ÇOK AMAÇLI GENETİK ALGORİTMA KULLANARAK DNA MİKRODİZİ VERİLERİNİN KÜMELENMESİ

Mustafa KAHRAMAN

Yüksek Lisans Tezi
Biyomühendislik Anabilim Dalı
Danışman: Doç. Dr. Mehmet KAYA

EYLÜL – 2010

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK AMAÇLI GENETİK ALGORİTMA KULLANARAK DNA MİKRODİZİ VERİLERİNİN KÜMELENMESİ

YÜKSEK LİSANS TEZİ

Mustafa KAHRAMAN

(07132102)

Tezin Enstitüye Verildiği Tarih: 15 Eylül 2010
Tezin Savunulduğu Tarih: 30 Eylül 2010

Tez Danışmanı: Doç. Dr. Mehmet KAYA
Diğer Jüri Üyeleri: Doç. Dr. H. Soner ALTUNDOĞAN
Yrd. Doç. Dr. A. Bedri ÖZER

EYLÜL – 2010

ÖNSÖZ

Yüksek Lisans eğitimimi yaptığım süre boyunca tez konumun belirlenmesi, şekillenmesi, yürütülmesi ve tez sürecinin düzenlenmesinde en derin bilgilerini ve bilim alanındaki tecrübelerini benden esirgemeyen, araştırma ve çalışma şevkini kazanmamı sağlayan danışman hocam Sayın Doç. Dr. Mehmet KAYA'ya teşekkürlerimi bir borç bilirim.

Ders aşamasında teorik bilgilerini ve yardımlarını esirgemeyen Prof. Dr. Emine ÜNSALDI, Doç. Dr. İbrahim TÜRKOĞLU, Doç. Dr. Ali KARCI'ya ve bu tez çalışması için proje desteği sağlayan FÜBAP'a teşekkürlerimi sunarım.

Hayatım boyunca benden desteklerini esirgemeyen ve bu günlere gelmemi sağlayan aileme de sonsuz teşekkür ederim.

Mustafa KAHRAMAN
ELAZIĞ – 2010

İÇİNDEKİLER

Sayfa No

ÖZET

DNA mikrodizi teknolojisi, önemli biyolojik süreçlerdeki binlerce gen ifadesi seviyelerinin aynı anda izlenmesini olanaklı kılar. Gen ifadesi verilerinde saklı örüntüleri açıklama fonksiyonel genomu anlama için muazzam bir fırsat sunar. Bununla birlikte, biyolojik ağların karmaşıklığı ve çok büyük miktarlardaki genler, oluşan veriyi anlama ve yorumlamada bazı problemleri de beraberinde getirir. Bu problemlerin çözümü için sunulan bir alternatif, kümeleme tekniklerinin kullanılmasıdır. Kümeleme, mevcut verideki ilginç örüntüleri tanımlama ve doğal yapıları açığa çıkarmak için kullanılan veri madenciliği yöntemlerinden biridir. Kümeleme analizi verilen bir veri kümesini belirlenmiş özelliklere göre gruplara parçalama çabasıdır. Böylece bir grup içindeki veri noktaları, farklı gruptaki noktalara göre birbirine daha çok benzerdir.

Kümeleme algoritmalarında küme sayıları genellikle önceden verilir. Fakat bir veri kümesi için uygun küme sayısının önceden tahmin edilmesi alanın uzmanı için zor bir işlemdir. Bu tez çalışmasında bu problemin üstesinden gelebilmek için çok-amaçlı genetik algoritma tabanlı bir kümeleme yöntemi önerilmiştir. Yöntem k-means kümeleme algoritmasıyla çok-amaçlı genetik algoritma sürecini birleştirir. Leukemia, Lymphoma ve Kolon kanseri veritabanlarına uygulanan yöntemin sonuçları literatürde çokça kullanılan Dunn, Davies Bouldien, Silhoutte, C, SD ve S-Dbw küme geçerlilik indeksleriyle karşılaştırılmış ve yüksek doğruluk oranlı etkin çözümler verdiğini göstermiştir.

Anahtar Kelimeler: DNA Mikrodizi Teknolojisi, Kümeleme, Çok-amaçlı Genetik Algoritma, K-means

SUMMARY

Clustering DNA Microarray Data via Multi-Objective Genetic Algorithm

DNA microarray technology has now made it possible to simultaneously monitor the expression levels of thousands of genes during important biological processes. Elucidating the patterns hidden in gene expression data offers a tremendous opportunity for an enhanced understanding of functional genomics. However, the large number of genes and the complexity of biological networks greatly increase the challenges of comprehending and interpreting the resulting mass of data. A first step toward addressing this challenge is the use of clustering data. Cluster analysis seeks to partition given data set into groups based on specified features so that the data points within a group are more similar to each other than the points in different groups.

Clustering algorithms in general need the number of clusters as a priori, which is mostly hard for domain expert to estimate. In this thesis, in order to overcome this problem, a multi-objective genetic algorithm based method is proposed. The method combines the K-means clustering algorithm with multi-objective genetic algorithm process. The experimental results conducted on Leukemia, Lymphoma and Colon cancer databases have been compared to Dunn, Davies Bouldien, Silhoutte, C, SD ve S-Dbw cluster validation indexes which are widely used in the literature. So, we demonstrate the applicability and effectiveness of the proposed clustering approach.

Keywords: DNA Microarray Technology, Clustering, Multi-objective Genetic Algorithm, K-means

ŞEKİLLER LİSTESİ

Sayfa No

Sayfa No

KISALTMALAR

cDNA	: Complementary DNA
Cy3	: Cyanine 3
Cy5	: Cyanine 5
DBSCAN	: Density-Based Spatial Clustering of Applications with Noise
DNA	: Deoxyribonucleic Acid
GA	: Genetic Algorithm
mRNA	: Messenger RNA
PCR	: Polymerase Chain Reaction
SAGE	: Serial Analysis of Gene Expression
RNA	: Ribonucleic Acid
SD	: Scat Distance
SNP	: Single-nucleotide polymorphism

1. GİRİŞ

Kümeleme tekniklerinin, gen fonksiyonları, gen düzenlemeleri, hücresel süreçler ve hücre alt tiplerini anlamada faydalı oldukları kanıtlanmıştır. Benzer ifade örüntülü genler (birlikte ifade edilmiş genler) benzer hücresel fonksiyonlarla birlikte kümelenebilir. Bu yaklaşım daha önceden bilgi elde edilemeyen birçok gen fonksiyonunu anlamaya yardımcı olabilir [1]. Dahası aynı küme içinde birlikte ifade edilmiş genler muhtemelen aynı hücresel süreçte de ihtiva edilirler ve bu genler arasındaki ifade örüntülerinin güçlü ilişkisi birlikte düzenliliği de gösterir. Aynı küme içerisindeki genlerin promoter bölgelerindeki ortak DNA sekanslarını arama, tanımlamak için her bir gen kümesine özgü düzenleyici motiflere ve önerilmek için cis-düzenleyici elemanlara izin verir [2]. Gen ifadesi verilerinin kümelenmesiyle düzenliliğin çıkarılması transkripsiyon düzenleyici ağ mekanizması ile ilgili hipotezlerin ortaya çıkmasına sebep olur [3]. Sonuç olarak uygun ifade profillerine göre farklı örnekleri kümeleme geleneksel morfolojik tabanlı yaklaşımlarla tanımlanması zor olan alt hücre tiplerini açığa çıkarabilir.

Kümeleme, küme olarak adlandırılan ayrık sınıflar kümesine veri nesnelerini gruplandırma işlemidir. Öyle ki bir sınıf içindeki nesneler birbirlerine göre yüksek benzerliğe sahip iken ayrı sınıflardaki nesneler birbirine daha az benzerler. Kümeleme eğiticisiz bir sınıflandırma örneğidir. Sınıflandırmaya veri nesnelerini sınıflar kümesine atayan bir prosedür olarak bakılabilir. Böylece kümeleme örüntü tanımadan veya ayrıklaştırma analizi ve karar analizi olarak bilinen istatistiğin alanlardan farklıdır.

Günümüzde tipik bir mikrodizi tecrübesi 10^3 yada 10^4 gen içerir. Bu sayının 10^6'ya kadar erişmesi bekleniyor. Bununla birlikte bir mikrodizi tecrübesinde ihtiva edilen örneklerin sayısı genellikle 100'den daha azdır. Gen ifadesi verilerinin karakteristiklerinden biri genleri ve örnekleri kümelemenin anlamlı olmasıdır. Bir yandan birlikte ifade edilmiş genler ifadesi örüntülerine dayalı olarak gruplandırılabilirken diğer bir yandan örnekler homojen gruplar içerisine parçalanabilir [4]. Bu şekilde her bir grup klinik sendromlar veya kanser tipleri gibi bazı belirli makroskopik fenotiplere uygun olabilir. Böyle örnek tabanlı kümelemede, örneklere nesneler olarak, genlere de özellik olarak bakılabilir.

Şimdiye kadar kümelemenin kullanacağı bileşenler için birçok kabul ve terminolojiler göz önüne alındı. Bu kabullerden biri de küme sayısının önceden bilinmesidir.

1.1. Tezin Amacı

Bu tezin amacı DNA mikrodizi verilerini örnek tabanlı kümelemek için küme sayısı önceden belirlenmeden çok-amaçlı genetik algoritmalara göre yeni bir yöntem geliştirmektir. Bir kümeleme işleminde iki önemli parametre vardır. Bunlardan biri küme içindeki nesnelerin olabildiğince birbirine benzemesi diğer bir deyişle hata miktarının minimum olması, diğeri ise anlaşılabilirliği açısından küme sayısının minimum olmasıdır. Bu iki parametre birbiriyle çelişir durumdadır. Yani küme sayısı artarken hata azalır, benzer şekilde küme sayısı azalırken hata artar. Önerilen yöntem daha önce geliştirilmiş olan hızlı genetik k-means algoritmasını [5] çok-amaçlı genetik algoritma süreci ile birleştirir. Bu sayede daha etkin ve doğruluk oranı yüksek bir kümeleme yöntemi ortaya çıkmış olur.

1.2. Tezin Yapısı

Bu tezin bundan sonraki bölümleri şu şekilde organize edilmiştir. Bölüm 2'de DNA mikrodizi teknolojisi hakkında temel bilgiler verilmiştir. Bölüm 3'de kümeleme ve kümeleme analizi anlatılmıştır. Bölüm 4, genetik algoritmalar hakkında temel bilgiler içerir. Bölüm 5, çok-amaçlı genetik algoritma kullanılarak geliştirilen kümeleme yöntemini verir. Bölüm 6'da önerilen yöntemin uygunluk ve doğruluğunu test etmek için kullanılacak küme geçerlilik indeksleri anlatılmıştır. Uygulama sonuçları Bölüm 7'de verilmiştir. Bölüm 8 ise tezin sonuç kısmıdır.

2. DNA MİKRODİZİ TEKNOLOJİSİ

İnsan da dâhil olmak üzere tüm ökaryot hücreli canlılarda kalıtsal materyali taşıyan ve sonraki nesillere aktarılmasını sağlayan sarmal DNA yapıları bulunmaktadır. Bir insan hücresindeki DNA yaklaşık 2 metre uzunluğundadır ve 46 kromozom üzerinde 3 milyar civarında baz çiftini içermektedir. Proteinler aminoasit dizilerinden oluşur ve her bir aminoasit tam olarak üç ardışık nükleotid içerir. Örneğin, ATG nükleotid dizisi 'methionine' aminoasidini kodlar. Proteinler hücre hayatı için büyük önem taşır. Çünkü bunlar; temel yapı taşları, besin parçalayıcı, enzim, bağışıklık sistemi, dönüştürücü, aktivatör vb. olarak işlev görür. Uzun DNA iplikleri üzerindeki bazı özelleşmiş bölgelere gen adı verilir. DNA'nın küçük bir bölümünü oluşturan genler, DNA sarmalı üzerinde yer alır. Genler insanın saç renginden ayakkabı numarasına ve hatta yakalanabileceği hastalıklara kadar kişinin yaşamını belirleyen tüm proteinlerin salgılanmasını sağlamaktadır. Bir insanda yaklaşık 50000 gen bulunmaktadır.

Günümüzde teknolojinin ve Genetik Biliminin hızla ilerlemesiyle canlıların DNA gen dizilimlerinin elde edilmesini sağlayan bazı teknikler geliştirilmiştir. *Mikrodizi Teknolojisi* genlerin genomlardaki dizilimlerinin ölçülmesini sağlayan bir tekniktir. Bu tekniğin en önemli özelliği binlerce hatta on binlerce farklı genin ifade düzeylerini aynı anda ve hızlı bir şekilde inceleme olanağı sunmasıdır.

Gen ifadesi kavramı; bir DNA dizisi olan genlerin fonksiyonel protein yapılarına dönüşmesi süreci için kullanılan bir terimdir [6].

Geçtiğimiz on yıl boyunca gelişmekte olan Mikrodizi yöntemi, aynı anda çok sayıda genin ifade örüntüsünün etkili bir şekilde analiz edilmesi için yeni bir teknik sunmaktadır. Mikrodizi teknolojisi; robotik araçlarla uygulanan ve otomatik hale getirilen işlemlerin kullanılması ile büyük ölçekli analizlerin verimliliğini yüksek oranda arttırmıştır. Böylelikle bu teknik, gen ifade örüntülerinin analizinde bir devrim niteliğindedir [7].

Genlerin ifade örüntülerini bilmek, onların işlevleri hakkında önemli bilgiler verir. Hücredeki hemen hemen bütün değişimler mRNA yapılarındaki değişikliklerle oluşmaktadır. Diğer taraftan çevresel etkenlerin değişimleri de genlerin ifadelerinde değişikliklere neden olmaktadır. Genler ve gen ürünleri arasındaki etkileşimi anlamak için

genomdaki genlerin ifade değişimlerini analiz etmek gerekmektedir [8,9]. Bu analiz için de şu ana kadar geliştirilmiş en etkili ve hızlı yöntemlerden biri DNA Mikrodizi tekniğidir.

2.1. DNA Mikrodizi Teknolojisinin Uygulama Alanları

Mikrodiziler başta genetik gözlem ve araştırmalar olmak üzere birçok alanda kullanılmakta hatta yeni alt bilim dallarının doğmasını ve gelişmesini sağlamaktadır.

DNA mikrodiziler, kanser gibi bazı hastalıklar altında gen ifade değişikliklerini incelemek amacıyla kullanılmaktadır. Tümör profilleri kullanarak DNA mikrodizileri, böyle karmaşık hastalıkların ilerleyişi ve gelişiminin takip edilmesine olanak sağlamaktadır.

DNA mikrodiziler kullanılarak bir çok kompleks hastalıklar için ilaç keşfi, potansiyel tanı ve hastalığın muhtemel seyrini etkileyen maddeler (prognostik biyomarker) hedef olarak incelenebilmektedir.

DNA mikrodiziler ile genellikle kan örmeklerindeki virüsleri ve diğer hastalıklara neden olan patojenleri belirlemek için kullanılır. Bu özelliğiyle bir patojen tespit yöntemi olarak kullanılabilir

DNA Mikrodizi verileri son zamanlarda kalıtsal işaretleyicileri tanımlamak için kullanılacağı gibi bir genotipleme aracı olarak ta kullanılabilmektedir.

DNA Mikrodizi teknolojisine dayanan tek nükleotid polimorfizm (SNP) çipleri, bir dizi veya çip yaklaşımı kullanarak tek nükleotid polimorfizmlerinin yüksek verimli profillenmesine imkan verir. Böylece bu profillenen polimorfizmlerin ilgili hastalıkla olan bağlantısı daha kolay ve çabuk tespit edilebilmektedir [10].

Mikrodiziler vasıtasıyla bir hücreye yapılan bir müdahalenin hücreyi nasıl etkilediği, hangi metabolik olayları uyardığı, hangilerini durduğu uygulama öncesine göre kıyaslanarak ortaya çıkarılabilir. Benzer yaklaşımla hastalıkların metabolizma üzerine etkilerinin belirlenmesi ve tedavi için hedef moleküllerin saptanması amacıyla hasta ve kontrol gruplarında mikrodiziler ile genel ifade analizleri yapılabilmektedir. SNP mikrodiziler yardımı ile hastalığa ya da hastalığa yatkınlığa neden olan genler saptanabilmektedir [11].

Mikrodizi teknolojileri kullanılarak üzerinde çalışılan diğer bir alan ise hücre içindeki genetik düzenlemelerin ve düzenleyici fonksiyonların analizidir. Örneğin elde edilen yeni bir genin, transkripsiyonu başlatan veya durduran bir transkripsiyon faktörü olup olmadığının incelemesi yapılabilmektedir.

Bir mRNA ya da gen ifadesi profilleme deneyinde binlerce genin ifade düzeyleri, belirli hastalıkların veya tedavilerin etkilerini anlamak için eş zamanlı olarak gözlenmektedir. Örneğin mikrodizi tabanlı gen ifade profillemede, hastalıklı doku veya hücrelerle hastalıklı olmayanların gen ifadelerinin karşılaştırılması yapılmaktadır. Böylece patojenlerden ya da diğer organizmalardan etkilenerek değişikliğe uğrayan gen ifadelerini belirlemek mikrodizi teknolojisi ile mümkün olabilmektedir [12].

Farklı hücreler ya da birbirine yakın benzer organizmaların sahip olduğu genom içeriğinin tespit edilmesinde, başka bir deyişle gen haritasının çıkarılmasında mikrodiziler kullanılmaktadır [13,14].

2.2. Mikrodizi Verilerinin Elde Edilmesi

Biyolojinin temel prensibine göre bir protein transkripsiyon ve translasyon adı verilen ardışık iki olay ile üretilir. Transkripsiyon esnasında DNA, kesin olarak belirlenmiş bir konumda geçici olarak açılarak ilgili genle eşleşir ve tamamlama (tümleme) yoluyla mRNA sentezlenir. mRNA kodlayıcı genin kodunu üzerinde taşır. Daha sonra mRNA, nükleustan dışarı çıkarak protein sentezini gerçekleştiren ribozoma girer ve ribozom bu mRNA'daki diziye göre proteini sentezler. Ribozom süreci translasyon olarak bilinir [15]. mRNA'nın bu özelliklerinden dolayı mikrodizi teknolojisi genellikle mRNA düzeyleri üzerinde çalışılan yapı sunmaktadır.

DNA karakterizasyonu yapan yöntemlerin çoğu, DNA'nın iki tamamlayıcı (komplementer) tek iplikten, kısmen veya tamamen tamamlayıcı olan çift sarmala şekillenme yeteneği olan hibridizasyona (baz eşleşmesi) dayanmaktadır. [16] Hibridizasyon, çözeltideki iki tamamlayıcı molekül arasında ya da çözeltideki bir molekül ile katı bir yüzey üzerinde hareketsiz bir tamamlayıcı molekül arasında oluşabilmektedir. DNA mikrodizi yöntemleri de tek iplikli moleküller ile çip yüzeyindeki tek iplikli dizeler arasındaki hibridizasyon tepkimelerinden yararlanmaktadır [17].

DNA mikrodizileri, genellikle düzlemsel katı bir yüzey üzerine serbest nükleik asit örneklerindeki DNA hedeflerinin organize bir şekilde sabitlenmesiyle oluşturulur. [18]

Bir mikrodizi deneyi, bir cam, naylon ya da (yarı iletken endüstrisinden uyarlanan ve Affymetrix tarafından kullanılan) kuvars silikon levha üzerine büyük bir cDNA (tamamlayıcı DNA) veya oligonükleotid DNA'nın dizi şeklinde sabitlenmesini gerektirir. Bu dizi daha sonra iki farklı floresan renk ile etiketlenen iki mRNA prob serisi ile tepkimeye girer. Probların hibridizasyonundan sonra mikrodizi, tüm spotların bir görüntüsünü elde etmek için genellikle bir lazer ışını ile taranır. Her spottaki floresan sinyalin yoğunluğu, o spottaki belirli bir dizi ile birleşmiş mRNA düzeylerinin bir ölçümü olarak kabul edilir. Tüm spotların görüntüsü, her spottaki DNA dizisi hakkındaki bilgi ile bağlantılı özel yazılım kullanılarak analiz edilir. Bu daha sonra, seçilen deneysel ve kontrol (referans) örnekleri için gen ifade düzeylerinin genel bir profilini üretir [7].

Bir mikrodizi deneyi kısaca aşağıdaki adımları içerir;

1. Mikrodizi hazırlama
2. Prob hazırlama ve hibridizasyon
3. Tarama
4. Düşük seviye veri analizleri
5. Yüksek seviye veri analizleri

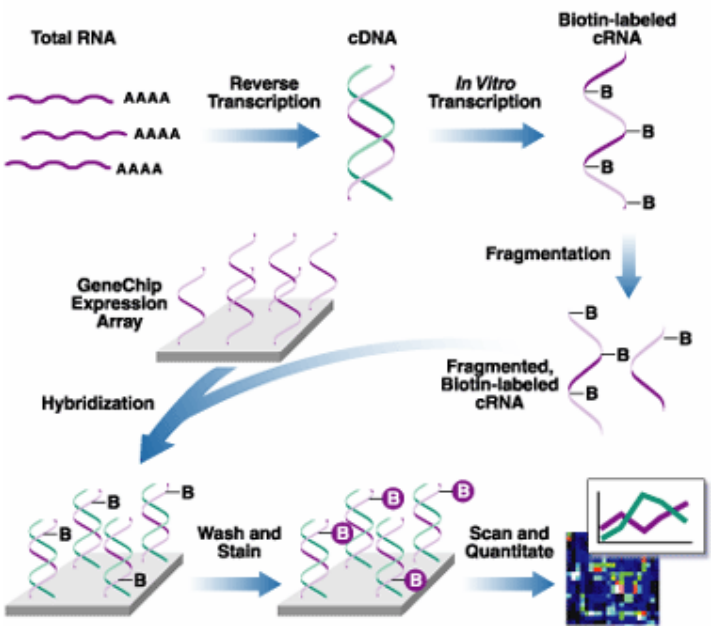
1. Adımda, deneyde kullanılacak mikrodizinin seçilmesi ve hazırlanması gerekmektedir. Dizi üzerine yerleştirilecek olan DNA dizilerinin yapısı, madde üzerine dizileri sabitleme tekniği ve bunların seçilmesi hassas olan adımlardır.[7] Genellikle en yaygın olarak kullanılan teknoloji cDNA mikrodizilerdir. cDNA, ifade düzeylerini doğrudan ölçemez. Daha çok iki örnekteki gen ifade düzeylerini kıyaslayarak dolaylı olarak ölçer. Böylece iki örnek arasındaki göreceli gen miktarını verir. Bu örneklerden biri daha önceden dizisi bilinen referans örneğimizdir. Aynı referans örnek kullanılarak birçok deney yapılabilir ve kıyaslanabilir.[15] Bu adımda hedef genler ve işlemde kullanılacak olan mikrodizi tespit edilir. Hedef genin ölçeğine göre uygun kapasiteye sahip mikrodizi seçilir. Günümüzde 25x75 mm slayt üzerine robotik baskı yöntemiyle 50000'den fazla DNA probları sabitlenebilir. Bu sayede yüksek yoğunluklu dizi yaklaşımı kullanılarak insan genomunun tamamını dizilemek mümkündür [19]. Kullanılacak slayt tipine karar verildikten sonra slayt işleme hazırlanır. İşleme hazırlık, slaytın kaplanması ve temizlenmesini içerir.

Genellikle cam mikroskop slaytları kullanılır. Daha önce tespit edilen cDNA'lar, hazırlanan cam slaytlar üzerine matris platformunda basılır. Bunun için *arrayer* adı verilen otomatikleşmiş bir robot kullanılır. Bu arrayer, slaytlar üzerine hafifçe vuran ve böylelikle küçük bir DNA miktarının bırakılmasını sağlayan baskı uçlarına sahiptir. Her vurulan alana spot (nokta) adı verilir. Her spot önceden belirlenmiş bir genle doldurulur. Spotlar matrislere benzer biçimde hem yatay hem de dikeyde eşit olarak hizalanmıştır. Baskı işlemi sırasında, baskı uçlarının keskinlikleri değişir. Bu da slayt üzerindeki spotların boyutunu ve geometrisini değiştirir. Bu nedenle baskı uçları periyodik olarak yenisi ile değiştirilir.

2. Adımda, probların hazırlanması ve hibridizasyon işlemleri gerçekleştirilir. Mikrodizi ile tepkimeye girmesinden sonra mRNA problarını hazırlamak için ilk adım, deneysel ve kontrol örneklerinden RNA popülasyonunun yalıtılmasıdır. Bunun için iki farklı ortamdaki örneklerden iki prob seçilir, mRNA çıkarılır ve ters transkripsiyon ile tamamlayıcı DNA'ya (cDNA) sentezlenir. Daha sonra cam bir tüp içerisinde cDNA transkripsiyon ile cRNA'ya dönüştürülür. Elde edilen bu RNA'lar biotin adı verilen Cy3 ve Cy5 ile iki farklı şekilde renklendirilerek (yeşil ve kırmızı) iki farklı floresan boya ile etiketlenir. Hazırlanan bu iki prob karışımı daha sonra mikrodizinin üzerine dökülür. Mikrodizi üzerindeki moleküllere tamamlayıcı olan RNA'lar, mikrodizi üzerinde bulunan ipliklerle hibridize olur. Bu nedenle spotlar ve problardaki genler birbirini tamamlayıcı ve bağlayıcıdır. Her bir gen yalnızca dizi üzerindeki tamamlayıcı noktası ile eşleşir. DNA dizisi daha sonra yıkanarak hibridize olmayan veya eşleşmeyen bölgeler dizi üzerinden uzaklaştırılır [7,15,19].

3. Adımda mikrodizi tarama işlemi yapılır. Prob yıkama ve hibridizasyondan sonra mikrodizi maddesinin (substrat), maddenin yapısına uygun olan teknik kullanılarak görüntüsü çıkarılır. Çipin yüksek yoğunluğu ile bu iş, genellikle çipin yüksek hassasiyette taranmasını gerektirir. RNA ile eşleşen spotlar, belli diziye hibridize olan etiketli RNA'ların seviyelerine göre sinyal gösterecektir. Bunun için hibridize olan genlerin miktarına, kırmızı ve yeşil floresan boyaların yoğunluğunun ölçülmesi ile karar verilebilir. Burada asıl önemli olan, örnekteki bir gen çok yüksek derecede ifade edilmişse o zaman onun miktarı fazladır ve yoğunluğu yüksektir. Yoğunlukların ölçülmesi ile ifade seviyeleri bu yöntemle ortaya çıkarılabilir. Yoğunluk ölçümü *tarayıcı (scanner)* denilen farklı bir donanım ile yapılmaktadır. Bu tarayıcı tüm slaytı tarar ve onun renkli görüntüsünü

oluşturur. Tarayıcının çıkışı, dizinin yüksek çözünürlüklü renkli görüntüsüdür [7,15]. Şekil 2.1'de mikrodizi deneyinin adımları görülmektedir.

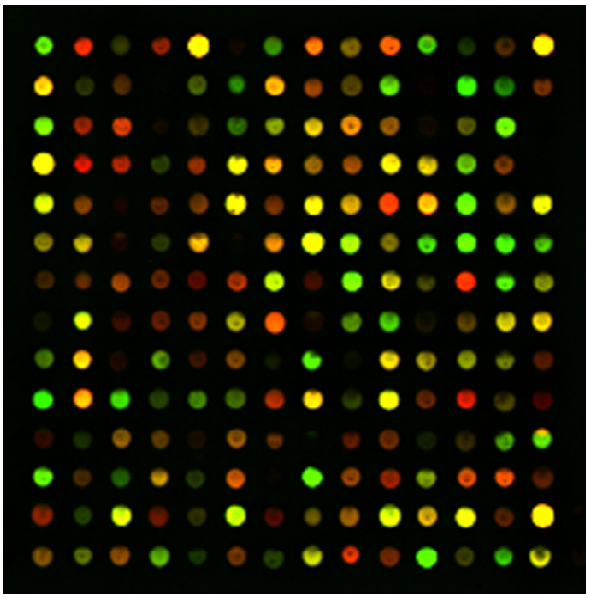


Şekil 2.1. Bir mikrodizi deneyinin adımları (Affimetrix web sitesinden)

4. Adım düşük seviye veri analizini içerir. Noktaların sınırlarını belirlemek için tarayıcıdan elde edilen renkli görüntü, görüntü analiz programları tarafından işlenir. Taranan görüntü, doğal gürültünün varlığından dolayı kusursuz değildir. Arka planın tamamen siyah olması istenir ancak hibridizasyon ve tarama hataları gürültüye sebep olur. İki kanal (kırmızı ve yeşil) ayrı ayrı ölçülür ve gen ifade seviyelerini temsil edecek şekilde birleştirilir. Eğer belirli bir nokta kırmızı ise, o zaman ona karşılık gelen gendeki kırmızı boyalı problar yeşil boyalı problardan daha fazladır. Eğer hemen hemen eşit ise renk sarıdır. Bir noktanın siyah olması demek, genin her iki probta da ortaya çıkmaması anlamına gelir.

Bu adımlardan sonra işlenmemiş veri kümesi elde edilir. Özet olarak işlenmemiş veri kümesinde slayt üzerindeki her bir gen için bir dizi vardır. Her dizideki kırmızı ve yeşil kanallar için iki yoğunluk seviyesi vardır. Aynı zamanda her bir dizide bu kanallar için iki arka plan rengi mevcuttur [15]. Şekil 2.2'de bir mikrodizi verisi görünmektedir.

5. Adımda işlenmemiş veri kümesindeki veriler yüksek seviye veri analizine tabi tutulur. Yüksek seviye veri analizi için kullanılan başlıca yöntemlerden biri kümeleme analizidir. Kümeleme analizi geleneksel olarak filogenetik araştırmalarda kullanılır ve mikrodizi analizi için de benimsenmektedir. Kümeleme analizinin mikrodizi teknolojisindeki amacı benzer profilli kümelere deney yapmak veya benzer gen gruplarını elde etmektir [7]. Kümeleme analizi sonraki bölümde ayrıntılı olarak incelenecektir.



Şekil 2.2. Mikrodizi verisi. Kırmızı spotlar, ilgili gene ait test cDNA sinyalinin referans cDNA sinyalinden yüksek, yeşil spotlar düşük, sarı spotlar ise eşit olduğunu gösterir

2.3. DNA Mikrodizi Teknolojisinin Avantajları ve Dezavantajları

DNA mikrodizi yöntemleri, diğer profilleme yöntemlerinden (SAGE, SH, PCR gibi) daha iyidir. Büyük bir potansiyele ve çok parlak bir geleceğe sahiptir. Özelliklerinden dolayı bu teknolojiyi kullanmak çok daha avantajlıdır [10,20,21]. Başlıca avantajları şunlardır;

- Kullanımı daha kolaydır.
- Bir defada binlerce geni analiz ettiği için yüksek verime sahiptir.
- Kısa sürede büyük miktarda veri oluşturabilir.

- Büyük ölçekli dizileme gerektirmez.
- Çok sayıda örnekten binlerce genin hesaplanmasına olanak sağlar. Hatta tek bir deneyde, tüm genomun gen ifadesini görüntüleme imkânı verir.
- Mikrodizi tekniği son derece kullanıcı dostudur. Radyoaktif ve zehirli kimyasallar içermez. Diziler kolaylıkla ve düşük maliyetle yenilenebilir.
- DNA mikrodizi teknolojisini kullanmak nispeten daha ucuzdur.
- Verilen bir dizi üzerinde temsil edilen düşük miktardaki mRNA kopyalarını belirlemede yeteri kadar hassastır.

Mikrodizilerin ilk başarıları ve olumlu sonuçları büyük bir heyecan yaratmıştır. Ancak buna rağmen bazı problemleri de beraberinde getirmiştir. Teknolojinin kurulduktan sonraki kullanım maliyeti düşüktür. Fakat ilk olarak teknoloji ekipmanlarının temini, kurulumu gibi başlangıç maliyetleri yaklaşık 60000 $ civarındadır. Bundan sonra bir mikrodizi kopyası başına maliyet küçüktür ve genellikle 100 $'dan daha azdır [20].

Mikrodizi teknolojisinin deneysel kısmı ve verilerin elde edilmesi çok iyi çalışır gibi görünmektedir [22]. Deneysel aşamalardan sonra işlenmemiş veriler elde edilir. Ancak asıl büyük zorluk bundan sonra başlamaktadır. Bu işlenmemiş verilerin yorumlanması ve her yönü ile analiz edebilecek biyoinformatik yazılımların tam olarak geliştirilememesi önemli dezavantajlardan biridir. Birçok bilim adamı ve araştırmacı bu konu üzerine yoğunlaşmıştır.

3. KÜMELEME VE KÜMELEME YÖNTEMLERİ

Kümeleme analizi özellikle son yılarda popüler olan çok değişkenli istatistiksel yöntemlerden biridir. Bu yöntem bilhassa bilim ve iş alanlarında, birçok durumda uygulanabilen, etkili ve kolay yorumlanabilen bir yöntem olma özelliğini taşımaktadır [23]. Kümeleme analizinin genel amacı, gruplandırılacak verileri benzerliklerine göre alt gruplara ayırarak onları incelemek ve açıklamaktır. Başka bir deyişle, çalışmada yer alan tüm bireyler veya nesneler arasındaki benzerlikler esas alınarak, benzer bireylerin aynı gruplarda veya kümelerde toplanması kümeleme analizinin esasını teşkil etmektedir [24].

Kümeleme işlemi için literatürde birçok yöntem ve algoritma bulunmaktadır. Bu yöntemler zamanla geliştirilerek daha başarılı bir şekilde kümelerin bulunması amaçlanmıştır. Bu yöntemlerin yaygın kullanılanlardan bazıları bu çalışmada incelenmiştir.

3.1. Kümeleme Analizi

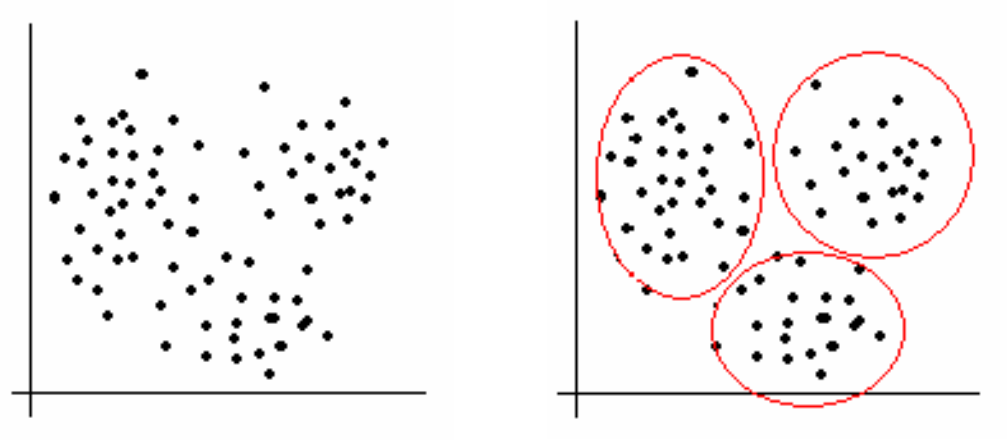
Kümeleme, öngörülecek alanların belirlenmesini ve birbirine benzeyen verilerin altkümelere ayrılmasını hedefler. Kümeleme analizinin hedefi, veri setinde doğal olarak meydana gelen alt sınıfları bulmaktır. Kümeleme, veri setinin, kümeler olarak adlandırılan sınıflar seti haline getirmek amacıyla bölümlenmesi sürecidir. Her kümenin üyeleri bazı ortak ilginç özellikleri paylaşmaktadır.

Kümeleme analizi, veriden saklı örüntülerin keşfi uygulamalarında sıkça kullanılmaktadır. Bu kümeleme analizini; bilimsel bilgi keşfi, bilgiye erişim, sayısal biyoloji, web kayıtları analizi, suç analizleri ve bunun gibi birçok alanda ideal bir yöntem olarak kullanılmasını mümkün kılmıştır.

Kümeleme analizi literatürde aynı zamanda Eğiticisiz Sınıflandırma (Unsupervised Classification) olarak da tanımlanmaktadır. Sınıflandırma analizinde bireylerin ya da nesnelerin daha önceden belirlenmiş olan sınıflardan hangisine dâhil olduğu tahmin edilmeye çalışılmaktadır. Kümelemede ise önceden tanımlanmış sınıflar (kümeler) bulunmaz. Mevcut bireylerin özelliklerine ve konumlarına göre, benzer bireylerin aynı

gruplarda yer aldığı kümeler oluşturulur. Bu sebeple bazı uygulamalarda kümeleme yöntemi, sınıflandırma yönteminin bir önişlemi olarak görev alabilmektedir.

Kümeleme analizi, ana kitlede yer alan yapıyı mümkün olduğunca ortak özellikli (homojen) ve kendi aralarında mümkün olduğunca farklı (heterojen) alt gruplara bölerek birimleri ve değişkenleri gruplamak, sınıflamak, kümelemek için yapılır [25].



(a) **(b)**

Şekil 3.1. Veri yığınlarının kümelere ayrılması

Kümeleme analizi uygulanmadan önce Şekil 3.1 (a)'da görüldüğü gibi karmaşık bir dağılıma sahip olan veri yığını, kümeleme analizi sonrasında benzer özelliklerine göre Şekil 3.1.(b)'de görüldüğü gibi kümelere ayrılmıştır [25].

3.2 Kümeleme Analizinin Başlıca Kullanım Alanları

Veri kümeleme güçlü bir gelişme göstermektedir. Veri tabanlarında toplanan veri miktarının artmasıyla orantılı olarak, kümeleme analizi son zamanlarda veri madenciliği araştırmalarında aktif bir konu haline gelmiştir [26]. Kümeleme, ticaret, biyoloji, tıp, psikoloji, sosyoloji, arşivcilik, internet, coğrafya gibi bilim ve iş sahalarında uygulama alanları bulmaktadır. Şimdi bunlardan bazılarına kısaca değinelim.

Biyoloji: Kümeleme, uzun yıllar boyunca bitkileri ve diğer canlıları taksonomi kapsamında belirli özelliklerine bağlı olarak cins, tür, sınıf, aile vb gibi hiyerarşik sınıflara

ayırmada kullanılmıştır. *Son yıllarda ise daha çok büyük miktardaki gen bilgilerini analiz etmek ve böylece aynı özelliklere sahip olan gen gruplarının bulunmasını sağlamak için kullanılmaktadır.*

Psikoloji ve Tıp: Bir hastalık veya sağlık durumundaki çeşitli değişimleri ortaya çıkarmak için kümeleme analizi kullanılabilir. Örneğin kümeleme, depresyonun değişik türlerinin belirlenmesinde kullanılmıştır. Ayrıca diğer hastalıkların zaman ve mekâna göre dağılımlarının ortaya çıkarılması gibi uygulamalarda da kullanılabilmektedir.

Ticaret ve Pazarlama: Ticaret ve Pazarlama, o anki müşteriler veya potansiyel müşterilerle ilgili büyük miktarda bilgi toplar. Kümeleme bu müşterileri daha küçük alt gruplara ayırmakta ve böylece daha ayrıntılı analiz ve pazarlama aktiviteleri yürütmekte kullanılabilir. Ürünü, markayı, pazarı, tüketici tercihlerini ve davranışlarını analiz etmek için kullanılabilir.

Bilgi Çıkarımı: İnternet dünyası milyarlarca web sayfası içermektedir. Arama motoruna yapılacak herhangi bir sorgu, geriye binlerce sayfa sonuç döndürebilir. Kümeleme, bu bilgilerin çeşitli gruplara ayrılmasında kullanılabilir. Böylece her grup sorgunun belli bir yönüne karşılık gelebilir. Örneğin bir "film" sorgusuna için sonuçlar, eleştiri, fragman, yıldızlar ve tiyatrolar olarak kümelere ayrılabilir. Böylece kullanıcının sonuçları daha iyi irdelemesine yardımcı olur.

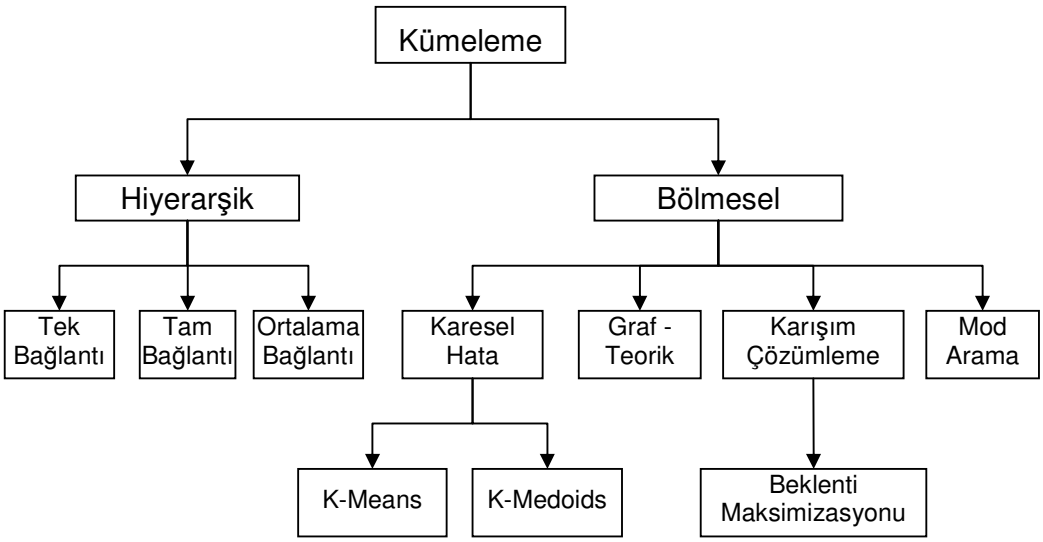
İklim: Şu ana kadar kümeleme analizi, kutupsal bölgelerin atmosferik basınçlarına ilişkin paternlerin ve kara iklimine önemli etkisi bulunan okyanus alanlarının bulunmasında kullanılmıştır [27].

Arşivcilik*:* Belgelerin, kitapların ve dokümanların belirli kriterler doğrultusunda arşivlenmesinde kümeleme analizi kullanılabilmektedir.

3.3. Bazı Kümeleme Yöntemleri

Üzerinde en çok tartışmanın yapıldığı konulardan birisi, kümeleme yöntemlerini birbirinden ayırma kriteridir. Kümeleme yöntemlerini sınıflandırmanın belli bir kuralı yoktur. Bazıları kümeleme türlerini, onların hiyerarşik (iç içe) olup olmadıklarına göre, bazıları kümelemenin keskin ya da bulanık olup olmadıklarına göre, bazıları da tam ya da

kısmi olmalarına göre sınıflara ayırırlar. Ancak genelde hiyerarşik ve bölmesel kümeleme şeklinde bir taksonomi daha yaygın olarak kabul görmüştür. Şekil 3.2. de görüldüğü gibi kümeleme yaklaşımlarını temelde hiyerarşik ve bölmesel olarak sınıflara ayıran bir taksonomi şeması görülmektedir [28].



Şekil 3.2. Kümeleme yaklaşımları için bir taksonomi

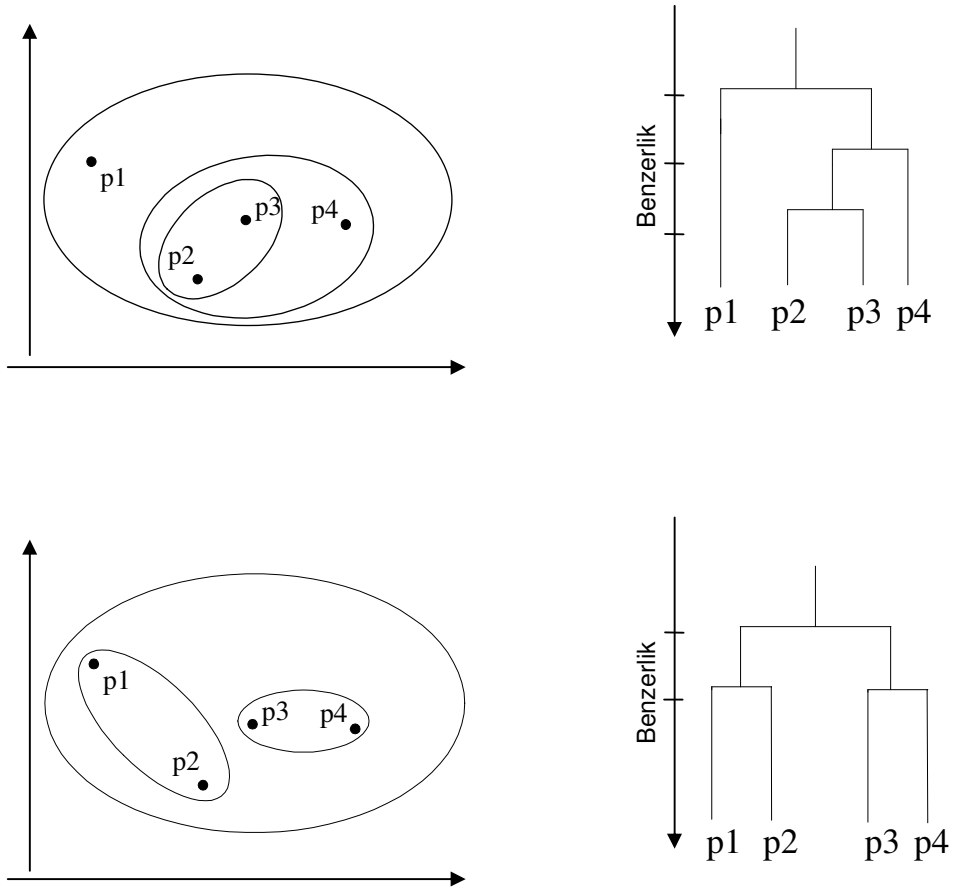
Bir bölmesel kümeleme, her bir veri nesnesinin yalnızca bir kümede bulunduğu, veri nesnelerinin kesişmeyen alt kümelere ayrılmasıdır. Kümelerin alt kümelere sahip olması durumunda ise hiyerarşik kümeleme yapılmış olur.

3.3.1. Hiyerarşik Kümeleme Yöntemleri

Hiyerarşik kümeler, ağaçlar şeklinde organize edilmiş iç içe geçmiş alt kümelerden oluşur. Yaprak düğümler dışında, ağaçtaki her bir düğüm (küme), kendi alt kümelerini kapsayan bir üst küme, ağacın kökü ise tüm nesneleri içeren tek bir kümeyi temsil eder [27]. Bu teknikte öncelikle bireyler ya da değişkenler arasındaki uzaklıklar hesaplanır. Daha sonra oransal uzaklıklar *dendogram* adı verilen ağaç grafiği üzerinde gösterilir. Dendogram yardımıyla da birbirine yakın birimler ya da değişkenler, birbirine yakınlık

oranları bakımından gruplandırıldıklarından kümelerin görsel algılanabilirlikleri de artmaktadır.

Şekil 3.3.'de iki boyutlu veri seti kullanılarak hiyerarşik bir kümeleme örneği gösterilmiştir. Şekilde iki adet alt kümeye ayrılabilen toplam dört adet noktanın dağılımı ve bu dört noktaya karşılık gelen dendogram yapısı görülmektedir. Şekillerde görüldüğü gibi küme sayısı arttıkça küme içi benzerlik seviyesi artmaktadır. Dendogram, farklı veri kümeleri veren farklı seviyelere ayrılabilmektedir.



Şekil 3.3. Veri noktalarına ait hiyerarşik kümeleme örnekleri ve dendogramları

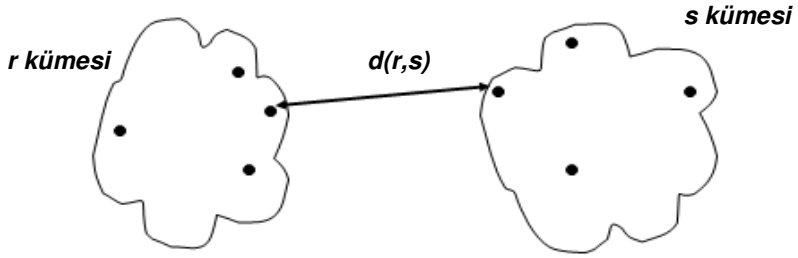
Tek-Bağlantı Kümeleme (En Yakın Komşu) Yöntemi:

Tek-bağlantı (Single-Link) yöntemi ilk olarak 1951'de Florek tarafından açıklanmıştır. Tek-bağlantı kümeleme analizi yöntemi, hiyerarşik kümeleme yöntemleri arasında en basit olanıdır. En yakın komşu yöntemi olarak da bilinir. Bu yöntemde iki kümenin nesneleri arasındaki mesafeler hesaplanır ve birbirine en yakın olan iki küme veya gözlem birleştirilir.

Tek-bağlantı kümelemede bu mesafe şu şekilde hesaplanır;

$$\mathbf{d}(r,s)=min(\mathbf{D}(i\ ,\ j)\ ) \qquad \textbf{(3.1)}$$

Burada ***i,*** ***r*** kümesine ait bir nesnedir, ***j*** ise ***s*** kümesine ait bir nesnedir. mümkün olan tüm nesne çiftleri (***i, j***) arasındaki mesafeler teker teker hesaplanır. Bu mesafeler içerisinden minimum mesafe (**r, s**) kümeleri arasındaki mesafe olarak adlandırılır. Başka bir deyişle kümeler arası mesafe mümkün olan en kısa mesafedir. Bu durum Şekil 3.4'de görülmektedir [27].



Şekil 3.4. Tek-bağlantı tekniği ile kümeler arası bağlantıların bulunması

Tek-bağlantı kümeleme algoritması şu şekildedir.

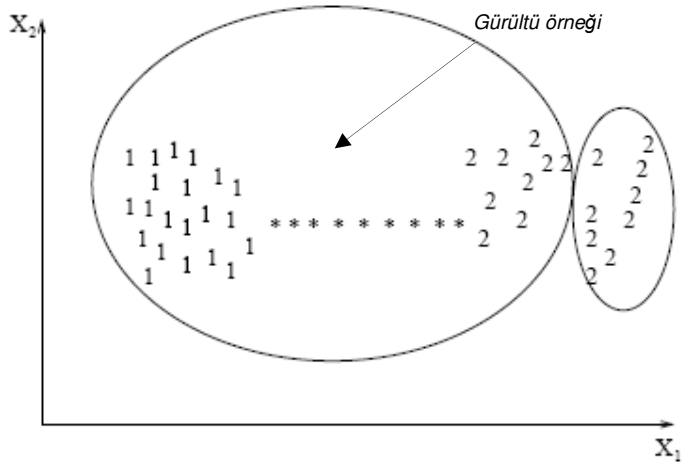
1. Adım: Tüm örnekleri kendi kümesine yerleştir. Kuralsızca oluşturulan örnek çiftleri kullanarak bir liste inşa et ve artan sırayla bu listeyi sırala.

2. Adım: Mesafelerin sıralanmış listesini kullanarak, bir biriyle d_k adıyla adlandırılan, en kısa mesafeyle bir birine bir graf kenarıyla örnek çifteleri birbiriyle bağlanır. Eğer tüm örnekler bir bağlantı grafının üyesi haline gelmiş ise durulur. Değilse adım tekrarlanır.

3. Adım: Sonuçta oluşan ürün tek bir bağlantılı graftır [28].

Bu yöntemin sağladığı en önemli avantaj, benzerlik matrisinin tekdüze oluşumlara karşı değişken olmaması ve veri setindeki bağlı değerlerden etkilenmemesidir. Bu metot birbirinden yeterince ayrık olan kümeleri belirlemede oldukça iyidir. Tek-bağlantı yöntemiyle elde edilen kümedeki değerler, diğer kümedeki değerlere göre birbirine daha çok benzerdirler. Elips şeklinde dağılmayan değerleri aynı kümede toplayabilen az sayıdaki kümeleme yöntemlerinden biridir. Örneğin U şeklinde dağılan noktalar, bu yöntemle aynı kümenin içinde toplanabilirler. Fakat bu şekilde oluşan bir kümenin zıt taraflarındaki değerler, birbirinden oldukça farklı olabilirler.

Tek-bağlantı yönteminin dezavantajı, birbirinden çok az farklı olan kümeleri ayırt etmede yetersiz kalmasıdır. Gürültüden de etkilenerek farklı kümeler elde edebilmektedirler. Zincirleme etkiyle ortaya çıkan bir durum nedeniyle çoğu kez kümeler düzgün yapılamayabilir ve birbirinden çok farklı üyeler aynı kümede bulunabilir. Şekil 3.5'de bu durum gösterilmiştir.



Şekil 3.5. Gürültü örneği (*) içeren bir veri setinin Tek-bağlantı tekniği ile kümelenmesi

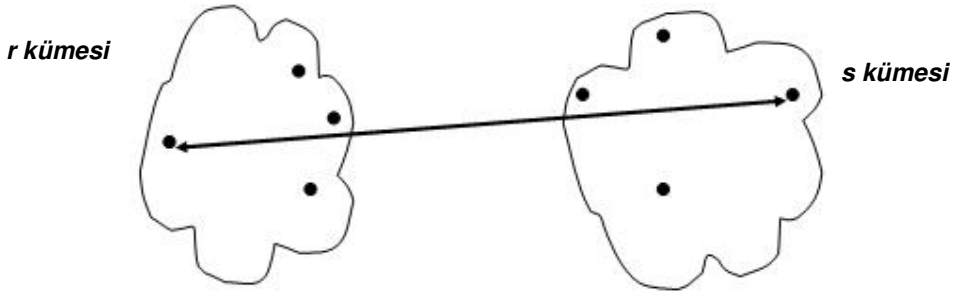
Tam-Bağlantı Kümeleme (En Uzak Komşu) Yöntemi:

Tam-bağlantı (Complete-Link) kümeleme yöntemi, tek-bağlantı kümeleme yönteminin tam tersi bir mantıkla kümeleme yapmayı sağlar. Bu yöntemin uygulanması tek bağlantı yöntemi kadar kolaydır. En uzak komşu yöntemi olarak da bilinir. Bu yöntemde kümeler arası mesafe, en uzak nesne çifti arasındaki mesafe olarak tanımlanır.

Bu yöntem gruplar arası mesafeyi şu şekilde hesaplar;

$$d(r, s) = maks\ (D(i, j))\qquad (3.2)$$

Burada ***i, r*** kümesine ait bir nesnedir, ***j*** ise ***s*** kümesine ait bir nesnedir. *G*ruplar arası mesafe, iki grubun nesneleri arasında oluşabilecek tüm bağlantı durumlarındaki en uzak mesafedir. Bu durum Şekil 3.6'de ifade edilmiştir.



Şekil 3.6. Tam-bağlantı tekniği ile kümeler arası bağlantıların bulunması

Tam-bağlantı algoritması şu şekildedir.

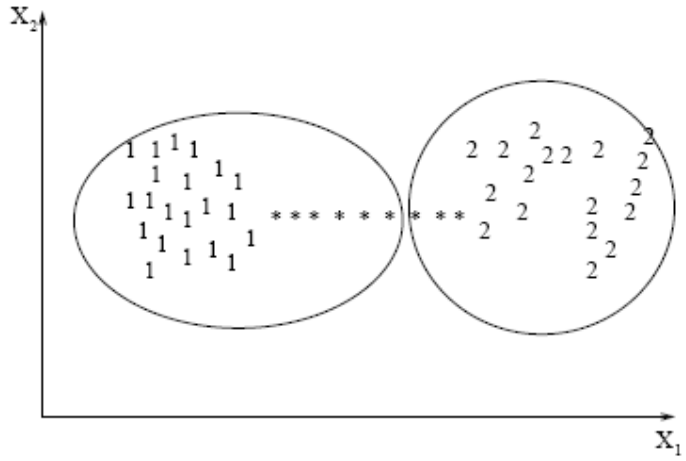
1.Adım: Tüm örnekleri kendi kümeleri içerisine yerleştir. Kuralsızca oluşturulan örnek çiftleri kullanarak bir liste inşa et ve artan sırayla bu listeyi sırala.

2.Adım: Bu sıralanmış listeyi kullanarak her örnek çiftini en kısa mesafe ile bağlanacak şekilde bir graf kenarıyla bir birine bağla. Eğer tüm örnekler tam bir graf içerisine bağlanmış ise dur. Değilse 2. adımı tekrarla.

3.Adım: Algoritmanın çıktısı bir birine tam olarak bağlanmış graftır.

Bu yönteme göre benzerlik matrisi tekdüze dönüşümlere karşı değişken değildir. Tam-bağlantı yöntemi, tek-bağlantı yönteminden daha katı kurallara sahiptir. Bu sebeple tam-bağlantı yöntemi X-Y koordinat sisteminde birbirine yakın noktaların elips şeklinde dağılım göstermesi durumunda bu değerleri kümelemede tek-bağlantı yöntemine göre daha iyi sonuçlar verir. Ayrıca zincirleme gürültü örneklerine karşı daha az duyarlıdır. Şekil 3.4'deki tek-bağlantı yöntemi ile yapılan gürültülü kümeleme örneği Şekil 3.7'da tam-

bağlantı yöntemi ile gerçekleştirilmiştir. Zincirleme etkiden daha az etkilenmiş ve kümeler daha düzgün ayrılmıştır.



Şekil 3.7. Gürültü örneği (*) içeren bir veri setinin Tam-bağlantı tekniği ile kümelenmesi

Ortalama Bağlantı Kümeleme Yöntemi:

Ortalama bağlantı tekniğinde kümeler arasındaki oluşabilecek nesneler arası bağlantıların tümünün ortalaması alınır.

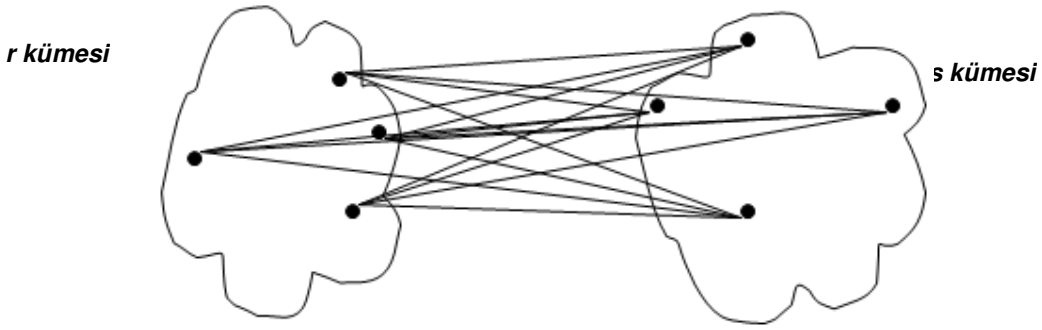
Bu yöntemde ortalama mesafe şu formül ile hesaplanır;

$$\mathbf{D(r,s) = T_{rs} / (N_r * N_s)} \qquad \mathbf{(3.3)}$$

Burada $T_{rs,}$ **r** ve **s** kümesindeki nesneler arasında oluşabilecek tüm mesafelerin toplamıdır. N_r ve N_s **r** ve **s** kümelerinin elaman sayılarıdır. Hiyerarşik kümelemenin tüm aşamalarında ***D(r,s)'***nin minimum değeri için r ve s kümeleri birleştirilir. Bu durum Şekil 3.8'de gösterilmiştir.

Ortalama bağlantı tekniği, yaygın olarak biyoloji biliminde kullanılmaktadır, bununla birlikte sosyal bilimlerde kullanımı da giderek artmaktadır. Genellikle tam bağlantı ve ortalama bağlantı tekniklerinde benzer dendogramlar oluşmaktadır. Ancak her bir

yöntemde uzaklık farklı tanımlandığı için birleştirmeler farklı seviyelerde ortaya çıkabilmektedir [29].



Şekil 3.8. Ortalama bağlantı tekniği ile kümeler arası bağlantıların bulunması

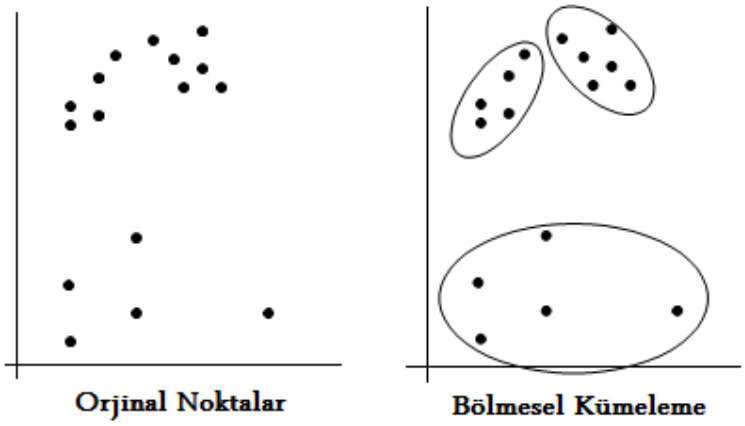
Genel olarak hiyerarşik kümele yöntemlerinin dezavantajı, gözlem birimi ya da değişken sayısı arttıkça yapılacak işlem miktarı da artmaktadır. Bu yüzden büyük boyutlu veri setlerinde çok zaman alıcı bir yöntemdir. Ancak son zamanlarda daha güçlü işlem kabiliyetine sahip bilgisayarların yaygınlaşması bu dezavantajı azaltmıştır.

Kümeleme sonucu elde edilecek küme sayısına, kullanıcı tarafından karar verilmemiş ise hiyerarşik kümeleme algoritmaları tercih edilmektedir.

3.3.2. Bölmesel Kümeleme Yöntemleri

Bölmesel kümeleme (partitional clustering) teknikleri, dendogram gibi iç içe bir kümeleme yapısı üzerinde çalışmak yerine tek seviyeli kümeleri bulan işlemler gerçekleştirirler [28]. Her bir küme, başka alt kümelere bölünmez ve diğer kümelerden kesin bir şekilde ayrılır. Şekil 3.9'da bölmesel bir kümeleme örneği görülmektedir [27].

Genellikle veri setinde bulunan nesne sayısı '**n**', kümeleme sonucu oluşturulacak olan ve kullanıcı tarafından belirlenen küme sayısı ise '**k**' ile gösterilir. Bölmesel kümeleme yöntemleri, veri setindeki bu **n** adet nesneyi **k** adet kümeye bölerler. Kümeleme sonucunda her nesne mutlaka bir kümenin üyesidir. Sonuçta aynı kümede bulunan nesnelerin özellikleri birbirine benzer, başka kümedeki nesneler ise farklıdır.



Şekil 3.9. Veri noktalarının bölmesel kümeleme tekniği ile kümelenmesi

Çalışmada oluşturulacak küme sayısı konusunda ön bilgi elde edişmiş ise ya da araştırmacı çalışma için gerekli küme sayısına karar vermiş ise bu durumda hiyerarşik yöntemler yerine hiyerarşik olmayan başka bir deyişle bölmesel kümeleme yöntemlerine başvurulur.

Bölmesel kümeleme yöntemleri, hiyerarşik yöntemlere göre daha büyük veri setlerine uygulanabilirler [30]. Bölmesel yöntemler, hem uygulanabilirliğin kolay, hem de verimli olması nedeniyle iyi sonuç üretirler. Ayrıca teorik geçerliliklerinin daha güçlü olması, bunların tercih edilme sebeplerindendir.

Küme Sayısının Belirlenmesi:

Bölmesel kümeleme yöntemlerinde, kümeleme analizinden sağlıklı bir sonuç elde edilebilmesi için değişkenlerin seçimi ve öncelikle küme sayısının doğru olarak belirlenmesi önemlidir. Çünkü gerçekçi olmayan bir k – küme sayısı kümeleme analizinin başarısını çok olumsuz bir şekilde etkileyebilmektedir. Bu yüzden küme sayısının belirlenmesi konusunda son zamanlarda yoğun çalışmalar yapılmaktadır. Veri setinin optimal küme ya da sınıf sayısının belirlenmesinde kullanılan küme geçerliliği ya da geçerlilik kriteri olarak adlandırılan çeşitli fonksiyonlar da literatürde yer almaktadır. Küçük örneklemelerde küme sayısının belirlenmesi için sık kullanılan bazı eşitlikler şunlardır.

1. ***n*** kümelenecek birey (nesne) sayısını, ***k*** küme sayısını göstermek üzere;

$$k = \sqrt{\frac{n}{2}} \tag{3.4}$$

olarak belirtilir.

2. Mariot tarafından önerilen yöntemde ise;

$$M = k^2 |W| \tag{3.5}$$

Burada en küçük M değerini veren küme sayısı gerçek küme sayısıdır. W ise grup içi kareler toplamı matrisidir.

3. Calinsky ve Horabaz tarafından geliştirilen yöntemde ise;

$$C = \frac{[iz(B)/k - 1]}{[iz(W)/(n = k)]} \tag{3.6}$$

eşitliğini en büyük yapan k değeri küme sayısıdır. Burada B ve W, sırasıyla gruplar arası ve grup içi kareler toplamı matrisleridir [29].

Karesel Hata Algoritmaları:

Bölmesel kümeleme yöntemlerinde kümeleri değerlendirmede en sık kullanılan kriter fonksiyonu, karesel hata (Squared Error) kriteridir. Karesel hata kriteri, yoğun ve ayrı kümelerle verimli çalışmaya uygundur. Bir **X** örnek setinin **Y** kümesi için karesel hata şu şekilde ifade edilir.

$$e^2(X, Y) = \sum_{j=1}^{K} \sum_{i=1}^{nj} \left\| x_i^{(j)} - c_j \right\|^2 \tag{3.7}$$

Burada $x_i^{(j)}$; *j.* kümeye ait olan *i.* örneği temsil eder. C_j ise *j.* kümenin merkezidir (centroid). Burada amaç, tüm örnekler arası Öklit (Euclidean) uzaklıkları ve küme merkezlerinin toplamı olan karesel hatanın minimize edilmesiyle, sabit sayıdaki kümeler için bir bölüm oluşturmaktır.

Genel olarak karesel hata algoritması ile kümeleme yöntemi şu şekilde yapılır;

1. Önceden belirlenmiş küme sayısı (*k*) kadar, küme merkezi olarak örneklerin başlangıç yerleri seçilir.

2. Her bir örnek, onun en yakın küme merkezine atanır ve yeni kümelerin merkezleri hesaplanır. Bu adım küme üyelerinin yerleri sabit kalıncaya kadar tekrarlanır.

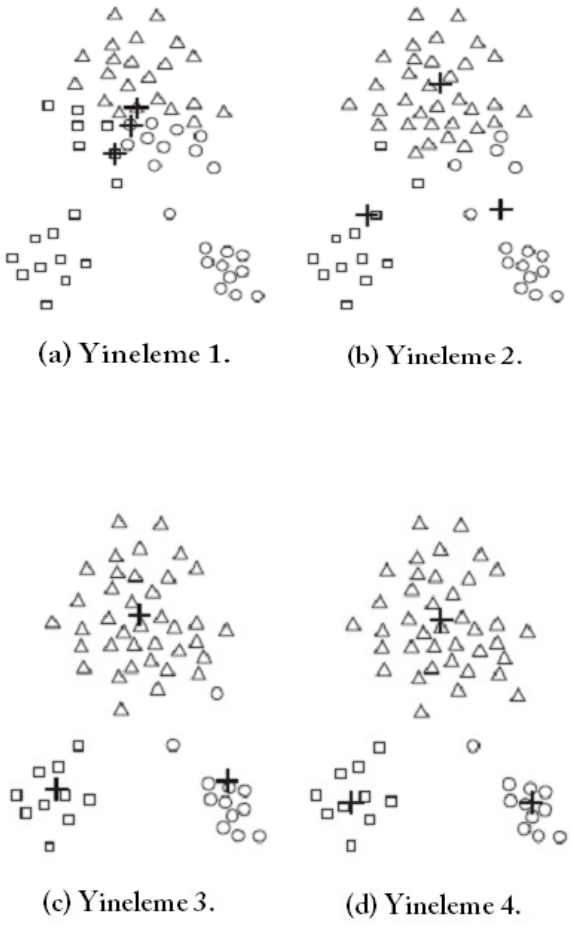
3. Bazı sezgisel (bulgusal) bilgiye dayalı olarak kümeler birleştirilir veya bölünür. İstenirse ikinci adım tekrarlanır.

K-Means, karesel hata kriterini en sık kullanan algoritmadır. K-Means algoritması popülerdir ve uygulaması kolaydır.

K-Means Kümeleme Yöntemi:

K-Means (K-Ortalama) kümeleme tekniği basittir. Öncelikle ***n*** adet nesne içinden rastgele olarak ***k*** tanesi seçilir. Bu seçilen nesnelerin her biri, bir kümenin merkezini veya orta noktasını temsil eder. Daha sonra geriye kalan nesnelerin her biri, kendine en yakın olan küme merkezine göre kümelere atanır. Başka bir ifadeyle bir nesne, hangi kümenin merkezine daha yakın ise o kümeye yerleşir. Ardından yeni oluşan her küme için ortalama (ağırlık merkezi) hesaplanır. Hesaplanan bu değer o kümenin yeni merkezi olur. Merkezlerin yerleri değiştiği için yeni merkezleri esas alarak nesneler yeniden en yakın küme merkezlerine atanırlar. Yukarıdaki işlemler hiçbir kümenin merkezi değişmeyene kadar veya nesnelerin üyelikleri sabit kalana kadar devam eder. Böylece her nesne kendi kümesini bulmuş olur.

Bir nesne grubunun Şekil 3.10'da görüldüğü gibi uzayda konumlanmış olduğu varsayılsın. Kullanıcının bu kümeleri üç kümeye ayırmak istediği farz edilirse k=3 olur. Şekil 3.10 (a)'da ilk önce rastgele üç nesne, üç kümenin merkezi olarak seçilmiş (+ işaretleri) ve diğer nesneler de bu merkezlere olan yakınlıklarına göre üç kümeye ayrılmışlardır. Bu ayrıma göre üç kümenin nesnelerinin yeni ortalaması alınmış ve bu değerler, (b)'de görüldüğü gibi kümelerin yeni merkezleri olmuştur. Bu adımdan bu adımdan sonra aynı işlemler yinelenerek (c) ve (d) adımlarında merkezlerden iki tanesi şeklin altındaki iki küçük nokta grubuna doğru kaydığı gözlenir. Artık herhangi bir değişiklik olmadığı için K-Means algoritması sonlandırılır ve böylece merkez noktaların doğal gruplandırılması belirlenmiş olur [27].



Şekil 3.10. K-Means algoritması kullanılarak üç kümenin bulunması [5].

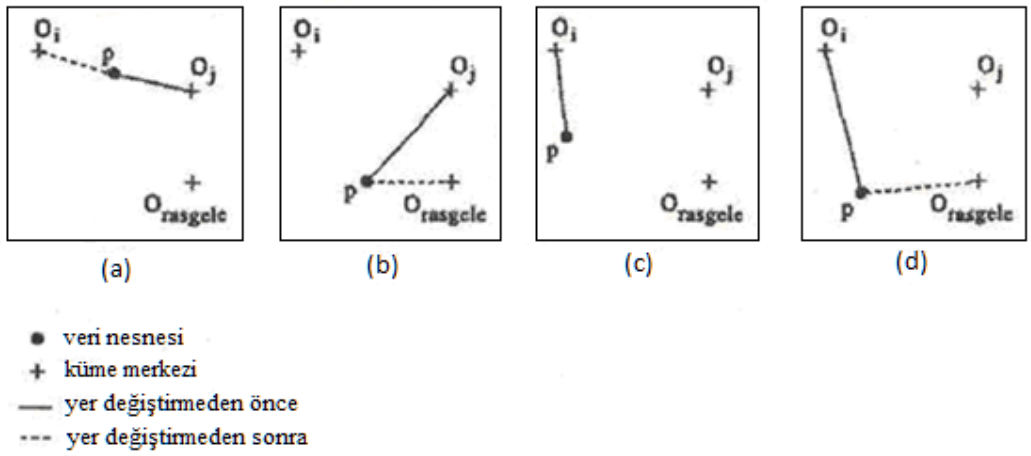
K-Means kümeleme yöntemi, sadece kümenin ortalamasının tanımlanabildiği durumlarda kullanılır. Kullanıcının *k* değerini yani oluşacak küme sayısını belirtme gerekliliği bir dezavantaj olarak görülebilir. Asıl önemli olan dezavantaj ise dışarıda kalanlar (outliers) olarak adlandırılan nesnelere karşı olan duyarlılıktır. Değeri çok büyük olan bir nesne dâhil olacağı kümenin ortalamasını ve merkez noktasını büyük oranda değiştirebilir. Bu değişiklik ise kümenin hassasiyetini bozabilir.

K-Medoids Kümeleme Yöntemi:

K-Means yönteminde kümenin merkezi, gerçek bir noktaya karşılık gelmeyebilir. Bu sorunu gidermek için kümedeki nesnelerin ortalamasını almak yerine, kümede merkeze en yakın olan nesne anlamına gelen *medoid* kullanılabilir. Bu işlem K-medoids yöntemi

olarak gerçekleştirilir. K-medoids yönteminde, küme merkezi gerçek bir nesneye karşılık gelmese bile merkeze en yakın olan nesne, küme merkezi olarak atanır. Bu nesneye *temsilci nesne* de denilebilir.

K-medoids kümeleme tekniğinin temel stratejisi ilk olarak n adet nesnede, merkezi temsil edecek bir medoid olan k adet küme bulmaktır. Geriye kalan nesneler kendilerine en yakın olan nesneyi bulmak için medoid, medoid olmayan her nesne ile yer değiştirir. Bu işlem en verimli medyan bulunana kadar devam eder [31].



Şekil 3.11. K-medoids yöntemi ile kümeleme örneği

Şekil 3.11'da **Oi** ve **Oj,** iki ayrı kümenin medoidlerini, $\mathbf{O_{rasgele}}$ rasgele seçilen ve medoid adayı olan bir nesneyi, **p** ise medoid olmayan bir nesneyi temsil etmektedir. Sekil 3.11 $O_{rasgele}$'nin, su anda medoid olan Oj'nin yerine geçip, yeni medoid olup olamayacağını belirleyen dört durumu göstermektedir.

(a): p nesnesi şu anda Oj medoidine bağlıdır (Oj medoidinin bulunduğu kümededir). Eğer Oj, $O_{rasgele}$ ile yer değiştirir ve p Oi 'ye en yakınsa, p nesnesi Oi 'ye geçer.

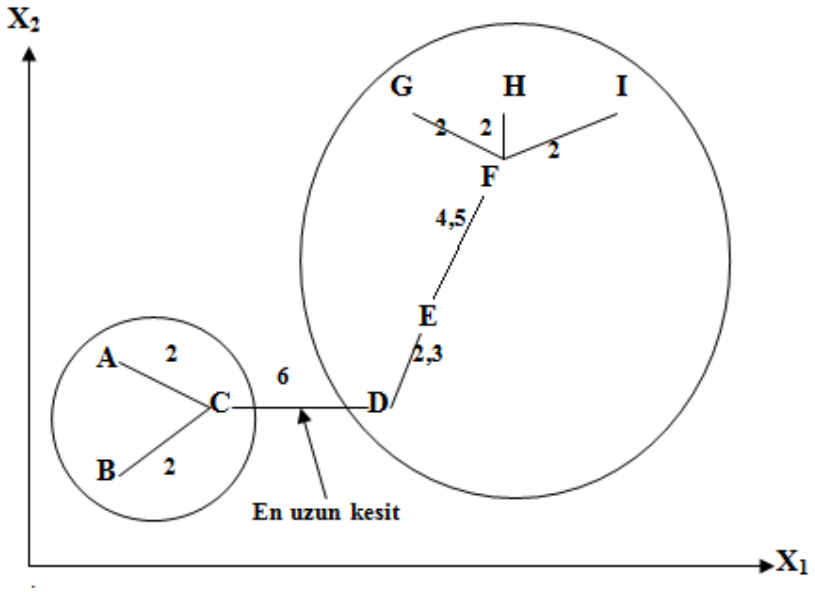
(b): p nesnesi şu anda Oj medoidine bağlıdır. Eğer Oj, $O_{rasgele}$ ile yer değiştirir ve p $O_{rasgele}$ 'ye en yakınsa, p nesnesi $O_{rasgele}$ 'ye geçer.

(c): p nesnesi şu anda Oi medoidine bağlıdır. Eğer Oj, $O_{rasgele}$ ile yer değiştirir ve p hala $O_{rasgele}$ 'ye en yakınsa, p nesnesi yine Oi 'ye bağlı kalır.

(d): p nesnesi şu anda Oi medoidine bağlıdır. Eğer Oj, $O_{rasgele}$ ile yer değiştirir ve p $O_{rasgele}$ 'ye en yakınsa, p nesnesi $O_{rasgele}$ 'ye geçer.

Graf – Teorik Kümeleme Yöntemi:

En iyi bilinen graf – teorik kümeleme algoritması, minimum örten ağaç (Minimal Spanning Tree – MST) yapısının bulunması ve küme oluşturmak için en uzun kesitin ağaçtan çıkarılması esasına dayanmaktadır [28]. Graf – teorik kümelere en iyi örnek komşuluk tabanlı kümelerdir. Bu kümelerde iki nesne, ancak arasında belirli bir mesafe varsa (yakınsa) birbirine bağlıdırlar. Bu durumda böyle kümelerde, bir nesne daima küme içindeki diğer bir nesneye, başka kümedeki herhangi bir nesneden daha yakındır [10].



Şekil 3.12. Kümeleri bulmak için minimum örten ağacın kullanımı

Şekil 3.12'de dokuz tane iki boyutlu noktadan elde edilmiş bir minimum örten ağaç örneği görülmektedir. Burada A, B, ….., I düğümleri veri nesnelerini temsil eder. İki düğüm arasındaki dallar (kesit) ise nesneler arasındaki bağlantıyı temsil eder. Şekilde görüldüğü gibi 6 birim ile en uzun kesit olan C – D arasındaki bağın çıkarılması ile {A, B, C} ve {D, E, F, G, H, I} olan iki farklı küme elde edilir. Burada büyük çaplı olan ikinci küme 4,5 birim uzunluğa sahip olan E – F arasındaki kesitin çıkarılması ile tekrar iki ayrı kümeye bölünebilir. Bu yöntemde kesit uzunlukları için bir eşik değeri belirlenir. Eşik değerinden büyük olan kesitler kümeden çıkarılarak yeni kümeler oluşturulur.

Bu kümeleme yöntemleri, küme şekillerinin düzensiz veya birbirine geçmiş durumda iken yararlıdır. Ancak gürültüye karşı duyarlıdırlar. İki küresel küme arasındaki küçük bir gürültü zinciri iki kümenin birleşmesine neden olabilir.

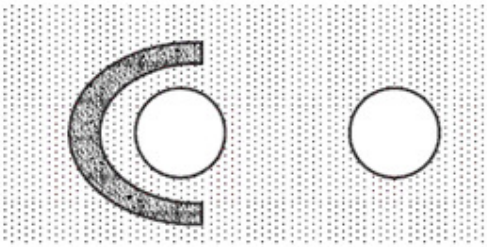
Bulanık Kümeleme Yöntemi:

Klasik kümeleme analizinde farklı kümelerin sınırları kesindir, yani bir örnek sadece bir kümeye aittir. Ancak pratikte bazı durumlarda verilerin ait olabileceği kümenin sınırları kesin olarak tanımlanamayabilir. Bu durumda, bir gözlem bir ya da daha fazla kümeye farklı üyelik dereceleri ile ait olabilirler [32]. Bulanık kümeler kümedeki birimin üyelik derecesi olarak tanımlanan 0 ile 1 arasındaki her bir örneği belirleyen fonksiyonlardır. Birbirine çok benzeyen örnekler aynı kümede yüksek üyelik ilişkisine göre yer alırlar. Bundan dolayı Bulanık Kümeleme Yöntemi, birimlerin kümeye ya da kümelere ait olabilme katsayılarını hesaplar. Üyelik katsayılarının toplamı daima 1'e eşittir. Böylelikle birim en yüksek üyelik katsayısına sahip olduğu kümeye atanır. Üyelik fonksiyonları, kümedeki elemanlar sürekli veya süreksiz olsun bir bulanık kümedeki bulanıklığı karakterize eden fonksiyonlardır. Klasik kümeleme yöntemlerinde ise her bir birim sıfır olmayan sadece bir üyelik katsayısına sahiptir ve bu değer daima 1 dir. Dolayısıyla klasik kesin kümeleme yöntemleri, bulanık çözümlemenin sınırlı bir durumudur.

***Yoğunluk-Tabanlı Kümeleme Yöntemi*:**

Yoğunluk-tabanlı (Density-based) kümelemede, alanlar veri yoğunluğunun fazla ve az olmasına göre belirlenir. Yoğunluk-tabanlı kümeleme yaklaşımı, düşük yoğunluklu bölgeler ile yüksek yoğunluklu bölgeler olarak ayrılan düzensiz şekilli kümeleri bulma yeteneğine sahip bir yöntemdir. Kümelerin içinde yer alan ortalamaları bozan çok büyük veya çok küçük değerlerden etkilenmez.

Şekil 3.13'de düşük yoğunluklu bir bölge tarafından sarılmış (gürültü), yüksek yoğunluklu bölgeler görülmektedir [27]. Yoğunluk-tabanlı kümeleme yaklaşımı ile kümeler, düşük yoğunluklu gürültülerden etkilenmeden bulunabilmektedir.



Şekil 3.13. Düşük yoğunluklu alanlarla ayrılmış yüksek yoğunluklu alanlar

Yoğunluk-tabanlı kümeleme yaklaşımı olarak en yaygın kullanılan DBSCAN yöntemidir.

DBSCAN:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise – Gürültülü uygulamaların yoğunluk tabanlı uzaysal kümelenmesi) yoğunluk-tabanlı kümeleme yapan basit ve etkin bir algoritmadır. Bu kısımda DBSCAN'in dayandığı merkez-tabanlı yaklaşımı tartışacağız.

Merkez-tabanlı yaklaşımda, veri kümesindeki bir nokta için yoğunluk o noktanın belirli bir mesafesinde bulunan (Eps yarıçapındaki) nokta sayısına bakılarak tahmin edilir. Bu sayı noktanın kendisini de içerir. Bu teknik 3.14. (a)'da gösterilmiştir. A noktasından Eps uzaklığındaki (yarıçapında) alanda bulunan noktaların sayısı ve noktanın kendisi toplamı 7'dir.

Bu yöntemin uygulanması kolaydır ancak bir noktanın yoğunluğu özel bir değer olan Eps'e bağlıdır. Bu yüzden eğer yarıçap büyük tutulursa her bir noktanın yoğunluğu veri kümesindeki nokta sayısına eşit olacaktır. Aynı şekilde yarıçap çok küçük tutulduğunda her bir noktanın yoğunluğu 1 (kendisi) olacaktır

Merkez-Tabanlı Yoğunluğa göre Noktaların Sınıflandırılması:

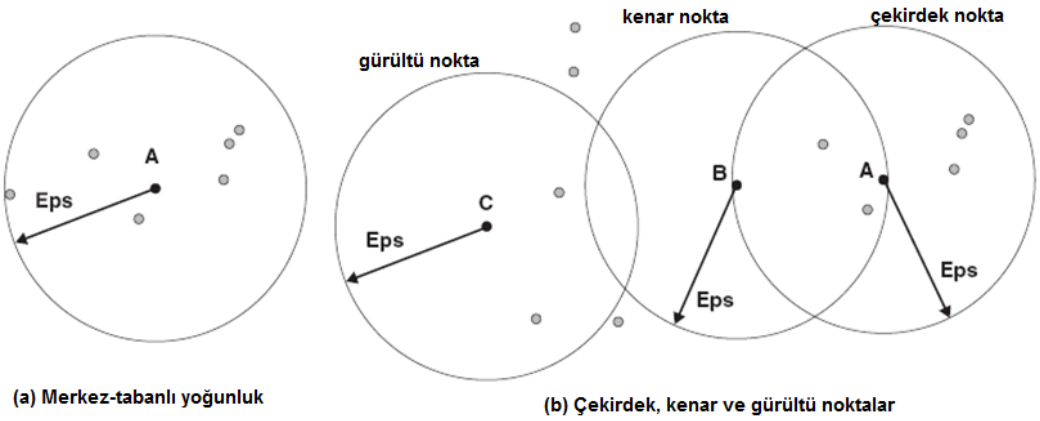
Merkez-tabanlı yaklaşım bizim noktaları üç farklı şekilde sınıflandırmamızı sağlar. Yoğunluğa göre oluşturulan alanın içersinde (***çekirdek nokta***), alanın kenarında (***kenar nokta***), alandan uzakta (***gürültü nokta***). Şekil 3.14. (b)'de çekirdek, kenar ve gürültü

noktaları iki boyutlu nokta kümesi kullanılarak gösterilmiştir. Aşağıda daha net bir açıklama verilmiştir.

Çekirdek Noktalar: Yoğunluk-tabanlı kümenin içersinde yer alan noktalardır. Bir noktanın çekirdek nokta olması için uzaklık fonksiyonu ve Eps ile hesaplanan komşu noktalarının kullanıcı tarafından belirlenen bir parametre olan MinPts'yi aşmaması gerekir. 3.14. (b)'de A noktası kullanıcının belirlediği Eps için MinPts < 7 ise çekirdek noktadır.

Kenar Noktalar: Kenar noktalar çekirdek nokta değildir ancak çekirdek noktaların komşuluk alanları içersinde yer alırlar. 3.14. (b)'de B bir kenar noktadır.Bir kenar nokta bazı çekirdek noktaların komşuluk alanları içersinde yer alır.

Gürültü Noktalar: Çekirdek nokta kümesine de kenar nokta kümesine de girmeyen noktalar gürültü noktalardır. 3.14. (b)'de C gürültü noktadır.



Şekil 3.14. Yoğunluk-tabanlı kümeleme için nokta türleri

***DBSCAN Algoritması*:**

Yukarıdaki tanımlara dayanarak DBSCAN algoritması şöyle açıklanabilir. Birbirine yeteri kadar yakınlıkta (en fazla Eps) olan iki çekirdek nokta aynı kümeye konur. Aynı şekilde çekirdek noktaya yeteri kadar yakınlıkta olan bir kenar nokta çekirdek noktayla aynı kümeye yerleştirilir.(Bir kenar noktanın başka bir kümedeki çekirdek noktaya olan uzaklığı da dikkate alınmalıdır.) Gürültü noktalar çıkartılır. Aşağıdaki algoritma orjinal DBSCAN ile aynı kümeleri bulacaktır ancak basit olması için optimize edilmiştir.

DBSCAN algoritması:

1-Tüm noktaları çekirdek, kenar ya da gürültü noktalar olarak işaretle.

2-Gürültü noktaları çıkar.

3-Birbiriyle Eps çapı içersindeki tüm çekirdek noktalar arasına bir kenar çiz.

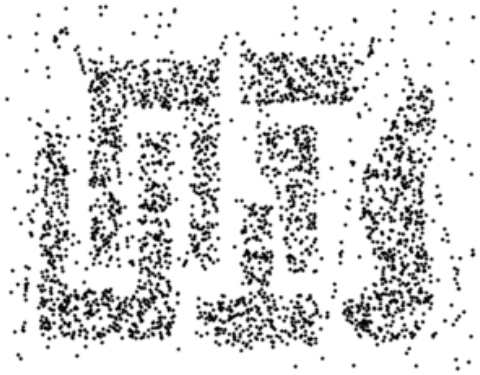
4-Her bağlı çekirdek nokta grubu için bir küme oluştur.

5-Çekirdek noktalara göre her bir kenar noktayı bir kümeye dahil et.

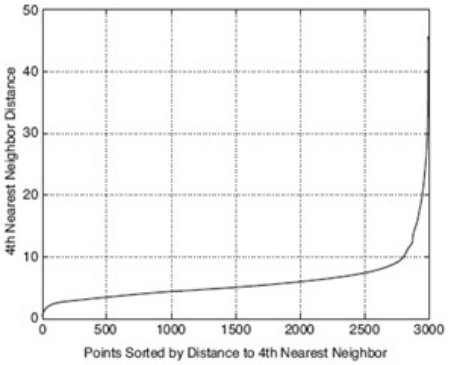
DBSCAN Parametrelerinin Seçimi:

Peki ***Eps*** ve ***MinPts*** parametreleri nasıl seçilecektir? Temel yaklaşım bir noktaya k. yakın komşunun uzaklık davranışıdır (k-dist). Bir öbekte yer alan noktalar için eğer k küme boyutundan büyük değilse, k-dist değeri küçük olmalıdır. Kümenin yoğunluğundan veya noktaların rastgele dağıtılmasından kaynaklanan farklılıklar oluşabilir. Ancak ortalamaya bakıldığında eğer küme yoğunlukları birbirinden çok farklı değilse farklılık oranı da çok büyük olmayacaktır. Bir kümeye ait olmayan, gürültü noktalara bakıldığında ise k-dist değerinin nispeten daha büyük olduğu görülecektir. Bu yüzden herhangi bir k değeri için tüm noktaların k-dist değerlerini hesaplayıp artan sıraya sokarsak, Eps'ye karşı gelen k-dist değerinde kesin bir değişiklik gözlenmelidir. Eğer bu uzaklık Eps olarak k ise MinPts olarak seçilirse, k-dist değeri Eps'den küçük olan noktalar çekirdek noktalar, diğerleri de kenar ya da gürültü noktalar olarak işaretlenir.

Şekil 3.15'de örnek bir veri seti görüyoruz. Verinin k-dist çizgesi de şekil 3.16'da verilmiştir. Eps değeri k değerine dayanarak hesaplansa da k'nın değişiminden çarpıcı bir şekilde etkilenmez. k değerinin çok küçük ya da çok büyük seçildiği durumlarda yanlış sonuçlar ortaya çıkabilir.k çok küçük seçildiğinde gürültü noktalar çekirdek nokta olarak etiketlenebilir aynı şekilde çok büyük seçilmesi durumunda çekirdek noktalar bile gürültü noktalar olarak işaretlenebilir.Orjinal DBSCAN algoritmasında k=4 kullanılır.Bu değer iki boyutlu veri kümeleri için kabul edilebilir bir değerdir.



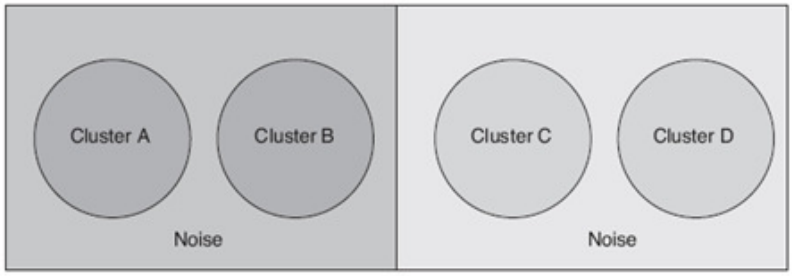
Şekil 3.15. Örnek veri



Şekil 3.16. Örnek verinin k-dist çizgesi

Değişen Yoğunluklu Kümeler:

Eğer kümelerin yoğunlukları büyük oranda farklıysa, DBSCAN için problem oluşturabilir. 3.17'deki gürültü içeren 4 kümeye bakalım. Kümelerin yoğunluğu ve gürültü alanları koyuluklarıyla ifade ediliyor. A ve B gibi yoğun iki kümenin gürültüsü C ve D kümelerininkiyle aynı yoğunluğa sahiptir. Eğer Eps eşiği yeterince düşükse, DBSCAN C ve D yi küme olarak bulabilir. A, B ve etrafındaki noktaları da tek bir küme olarak bulacaktır. Eps eşiği yeterince büyükse, A ve B yi ayrı küme olarak bulur ve etrafındaki noktaları gürültü olarak işaretlenir. Ancak C, D ve etrafındaki noktalar da gürültü olarak işaretlenecektir.



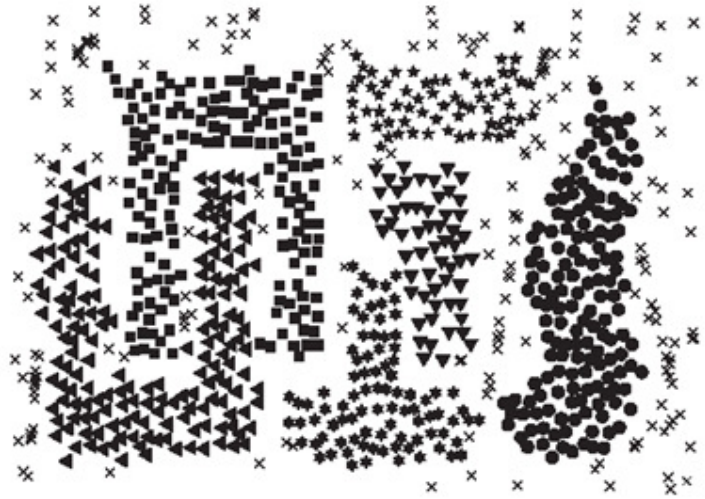
Şekil 3.17. Gürültü içinde bulunan dört küme

Örnek:

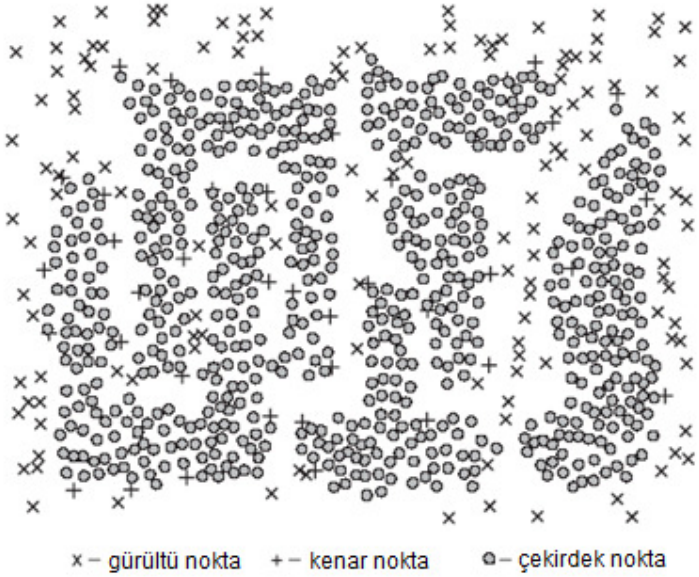
DBSCAN'in kullanımını anlayabilmek için iki boyutlu karmaşık bir veri setinde, DBSCAN'in kümeleri nasıl bulduğunu inceleyeceğiz. Veri seti 3000 adet iki boyutlu noktadan oluşuyor. Bu veri için Eps eşiği her bir noktanın 4. en yakın komşuluklarının sıralanıp işaretlenerek kesin artışın başladığı değerin çıkarılmasıyla bulunacaktır.(Şekil 3.16) Eğrinin köşesi olan değeri Eps olarak seçtik (Eps=10). MinPts=4, Eps=10 iken DBSCAN ile bulunan kümeler Şekil 3.18 (a)'da gösterilmiştir. Çekirdek, kenar ve gürültü noktalar Şekil 3.18 (b)'de gösterilmiştir.

Güçlü ve Zayıf Noktalar:

DBSCAN kümenin yoğunluk-tabanlı tanımını kullandığından gürültüye dayanıklıdır. Farklı şekilde ve büyüklükte kümeleri ele alabilir. Bu yüzden, Şekil 3.15'de gösterildiği gibi DBSCAN algoritması K-Means ile bulunamayan birçok kümeyi bulabilir. Önceden bahsedildiği gibi kümeler arasındaki yoğunluk farkı büyükse DBSCAN algoritması problem yaşayacaktır. Ayrıca yüksek boyutta verilerde de problem yaşayacaktır çünkü böyle veriler için yoğunluk belirlemek oldukça zordur. Sonuç olarak, DBSCAN algoritmasında yakın komşulukların aranması tüm çiftlerin yakınlıklarının hesaplanmasını gerektiriyorsa (genelde yüksek boyutlu verilerde oluşan bir durumdur) maliyeti yüksek olacaktır.



(a) DBSCAN ile kümelerin bulunması



(b) Çekirdek, kenar ve gürültü noktalar

Şekil 3.18. İki boyutlu 3000 noktanın DBSCAN ile kümelenmesi

Genetik Algoritmalar ile Kümeleme (Evrimsel Kümeleme):

Genetik Algoritmalar, evrimsel hesaplama yöntemlerine dayanan bir kümeleme yöntemidir. Darwin'in evrim teorisinden ilham alınarak geliştirilmiştir. Genetik algoritmalar, problemin çözümü için doğa ananın kullandığı benzer süreçlerin taklit

edilmesiyle elde edilen bir yöntemdir. Bu algoritmalar evrim teorisinde olan seleksiyon (seçim), kombinasyon, rekombinasyon, mutasyon gibi bazı işlemleri ve adımları kullanır [33]. Bu algoritmaların birçoğu çok çeşitli alanlardaki kümeleme problemlerini çözmek için geliştirilmiştir. Özellikle bunların bazıları biyoinformatik araştırmalarda *gen ifade mikrodizi verisindeki* küme yapılarını ortaya çıkarmak için kullanılmaktadır [34].

Genetik Algoritmalar, doğal biyolojik evrime mecazi olarak benzeyen varsayımsal bir genel arama yöntemidir. Daha iyi çözümleri elde etmek için; bir popülasyon üzerindeki potansiyel çözümlerin en uygununun hayatta kalması esasına dayanarak çalışır. En uygun çözüm, yapılan yinelemelerin sonucunda elenmeden hayatta kalmayı başarabilen çözümdür [35].

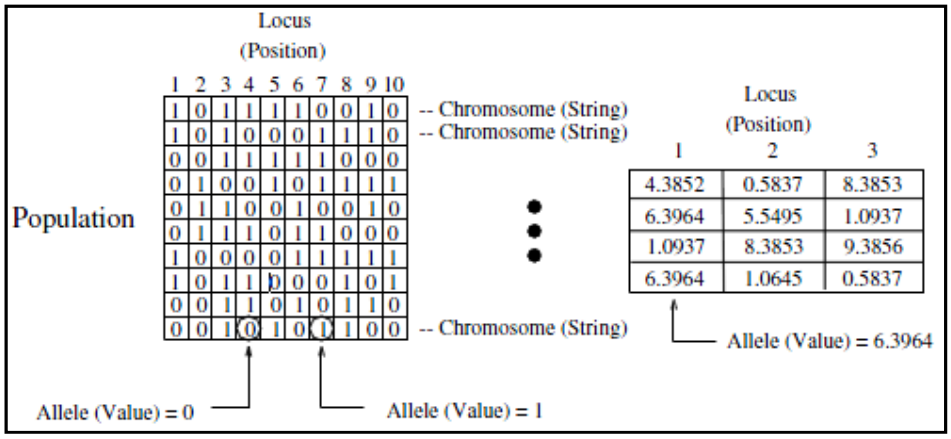
Genetik Algoritmalar, problemin ele alındığı ortamda bulunan uygun ve güçlü çözümlerin yaşatılarak, iyiler arasından daha da iyileştirilmiş çözümler üretmeyi amaçlayan bir yöntemdir. Ancak her problem için en uygun sonucu bulduğunu garanti edemez.

Genetik Algoritmaların Çalışması:

Algoritma, *popülasyon* adı verilen ve *kromozomlarla* temsil edilen bir çözüm kümesi ile başlamaktadır. Bir popülasyondaki kromozomlar, yeni popülasyonların oluşturulmasında kullanılmaktadır. Önceden belirlenen bir *uygunluk (fitness) fonksiyonu*, kromozomun bir sonraki nesilde hayatta kalma olasılığını belirler. Mevcut kromozomlar kullanılarak onlardan yavru kromozomlar elde edilir. Buradaki amaç mevcut popülasyondan daha iyi bir popülasyon elde etmektir. Daha uygun olan kromozomların yinelemeler sonucunda hayatta kalma ihtimali daha yüksektir. Bu işlemler, en uygun popülasyonun (çözüm kümesinin) elde edilmesine kadar devam etmektedir [37].

Başka bir deyişle bir birey, bir problem için kodlanmış bir çözümdür. Tipik olarak bir birey, biyolojik genotipe karşılık gelen bir string (1001011 gibi) olarak temsil edilir. Bu genotip, bir fenotipe (kalıtımla oluşan dış görünüş) kodlandığında bir bireysel organizmayı tanımlar. Bir genotip bir veya daha fazla kromozomdan oluşur. Her kromozom, genetik alfabeden oluşan belirli değere sahip olan ayrı genlerden (alel) meydana gelir. Bir lokus, kromozom içindeki bir genin pozisyonunu tanımlar. Böylece her birey, ele alınan fonksiyon için girdi olarak kullanılan parametre kümesine kodlanır. Sonuç olarak, verilen

bir kromozom kümesi popülasyon olarak adlandırılır. Bu kavramlar hem ikilik hem de gerçek değerli kromozomlar için Şekil 3.19'da gösterilmektedir [38].



Şekil 3.19. Genel bir Evrimsel Algoritma Veri Yapısı

Yukarıda belirtildiği gibi Genetik Algoritmada popülasyon, 1001011 gibi ikilik tamsayılardan oluşan bireylerin bir kümesidir. Her birey bir canlının kromozomunu temsil eder. En uygun bireyin nasıl belirleneceğini belirleyen bir uygunluk fonksiyonu ve bireyleri yeniden üretmek için popülasyondan seçen diğer bir fonksiyon vardır. Seçilen iki kromozom çaprazlanır ve tekrar bölünür. Daha sonra iki yeni birey mutasyona uğrar. Süreç belli bir sayıda tekrarlanır [36].

Temel Genetik Algoritma yapısı şu şekildedir;

1. **Başlangıç:** n kromozomdan oluşan rastgele bir popülasyon üretilir. (problemin olası tüm çözümleri için)
2. **Uygunluk (fitness):** Popülasyondaki her bir x kromozomu için f(x) uygunluk fonksiyonu hesaplanır.
3. **Yeni Popülasyon:** Yeni popülasyon tamamlanıncaya kadar aşağıdaki adımlar tekrarlanarak yeni popülasyon oluşturulur.
    a. **Seçim (Seleksiyon):** Popülasyondan uygunluklarına göre iki ana kromozom seçilir. Daha iyi uygunluk daha fazla seçilme şansı demektir

b. **Çaprazlama (Crossover):** Çaprazlama olasılığı ile ana kromozomlar, yeni yavru oluşturmak için birbirleriyle çaprazlanır. Eğer çaprazlama yapılmaz ise yavrular, ana bireylerin aynısı olur.
c. **Mutasyon:** Bir mutasyon olasılığı ile her lokustaki (kromozomdaki pozisyon) yeni yavrular mutasyona uğrar.
d. **Kabul:** Yeni yavru, yeni popülasyona yerleştirilir.

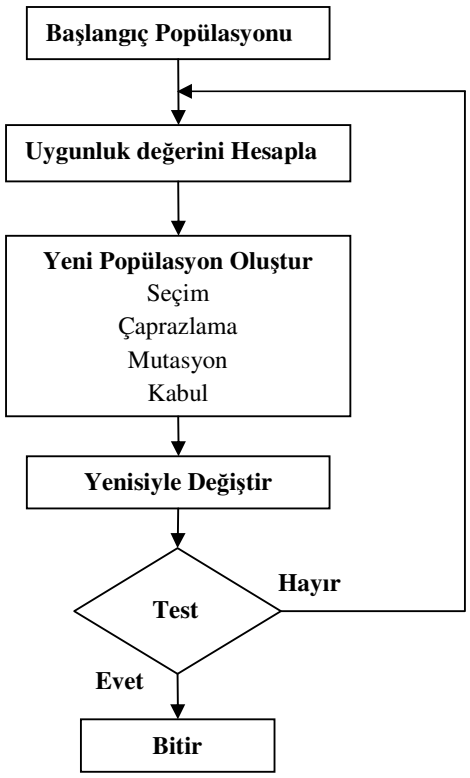
4. **Yenisiyle Değiştirme (Replace):** Algoritmanın tekrar çalıştırılmasında üretilen yeni popülasyon kullanılır.
5. **Deneme (Test):** Bitiş durumu sağlanırsa, durup popülasyondaki en iyi çözüm döndürülür.
6. **Döngü (Loop):** 2. Adıma gidilir [37].

Şekil 3.20'de temel bir genetik algoritmanın akış şeması görülmektedir.

Burada da görüldüğü gibi Temel Genetik Algoritma yapısı çok geneldir. Çeşitli problemlerde farklı şekilde uygulanabilen birçok türleri vardır.

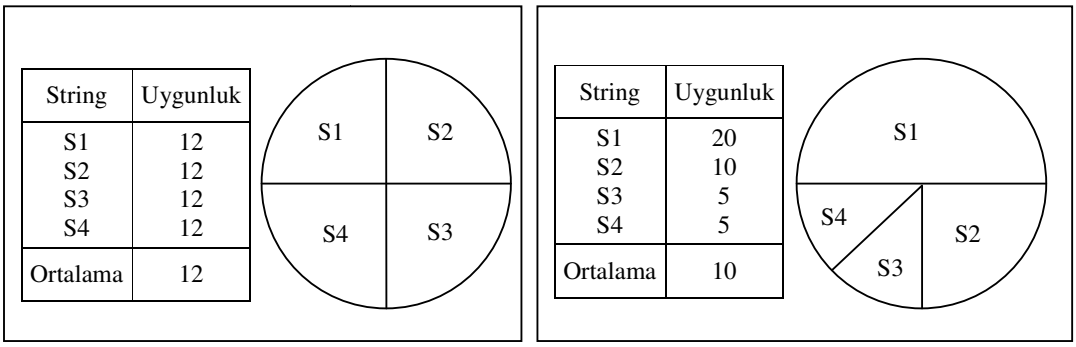
Sorulması gereken ilk soru kromozomların oluşturulması için hangi tip kodlamanın seçileceğidir. Bu, Genetik Algoritmanın iki temel operatörü olan mutasyon ve çaprazlama ile bağlantılıdır.

Bir sonraki soru, çaprazlama için ana bireylerin nasıl seçileceğidir. Bu pek çok farklı yöntemle yapılabilir. Ancak ana düşünce, daha iyi ana bireylerin (daha iyi ana bireyler daha iyi yavrular üreteceği düşüncesinden) seçilmesidir. Sadece yeni yavrularla yeni popülasyonu oluşturmak, en son popülasyondaki en iyi kromozomun kaybedilmesine neden olabileceği düşünülebilir. Bu yüzden *seçkinlik (elitizm)* adı verilen yöntem kullanılır. Bu yöntemde en iyi çözüme sahip kromozom veya kromozomlar yeni popülasyona değişikliğe uğramadan kopyalanırlar. Böylece en iyi bulunan çözüm, çalışmanın sonuna kadar hayatta kalabilir.



Şekil 3.20. Temel Genetik Algoritma Akış Şeması

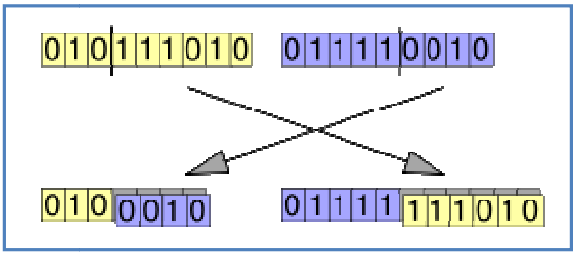
Popülasyondan ana bireylerin seçimi için *rulet-çemberi, sıralama, turnuva* gibi bazı seçim teknikleri mevcuttur. Ortalamanın üzerindeki bireyler, sonraki neslin (yeni popülasyon) üyesi olmak için seçilirler. Seçim parametreleri, daha fazla uygunluk değerine sahip ve neslin başarısında daha fazla yavru verme olasılığı olan stringleri verir. Şekil 3.21 dört tane bireyin iki farklı popülasyonda rulet-çemberi seçim işlemini gösterir. Burada her birey, uygunluk değerine orantılı olarak çemberin üzerinde yer kaplar [38].



Şekil 3.21. Rulet-Çemberi Seçimi

Literatürde farklı türde çaprazlama yöntemleri vardır. Çaprazlama, genellikle ana bireylerdeki seçilen genler üzerinde işlem yapar ve yeni yavru bireyler oluşturur. Bunu gerçekleştirmek için en basit şekilde rastgele bir veya birden fazla kesme noktası seçilmesi ve bu noktadan önceki her şeyi ilk ana bireyden, sonraki her şeyi ikinci ana bireyden alıp birleştirilerek yavru bireylerin oluşturulmasıdır.

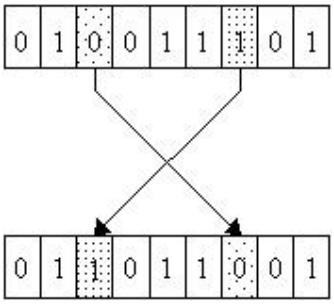
Şekil 3.22'de örnek bir çaprazlama gösterilmektedir [39].



Şekil 3.22. Çaprazlama örneği

Çaprazlama işlemi yapıldıktan sonra mutasyon işlemine geçilir. Mutasyon işleminin amacı, popülasyondaki bütün çözümlerin bir yerel optimum değerine düşmesinin önüne geçmektir. Mutasyon işleminde, çaprazlama sonucu elde edilen yavru bireyi rastgele olarak değiştirmektir. İkilik kodlamada rastgele seçilmiş bir ya da birkaç bit değerinden 1 değerine sahip biti 0'a, 0 değerine sahip biti 1'e değiştirmek bir mutasyondur.

Şekil 3.23'te bireyde iki bitin değiştirildiği bir mutasyon örneği gösterilmektedir [39].



Şekil 3.23. Mutasyon örneği

4. ÇOK-AMAÇLI GENETİK ALGORİTMA KULLANARAK GELİŞTİRİLEN KÜMELEME YÖNTEMİ

4.1. Çok – Amaçlı Optimizasyon

Çok amaçlı optimizasyon tek amaçlı optimizasyondan farklı olarak, aynı anda birbirleriyle çelişen amaçların (performans ve maliyet gibi) olduğu problemlerle uğraşır. Örneğin, karmaşık bir donanım tasarımı düşünüldüğünde en yüksek performans hedeflenirken maliyetlerin de minimize edilmesi gerekmektedir. Yukarıda bahsedildiği gibi birden fazla amaç olduğunda bunlardan bazıları kısıt olarak tanımlanabilir. Örneğin, bir sistem yüksek performans ve düşük maliyet ile optimize edilirken, sistemin belirli boyutları geçmemesi ayı bir optimizasyon kriteri olarak tanımlanabilir. Böylece, bir çok amaçlı optimizasyon problemi aşağıdaki gibi formüle edilebilir [40] ve bir amaçlı optimizasyon problemi genel olarak şunları içerir:

- bir parametre kümesi (karar değişkenleri), ***a***
- amaç fonksiyonları kümesi, ***b***
- kısıtlar kümesi, ***c***
- ve amaç fonksiyonları ve kısıtlar karar değişkenlerinin fonksiyonlarıdır.

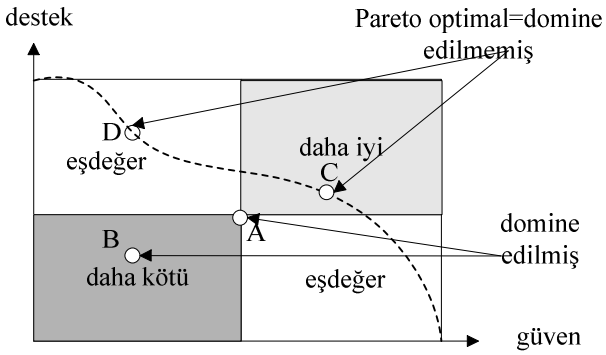
Matematiksel olarak bir optimizasyon problemi aşağıdaki şekilde gösterilebilir:

$$\begin{aligned} &\text{min/max} && y = f(x) = (f_1(x), f_2(x),..., f_b(x)) \\ &\text{sınırlamalar} && e(x) = (e_1(x), e_2(x),..., e_c(x)) \leq 0 \\ &\text{şartlar} && x = (x_1, x_2,..., x_a) \in X \\ & && y = (y_1, y_2,..., y_b) \in Y \end{aligned} \qquad \textbf{(4.1)}$$

Burada x karar vektörü, y amaç vektörü, X karar uzayı, Y amaç uzayı olarak isimlendirilir ve $e(x) \leq 0$ kısıtları olası çözümleri belirler.

Yukarıdaki tanımlamalar dikkate alınarak, amaçlarımız performans (f1) ve maliyet fonksiyonunun zıttı olan ucuzluk (f2) fonksiyonlarının sınırlayıcı şartları (e1) altında maksimize edilmesidir.

O zaman, optimum dizayn, maksimum performansı, minimum maliyet ile sağlar ve kısıtları aşmaz. Eğer böyle bir çözüm olsaydı, yani her bir amaç için optimum olan çözüm diğeri için de optimum olsaydı aslında sadece tek amaçlı optimizasyon problemi çözmemiz gerekirdi. Fakat çok-amaçlı optimizasyonu zorlaştıran genelde bu amaç fonksiyonlarının optimum çözümlerinin birbirinden farklı olmasıdır.



Şekil 4.1. Pareto'nun optimalite kavramı

Şekil 4.1'de amaç fonksiyonları olarak birliktelik kurallarının destek ve güven değerleri gösterilmektedir. Şekil'de gösterildiği gibi bir çözümde yer alan karar vektörü amaç değerleri açısından bir diğerinden daha iyi, daha kötü, eşit veya farksız olabilir. Daha iyi olması demek çözümün hiçbir amaç değerinin diğer çözümlerin amaç değerlerlerinden düşük olmaması ve en az bir amaç değerinde daha yüksek olması demektir. Örneğin şekilde B ile temsil edilen çözüm A çözümünden daha kötü ve C çözümü ise A çözümünden daha iyidir. Fakat C ile D çözümleri biri diğerinden daha iyi diyemeyiz. Bunun sebebi amaç değerlerinden birinin bir çözümde, diğerinin ise diğer çözümde yüksek olmasıdır.

Bu kavramı kullanarak optimum çözümü arama uzayında hiçbir diğer çözüm tarafından domine edilemeyen çözüm olarak tanımlayabiliriz. Böyle bir çözümü Pareto'nun optimum çözümü olarak tanımlayabiliriz. Şekilde noktalı çizgiyle gösterilen çözümler ise Pareto'nun optimum çözüm kümesidir.

Böyle bir optimizasyon probleminde amaçlar birbiriyle çakışır ve aynı anda optimize edilemezler. Bunun yerine tatmin edici bir tercih çözümü bulunmalıdır. Bu nedenle uygun tercih seçimi için bir karar verme süreci gerekir.

4.2. Çok-Amaçlı Genetik Algoritmalar

Gerçek dünya problemleri karmaşıktır ve iyi bir çözüm için birden fazla amacın sağlanması gerekmektedir. Birçok proje tek bir fonksiyona dayalı bir yaklaşımla çözüm getirmektedir. Bu basit yaklaşım pek çok durumda çok etkin değildir. İlk olarak amaçlar sık sık birbirleriyle çatışabilmektedir. İkinci olarak amaçlar sık sık aday çözüm kalitesine uygun olmayabilir ve farklılıklar gösterebilir. Çok amaçlı GA'larda bu iki durum da minimize edilmektedir. Genetik algoritmanın doğal ve evrimsel yapısı çok amaçlı yapı için de uygundur.

Pareto en uygun çözümü, çok amaçlı GA'nın doğasına çok uygundur [2]. Goldberg'in [3] seçme algoritması ise [4, 5]'de önerilen çok amaçlı evrimsel algoritmayı kullanır.

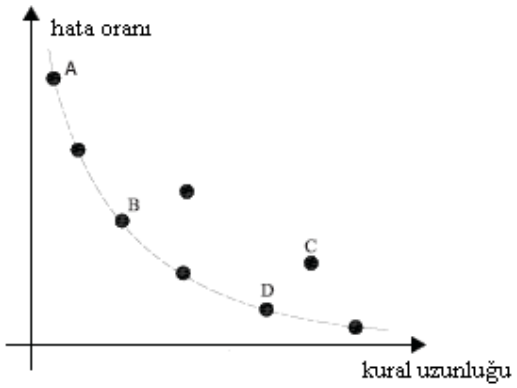
Çok amaçlı genetik algoritmalarda birden fazla amaca aynı anda bakılır. Yani tek bir en uygun çözüm yoktur. Amaçlar arasında bir tercih yapılarak bir çözüm kümesi seçilebilir [5] . Bu yöntemle kullanıcı spesifik problem için olası çözümlerden birini seçebilir. Böylece kullanıcı yüksek kaliteli çözümleri inceleyerek amaçlar arasında bir tercih yaparak bir çözüm kümesi seçme fırsatı bulacaktır. Bu yöntem kullanıcıyı tek bir çözüm kümesine zorlamaktan daha iyi bir yöntemdir.

Çok amaçlı GA'ların tercih edilmesinin nedenleri şunlardır:

(i) Genetik algoritmalar arama uzayının geniş olmasından dolayı güçlü bir algoritmadır.

(ii) Genetik algoritmalar global bir arama yöntemidir ve greedy arama metotlarından daha kolay bir şekilde nesneler ile iletişime girmektedir.

(iii) Genetik algoritmalar, çok amaçlı problemleri çözmek için bir aday çözüm kümesiyle çalışır [5].



Şekil 4.2. İki amaçlı problem örneği

Şekil 4.2'de A çözümü düşük kural uzunluğuna sahip ama hata oranı yüksek, D çözümü ise düşük hata oranına ancak yüksek kural uzunluğuna sahiptir. Her iki amaç önemli olduğundan; A çözümü D'den daha iyidir ya da tam tersi söylenemez. Diğer taraftan C çözümü D'den daha kötüdür denebilir.

4.3. Önerilen Kümeleme Yöntemi

Önerilen yöntemde her bir birey N uzunluklu bir kromozomdan ibarettir. Kromozomdaki her bir gen $\{1,2,\cdots,k\}$ kümesinden bir değer alır. Burada k toplam küme sayısıdır. Böylece her bir gen değeri ilgili örüntünün ait olacağı kümeyi gösterir. Her bir kromozom popülasyonda bir çözüm kümesini verir.

Başlangıçta mevcut jenerasyonun bütün üyelerine sıfır değeri atanır. Her bir kromozom 1 ile kullanıcı tarafından verilen maksimum küme sayısı aralığında değerler ile doldurulur. Daha sonra, önceden belirlenen kromozom sayısına göre başlangıç popülasyonu oluşturulur. Başlangıçta örnekler her bir kümeye rastgele atanır. Bu yapılırken bütün kümelerin mutlaka örnekler içermesine dikkat edilir.

4.3.1. Yöntem için Kullanılan Çok – Amaçlar

Bu tezde bir bireyin uygunluğu birbiriyle çelişen 2 farklı amaç ile bulunur. Bunlar:

Küme değişimi içindeki toplam hata: N örüntü noktasını K küme içine parçalamak için kullanılan amaçlardan biri küme değişimi içindeki toplam hatayı minimize etmektir. Bu değer aşağıdaki formül ile bulunur.

$$Toplam\ hata = \sum_{n=1}^{N}\sum_{d=1}^{D} X_{nd}^2 - \sum_{k=1}^{K}\frac{1}{Z_k}\sum_{d=1}^{D} SF_{kd}^2 \qquad \textbf{(4.2)}$$

Burada $X_1, X_2, \cdots X_N$, N tane örnek, X_{nd}, X_n örüntüsünün d'ninci özelliğidir. SF_{kd}, k'nıncı kümede (G_k) bütün örüntülerin d'ninci özelliklerinin toplamı ve Z_k, k'nıncı kümede örüntü sayısıdır. SF_{kd} aşağıdaki gibi bulunur:

$$SF_{kd} = \sum_{\overline{X_n}\in G_k} X_{nd}.\ (d = 1,2,\cdots,D) \qquad \textbf{(4.3)}$$

Küme sayısı: Önerilen yöntemde diğer amaç fonksiyon ise küme sayısını minimize etmektir. Küme sayısı ne kadar az olursa toplam hata da o kadar yüksek çıkacaktır. Zaten bu iki amaç birbiriyle çelişki halindedir.

4.3.2. Genetik Operatörler

Önerilen yöntemde gen havuzundaki iki birey seçilerek önceden tanımlanmış olasılığa göre tek noktalı çaprazlama kullanılmıştır. Mutasyon operatörü ise arama uzayının daha fazla keşfini sağlamak ve lokal optimuma yakalanmamak için kullanılmıştır. Genelde mutasyon önceden verilen bir olasılıkla bir bireyin belirli bir pozisyonunun değerini değiştirmekle gerçekleşir. Bu yöntem, kodlama yöntemine göre üç mutasyon operatörünü kullanır.

Başlangıç pozisyonunu sağa doğru kaydır: Rastgele seçilmiş bir genin başlangıç pozisyon değerini bir artır.

Başlangıç pozisyonunu sola doğru kaydır: Rastgele seçilmiş bir genin başlangıç pozisyon değerini bir azalt.

Rastgele değiştirme: Mutasyonun bu türünde öncelikle küçük bir tamsayı üretilir. Daha sonra bu sayı bireyin her bir farklı alanının şu anki değerine eklenir ya da çıkarılır. Bu şekilde yeni alan değerleri elde edilir. Yalnız bu yeni değerlerin, alanın alt ve üst limit değerlerini geçmemesine dikkat edilir.

4.3.3. Algoritma

Başlangıç değerleri:

Popülâsyon boyutu: 200
Çaprazlama olasılığı: 0,8
Mutasyon olasılığı: 0,3
Durdurma kıstası: 100 iterasyon
1.....for 1 to 100 do
2.....Genetik_algoritma ile başlangıç popülasyonu üretilir
3.....Üretilen başlangıç popülasyonuna göre uygunluk değerlendirmeleri yapılır
4.....Maksimum iterasyon sayısına erişilip erişilmediği control edilir
5.....Seçme kriterlerine uygun olan kromozomlar ve kurallar seçilir
6……..K-means kümeleme algoritması çalışır.
7.....Uygunluk değerlendirmesi yapılır
8.....İyi genler çaprazlanarak kötü olan genlerin yerine yerleştirilir,
9.....Mutasyon yapılır
10.....Sıralama mekanizması çalışır
11.....İstenen değerler bulununca veya döngü bitince işlem bitirilir.

Şekil 4.3. Önerilen algoritma

5. KÜME GEÇERLİLİK YÖNTEMLERİ

Kümeleme eğiticisiz bir sınıflandırma problemidir. Veriler kümelendikten ve alt gruplara parçalandıktan sonra geçerliliğinin kontrol edilmesi gerekir. Kümeleme algoritmaları tarafından yaygın bir şekilde kabul edilen kriter küme içindeki örnekler yoğun olmalı kümeler arası ayrışma ise iyi sağlanmalıdır. Bu maksatla ilgili kriterin geçerliliği sağlanmalı ve optimum kümeler bulunmalıdır. Temel olarak, bir kümeleme işleminde küme sayısı önceden verilir. Bu tezde geliştirilen yaklaşımda ise kümeleme sonuçları için pareto-optimal çözüm kümesi elde edilir. Bu sayede çözüm kümesinde, küme sayısına göre geçerlilik indeks değer değişimlerinin bütün resmi bir arada görülerek uygun kümelemeye kolay bir şekilde karar verilebilir.

Bu tezde, elde edilen kümelerin geçerliliğini görmek için yaygın olarak kullanılan 6 farklı küme geçerlilik tekniği göz önüne alınmıştır. Bunlar, Dunn indeks [6], Davies Bouldien indeks [7], Silhoutte indeks [8], C indeks [9], SD indeks [10] ve S-Dbw indeksdir [11]. Geçerlilik sonuçlarına göre optimum küme sayıları belirlenecektir.

SD geçerlilik indeksi

SD geçerlilik indeks tanımı, kümeler için ortalama dağılma ve kümeler arasında toplam ayrılma kavramlarına dayalıdır. Kümeler için ortalama dağılma aşağıdaki gibi formülize edilebilir:

$$Dağılma(n_c) = \frac{1}{n_c}\sum_{i=1}^{n_c} \|\sigma(v_i)\| \Big/ \|\sigma(x)\| \qquad \textbf{(5.1)}$$

Burada $\sigma(v_i)$, küme merkezlerinin ortalama standart sapmasıdır (bütün noktalar arasında Öklid uzaklığının ortalaması) ve $\sigma(x)$, bütün veri noktalarının ortalama standart sapmasıdır. Kümeler arası toplam ayrılma ise aşağıdaki gibi tanımlanır:

$$Uzaklık(n_c) = \frac{D_{max}}{D_{min}} \sum_{k=1}^{n_c} \left(\sum_{i=1}^{n_c} \|v_k - v_z\| \right)^{-1} \quad \textbf{(5.2)}$$

Burada $D_{max} = max(\|v_i - v_j\|)$, $\forall i,j \in \{1,2,3,\cdots,n_c\}$ küme merkezleri arasındaki maksimum uzaklık ve $D_{min} = min(\|v_i - v_j\|)$, $\forall i,j \in \{1,2,3,\cdots,n_c\}$ ise küme merkezleri arasındaki minimum uzaklıktır. Böylece SD indeksi aşağıdaki denklem kullanılarak hesaplanabilir:

$$SD(n_c) = \alpha . Dağılma(n_c) + Uzaklık(n_c) \quad \textbf{(5.3)}$$

Burada α ağırlık faktörüdür.

Yukarıdaki denklemde, $Dağılma(n_c)$ kümelerin ortalama yoğunluğunu gösterir. Bu değerin küçük olması yoğun kümelerin elde edilmesi demektir. $Uzaklık(n_c)$, n küme arasındaki toplam ayrılmayı verir. SD'nin iki terimi de farklı aralıklara sahip olduğundan bir ağırlık faktörüne ihtiyaç duyulur. Sonuç olarak, SD indeksini minimize eden küme sayısı optimum değerdir.

S_Dbw geçerlilik indeksi

S_Dbw kümelerin yoğunluğuna (küme içindeki değişme) ve kümeler arasındaki yoğunluğa bağlı olarak formülize edilir. Kümeler arası yoğunluk aşağıdaki gibi tanımlanır:

$$Yoğunluk\_bw = \frac{1}{n_c \times (n_c - 1)} \sum_{i=1}^{n_n} \left(\sum_{\substack{j=1 \\ i \neq j}}^{n_c} \frac{Yoğunluk(u_{ij})}{max\{Yoğunluk(v_i), Yoğunluk(v_j)\}} \right) \quad \textbf{(5.4)}$$

Burada v_i ve v_j, c_i ve c_j kümelerinin merkezleridir; u_{ij}, v_i ve v_j küme merkezleri tarafından tanımlanan hat parçasının orta noktasıdır. Buna göre $Yoğunluk(u)$ aşağıdaki denklem ile bulunur:

$$Yoğunluk(u) = \sum_{l=1}^{n_{ij}} f(x_l, u) \tag{5.5}$$

Burada n_{ij}, c_i ve c_j kümesine ait kayıtların sayısıdır. Yani $x_l \in c_i$ ve $c_j \in S$'dir. $f(x,u)$ fonksiyonu aşağıdaki gibi bulunur:

$$f(x,u) = \begin{cases} 0, Eğer\ d(x,u) > ortstsap \\ 1, diğer\ durumlarda \end{cases} \tag{5.6}$$

Burada $ortstsap$ kümelerin ortalama standart sapmasıdır.

Kümeler arası yoğunluk, kümelerin yoğunluğuyla ilgili olarak kümeler arasındaki bölgede ortalama yoğunluğu değerlendirir. Küme içindeki değişme, kümelerin ortalama dağılmasını ölçer. Bu değer daha önceki SD indeks kısmında tanımlanmıştı.

Sonuç olarak S-Dbw aşağıdaki denklemle hesaplanır:

$$S\_Dbw(n_c) = Dağılma(n_c) + Yoğunluk\_bw(n_c) \tag{5.7}$$

S-Dbw'nin tanımı yoğunluk ve ayrılmayı birlikte ele alır. İndeks değerini minimize eden küme sayısı optimum değerdir.

Dunn geçerlilik indeksi

Dunn indeks aşağıdaki formül ile belirlenir:

$$D_{n_c} = \min_{i=1,\ldots,n_c} \left\{ \min_{j=i+1,\ldots,n_c} \left[\frac{\frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x,y)}{\max_{k=1,\ldots,n_c} \left(\frac{\sum_{x \in C_k} d(x,c_k)}{|C_k|} \right)^2} \right] \right\} \tag{5.8}$$

Burada c_i, belirli bir parçanın i'ninci kümesini temsil eder. $d(x, y)$, x ve y veri noktaları arasındaki uzaklıktır. x, i'ninci kümeye ait iken y j'ninci kümeye aittir. $d(x, c_k)$, ait olduğu küme merkezine x veri noktasının uzaklığıdır. $|c_k|$, k kümesindeki veri noktalarının sayısıdır.

Ölçünün ana amacı kümeler arası uzaklıkları maksimize etmek, küme içi uzaklıkları ise minimize etmektir. Bundan dolayı D'yi maksimize eden küme sayısı optimum sayı olarak ele alınır.

Davies Bouldien geçerlilik indeksi

DB indeksi aşağıdaki denklem kullanılarak hesaplanır:

$$DB = \frac{1}{n}\sum_{i=1}^{n} max_{i \neq j}\left\{\frac{S_n(Q_i) + S_n(Q_j)}{S(Q_i, Q_j)}\right\} \quad \textbf{(5.9)}$$

Burada n küme sayısı, S_n ise kümeden küme merkezlerine bütün nesnelerin ortalama uzaklığıdır. $S(Q_i, Q_j)$ kümelerin merkezleri arasındaki uzaklığı verir.

Davies-Bouldin indeksi küme içi dağılmanın toplamının küme ayrılma arasına oranın bir fonksiyonudur. Küçük bir değer olduğu zaman iyi bir kümeleme sergiler.

Silhouette geçerlilik indeksi

Aşağıdaki formül Silhouette indeksini hesaplama için kullanılır:

$$S(i) = \frac{(b(i) - a(i))}{max\{a(i), b(i)\}} \quad \textbf{(5.10)}$$

Burada $a(i)$ i'ninci nesnenin aynı küme içindeki bütün diğer nesnelere ortalama farklılığıdır. Öklid uzaklığı farklılığı hesaplamada kullanılır. $b(i)$, i'ninci nesnenin en yakın kümedeki bütün nesnelere ortalama farklılığıdır. Formül Silhouette değerinin $[-1,1]$ aralığında gezinebileceğini söyler:

- Silhouette değeri 1'e yakın ise bu demektir ki örnek çok uygun bir kümeye atanmıştır.

- Silhouette değeri sıfır civarında ise bu demektir ki örnek her iki kümeden eşit uzaklıktadır ve diğer herhangi yakın bir kümeye de atanabilir.
- Silhouette değeri -1'e yakın ise bu demektir ki örnek yanlış sınıflandırılmıştır.

En büyük ortalama Silhouette değerli parçalama en iyi kümeleme demektir.

C indeksi

C indeksi aşağıdaki formül ile bulunur:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{5.11}$$

Burada S aynı kümeden bütün örüntü çiftlerinin uzaklıkları toplamıdır. l, bu çiftlerin sayısı olsun. O zaman S_{min}, örüntülerin bütün çiftleri göz önüne alındığı zaman l en küçük uzaklıklar toplamıdır. S_{max} ise bütün çiftlerin dışındaki l en büyük uzaklıklar toplamıdır. Bu durumda C'nin küçük bir değeri iyi bir kümelemenin varlığına işaret eder.

6. UYGULAMA SONUÇLARI

Tezin bu bölümünde, geliştirilen çok-amaçlı genetik algoritma tabanlı kümeleme yöntemin performans ve verimliliğini değerlendirmek ve elde edilen sonuçları küme geçerlilik indeksleriyle test etmek için bazı deneysel tecrübeler yapılmıştır. Tecrübelerin tamamı Pentium 1,8 GHz işlemcili, 1 GB RAM'e sahip ve Windows Vista işletim sistemine sahip bir bilgisayar üzerinde C++ kullanılarak gerçekleştirilmiştir.

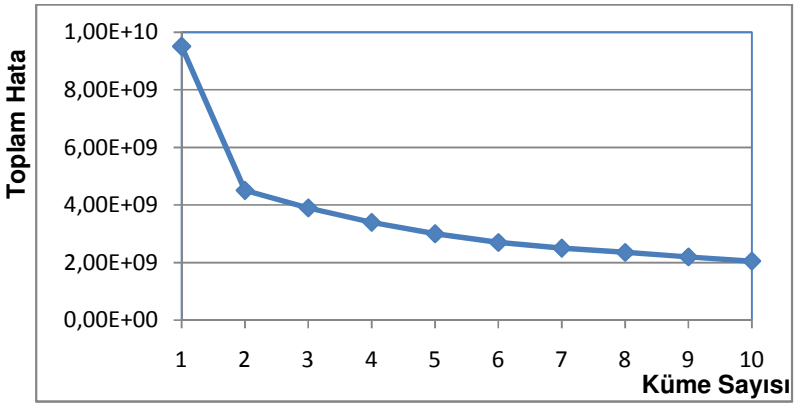
Önerilen yöntemin uygunluğu ve performansı 3 farklı gen ifade veri kümesi kullanılarak test edildi. Bu veri kümeleri, Leukemia [12], Lymphoma [13] ve Kolon kanseridir [14].

6.1. Leukemia veri kümesi

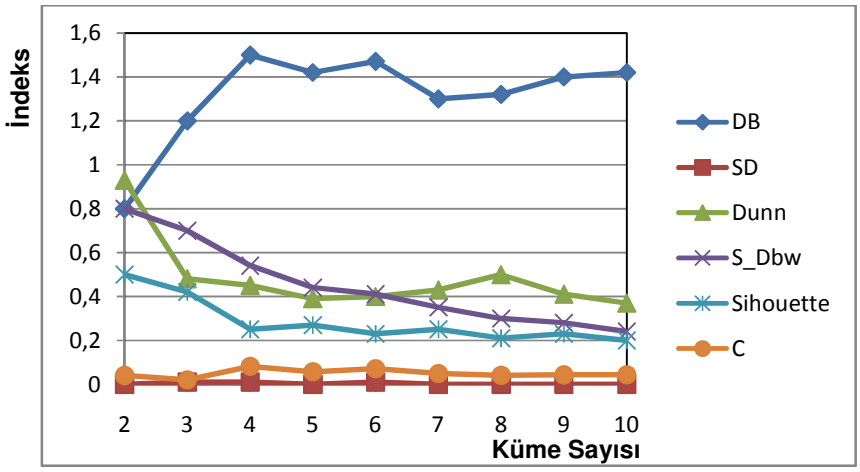
Bu veri kümesi Golub ve diğ. [12] tarafından tanıtılmıştır. 47 ALL-lösemi ve 25 AML-lösemi hasta için 7129 genin ifade seviyelerini içerir.

Şekil 6.1'de çok-amaçlı genetik algoritma yöntemiyle elde edilen pareto front çözümler gösterilmiştir. Şekilde görüldüğü gibi küme sayısı artarken toplam hata miktarı azalmaktadır. Optimum küme sayısı olarak 2 bulunmuştur. Çünkü bu noktadan sonra küme sayısı artarken hata miktarındaki azalış o kadar hızlı olmamaktadır.

Şekil 6.2'de ise Leukemia verisi için küme geçerlilik indeks sonuçları verilmiştir. Bu şekilde de görüleceği gibi S_Dbw hariç bütün geçerlilik analizlerinde optimum küme sayısı 2 olarak bulunmuştur.



Şekil 6.1. Leukemia verisi için önerilen yöntemin bulduğu çözümler

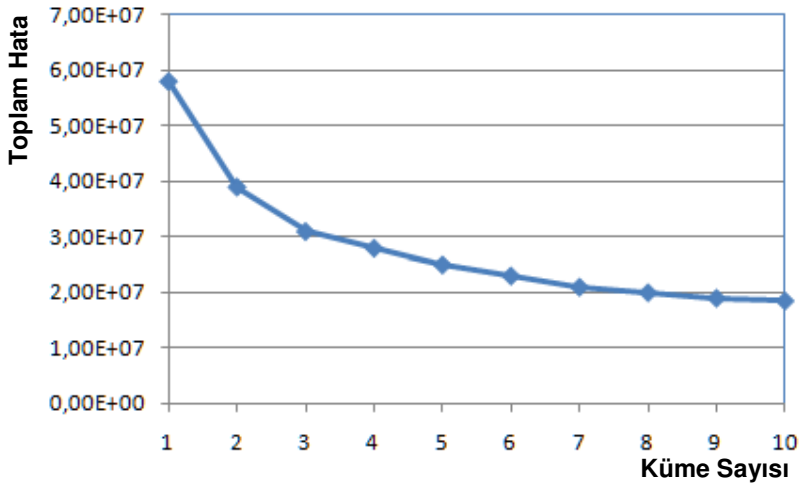


Şekil 6.2. Leukemia verisi için küme geçerlilik sonuçları

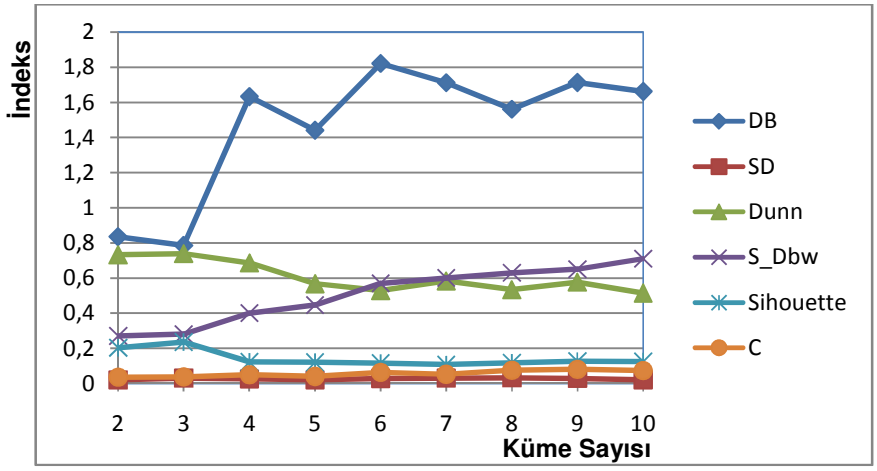
6.2. Lymphoma veri kümesi

Alizadeh ve diğ. [13] tarafından tanıtılan Lymphoma veri kümesi 3 farklı sınıftan 62 hasta için 4026 genin ifade seviyesini içerir. Bu hastalardan 42 tanesi dağınık geniş B-hücreli kötü huylu lenf tümörü, 9 tanesi küçük kese kötü huylu lenf tümörü ve 11 tanesi de kronik lenfoside sahiptir.

Şekil 6.3. Lymphoma veri kümesi üzerinde önerilen yöntemle bulunan küme sayılarını ve bunlara karşılık gelen toplam hata miktarlarını veren pareto front çözümleri gösterir.



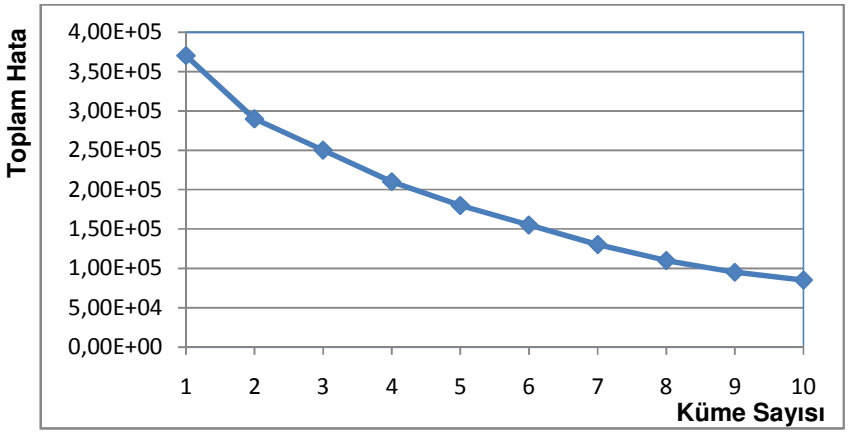
Şekil 6.3. Lymphoma verisi için önerilen yöntemin bulduğu çözümler



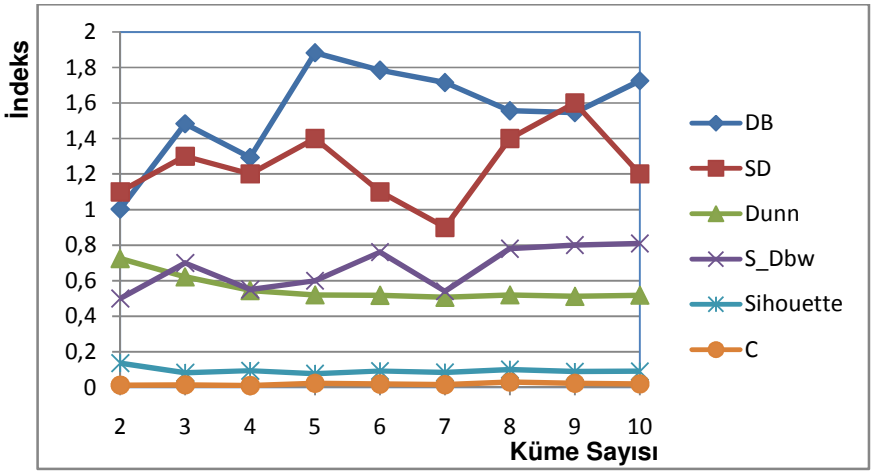
Şekil 6.4. Lymphoma verisi için küme geçerlilik sonuçları

6.3. Kolon Kanser veri kümesi

Kolon veri kümesi Alon ve diğ. [14] tarafından tanımlanan ve kullanıma açık olan bir gen ifade veri kümesidir. Veri kümesi 2 sınıftan 62 hasta için 2000 gen ifade seviyesini içerir. Bu hastalardan 22 tanesi sağlıklı iken 40 tanesi ise kolon kanserine sahiptir. Bu veri kümesi Leukemia kadar düzenli parametrelere sahip değildir. Bu nedenle sınıflandırma uygunluğu çok daha düşüktür.



Şekil 6.5. Kolon Kanser verisi için önerilen yöntemin bulduğu çözümler



Şekil 6.6. Kolon Kanser verisi için küme geçerlilik sonuçları

7. SONUÇLAR

Bu tez çalışmasında DNA mikrodizi verilerini daha uygun bir şekilde kümelemek için çok-amaçlı genetik algoritma tabanlı bir yöntem önerilmiştir. Bu yöntem daha önce önerilen hızlı k-means genetik algoritma yaklaşımının çok amaçlı genetik algoritma biçimine dönüştürülmüş halidir. Yöntemde kullanılan çok-amaçlar küme değişimi içindeki toplam hatanın minimize edilmesi ve küme sayısının da minimize edilmesidir. Bir kümeleme işleminde küme içi elemanların birbirine benzeme değerini ölçen toplam hata miktarı ne kadar düşük olursa o kümeleme yöntemi o kadar etkindir. Kümeleme işleminde diğer bir parametre ise uygun küme sayısının bulunmasıdır. Her zaman için istenen küme sayısının olabildiğince az olmasıdır. Bu sayede birbirine benzeyen örnekler olabildiğince bir araya gelir. Buna karşılık az küme sayısıyla küme işini tamamlamak toplam hatayı da artırır. Bu şekilde her iki amaç için bir ikilem oluşur. Zaten tezin amacı da bu ikilemi gözeten en uygun çözümler kümesini bulmaktır.

Geliştirilen yöntemin uygunluğu literatürde kullanılan 6 önemli küme geçerlilik indeksiyle test edilmiştir. Bu maksatla öncelikle kümeleme işi için zor sayılabilecek 3 farklı veritabanı seçilmiştir. Bu veritabanları DNA mikrodizi veritabanlarından Leukemia, Lymphoma ve Kolon kanser veritabanlarıdır. Önerilen yöntem bu veritabanlarına uygulanarak uygun küme sayıları bulunmuştur. Daha sonra da küme geçerlilik indeksleriyle karşılaştırılmıştır. Deneysel tecrübelerden elde edilen sonuçlarda önerilen yöntemin her üç veritabanı içinde en uygun küme sayılarını bulduğu gözlenmiştir. Leukemia veritabanı için sadece S_Dbw indeksi diğerlerinden farklı olarak küme sayısını 2 bulmamıştır. Benzer şekilde Lymphoma veritabanında ise sadece C indeksi küme sayısını 3 bulamamıştır. Önerilen yöntem ve diğer bütün küme geçerlilik indeksleri küme sayısı olarak 3'ü yakalayabilmiştir. Son olarak Kolon kanseri veritabanında ise yine küme sayısı 2 olarak önerilen yöntem tarafından belirlenmiştir. Bu veritabanı diğer iki veritabanından daha az düzenli olanıdır. Bu nedenle kümeleme işi daha zordur. Buna rağmen küme geçerlilik indekslerinden SD ve S_Dbw hariç diğer indeksler 2 sayısını bulabilmiştir.

Tezin bundan sonraki safhalarında önerilen çok-amaçlı genetik algoritmanın amaç fonksiyonları ve genetik algoritma süreci geliştirilerek daha uygun çözümler yakalanabilir. Bu sayede daha düzensiz veritabanları için kümeleme işlemi kolaylaşabilir.

KAYNAKLAR

[1] **Tavazoie, S. Hughes, D. Campbell, M.J. Cho, R.J. and Church**, **G.M.** 1999. Systematic Determination of Genetic Network Architecture, Nature Genetics, pp. 281-285.

[2] **Brazma A. and Vilo**, **J.** June 2000. Minireview: Gene Expression Data Analysis, Federation of European Biochemical Soc., **480,** 17-24.

[3] **D'haeseleer, P. Wen, X. Fuhrman, S. and Somogyi**, **R.** 1998. Mining the Gene Expression Matrix: Inferring Gene Relationships From Large Scale Gene Expression Data, Information Processing in Cells and Tissues, pp. 203-212.

[4] **Eisen, M.B. Spellman, P.T. Brown, P.O. and Botstein**, **D.** Dec. 1998. Cluster Analysis and Display of Genome-Wide Expression Patterns, Proc.Nat'l Academy of Science, **95,** no. 25, pp. 14863-14868.

[5] **Lu**, **Y.** et al, 2004. FGKA: A Fast Genetic K-means Clustering Algorithm, Proceedings of ACM Symposium on Applied Computing, Nicosia, Cyprus, pp.162-163.

[6] http://tr.wikipedia.org/wiki/Gen_ifadesi Gen İfadesi. 25 Mart 2010.

[7] **Korol, A. B**. Jan – 2003 Review: Microarray Cluster Analysis and Applications, Institute of Evolution, University of Haifa.

[8] **Karaca M, Onus A.N.** 2004. "Array gen expresyon teknolojisi ve bitkisel üretimde kullanımı." Alatarım, **3,** 5–10.

[9] **Alberts B, Johnson A, Lewis J, Raff M,Roberts K, Walter P.** 2002. Molecular Biology of Cell. 4thed. Garland Science, New York

[10] http://www.microarraystation.com/dna-microarrays-info/ Information on DNA Microarrays. 26 Mart 2010.

[11] **Özdağ, H.** 2006. Genetikten genombilime geçişte transkriptom analizleri. *Türk Farmakoloji Derneği Eğitim Sempozyumu* Ankara Üniversitesi Biyoteknoloji Enstitüsü. 24-26.

[12] **Adomas A, Heller G, Olson A, Osborne J, Karlsson M, Nahalkova J, Van Zyl L, Sederoff R, Stenlid J, Finlay R, Asiegbu FO.** 2008. Comparative analysis of transcript abundance in Pinus sylvestris after challenge with a saprotrophic, pathogenic or mutualistic fungus. Tree Physiol. **28** (6), 885–897

[13] **Pollack JR, Perou CM, Alizadeh AA, Eisen MB, Pergamenschikov A, Williams CF, Jeffrey SS, Botstein D, Brown PO**. 1999. Genome-wide analysis of DNA copy-number changes using cDNA microarrays. Nat Genet **23 (1),** 41–46.

[14] **Moran G., Stokes C., Thewes S., Hube B., Coleman DC., Sullivan D**., 2004. Comparative genomics using Candida albicans DNA microarrays reveals absence and divergence of virulence-associated genes in Candida dubliniensis. Microbiology **150 (Pt 10)**, 3363–3382.

[15] **Abul, O.** 2005. Controllıng Discrete Genetic Regulatory Networks, *PhD Thesis, METU.* Department of Computer Engineering, Ankara.

[16] **Eickhoff H., Schneider U., Nordhoff E., Nyarsik L., Zehetner G., Nietfeld W., Lehrach H.,** 2002. Technology Development for DNA Chips. Eds; Grigorenko EV. In. DNA Arrays Technologies and Experimental Strategies. 2nd Ed, United States of America: CRC Press, 1-9.

[17] **Schena, M., Davis, RW.,** 2001. Genes, Genomes, and Chips. Eds; Schena M. In.DNA Microarrays A Practical Approach, New York: Oxford University Press, 1-16.

[18] http://www.worldscibooks.com/etextbook/6712/6712_chap01.pdf DNA Microarray Technology and Data Analysis in Cancer Research. 28 Mart 2010.

[19] http://www.microarraystation.com/dna-microarray-protocol/ DNA Microarray Protocol. 29 Mart 2010.

[20] http://arabidopsis.info/students/microarrays/advantages.html DNA microarray advantages. 30 Mart 2010.

[21] **Ulrike A. N.,** Gene expression profiling in plants using cDNA microarrays, DNA microarrays. chpt. 3. P. 27.

[22] **COBB, K.,** Fall 2006. Microarrays: The search for meanig in a vast sea of data, Biomedical computation review.

[23] **Anderberg, M. R.,** 1973. Cluster Analysis for Application, Academic Press, New York.

[24] **Tatlıdil, H.,** 1996. Uygulamalı Çok Değişkenli Analiz, Ankara.

[25] **Otbiçer, T.,** 2004. Ölçmede Kümeleme Analizi Uygulamaları, Ankara.

[26] **Özekes, S.** 2003. Veri Madenciliği Modelleri ve Uygulama Alanları, İstanbul Ticaret Üniversitesi Dergisi. **3,** 65-82.

[27] **Tan, P.,** 2004. Steinbach, M., Kumar, V., Cluster Analysis: Basic Concepts and Algorithm, Introduction to Data Mining, chpt 8.

[28] **Jain A. K., Murty M. N. ve Flynn P. J.,** 1999. Data Clustering: A Review, ACM Computing Surveys, 31, 3.

[29] **Çakmak, Z., Uzgören, N., Keçek, G**., Kümeleme Analizi Teknikleri ile İllerin Kültürel Yapılarına Göre Sınıflandırılması ve Değişimlerinin İncelenmesi. http://sbe.dpu.edu.tr/12/15-36.pdf, 30 Mart 2010.

[30] **Johnson, R. A., Wichern. D. W.,** 1988. Applied Multivariate Statistical Analysis, (2^{nd} Ed.) Prentice Hall, Englewood Cliffs, New Jersey.

[31] **Han, J. Kamber, M.,** 2000. Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, 1st Ed., San Francisco, USA.

[32] **Dalkılıç, T. E.,** 2005. Switching Regresyon'da Bulanık Sinir Ağları Yaklaşımı ile Parametre Tahmini, Doktora Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[33] http://www.ai-junkie.com/ga/intro/gat1.html Genetic Algorithms in Plain English. 01 Nisan 2010.

[34] **Patrick C. H. Ma, Keith C. C. Chan, Xin Yao, Fellow, IEEE, and David K. Chiu Y.** JUNE 2006. An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis-IEEE Transactions on Evolutionary Computation, vol. **10**, no. 3.

[35] **Chipperfield, A., Fleming, P., Pohlheim, H., Fonseca C.,** Genetic Algorithm Toolbox User's Guide For Use with MATLAB. http://www.sheffield.ac.uk/content/1/c6/03/35/06/manual.pdf, 02 Nisan 2010.

[36] **Prebys, E. K.,** 1999. The Genetic Algorithm in Computer Science. MIT Undergraduate Journal of Mathematics, 165-170.

[37] http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php, Genetic Algorithms. 03 Nisan 2010.

[38] **Carlos A. Coello C, Gary B. Lamont and David A. Van Veldhuizen**, 2007. Evolutionary Algorithms for Solving Multi-Objective Problems. Second Edition.

[39] http://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.Cut.and.Splice.svg Crossover. 5 Nisan 2010.

[40] **Kaya M.,** 2005. Multi-objective genetic algorithm based approaches for mining optimized fuzzy association rules, Soft Computing, 578-586.

[41] **Goldberg D. E.,** 1989. Genetic algorithms in search optimization and machine learning, Addison-Wesley, Reading, MA.

[42] **Goldberg D. E.,** 2002. Design of innovation: Lessons from and for competent genetic algorithms, Kluwer Academic Publishers, Boston.

[43] **Coello C. A., Veldhuızen D. A. Van, ve Lamont G. B.,** 2002. Evolutionary algorithms for solving multi-objective problems, Kluwer Academic Publishers, Boston, MA.

[44] **DEB K.,** 2001. Multi-objective optimization using evolutionary algorithms. John Wiley and Sons, Chichester, UK., 518.

[45] **Dunn, J.** 1974. Well separated clusters and optimal fuzzy partitions. Journal of Cybernetics, Vol.4, 95-104.

[46] **Davies, D.L. and Bouldin, D.W.,** 1979. A cluster separation measure, IEEE Transactions on Pattern Recognition and Machine Intelligence, **1,** 224-227.

[47] **Rousseeuw, P.J.,** 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Comp App. Math, **20,** 53-65.

[48] **Hubert, L. and Schultz, J.,** 1976. Quadratic assignment as a general data-analysis strategy, British Journal of Mathematical and Statistical Psychologie, Vol.29, 190-241.

[49] **Halkidi, M. Vazirgiannis, M. and Batistakis**, **I.,** 2000. Quality scheme assessment in the clustering process, Proceedings of PKDD, Lyon, France,

[50] **Halkidi, M. Vazirgiannis, M.,** Nov. 2001. Clustering Validity Assessment: Finding the optimal partitioning of a data set, Proceedings of IEEE ICDM, California.

[51] **Golub, T. R. et al**, 1999. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science , 286, 531-537.

[52] **Alizadeh, A.A. et al.,** Feb. 2000. Distinct types of diffuse large B-Cell Lymphoma Identified by Gene Expression Profiling, Nature, vol. 403, 503-511.

[53] Alon, U. Barkai, N. Notterman, D.A. Gish, K. Ybarra, S. Mack, D. and Levine, A.J., June 1999. "Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Array," Proc. Nat'l Academy of Science, vol. 96, no. 12, 6745-6750.

EKLER

EK: BİLGİSAYAR PROGRAMI

```
#include "clalg.h"
clalg::clalg(conf * c, databin<USED_DATA_TYPE>*b,
evaluation * ev) {
   bin = b;
   par = c;
   e = ev;}
clalg::~clalg() {
   bin = NULL;
   par = NULL;}
clustering * clalg::getclustering() {
   if (clust == NULL) {
            return NULL;     }
   return clust; }
#include "clustering.h"
 clustering::clustering(conf * c) {
    par = c; }
clustering::~clustering() {
    if (centres != NULL) {
            for (int i=0; i<num; i++) {
               delete[] centres[i];            }
            delete [] centres;     }
    if (partition != NULL) {
            delete [] partition;    } }
void clustering::init(int n) {
    num = n;
    centres = new USED_DATA_TYPE*[num];
    for (int i=0; i<num; i++) {
            centres[i] = new USED_DATA_TYPE[par-
>bindim];     }
    partition = new int[par->binsize]; }
int & clustering::operator[](int i) {
    return partition[i]; }
void clustering::newcentre(int index,
data<USED_DATA_TYPE> * coords) {
     for (int i=0; i<par->bindim; i++) {
            centres[index][i] = (*coords)[i];
    } }
#ifndef DATABIN_JH_2003
#define DATABIN_JH_2003
#include "conf.h"
#include "tmatrix.h"
#include "databin.h"
#include <fstream>
#include <iostream>
#include "math.h"
#include "pesa2.h"
#include "gasdev.h"
#include "pca.h"
#include "string.h"
using namespace std;
template <class BINTYPE> class databin;
template <class DBINTYPE> class docbin;
template <class DATATYPE>
class data {
    friend class databin<DATATYPE>;
 private:
   DATATYPE * vector;
  protected:
   conf * par;
 public:
   int color;
   int cluster;
  public:
   ~data();
   data(conf * c, DATATYPE * d);
   data(conf * c, DATATYPE * dat, int col, int cl);
   const int length();

DATATYPE square(DATATYPE x);
   DATATYPE &operator[](const int i);
   const DATATYPE distanceto(data<DATATYPE> &
d);
   void add(data<DATATYPE> & d);
   void div(int i);
   void set(data<DATATYPE> & d);
   void set(DATATYPE * d);};
template <class BINTYPE>
class databin {
 protected:
   conf * par;
   data<BINTYPE> ** bin;
 public:
   tmatrix<BINTYPE> * distancematrix;
   BINTYPE * mean;
   BINTYPE * std;
  BINTYPE * minvalue;
   BINTYPE * maxvalue;
   BINTYPE ** memmatrix;
   int * color;
   char ** label;
   public:
  databin(conf * c);
  databin(conf * c, char * name, int size, int dim);
  ~databin();
  const BINTYPE d(const int index1, const int index2);
  const BINTYPE precomputed_d(const int index1, const
int index2);
  data<BINTYPE> & operator[](const int i);
  void permutate();
  void uniformprescription();
  int find(char * templabel, char ** classlabel, int &
labelctr);};
template <class DATATYPE>const int data
<DATATYPE>::length() {
   return par->bindim;
}
template <class DATATYPE>data <DATATYPE>::
data(conf * c, DATATYPE * dat) {
   par = c;
   vector = new DATATYPE[par->bindim];
   color = 0;
   cluster = 0;
        for (int i=0; i<par->bindim; i++) {
            vector[i] = dat[i];    }}
```

```
template <class DATATYPE>data <DATATYPE>::
data(conf * c, DATATYPE * dat, int col, int cl ) {
   par = c;
   vector = new DATATYPE[par->bindim];
   color = col;
   cluster = cl;

   for (int i=0; i<par->bindim; i++) {
            vector[i] = dat[i];   }}
template <class DATATYPE>data
<DATATYPE>::data(conf * c)  {
   par = c;
   vector = new DATATYPE[par->bindim];
   color = 0;
   cluster = 0;
      for (int i=0; i<par->bindim; i++) {
            vector[i] = 0;    }}
template <class DATATYPE>data
<DATATYPE>::~data() {
   delete [] vector;}

template <class DATATYPE> DATATYPE data
<DATATYPE>::square(DATATYPE x) {
   return x*x;}
template <class DATATYPE> DATATYPE &data
<DATATYPE>::operator[](const int i) {
   return vector[i];}
template <class DATATYPE>const DATATYPE data
<DATATYPE>::distanceto(data<DATATYPE> & dd) {
  if (par->distance == EUCLIDEAN) {
   data & d = (data &)dd;
   DATATYPE result = 0.0;
   for (int i=0; i<par->bindim; i++) {
     result += square(d.vector[i] - vector[i]);    }
     return sqrt(result);  }
  else if (par->distance == COSINE) {
   data & d = (data &)dd;
   DATATYPE result = 0.0;
   DATATYPE r1 = 0.0;
   DATATYPE r2 = 0.0;
   for (int i=0; i<par->bindim; i++) {
            result += d.vector[i] * vector[i];
            r1 += d.vector[i] * d.vector[i];
            r2 += vector[i] * vector[i];     }
1.0 - 0.5*(1.0 + result / sqrt(r1*r2)) << endl;
   if (r1 == 0 || r2 == 0) return 0.0;
   else return 1.0 - 0.5*(1.0 + result / sqrt(r1*r2));
  }
  else if (par->distance == CORRELATION) {
   data & d = (data &)dd;
   DATATYPE result = 0.0;
   DATATYPE r1 = 0.0;
   DATATYPE r2 = 0.0;
   DATATYPE av1 = 0.0;
   DATATYPE av2 = 0.0;
   for (int i=0; i<par->bindim; i++) {
     av1 += vector[i] * vector[i];
     av2 += d.vector[i] * d.vector[i];    }
   av1 /= double(par->bindim);
   av2 /= double(par->bindim);

   for (int i=0; i<par->bindim; i++) {
            result += (d.vector[i]-av2) * (vector[i]-av1);
            r1 += square(d.vector[i]-av2);
            r2 += square(vector[i]-av1);     }
   if (r1 == 0 || r2 == 0) return 0.0;
   return 0.5*(1.0 - (result / sqrt(r1*r2)));  }
  else if (par->distance == GAUSSIAN) {
   data & d = (data &)dd;
   DATATYPE result = 0.0;
   for (int i=0; i<par->bindim; i++) {
            result += square(d.vector[i] - vector[i]);
   }
   return 1-exp(-result);  }
  else if (par->distance == JACCARD) {
   data & d = (data &)dd;
   DATATYPE result = 0.0;
   DATATYPE r1 = 0.0;
   DATATYPE r2 = 0.0;

   for (int i=0; i<par->bindim; i++) {
      result += d.vector[i] * vector[i];
            r1 += d.vector[i] * d.vector[i];
            r2 += vector[i] * vector[i];     }
   DATATYPE denom = r1+r2-result;
   if (denom == 0) return 0;
   else return 0.5*(1.0 - result / denom);   }
  else {
   cerr << "NO valid distance function selected" << endl;
}}
template <class DATATYPE>void data
<DATATYPE>::add(data<DATATYPE> & d) {
   for (int i=0; i<par->bindim; i++) {
            vector[i] += d.vector[i];
   }}
template <class DATATYPE> void
data<DATATYPE>::set(data<DATATYPE> & d) {
     for (int i=0; i<par->bindim; i++) {
              vector[i] = d.vector[i];
     }}
template <class DATATYPE> void
data<DATATYPE>::set(DATATYPE * d) {
     for (int i=0; i<par->bindim; i++) {
              vector[i] = d[i];
     }}
template <class DATATYPE>void data
<DATATYPE>::div(int divisor) {
   for (int i=0; i<par->bindim; i++) {
            vector[i] /= double(divisor);
   }}
template <class BINTYPE>databin
<BINTYPE>::~databin() {
  delete [] std;
  delete [] mean;
  delete [] minvalue;
  delete [] maxvalue;
   for (int i=0; i<par->binsize; i++) {
            delete bin[i];
            delete memmatrix[i];    }
   delete [] bin;
   delete [] memmatrix;
   if (distancematrix != NULL) delete distancematrix;
}
template <class BINTYPE> const BINTYPE databin
<BINTYPE>::precomputed_d(const int index1, const int
index2) {
   return (*distancematrix)(index1,index2);}
```

```
template <class BINTYPE>inline const BINTYPE databin
<BINTYPE>::d(const int index1, const int index2) {
   return bin[index1]->distanceto(*bin[index2]);
}
template <class BINTYPE>inline data<BINTYPE> &
databin <BINTYPE>::operator[](const int i) {
   return *bin[i];}
used (identified by name)
template <class BINTYPE>databin
<BINTYPE>::databin(conf * c, char * name, int size, int
dim) {
  using namespace std;
   int clust = size;
  ifstream input(name);
   if (! input){
    cerr << "Error while trying to open file " << name <<
endl;
     exit(0);    }
   par = c;
   par->bindim = dim;
   par->binsize = size;
   par->kclusters = clust;
   par->num_cluster = clust;
   int labelctr = 0;
   char ** classlabels = new char *[size];
   for (int i=0; i<size; i++) {
    classlabels[i] = new char[100];    }
   par->size_cluster = new int[clust];
   for (int i=0; i<clust;i++) {
    par->size_cluster[i] = 0;    }
   color = new int[par->binsize];
   bin = new data<USED_DATA_TYPE>*[par->binsize];
   mean = new USED_DATA_TYPE[par->bindim];
   std = new USED_DATA_TYPE[par->bindim];
   minvalue = new USED_DATA_TYPE[par->bindim];
   maxvalue = new USED_DATA_TYPE[par->bindim];
   label = new char *[size];
   for (int i=0; i<size;i++) {
    label[i] = new char[1000];    }
   for (int k=0; k<par->bindim; k++) {
            maxvalue[k] = -100000000.0;
            minvalue[k] = 100000000.0;    }
   for (int i=0; i<par->bindim; i++) {
            mean[i] = 0;
            std[i] = 0;    }

   USED_DATA_TYPE ** temp = new
USED_DATA_TYPE*[par->binsize];
   for (int i=0; i<par->binsize; i++) {
            temp[i] = new USED_DATA_TYPE[par-
>bindim];
   }

 char templabel[100];
   par->num_cluster = 0;
   for (int i=0; i<par->binsize; i++) {
            for (int j=0; j<par->bindim; j++) {
              if ( !input.eof() ) {
                input >> temp[i][j];
                //            cerr << temp[i][j] << " ";
                mean[j] += temp[i][j];
              }
              else {
                cerr << "Error in input line " << i << " " << "
at column " << j << endl;
                cerr << "Size = " << par->binsize << " and
dimensionality = " << par->bindim << " given do not
correspond to the real size of input file " << name << endl;
                exit(0);  }}
            input >> label[i];
            strcpy(templabel,label[i]);
            color[i] = find(templabel, classlabels, labelctr);
            par->size_cluster[color[i]]++;
            par->num_cluster = (int)max(par->num_cluster,
color[i]+1);    }
   int dummy;
   input >> dummy;
   if ( !input.eof() ) {
    cerr << "Error at the end of input" << endl;
    cerr << "Size = " << par->binsize << " and
dimensionality = " << par->bindim << " given do not
correspond to the real size of input file " << name << endl;
     exit(0);    }
   for (int j=0; j<par->bindim; j++) {
            mean[j] /= double(par->binsize);    }
   for (int j=0; j<par->bindim; j++) {
            for (int i=0; i<par->binsize; i++) {
              double diff = temp[i][j]-mean[j];
              std[j] += diff*diff;
            }
            std[j] /= double(par->binsize);
            std[j] = sqrt(std[j]);    }
   int ctr = 0;
   for (int i=0; i<par->binsize; i++) {

              for (int j=0; j<par->bindim; j++) {
                if (par->normalize == true) {
                  temp[i][j] -= mean[j];
                  if (std[j] != 0) temp[i][j] /= std[j];
                }  }
bin[i] = new data<USED_DATA_TYPE>(par, temp[i],
color[i]+1, color[i]);
data<USED_DATA_TYPE>(par, temp[i]);
              ctr++;    }
      for (int i=0; i<par->binsize; i++) {
    for (int k=0; k<par->bindim; k++) {
            maxvalue[k] = max(maxvalue[k],
((*(bin[i]))[k]));
            minvalue[k] = min(minvalue[k],
((*(bin[i]))[k]));     }    }

   if (par->normalize == true) {
    for (int j=0; j<par->bindim; j++) {
            mean[j] = 0.0;
            std[j] = 1.0;     }    }
   distancematrix = new
tmatrix<USED_DATA_TYPE>(par->binsize);

  par->mu = 0.0;
   par->maxd = 0.0;
   par->mind = 0.0;
   for (int i=0; i<par->binsize; i++) {
            for (int j=0; j<i; j++) {
              (*distancematrix)(i,j) = bin[i]-
>distanceto(*(bin[j]));
              par->mu += (*distancematrix)(i,j);
              if (i==1 && j==0) {
```

```
                par->mind = (*distancematrix)(i,j);
                par->maxd = (*distancematrix)(i,j);
              }
              else {
                if ((*distancematrix)(i,j) < par->mind) {
                        par->mind = (*distancematrix)(i,j);
                }
                BINTYPE d = (*distancematrix)(i,j);
                if (d < 0) d =-d;
                if (d > par->maxd) {
                        par->maxd = d;
                }           }             }    }
   for (int i=0; i<par->binsize; i++) {
     for (int j=0; j<i; j++) {
            (*distancematrix)(i,j) = ((*distancematrix)(i,j)-
par->mind)/(par->maxd-par->mind);      }    }
   par->mu /= 0.5*(par->binsize-1)*par->binsize;
   memmatrix = temp;
   par->kclusters = par->num_cluster; }

  template <class BINTYPE> void databin
<BINTYPE>::uniformprescription() {
  long idum = 732832;

  if (distancematrix != NULL) delete distancematrix;
USED_DATA_TYPE ** datamatrix = new
USED_DATA_TYPE * [par->binsize];

  for (int i=0; i<par->binsize; i++) {
     datamatrix[i] = new USED_DATA_TYPE[par-
>bindim];    }
   for(int t=0;t<par->binsize;t++)      {
            for (int j=0; j<par->bindim; j++) {
              datamatrix[t][j] = memmatrix[t][j];
            }      }
      pca(datamatrix, par->binsize, par->bindim);
     for(int t=0;t<par->binsize;t++)
     {
            for (int j=0; j<par->bindim; j++) {
              (*(bin[t]))[j] = datamatrix[t][j];      }      }

   for (int i=0; i<par->binsize; i++) {
     delete [] datamatrix[i];
   }
   delete [] datamatrix;

     distancematrix = new tmatrix<BINTYPE>(par-
>binsize);
   if (distancematrix == NULL) {
            cerr << "Databin: Memory allocation failed" <<
endl;
            exit(0);    }
       par->mu = 0.0;
   par->max = 0.0;
   for (int i=0; i<par->binsize; i++) {
            for (int j=0; j<i; j++) {
               (*distancematrix)(i,j) = bin[i]-
>distanceto(*(bin[j]));
               par->mu += (*distancematrix)(i,j);
               par->max = max(par->max,
(*distancematrix)(i,j));
            }
   }
   par->mu /= 0.5*(par->binsize-1)*par->binsize;
}
template <class BINTYPE> void databin
<BINTYPE>::permutate() {
  BINTYPE tempval;

   for (int i=0; i<par->binsize; i++) {
            int j = int(mydrand()*(par->binsize));
            data<BINTYPE> * temp = bin[i];
            bin[i] = bin[j];
            bin[j] = temp;
            for (int k=0; k<par->binsize; k++) {
               if ((k != i)  && (k != j)) {
                        tempval = (*distancematrix)(i,k);
                        (*distancematrix)(i,k) =
(*distancematrix)(j,k);
                        (*distancematrix)(j,k) = tempval;
               }         }    }}

template <class BINTYPE> int databin
<BINTYPE>::find(char * templabel, char ** classlabel,
int & labelctr) {
   for (int i=0; i<labelctr; i++) {
            if (strcmp(classlabel[i],templabel) == 0) {
               return i;
            }    }
   strcpy(classlabel[labelctr], templabel);
   labelctr++;
   return labelctr-1;
}
#endif
#include "kmeans.h"
#include "random.h"
#define MAXIT 1
static long idum = 258954802;
double kmeans::square(double x) {
   return x*x;}

kmeans::kmeans(conf * par,
databin<USED_DATA_TYPE> * bin, evaluation * e):
   clalg(par, bin, e) {
   partition = new int[par->binsize];
   center = new data<USED_DATA_TYPE> * [par-
>kclusters];}

kmeans::~kmeans() {
   delete [] partition;
   for (int i=0; i<par->kclusters; i++) {
            delete center[i];    }
   delete [] center;
   delete clust;   }
void kmeans::init() {
USED_DATA_TYPE * proto = new
USED_DATA_TYPE[par->bindim];

   for (int i=0; i<par->kclusters; i++) {
            for (int k=0; k<par->bindim; k++) {
              proto[k] = 0;         }
            center[i] = new
data<USED_DATA_TYPE>(par, proto);    }
   delete [] proto;}
void kmeans::run() {
   clustering * bestclust = NULL;
   int changes = TRUE;
   int iteration_ctr = 0;
```

```
   double minvar = -1.0;
   int * mem_ctr = new int[par->kclusters];
      for (int i=0; i<MAXIT; i++) {
            for (int i=0; i<par->binsize; i++) {
               partition[i] = int(ran0(&idum)*par-
>kclusters);            }
            clust = new clustering(par);
            iteration_ctr = 0;
            changes = TRUE;
            while ((changes == TRUE) && (iteration_ctr <
10)) {



            for (int i=0; i<par->kclusters; i++) {
                     mem_ctr[i] = 0;          }
               iteration_ctr++;
               changes = FALSE;

               int * ctr = new int[par->kclusters];
               for (int i=0; i<par->kclusters; i++) {
                     ctr[i] = 0;
               }

               for (int i=0; i<par->kclusters; i++) {
                 for (int j=0; j<par->bindim; j++) {
                     (*(center[i]))[j] = 0;
                 }
               }
               for (int i=0; i<par->binsize; i++) {
                     center[partition[i]]->add((*bin)[i]);
                     ctr[partition[i]]++;
               }
               for (int i=0; i<par->kclusters; i++) {
                     if (ctr[i] != 0) {
                        center[i]->div(ctr[i]);

                     }
               }
               delete [] ctr;

               for (int i=0; i<par->binsize; i++) {
                     int temp = partition[i];
                     partition[i] = 0;
                     USED_DATA_TYPE mind =
center[0]->distanceto((*bin)[i]);
                     for (int j=0; j<par->kclusters; j++) {
                        USED_DATA_TYPE d = center[j]-
>distanceto((*bin)[i]);

                        if (d < mind) {
                              partition[i] = j;
                              mind = d;
                        }
                     }
                     mem_ctr[partition[i]]++;
                     if (temp != partition[i]) {
                        changes = TRUE;
                     }
               }

                     for (int i=0; i<par->kclusters; i++) {
                     //cout << mem_ctr[i] << endl;
                     if (mem_ctr[i] == 0) {
                        //            cout << "Empty cluster "
<< endl;
                        changes = TRUE;
                        int index = int(ran0(&idum)*par-
>binsize);
                        for (int j=0; j<par->bindim; j++) {
                              (*(center[i]))[j] =
(*bin)[index][j];
                        }
                        partition[index] = i;
                     }
               }
            }
            clust->init(par->kclusters);

            for (int i=0; i<par->binsize; i++) {
               (*clust)[i] = partition[i];
            }

            for (int i=0; i<par->kclusters; i++) {
               clust->newcentre(i, center[i]);
            }

            e->init(bin,clust);
            double var = e->variance();
            if ((minvar == -1) || (var < minvar)) {
               minvar = var;

               if (bestclust != NULL) delete bestclust;
               bestclust = clust;
            }
            else {
               delete clust;
            }

   }

   int * kctr = new int[par->kclusters];
   for (int i=0; i<par->kclusters; i++) kctr[i] = 0;
   for (int i=0; i<par->binsize; i++) {
            kctr[(*bestclust)[i]]++;
   }
   clust = bestclust;

   delete [] mem_ctr;


}
```

ÖZGEÇMİŞ

Mustafa KAHRAMAN, 1982 yılında Manisa'da dünyaya geldi. İlkokulu burada tamamladıktan sonra, ortaokul ve liseyi İzmir'de bitirdi. 2001 yılında başladığı Fırat Üniversitesi T.E.F Elektronik ve Bilgisayar Eğitimi Bölümü, Elektronik Öğretmenliğinden 2005 yılında mezun oldu. 2007 yılında F.Ü Fen Bilimleri Enstitüsü, Biyomühendislik Ana Bilim Dalında yüksek lisans eğitimine başlamış ve halen devam etmektedir.