., A Comparative Study on Binary Artificial Bee Colony Optimization Methods for Feature Selection, Focus on Swarm Intelligence Research and Applications, Nova Science Publishers, 2017.