Necmettin Erbakan Üniversitesi
Fen ve Mühendislik Bilimleri Dergisi
Journal of Science and Engineering (NEJSE)
DOİ: https://doi.org/10.47112/neufmbd.2022.8

Cilt: 4 Sayı:1 Yıl: 2022 | Araştırma Makelesi / Research Article | E-ISSN: 2667- 7989

# Ağaç Tohum Algoritmasının Kümeleme Problemlerine Uygulanması

**Abdülkadir PEKTAŞ** [1] **Onur İNAN** [2]

[1] Necmettin Erbakan Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Konya, Türkiye, apektas@erbakan.edu.tr *(Sorumlu Yazar/ Corresponding Author)*
[2] Selçuk Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği Bölümü, Konya, Türkiye, oinan@selcuk.edu.tr

**Makale Bilgileri**

**Makale Geçmişi**
**Geliş: 17.01.2022**
**Kabul: 27.03.2022**
**Yayın: 30.06.2022**

**Anahtar Kelimeler:**
Ağaç tohum algoritması, kümeleme, optimizasyon

**ÖZET**

Toplanan verilerin tutarlı bir şekilde gruplandırılması, veriden anlamlı bilgilerin çıkarılabilmesi için önemli bir adımdır. Verilerin gruplara ayrılması sınıflandırma ve kümeleme olmak üzere iki farklı işlem ile gerçekleştirilir. Sınıflandırma işleminde verilerin ayrılacağı alt grupların etiketleri ve dolayısıyla sınıf sayıları belirli iken, kümeleme işleminde verilerin ait olduğu gruplar belli değildir. Bu nedenle kümeleme işlemi danışmansız bir makine öğrenmesi işlemidir. Kümeleme işleminde veri elemanları birbirlerine göre benzerlik ve farklılıklarına göre alt gruplara ayrılırlar. Optimizasyon algoritmaları bir problemin uygun çözümler içerisinden en iyi sonucu bulmaya çalışan ve doğrusal olmayan problemlerin çözümünde etkili yöntemlerdir. Kümeleme işlemleri için geliştirilmiş ve yaygın olarak kullanılan geleneksel kümeleme yöntemlerinin yanında olası çözümler arasında uygun çözüme ulaşmaya çalışan karakteristik özelliklerinden dolayı optimizasyon algoritmaları da kümeleme problemlerinin çözümü için kullanılır hale gelmiştir. Bu çalışmada yakın zamanda geliştirilen bir optimizasyon algoritması olan Ağaç Tohum Algoritması 17 farklı kümeleme problemine uygulanmış ve elde edilen sonuçlar geleneksel kümeleme yöntemleri olan K-Ortalamalar, K-Medoidler ve Bulanık C-Ortalamalar yöntemlerinin performansları ile karşılaştırılmıştır. Ağaç Tohum Algoritmasının kümeleme işleminde geleneksel yöntemlere göre daha başarılı sonuçlara ulaştığı görülmüştür.

# Application of Tree Seed Algorithm on Clustering Problems

**Article Info**

**Article History**
**Received: 17.01.2022**
**Accepted: 27.03.2022**
**Published: 30.06.2022**

**Keywords:**
Clustering, optimisation, tree seed algorithm

**ABSTRACT**

Separating gathered data into consistent subgroups is an important phase to retrieve meaningful information from data. Separating data into groups can be performed with two different operation which are classification and clustering. While the labels of data groups and so the number of subgroups is known in classification, class labels are not known in clustering process. So, clustering is an unsupervised machine learning operation. In clustering process, data objects are separated into subgroups according to similarity and dissimilarity among data samples. Optimization algorithms are useful method which try to reach optimum solution among possible solutions and are effective for solving nonlinear problems. Besides commonly used traditional clustering methods, optimization methods are recently applied on clustering problems because of their characteristics for reaching optimum solutions among feasible solutions space. In this study a recently developed optimization algorithm called Tree Seed Algorithm is applied on 17 different clustering problems and the results are compared with the performances of traditional clustering methods like K-Means, K-Medoids and Fuzzy C-Means. It is seen that Tree Seed Algorithm has better performance than traditional methods for clustering problems.

**Atıf/Citation:** Pektaş, A. & İnan, O. (2022). Ağaç tohum algoritmasının kümeleme problemlerine uygulanması, *Necmettin Erbakan Üniversitesi Fen ve Mühendislik Bilimleri Dergisi*, 4(1), 1-10.

## GİRİŞ (INTRODUCTION)

Teknolojinin ilerlemesiyle birlikte veri kaydı yapılan alanların artmasının yanında, bir alanda daha fazla veri kaydedilmeye başlanması sonucu kayıt altına alınan veri miktarında büyük bir artış meydana gelmiştir. Kayıt altına alınan çok sayıda veriden anlamlı ve tutarlı bilgilerin çıkarılabilmesinin önemi her geçen gün artmaktadır. Çok sayıda veri içerisinden gereksiz ve hatalı verilerin ayıklanması, verilerin özelliklerine göre gruplandırılması gibi işlemler, toplanan veriden kullanışlı bilgiler elde edilebilmesi açısından önemlidir. Önemli bir veri madenciliği işlemi olan kümeleme, sınıf etiketleri bulunmayan veri topluluklarının, veri elemanlarının özelliklerinden yola çıkarak gruplara ayrılmasıdır. Kümeleme sonucunda birbirleriyle benzer olan veri elemanları aynı grup altında toplanırken, birbirlerinden farklı özelliklere sahip veri elemanları farklı gruplara dağıtılmış olur. Optimizasyon, bir problemin en iyi çözümüne ulaşmayı hedefler. Optimizasyon algoritmaları problemin uygun çözüm kümesinden en iyi çözümü bulmak üzere geliştirilmiş yöntemlerdir. Doğrusal olmayan yapısı ve olası çözümleri içeren çözüm uzayının boyutunun büyük olması sebebiyle kümeleme problemleri, optimizasyon algoritmaları kullanılarak çözümlenebilir özelliktedir.

DW van der Merwe ve AP Engelbrecht kümeleme problemleri üzerinde iki farklı Parçacık Sürü Optimizasyon Algoritması yaklaşımı uygulamış ve Parçacık Sürü Optimizasyon Algoritması'nın geleneksel kümeleme yöntemlerinden daha etkili sonuçlara ulaşma potansiyeli olduğunu ortaya koymuştur [1]. Nasiri ve Khiyabani kümeleme problemlerine Balina Optimizasyon Algoritması uygulamış ve bu algoritmanın performansını Parçacık Sürü Optimizasyonu, Diferansiyel Gelişim Algoritması, Genetik Algoritma ve geleneksel bir kümeleme yöntemi olan K-Ortalamalar yöntemi ile karşılaştırmış, Balina Optimizasyon Algoritmasının kümeleme problemlerinde başarılı bir şekilde kullanılabileceğini belirtmiştir [2]. Fathian, Amiri ve Maroosi sürü-tabanlı bir algoritma olan Bal Arısı Çiftleşme Algoritmasını üç farklı kümeleme veri kümesi üzerinde uygulamış, verilen veri kümeleri üzerinde Bal Arısı Çiftleşme Optimizasyonu'nun Genetik Algoritma, Karınca Kolonisi Algoritması, Tavlama Benzetimi metodu ve Tabu Arama metodundan daha iyi sonuçlar verdiğini ortaya koymuştur [3]. Niknam ve diğerleri Parçacık Sürü Optimizasyonu ile Tavlama Benzetimi metodlarını bir araya getirerek oluşturdukları hibrid modeli kümeleme problemleri üzerine uygulamış ve sunulan hibrid metodun hem k-Ortalamalar yönteminden ve Parçacık Sürü Optimizasyonu ile Tavlama Benzetini metodunun kendi başlarına ulaştıkları sonuçlardan daha iyi sonuçlara ulaştığını, hem de bu metodun çözüm adımlarının daha hızlı yakınsayarak problemin daha hızlı bir şekilde çözüldüğünü belirtmişlerdir [4]. Kushwaha ve diğerleri manyetik kuvvetlerden esinlenerek Manyetik Optimizasyon Algoritması alıyla yeni bir algoritma geliştirmiş ve bu optimizasyon algoritmasının kümeleme problemlerinde yerel minimum noktalarından kaçınma işleminde başarılı olduğunu ortaya koymuştur [5]. Maulik ve Mukhopadhyay Tavlama Benzetimi ve Yapay Sinir Ağları yöntemlerini bir araya getirerek oluşturduğu hibrid modelin kümeleme problemlerinde diğer kümeleme yöntemlerinden daha iyi performans sergilediğini belirtmiştir [6]. Selim ve Alsultan Tavlama Benzetimi yöntemini kümeleme problemlerini çözmek için uygulmaış ve Tavlama Benzetimi yönteminin kümeleme problemlerinde global minimum değere ulaşabildiğini belirtmiştir [7]. Maulik ve Bandyopahyay Genetik Algoritmanın arama yetenğini kullanarak uygun küme merkezlerini bulduğu çalışmasında dördü yapay, üçü gerçek hayat veri kümesi olmak üzere toplamda yedi farklı veri kümesinde Genetik Algoritma tabanlı bir kümeleme işleminin K-Ortalamalar kümeleme yönteminden daha iyi sonuçlar verdiğini ortaya koymuştur [8].

## MATERYAL VE METOT (MATERIALS AND METHODS)

### *Veri Kümeleri*

Bu çalışmada UCI (University of California, Irvine) Makine Öğrenmesi veri tabanında bulunan 17 farklı sınıflandırma ve kümeleme verisi kullanılmıştır [9]. Böylece farklı özellik ve küme merkezi sayılarına sahip az ve çok kayıt içeren verilerde yöntemin performansını incelemek hedeflenmiştir. Bir

problemde kullanılacak küme merkezi sayısı problemin veri kümesinde bulunan benzersiz sınıf etiketi sayısı olarak seçilmiştir. Kullanılan veri kümelerine ait özellikler Tablo 1'de verilmiştir.

**Tablo 1**. *Çalışmada kullanılan veri kümelerine ait özellikler*

| Veri Kümesi | Küme Merkezi Sayısı | Özellik Sayısı | Kayıt Sayısı |
|---|---|---|---|
| Balance | 3 | 4 | 625 |
| Btissue | 6 | 9 | 106 |
| Credit | 2 | 14 | 690 |
| Dermatology | 6 | 34 | 366 |
| Diabets | 2 | 8 | 768 |
| *E. coli* | 5 | 7 | 327 |
| Glass | 6 | 9 | 214 |
| Heart | 2 | 13 | 270 |
| Hepatit | 2 | 21 | 155 |
| Iris | 3 | 4 | 150 |
| Parkinson | 2 | 22 | 195 |
| Seeds | 3 | 7 | 210 |
| Somerville | 2 | 6 | 143 |
| Spect | 2 | 22 | 267 |
| Thyroid | 3 | 5 | 215 |
| User Modeling | 4 | 5 | 258 |
| Wine | 3 | 13 | 178 |

### *Kümeleme Yöntemleri*

*K-Ortalamalar Kümeleme Yöntemi*

K-Ortalamalar yöntemi 1967 yılında MacQueen tarafından önerilen [10] ve önerildiği zamanda en basit danışmansız öğrenme yöntemlerinden biri olan [11] bir yaygın kümeleme yöntemidir. K-Ortalamalar yöntemi kolay uygulanabilir, hızlı ve kümeleme problemlerinde genel olarak iyi sonuç vermesi sebebiyle en popüler kümeleme yöntemidir [12]. K-Ortalamalar yöntemi birbirinden bağımsız iki farklı aşamada gerçekleşir, bu aşamaların birincisi sayısı önceden belirlenmiş olan küme merkezlerinin seçilmesi, diğeri ise veri kümesinde bulunan elemanların seçilen küme merkezlerine atanmasıdır [11]. Küme merkezleri, kendilerine atanan veri kümesi elemanlarına göre güncellenir ve veri kümesi elemanları kendilerine en yakın güncellenmiş küme merkezlerine yeniden atanır. Bu işlemler iteratif olarak en uygun sonuca ulaşana dek devam eder.

*K-Medoidler Kümeleme Yöntemi*

K-Medoidler yöntemi Kaufman ve Rousseuw tarafından 1987 yılında, K-Ortalamalar yönteminin gürültülü ve sipesifik verilerde gösterdiği aşırı hassasiyeti ortadan kaldırmak amacıyla önerilmiştir [13]. K-Medoidler yönteminde küme merkezleri K-Ortalamalar yönteminden farklı olarak veri kümesi elemanlarından seçilir [14]. Yöntemin asıl amacı, her bir kümeyi temsil edebilecek merkezi bir veri kümesi elemanını saptayabilmektir. K-Meodidler yöntemi ile kümeleme işlemi yapılırken her bir veri elemanı kendine en yakın küme merkezine atanır. Bir küme içerisindeki tüm elemanlar, küme merkezi ile yer değiştirilerek her bir işlem için maliyet fonksiyonu hesaplanır. Yer değiştirme işlemi sonucunda daha düşük maliyetli bir küme merkezi seçimine ulaşıldığı durumda küme merkezi güncellenir. Bu işlemler maliyet fonksiyonu grafiği yakınsayana dek iterasyonlar boyunca devam eder.

*Bulanık C-Ortalamalar Kümeleme Yöntemi*

Bulanık C-Ortalamalar yöntemi Dunn tarafından 1973'te ortaya atılmış [15], 1981 yılında Bezdek tarafından genellenmiş [16], yumuşatılmış K-Ortalamalar metodu olarak da bilinen bir

kümeleme yöntemidir. Diğer geleneksel kümeleme yöntemlerinde adımlar boyunca her bir veri elemanı uzaklığı ne olursa olsun belirlenmiş küme merkezlerinden yalnızca birine ait kabul edilirken, Bulanık C-Ortalamalar yönteminde veri elemanları küme merkezlerine belli bir oran dahilinde ait kabul edilirler. Bir küme merkezine yakın olan bir veri kümesi elemanı o küme merkezine yüksek bir oranda dahil kabul edilirken, uzak olduğu küme merkezlerine daha düşük derecelerle dahil edilir. Bir veri kümesi elemanının tüm küme merkezlerine olan aidiyet oranları toplamı 1'dir. Bulanık C-Ortalamalar yönteminin bu yaklaşımı, diğer kümeleme yöntemlerinin keskin sınırlarından kaynaklanan hatalardan kaçınmasına yardımcı olur [17]. Bulanık C-Ortalamalar yönteminde minimize edilmeye çalışılan maliyet fonksiyonu aşağıda verilmiştir.

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{k} u_{ij}^{m} \left\| x_{i-} c_j \right\|^2 \tag{1}$$

Verilen denklemde N veri kümesi eleman sayısını, k küme merkezi sayısını, m algoritma içerisinde kullanılan 1'den büyük bir sabit sayıyı, $u_{ij}$ i numaralı veri kümesi elemanı $x_i$ ile j numaları küme merkezi arasındaki aidiyet oranını, $c_j$ de j numaralı küme merkezini göstermektedir. Bulanık C-Ortalamalar yönteminin başında aidiyet oranı değerleri aşağıda verilen denklem ile rastgele olarak belirlenir.

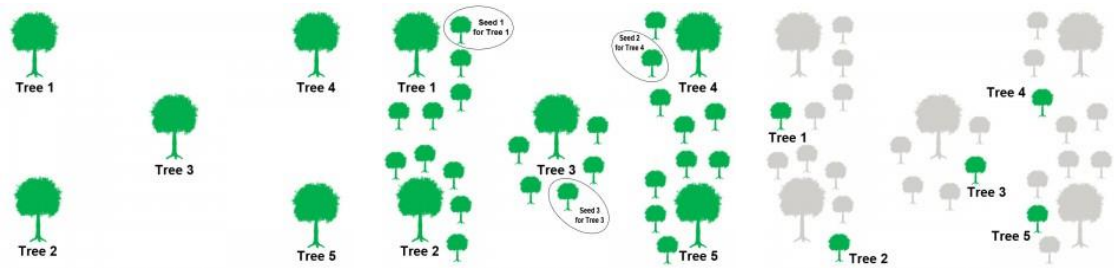
$$C_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} . x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \tag{2}$$

Küme merkezleri belirlendiğinde aidiyet dereceleri matrisi u, her bir uygulama adımında aşağıda verilen denklem ile güncellenir.

$$u_{ij} = \frac{1}{\sum_{n=1}^{k} \left( \frac{\left\| x_i - C_j \right\|^2}{\left\| x_i - C_n \right\|^2} \right)^{\frac{2}{m-1}}} \tag{3}$$

**Ağaç Tohum Algoritması**

Ağaç Tohum Algoritması 2015 yılında M. S. Kıran tarafından geliştirilmiş bir sürekli optimizasyon yöntemidir [18]. Ağaçların gelişme, üreme ve hayatta kalma davranışlarından esinlenilerek geliştirilmiştir. Bir popülasyon içerisindeki ağaçlar nesillerini devam ettirmek için tohum üretirler. Bu tohumlardan meydana gelen çocuk ağaçlar yetiştikleri noktanın ışık, nem, rüzgâr gibi fiziksel özelliklerine ve kendilerinde bulunan genetik özelliklere bağlı olarak farklı şekillerde gelişim gösterirler. Çocuk ağaçlar içerisinde güçlü özelliklere sahip olanlar büyür ve kendisini oluşturan ana ağaçtan daha yükseğe uzanarak daha iyi güneş ışığı alacak konuma gelir. Zayıf yapıdaki çocuk ağaçlar ise gölgede kalarak yok olurlar. Bu doğal seçilim süreci boyunca bir ağaç popülasyonunda sürekli en güçlü özelliklere sahip ağaçlar hayatta kalır ve bir sonraki nesilde kendilerinden daha güçlü bireyler oluşursa yerlerini onlara bırakırlar. Ağaç Tohum Algoritmasının esinlendiği ağaç davranışları Şekil 1'de gösterilmiştir [19].




**Şekil 1:** Ağaç tohum algoritması işleyişi *"**a)** Başlangıçta oluşturulan ağaçlar **b)** Her bir ağaç tarafından üretilen tohumlar **c)** Her bir ağacın tohumları arasında en iyiler hayatta kalır"*

Ağaç Tohum Algoritmasının tohum üretme aşamasında her bir ağaç belirli sınırlar içerisinde birbirlerinden farklı sayıda tohum üretebilirler, görselde her bir ağaç tarafından 5 adet tohum üretilmiştir.

Ağaç Tohum Algoritmasının matematiksel modelinde ağaçların bulunduğu ortam problemin olası çözümlerini içeren çözüm uzayını, ağaçlar ise problemin olası çözümlerini simgeler. Ağaçlar, çözüm uzayı içerisinde arama yapmak için tohumlar üreterek en iyi çözüme ulaşmaya çalışırlar. Başlangıçta üretilen ağaçlar çözüm uzayı sınırları içerisinde olmak üzere aşağıda verilen denkleme göre rastgele yerleştirilir.

$$T_{ij} = L_{j,min} + r_{ij}\left(H_{j,max} - L_{j,min}\right) \tag{4}$$

$L_{j,min}$ ve $H_{j,max}$ sırasıyla problemin j boyutundaki minimum ve maksimum sınırları, $r_{i,j}$ ise bu boyut için üretilmiş [0,1] aralığında bir rastgele sayıyı göstermektedir. Tohum üretim aşamasında her bir ağaç tohum üretir. Her bir ağaçtan üretilen tohum sayıları farklı olabilir ve üretilen tohumun sayısı problemin çözümü için kullanılan ağaç popülasyonunun büyüklüğüne bağlı bir değişkendir. Ağaç Tohum Algoritmasının kontrol parametrelerinin etkileri analiz edildiğinde bir ağaç tarafından üretilen tohum sayısının alt sınırı popülasyon büyüklüğünün %10'u, üst sınırı ise popülasyon büyüklüğünün %25'i olarak hesaplanmıştır. [18] Tohum üretim aşaması Ağaç Tohum Algoritmasının güncelleme aşamasıdır. Önceden tanımlı[20] bir parametre olan arama eğilimi (search tendency, ST) değerine göre tohum üretimi için iki farklı denklemden biri kullanılır. ST parametresi algoritmanın arama odağını belirleyen bir parametredir. Yüksek ST parametresi kuvvetli bir yerel arama sağlayarak maliyet fonksiyonunda hızlı bir yakınsama sağlarken, düşük ST parametresi algoritmanın global arama yeteneğini artırıp sonuca ulaşma süresini uzatır [18]. St parametresine göre içlerinden biri seçilerek tohum üretiminde denklemler aşağıda verilmiştir.

$$S_{ij} = T_{ij} + a_{ij} \times \left(B_j - T_{rj}\right) \tag{5}$$

$$S_{ij} = T_{ij} + a_{ij} \times \left(T_{ij} - T_{rj}\right) \tag{6}$$

Denklemlerde $B_j$ şimdiye kadar ulaşılmış en iyi sonuca sahip ağacın j numaralı boyuta ait değerini, $a$ -1 ile 1 arasında rastgele üretilmiş boyutlandırma faktörünü, $T_{rj}$ ise ağaç popülasyonu içerisinden seçilen rastgele bir ağacın j boyutundaki değerini ifade eder. Bu denklemlerden hangisi ile yeni bir tohum oluşturulacağına karar vermek için tohum üretim aşamasında [0,1] aralığında rastgele bir değer üretilir. Üretilen bu rastgele sayı, önceden [0,1] aralığında üretilmiş ST parametresinden küçükse 5. Denklem ile tohum üretim işlemi yapılır, böylece şimdiye kadar elde edilmiş en iyi sonuca sahip ağaca ait değerler göz önüne alınarak algoritma adımı güncellenmiş olur. Üretilen sayının ST parametresinden büyük olması durumunda 6. Denklem kullanılarak tohum üretim işlemi yapılır, böylece çözüm uzayı

içerisindeki rastgele bir başka ağaca ait değerler kullanılarak algoritmanın çözüm uzayı içerisinde sıçramalar yapması sağlanarak yerel minimum değerlerde takılı kalmasının önüne geçilir [18,21]. Üretilen tohumların maliyet fonksiyon değerleri hesaplandıktan sonra bir ağaç tarafından üretilen tohumlar içerisinde en iyi sonuç veren tohumun, kendisini oluşturan ağaçtan daha iyi bir performans gösterip göstermediğine bakılır. Eğer bir ağacın ürettiği tohumlar arasında en iyi sonuç veren tohum, ağacın kendisinden de daha iyi sonuç veriyorsa ana ağaç ile en iyi sonuca sahip tohum yer değiştirilerek bir sonraki uygulama adımına en iyi tohumla devam edilir. Böylece bir iterasyon içerisinde popülasyon büyüklüğü her bir ağaç tarafından üretilen tohumlar sebebiyle artarken iterasyon sonunda bir sonraki iterasyona bir ana ağaçtan üretilen çözümler arasında en iyisi ile devam edilerek yine başlangıçtaki popülasyon büyüklüğü ile devam edilir. Böylece algoritma yinelendikçe her adımda daha kalabalık bir popülasyonla güçlü bir arama sağlarken, yeni adıma yalnızca en iyi çözümlerle devam ederek problemin karmaşıklığının artmasının önüne geçilir.

Ağaç tohum algoritmasında ağaçların üretimi keşif (exploration) ve işleme (exploitation) adı verilen iki aşamada gerçekleştirilir. Keşif aşamasında başlangıç ağaçları çözüm uzayı içerisine rastgele dağıtılarak geniş bir arama alanı oluşturulur ve verimi yüksek çözümler elde edilir. İşleme aşamasında ise var olan ağaçlardan bahsedilen yöntemlerle üretilen tohumlar sayesinde çözüm uzayı içerisinde arama işlemi gerçekleştirilir. Keşif ve işleme mekanizmaları sayesinde Ağaç Tohum Algoritması uygulama adımları boyunca yerel ve global olarak en iyi sonuca ulaşmayı hedefler [22]. Ağaç Tohum Algoritmasının her bir iterasyonunda her bir ağaçtan orman büyüklüğünün %10'u ile %25'i arasında tohum üretilip, her bir tohum için maliyet fonksiyonu hesabı yapılacağından, algoritma bitirme şartı iterasyon sayısı yerine maliyet fonksiyonunun çağırılma sayısı olarak belirlenmiştir. Algoritmanın bitirme şartı problem boyutunun 10000 ile çarpılması ile bulunur.

$$Max_{FEs} = D \times 10000 \tag{7}$$

Ağaç Tohum Algoritmasının adımları Şekil 2'de gösterilmiştir [18].

***Adım 1**: Algoritmanın hazırlanması*
- Popülasyon büyüklüğünü belirle. (N)
- Arama eğilimini (ST) belirle.
- Problemin boyutunu belirle. (D)
- 5.Denklem yardımıyla başlangıç ağaçlarını oluştur.
- En iyi çözüme sahip ağacı belirle.

***Adım 2**: Tohumlarla arama*
- **FOR** her bir ağaç için
  - Ağaç tarafından üretilecek tohum sayısını (NS) belirle.
  - **FOR** her bir tohum için
    - **FOR** problemin her bir boyutu için
      - **IF** (rand<ST)
        - Çocuk ağacın konumunu 5. Denklem ile belirle.
      - **ELSE**
        - Çocuk ağacın konumunu 6. Denklem ile belirle.
      - **END IF**
    - **END FOR**
  - **END FOR**
  - En iyi sonuç veren tohumu belirle ve ana ağaçla karşılaştır.
  - Eğer tohum ana ağaçtan daha iyi sonuç veriyorsa, yer değiştir.
- **END FOR**

***Adım 3**: En iyi çözümün seçilmesi*
- Popülasyonun en iyi çözümünü belirle.
- En iyi çözüm, önceki adımın en iyi çözümünden daha iyi ise, en iyi çözümü güncelle.

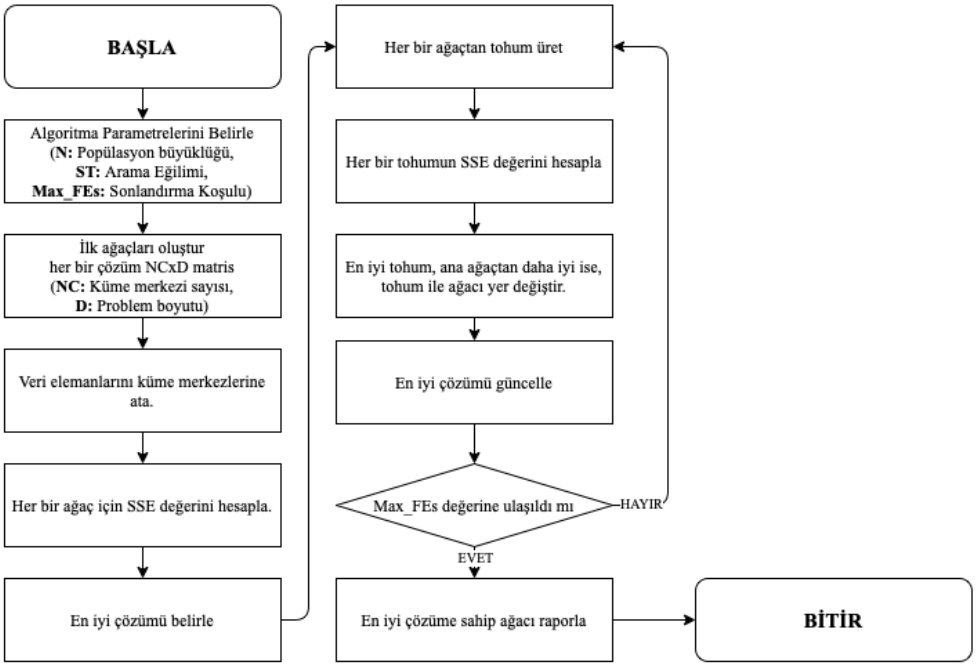
***Adım 4**: Bitirme şartı kontrolü*
- Bitirme şartı sağlanmıyorsa, Adım 2'ye git.

***Adım 5**: Raporlama*
- En iyi çözümü kaydet ve raporla.

**Şekil 2:** Ağaç tohum algoritmasının algoritmik yapısı

**Uygulama**

Ağaç Tohum Algoritmasının bir kümeleme probleminin çözümüne uygulanması için başlangıçtaki ağaç sayısı 20 olarak belirlendi. Kümeleme probleminin olası çözümleri küme merkezlerinin konumları olacak şekilde belirlendi. Başlangıç ağaçları, problemdeki verilerin minimum ve maksimum sınırları arasında olacak şekilde, NC küme merkezi sayısını, D problemin boyutunu göstermek üzere NCxD boyutunda matrisler olarak seçildi. Her bir ağaç için, veri kümesindeki her bir veri elemanı kendisine en yakın küme merkezine atandı ve maliyet fonksiyonu her bir veri elemanının ait olduğu kümenin merkezine olan uzaklıklarının toplamı olarak hesaplandı. Uzaklık hesaplamaları yapılırken Öklid uzaklığı ve toplam kare hatası (SSE, Sum Squared Error) yöntemleri kullanıldı. Arama eğilimi (ST) parametresi 0.1 olarak belirlendi. Uygulama boyunca izlenen adımlar Şekil 3'te görülebilir.




**Şekil 3:** Çalışmada izlenen adımlara ait akış şeması

## BULGULAR (RESULTS)

UCI (University of California, Irvine) Makine Öğrenmesi veri tabanından alınan 17 farklı veri kümesi üzerinde uygulanan kümeleme işlemleri sonucunda elde edilen toplam Öklid uzaklıklarının karesi (SSE) değerleri ve yöntemlerin sıralamaları Tablo 2'de gösterilmiştir. Tabloda her bir veri kümesi üzerinde ulaşılan en iyi sonuç kalın yazı tipi ile belirtilmiş ve yöntemlerin ulaştıkları sonuçlara göre ulaştıkları sıralamalar sonuçların altında gösterilmiştir. Tablonun sonunda her yöntemin ulaştıkları sıralamaların ortalama değeri gösterilerek en iyi sıralama ortalamasına sahip yönteme ait değer yine kalın yazı tipi ile belirtilmiştir.

Tabloda da görüleceği üzere, Ağaç Tohum Algoritması genel olarak ilgili veri kümeleri üzerinde geleneksel kümeleme yöntemlerine göre daha iyi sonuçlar vermektedir. Ağaç Tohum Algoritması ile gerçekleştirilen kümeleme işlemleri K-Medoidler ve Bulanık C-Ortalamalar yöntemlerinden daha başarılı olmakla birlikte K-Ortalamalar yönteminin en başarılı kümeleme işlemini gerçekleştirdiği durumlarda Ağaç Tohum Algoritması K-Ortalamalar yönteminin ulaştığı başarıya yakın sonuçlar elde etmektedir.

Tabloda bazı veri kümeleri üzerinde geleneksel yöntemlerin birbirleri ile aynı sonuçlara ulaştığı durumlar dikkat çekmektedir. Diabetes, parkinson, seeds ve spect veri kümelerinde K-Ortalamalar ve Bulanık C-Ortalamalar yöntemlerinin aynı sonuçları bulması, her iki yöntemin de çözüm olarak aynı noktaları küme merkezi olarak seçtiklerini gösterebilir. Tabloda da görüldüğü üzere, Ağaç Tohum Algoritması bu veri kümeleri üzerinde diğer kümeleme yöntemlerinin ulaştığı sonuçlardan daha iyi sonuçlara ulaşmıştır. Buradan yola çıkılarak K-Ortalamalar ve Bulanık C-Ortalamalar yöntemlerinin bu veriler üzerinde kümeleme işlemi yaparken aynı lokal minimum değere ulaştıkları, Ağaç Tohum Algoritmasının bu lokal minimum değere takılmayarak daha uygun sonuca ilerlediği söylenebilir.

**Tablo 2.** *Değerlendirme Sonuçları ve Sıralamaları*

| VERİ KÜMESİ | K-ORT. | K-MED. | B.C-ORT. | TSA |
|---|---|---|---|---|
| Balance | **1423.8514 1** | 1686.4799 3 | 1722.2446 4 | 1424.579 2 |
| Btissue | **130649.5537 1** | 143417.2244 2 | 145764.7385 3 | 149145.687 4 |
| Credit | 777510.5758 4 | 562284.0924 2 | 759180.4699 3 | **557390 1** |
| Dermatology | **2034.0428 1** | 2864.966 3 | 5196.3797 4 | 2705.169 2 |
| Diabetes | 74604.324 3 | 73172.0714 2 | 74604.324 3 | **48189.62 1** |
| E.Coli | **65.9877 1** | 145.9961 4 | 108.4402 3 | 71.582 2 |
| Glass | **215.3564 1** | 311.0533 3 | 400.9818 4 | 284.0928 2 |
| Heart | 10700.8385 2 | 11814.2091 3 | 13943.6643 4 | **10644.86 1** |
| Hepatit | 9973.9731 2 | 10376.5599 3 | 12755.0747 4 | **9458.369 1** |
| Iris | 97.3529 2 | 183.6139 4 | 106.3591 3 | **96.7811 1** |
| Parkinson | 17081.5962 2 | 17120.1081 3 | 17081.5962 2 | **11461.78 1** |
| Seeds | 754.3226 2 | 768.0356 3 | 754.3226 2 | **580.0928 1** |
| Somerville | 281.3565 2 | 327.701 4 | 317.1949 3 | **280.991 1** |
| Spect | 557.5988 2 | 633.544 3 | 557.5988 2 | **551.798 1** |
| Thyroid | 2001.6358 2 | 2097.6815 3 | 2812.4999 4 | **1885.215 1** |
| U. Modeling | **97.9459 1** | 152.5859 4 | 130.8699 3 | 98.068 2 |
| Wine | 16555.6794 2 | 17656.6765 4 | 17128.4579 3 | **16320.03 1** |
| **Ortalama Sıra** | **1.82** | **3.12** | **3.18** | **1.47** |

Optimizasyon yöntemlerinin uygun çözümler içerisinde sürekli daha iyiyi arama özelliklerine ek olarak Ağaç Tohum Algoritmasının bu şekilde lokal optimum değerlerden kaçınabilmesi, tohum üretme aşamasındaki güncelleme fonksiyonlarından birinde popülasyon içerisinde rastgele seçilen bir başka ağacın matematiksel konumunun göz önüne alınarak tohum üretilmesi ve dolayısıyla algoritmanın arama uzayı içerisinde sıçramalar yaparak ilerlemesinin sağlanmasından kaynaklanmaktadır.

Çalışmada kullanılan yöntemlerin tüm veriler üzerindeki kümeleme performanslar içerisindeki ortalama sıra dikkate alındığında Ağaç Tohum Algoritmasının bu problemler üzerinde diğer 3 geleneksel kümeleme yönteminden daha başarılı bir kümeleme işlemi yaptığı görülmektedir.

## SONUÇ (CONCLUSION)

Bu çalışmada kümeleme problemlerine Ağaç Tohum Algoritması uygulanmış ve elde edilen sonuçlar, kümeleme işleminde yaygın olarak kullanılan geleneksel yöntemlerle karşılaştırılmıştır. Ağaç Tohum Algoritmasının keşif ve işleme karakteristikleri sayesinde yerel ve global olarak daha iyi bir arama sağladığı ve geleneksel kümeleme yöntemlerinin ulaştıkları sonuçlardan daha iyi sonuçlara ulaşabildiği ortaya konmuştur. Kümeleme problemlerinde elde edilen ilham verici sonuçlar optimizasyon algoritmalarının farklı problemlere uygulanabilme ve doğrusal olmayan problemlerde sonuca ulaşma becerilerinin kümeleme problemlerinde alternatif bir çözüm yöntemi olarak kullanılabileceğini ve Ağaç Tohum Algoritmasının da bu alanda kullanılabilecek bir optimizasyon yöntemi olduğunu göstermektedir.

## KAYNAKÇA (REFERENCES)


[1] D.W. Van Der Merwe, A.P. Engelbrecht, Data clustering using particle swarm optimization, *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*. 1 (2003) 215–220. doi:10.1109/CEC.2003.1299577.

[2] J. Nasiri, F.M. Khiyabani, A whale optimization algorithm (WOA) approach for clustering, *Cogent Mathematics & Statistics*. 5 (2018) 1483565. doi:10.1080/25742558.2018.1483565.

[3] M. Fathian, B. Amiri, A. Maroosi, Application of honey-bee mating optimization algorithm on clustering, *Applied Mathematics and Computation*. 190 (2007) 1502–1513. doi:10.1016/j.amc.2007.02.029.

[4] T. Niknam, B. Amiri, J. Olamaei, A. Arefi, An efficient hybrid evolutionary optimization algorithm based on PSO and SA for clustering, *Journal of Zhejiang University: Science A*. 10 (2009) 512–519. doi:10.1631/jzus.A0820196.

[5] N. Kushwaha, M. Pant, S. Kant, V.K. Jain, Magnetic optimization algorithm for data clustering, *Pattern Recognition Letters*. 115 (2018) 59–65. doi:10.1016/j.patrec.2017.10.031.

[6] U. Maulik, A. Mukhopadhyay, Simulated annealing based automatic fuzzy clustering combined with ANN classification for analyzing microarray data, *Computers and Operations Research*. 37 (2010) 1369–1380. doi:10.1016/j.cor.2009.02.025.

[7] S.Z. Selim, K. Alsultan, A simulated annealing algorithm for the clustering problem, *Pattern Recognition*. 24 (1991) 1003–1008. doi:https://doi.org/10.1016/0031-3203(91)90097-O.

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

[9] D. Dua, C. Graff, {UCI} Machine Learning Repository, (2017). http://archive.ics.uci.edu/ml.

[10] J. MacQueen, Some methods for classification and analysis of multivariate observations, içinde: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967: ss. 281–297.

[11] N. Shi, X. Liu, Y. Guan, Research on k-means clustering algorithm: An improved k-means clustering algorithm, *3rd International Symposium on Intelligent Information Technology and Security Informatics, IITSI 2010*. (2010) 63–67. doi:10.1109/IITSI.2010.74.

[12] M. Karakoyun, A. Babalik, Data Clustering with Shuffled Leaping Frog Algorithm (SFLA) for

Classification, (2015). doi:10.15242/iae.iae0815009.

[13] E. Dinçer, Veri Madenciliğinde K-Means Algoritması Ve Tıp Alanında Uygulanması, (2006) 73–77.

[14] M. Karakoyun, A. Saglam, N.A. Baykan, A.A. Altun, Non-locally color image segmentation for remote sensing images in different color spaces by using data-clustering methods, *5th International Conference on Advanced Technology & Sciences (ICAT'17)*. (2017) 6–12.

[15] J.C. Dunn, A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics*. 3 (1973) 32–57. doi:10.1080/01969727308546046.

[16] J.C. Bezdek, Objective Function Clustering BT - Pattern Recognition with Fuzzy Objective Function Algorithms, içinde: J.C. Bezdek (Ed.), *Springer US*, Boston, MA, 1981: ss. 43–93. doi:10.1007/978-1-4757-0450-1_3.

[17] T. Velmurugan, Performance based analysis between k-Means and Fuzzy C-Means clustering algorithms for connection oriented telecommunication data, *Applied Soft Computing Journal*. 19 (2014) 134–146. doi:10.1016/j.asoc.2014.02.011.

[18] M.S. Kiran, TSA: Tree-seed algorithm for continuous optimization, *Expert Systems with Applications*. 42 (2015) 6686–6698. doi:10.1016/j.eswa.2015.04.055.

[19] M.S. Kiran, Tree Seed Algorithm, (y.y.). http://mskiran.kisisel.selcuk.edu.tr/tsa/ (erişim 12 Şubat 2020).

[20] A.C. Cinar, M.S. Kiran, Similarity and Logic Gate-Based Tree-Seed Algorithms for Binary Optimization, *Computers and Industrial Engineering*. 115 (2018) 631–646. doi:10.1016/j.cie.2017.12.009.

[21] A. Babalik, A.C. Cinar, M.S. Kiran, A modification of tree-seed algorithm using Deb's rules for constrained optimization, *Applied Soft Computing Journal*. 63 (2018) 289–305. doi:10.1016/j.asoc.2017.10.013.

[22] M. Aslan, M. Beskirli, H. Kodaz, M.S. Kiran, An improved tree seed algorithm for optimization problems, *International Journal of Machine Learning and Computing*. 8 (2018) 20–25. doi:10.18178/ijmlc.2018.8.1.657.