ÇOK AMAÇLI GENETİK ALGORİTMA İLE KATEGORİK VERİLERİN SINIFLANDIRILMASI

KAYHAN DURSUN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2012
ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver Kaynak
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan Doğdu
Anabilim Dalı Başkanı

Kayhan DURSUN tarafından hazırlanan ÇOK AMAÇLI GENETİK ALGORİTMA İLE KATEGORİK VERİLERİN SINIFLANDIRILMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Tansel ÖZYER
Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Reza HASSANPOUR _____

Üye : Yrd. Doç. Dr. Mehmet TAN _____

Üye : Yrd. Doç. Dr. Tansel ÖZYER _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Kayhan DURSUN

Üniversitesi: TOBB Ekonomi ve Teknoloji Üniversitesi

Enstitüsü: Fen Bilimleri

Anabilim Dalı: Bilgisayar Mühendisliği

Tez Danışmanı : Yrd. Doç. Dr. Tansel ÖZYER

Tez Türü ve Tarihi: Yüksek Lisans – Temmuz 2012

KAYHAN DURSUN

ÇOK AMAÇLI GENETİK ALGORİTMA İLE KATEGORİK VERİLERİN SINIFLANDIRILMASI

ÖZET

Bilgisayar bilimlerinde verilerin sınıflandırılması işlemi yıllardır üzerine yoğunlaşılıp, pek çok yöntemin öne sürüldüğü bir konudur. Bu konu, verilerin belirli öznitelikleri kullanılarak en uygun şekilde kümelenmeleri işlemi üzerine yoğunlaşmaktadır. Bu sayede birbirine benzeyen verileri birbirleri ile aynı öbeğe koyulabilmekte ve bu sonuçlar üzerinden belli çıkarımlar yapılabilmektedir. Bu alan, ele alınan veri kümelerinin durumlarına göre derinleşebilmektedir. Örneğin büyük boyutlu ve küçük boyutlu verilerin öbeklendirilmeleri birbirinden çok farklı şekilde özelleşebilmektedir.

Öbeklendirmenin türüne göre ise, çok amaçlı öbeklendirme son yıllarda çok revaçta olan bir problemdir. Çok amaçlı öbeklendirme, birbirinden farklı amaçları aynı anda ele alarak verilerin kümeleme sonuçlarını çıktı olarak sunar.

Bu tezde ise temel olarak çok amaçlı bir yapı ile kategorik veri kümelerinin öbeklendirilmesi üzerinde çalışılmıştır. Elde edilen sonuçlar tek amaçlı bir şekilde öbekleme yapan k-mod öbeklemesinin sonuçları ile karşılaştırılmıştır. Önerilen yöntemin her durumda daha doğal ve başarılı öbekleme sonuçlarına ulaştığı saflık ölçüm metriği ele alınarak ortaya konulmuştur.

Çalışmanın ileriki aşamalarında ise büyük boyutlu veri kümeleri işin içine girince çıkan problemler ele alınıp, bu durumlara çözümler getirilmiştir. Bu koşullar için ise geliştirilen yöntem, veriler üzerinden sıklık eleman kümelerinin elde edilip, bu

kümeler üzerinden verimli şekilde öbekleme yapılması işlemidir. Sonuçta elde edilen çıktılar ise önerilen yöntemin verimliliğini ve tutarlılığını ortaya koymaktadır.

Anahtar Kelimeler: öbekleme, sınıflandırma, kategorik veriler, çok amaçlı öbekleme, genetik algoritma, sıklık eleman kümeleri, küçük ve büyük ölçekli veri kümesi

University: TOBB Economics and Technology University

Institute: Institute of Natural and Applied Sciences

Science Programme: Computer Engineering

Supervisor: Assistant Prof. Dr. Tansel ÖZYER

Degree Awarded and Date: Master of Science – July 2012

KAYHAN DURSUN

CLUSTERING CATEGORICAL DATASETS USING MULTI OBJECTIVE GENETIC ALGORITHM

ABSTRACT

The process of classifying data in computer science has been the interest of lots of practices in recent years and lots of techniques have been suggested about this topic. This topic concentrates on how to classify these data appropriately using their attribute values. As a result of this, data which are similar can be put in same or near clusters and there can be made some inferences from these clustering results. These area can deepen with the status of data sets. For example classification of large and small scale data specialize very differently.

Multi objective clustering as a kind of clustering is a trending topic in recent years. This type of clustering takes into account more than one objective and gives clustering results of data as an output.

In this thesis, it has been studied on basically how to classify categorical data sets with a multi objective structure. Obtained results have been compared with the results of k-mod algorithm which do the clustering process with one objective. Using the purity measure, it has been shown that in every condition, the proposed procedure gives better clustering results.

In further steps of the work, the problems occurring with the clustering of large scale data investigated and some solutions have been proposed for them. For these

conditions, the procedure that has been proposed is to generate frequent item sets from the data and classifying these frequent item sets efficiently. In the end, outputs that have been gathered illustrate the efficiency and consistency of this procedure.

Keywords: clustering, classification, categorical data sets, multi objective clustering, genetic algorithm, frequent item sets, small scale data set, large scale data set

TEŞEKKÜR

Bu tezin hazırlanmasında yardım ve katkılarıyla beni yönlendiren değerli Hocam Yrd. Doç. Dr. Tansel Özyer'e, yüksek lisans eğitimim boyunca bana değerli katkılarda bulunan TOBB Ekonomi ve Teknoloji Bilgisayar Mühendisliği bölümü Hocalarıma, tez çalışmalarım boyu bana büyük yardımları olan değerli arkadaşım Onur Can Sert'e, yüksek lisans çalışmalarım boyunca bana maddi konularda destek verip verimli bir şekilde çalışmama katkıda bulunan TÜBİTAK'a ve en önemlisi beni bugünlere getiren değerli aileme teşekkürü borç bilirim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Kısaltma	Açıklama
NSGA	Non – dominated Sorting Genetic Algorithm
VTBK	Veritabanı Bilgi Keşfi
WCD	Weighted Coverage Density
EWCD	Expected Weighted Coverage Density
BAKY	Beklenen Ağırlıklı Kapsam Yoğunluğu
OK	Örtüşme Katkısı
BE	Beklenen Entropi

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simge	Açıklama
$\mathbf{WCD(C_k)}$	k öbeği için beklenen kapsam yoğunluğu değeri
$\mathbf{I_{kj}}$	k. öbekteki j. öznitelik
$\mathbf{N_k}$	k. öbekteki toplam eleman sayısı
$\mathbf{S_k}$	Frekans toplamı (k kümesindeki elemanların frekans toplamı)
$\mathbf{C^k}$	İçerisinde k adet öbek tutan öbek kümesi
d(X, Y)	X ve Y veri elemanları arasındaki uzaklık
δ(xj,yj)	X ve Y elemanlarının j. öznitelikleri arasındaki uzaklık değeri
S(s)	Sıklık kümesi s‘ in önemlilik değeri
E(Aj)	Bir veri kümesindeki j. kolonun entropi değeri
E(S)	S veri kümesinin entropi değeri
$\mathbf{BE(C^k)}$	k adet öbekten oluşan kümenin beklenen entropi değeri
R(p, q)	p ve q sıklık eleman kümeleri arasındaki fazlalık değeri
g(f)	f sıklık eleman kümesinin kazanım değeri
O(F)	F sıklık eleman kümesi topluluğunun örtüşme değeri
OK(f)	f sıklık eleman kümesinin örtüşme katkısı değeri

BÖLÜM 1

1. GİRİŞ

Öbekleme, veriyi, iyi ve sıkı bir şekilde birbirinden ayrılmış gruplara bölme işlemidir. Kaç adet gruba ayrım yapılacağı ise ön bilgi olarak verilmemektedir. Literatürde çok geniş bir şekilde yapılmış çalışmalar bulunmaktadır. Örn. k-means [1,2] nümerik verilerin öbeklendirilmesi ile ilgili kabul görmüş bir sınıflandırma yöntemidir. K-mod öbeklemesi ise k-means'in kategorik veriler üzerine geliştirilmiş sürümüdür. Ortalama (mean) kavramının yerine, özniteliklerin uygunluk ölçütünü dikkate alan mod kavramı monte edilmiştir.[1,2] ROCK sınıflandırması kategorik veriyi sınıflandırmak için hiyerarşik bir yapı izler.[3] İlk olarak her veri, ayrı bir öbek gibi alındıktan sonra bir uzaklık ölçütü yardımı ile birbirine yakın olan öbeklerin birleştirilmesi mantığı temel alınmaktadır. LIMBO, bilgi darboğazı mekanizması kullanarak tüm öbekleri ağaç yapısı altında saklar ve geriye kalan veriler en yakın öbek altından elde edilir. [4] CACTUS öbeklendirmesi hızlı özetleme tekniklerini kullanır ve veri kümesi üzerinde iki seferli bir tarama işlemi sonunda öbekleme yapar. [5] COOLCAT algoritması entropi ölçümüne dayanır ve ilk öbek merkezlerini, birbirinden en uzak şekilde duran veriler olarak belirler.[6] Daha sonra ise geriye kalan her bir veriyi kendisine en yakın gözüken sınıfa atar. STIRR algoritması dinamik bir ortamda yinelemeli şekilde çalışarak öbekleme yapar.[7] CLOPE algoritması bir eğim hesabı yapmak için uygunluk grafiği tabanlı genişlik ve yükseklik bilgisi kullanır.[8] Bu yöntemin bir varyasyonu ise uygunluk grafiği üzerinden elde edilen bir beklenen ağırlıklı kapsama yoğunlu hesabını temel alır.[9]

Bütün bu öbekleme algoritmaları tek bir amaç fonksiyonunu temel alarak veriyi öbeklere ayırır. Bu algoritmaların aslında birden çok amaç fonksiyonuna ihtiyaç duyduğu aşikârdır. Sonuç olarak, gerçek hayatta birbiriyle çakışan birden çok durumu dikkate almak, insanların karar mekanizmalarının çalışma mantığıyla daha uygun bir paralellik gösterecektir. Birden çok amaç, olası çözümlere ulaşmak için daha çok alternatifi ortaya koyacakken, karar kısmını ise insan karar

mekanizmalarına bırakacaktır. Arama uzayında, tüm amaçların eşzamanlı olarak optimize edilmesi imkânı, çok amaçlı yapıyı tek amaçlı bir yapıya göre üstün kılmaktadır. [10]

Çok amaçlı evrimsel yaklaşımlar önceleri üzerine çalışılmış ve kabul görmüştür. Çok amaçlı mantıkla çalışan genetik algoritmalar nümerik verileri öbeklendirmek için geliştirilmiştir.[11,12] Örneğin baskın olmayan sıralama genetik algoritması[1] alternatif çözümler veren baskın küme sonuçlarını ortaya çıkarmak için kullanılmıştır.[13]

Bu tez çalışması ile bu konuya yapılan katkılar aşağıdaki gibi sıralanabilir:

- Algoritma kategorik veriler üzerinde de çalışacak şekilde geliştirilmiştir.
- Yinelemeli yaklaşım öbeklemesinden farklı olarak, küçük boyuttaki veri kümelerinin içerdiği veriler bit gösterimi ile temsil edilmiştir.
- Bir havuzda toplanmış olan, verilerin aralarındaki birliktelik sonuçları, yapılan öbeklemenin çıktıları olarak kabul edilmiştir. Bu sonuçlar, aynı anda birden çok amaç fonksiyonunun çalıştırılması ile elde edilen doğal sonuçlar olup, alternatif öbekleme sonuçları NSGA[13] algoritması tarafından bulunmuştur.
- Sonuçlar, saflık[2] metriği ile test edilmiştir.
- Büyük ölçekli verilerin öbekleme işlemlerindeki sorunlar göz önüne alarak, yöntemin bu tarz veri kümeleri söz konusu olduğunda verimli şekilde öbekleme sonuçları vermesi sağlanmıştır. Bu doğrultuda, veri kümesini temsil eden sıklık eleman kümeleri çıkartılıp, aralarından birbiri ile en az çakışmaya sahip olan ve hitap ettiği veri sayısı en geniş olan kurallar seçilmiştir. Öbekleme işlemi bu seçilen kurallar üzerinden yapılmıştır.

[1] ing: *NSGA*
[2] ing: *Purity*

- Kategorik veriler öbeklendirilirken bir başka problem olan, ayrılacak öbek sayısının kullanıcı tarafından belirlenmesinin önüne geçilmiştir. Veri kümesinin bölünmesi istenen en küçük öbek sayısı belirlendikten sonra, verinin daha fazla sayıda öbeklere bölünmesi, önceki öbekleme sonuçları ve verilerin eski öbek merkezlerine göre olan uzaklık bilgileri kullanılarak sağlanmıştır.

Bu tez çalışması şu şekilde düzenlenmiştir: Bölüm 1'de, tez çalışmasının temelleri ve yapılan çalışmayı kısaca özetleyen giriş bölümü bulunmaktadır. Bölüm 2'de, kullanılan yöntemler hakkında bilgi verilerek, yapılan çalışmalara değinilmiştir. Bölüm 3' te bu çalışmada geliştirilen çok amaçlı öbekleme yapan genetik algoritma anlatılmıştır. Bölüm 4' te büyük ölçekli veri kümelerinin öbeklemesi için geliştirilmiş yöntemden detaylıca bahsedilmiştir. Bölüm 5 'te yapılan deneylerden bahsedilirken, bu bölüm iki ana başlık altında incelenmiştir. İlk kısmında küçük ölçekli veri kümeleri ile yapılmış deneylere değinilirken, ikinci bölümde büyük ölçekli veri kümeleri üzerine yapılan deneylerin sonuçları ortaya konulup çıkan tüm bu sonuçlar yorumlanmıştır. Sonuncu ve 6. Bölüm'de ise bu tez çalışmasında yapılan çalışma ve elde edilen sonuçlar özetlenip, tez sonlandırılmıştır.

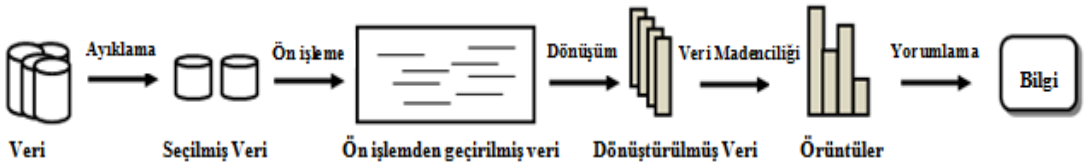
BÖLÜM 2

2. İLGİLİ ÇALIŞMALAR

Bu bölümde tez çalışmasında geliştirilen yöntemin temellerinde kullanılan kavramlar ile ilgili bilgiler verilecektir.

2.1. Veri Madenciliği

Kurumlar kendi verilerini depolamak ve depolanan bu verilerle analiz ve inceleme yapmak amacıyla pek çok değişik bilgi kaynağı kullanırlar. Verinin toplanıp, depolanması özellikle son yıllarda gelişen teknolojik imkânlar sayesinde artık daha kolay olmuştur. Bu sayede veriler çok büyük bir hızla toplanmaya başlanmış ve bunun doğal bir sonucu olarak, çok büyük boyuttaki veri bankaları ortaya çıkmıştır. Bu büyüklükte malzemeyi işlemek, mevcut istatistikî yöntemler ve veritabanı inceleme araçları ile artık çok zor hale gelmiştir. Bu duruma çözüm olarak ise devasa büyüklükteki veritabanlarını incelemekte başarılı olan ve bilgi keşif süreci adı verilen bir yöntem öne sürülmüştür. Veritabanlarındaki bilginin elde edilip işlenebilmesi için gerekli olan süreç veritabanı bilgi keşfi (VTBK) kavramı ile açıklanmıştır.[14] Şekil 2.1 VTBK sürecindeki temel aşamaları göstermektedir.



Şekil 2.1. Veri tabanlarında bilgi keşfi süreci (VTBK)

Şekilde de görülen ayıklama, ön işleme ve dönüşüm aşamaları ham veriyi veri madenciliği işlemlerinde uygun hale gelmesi için gereklidir. Çıktının yorumlanma kısmı ise analiz aşamasında ele alınmaktadır. Veri madenciliği kısmı, VTBK sürecinin temel aşamasıdır.

İstatistik, veritabanı, görselleştirme ve yapay zekâ gibi pek çok alanın kesişim noktası veri madenciliğidir. Daha önceden bilinmeyen yapıların büyük bir detay kümesinin içerisinden çıkartılıp keşfedilmesi için, büyük veri kümelerinin işlenmesini temel alan tüm metot ve yöntemlerin ortak buluştuğu nokta veri madenciliğidir.

Pek çok teknikten, veri madenciliğinin amacı için yararlanılabilir. İlişki kural madenciliği, sıralı kural madenciliği, sınıflandırma, kümeleme bunlardan bazılarına örnek olarak verilebilir. Bu tezdeki çalışmada, temel olarak ele alınan ise bir veri kümesinin en verimli şekilde nasıl öbeklendirileceğidir. Bu doğrultuda, öne sürülen yönteminin çözümünde genetik algoritma yapısı kullanılmıştır. Bu sebeple, bu bölümün geri kalan kısmında genetik algoritma mantığı ve bunun yanı sıra çözüm metodunda faydalanılan bazı metotlara değinilecektir.

2.2. Genetik Algoritma

Genetik algoritmanın temellerini, Michigan Üniversitesi'nde yaptığı çalışmalar ile John Holland atmıştır. Genetik algoritmalar evrimsel bir yaklaşım kullanmaktadır.[15] Bir genetik algoritmada, bir popülasyon, olası bir çözüm kümesi olarak kullanılır ve her çözümün bir uygunluk[3] değeri vardır. Popülasyon, her yeni jenerasyonda Darwin yaklaşımını kullanarak evrimleşir. Genetik algoritmalar özellikle arama ve en iyileme problemlerinde başarılı bir şekilde kullanılmaktadır. Optimal sonuçları bulma gibi avantajlara sahiplerdir. Bir genetik algoritma, her zaman en genel optimum çözümü bulamasa da, karmaşık arama uzaylarında optimum sonuca yakın sonuçlar bulmayı garanti eder. Koza, genetik algoritmanın karmaşık, lineer olmayan ve çok boyutlu uzaylarda verimli ve hızlı şekilde çalıştığını belirtmektedir.[16] Genetik algoritma ile çalışılırken en ilginç nokta ise problemin kapsam kümesinin veya uygunluk fonksiyonun nasıl çalıştığının bilinmesine gerek olmadığıdır.

[3] ing: *fitness*

Genetik algoritmanın her neslinde, popülasyonlar, üreme, eşeysel üreme ve mutasyon aşamalarından sonra, uygunluk fonksiyonuna göre eliminasyon aşamaları yer alır. Tüm bu aşamalar, gerçek hayattaki bir popülasyonun evrim aşamaları temel alınarak ortaya çıkarılmıştır. Popülasyondaki her birey, arama uzayındaki bir çözümü temsil eder. Uygunluk değeri, bireyin çözüm ile ne kadar uyumlu olduğunu gösterir. Biyolojik evrimden esinlenerek ortaya çıkarılan tüm işleçler, daha iyi bireyler ortaya çıkartmak amacıyla kullanılmaktadır. Doğal yaşam açısından düşünülürse, tüm döller çaprazlamalar sonucu ortaya çıkarken, mutasyona uğrama ihtimalleri vardır. Gerçek yaşamdaki bu süreç genetik algoritma mantığıyla tamamen paralellik göstermektedir.

Verilen probleme göre, tüm arama uzayı olası çözümler ile doludur. Kodlama ve değerlendirme genetik problem tarafından özel olarak incelenen iki temel kısımdır. Her bir birey, kromozom olarak adlandırılır ve sabit uzunlukta bir dizgi şeklinde tanımlanır. Bu dizgilere gen adı verilir. Genler geleneksel olarak bitler ile kodlanır, böylece bir kromozom (0, 1) bitlerinden oluşan dizgi haline gelir. Aşağıdaki algoritma, genetik algoritmanın genel bir sözde kodunu vermektedir.

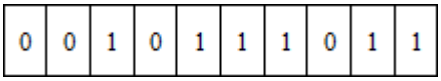
Algoritma 1 *Genetik Algoritma*

```
Popülasyonu oluştur.
Popülasyondaki bireylerin uygunluk değerini hesapla.
Sonlanma kriteri sağlanmadığı sürece
{
    Yeniden üretim için popülasyondan ataları seç
    Çaprazlamayı uygula
    Mutasyonu uygula
    Popülasyondaki bireylerin uygunluk değerini hesapla.
}
```

Algoritma 1' de, popülasyon oluşturulduktan sonra, ilk neslin bireyleri üretilmektedir. Popülasyon boyutu ve sonlanma kriteri, genetik algoritmanın iki temel parametreleridir. Sonlanma kriterine örnek olarak, en fazla iterasyon sayısı, uygunluk değerine verilecek bir eşik sınırı veya popülasyondaki en iyi veya en kötü uygunluk değerinin iyileşmemesi, örnek olarak verilebilir.

Genetik işlem, sonlanma kriteri sağlanmadığı sürece devam eder. Sağlanmadığı zaman, popülasyonun üyeleri yeniden üretim için seçilir ve bireyler yeniden üretim için çiftleşir ve döl bireylerini eşeysel eşleşme ve mutasyon sonucu oluştururlar. Daha sonra ise döl bireyleri ölçüme sokulurlar. Algoritma 1' in ana adamları ise aşağıdaki gibi anlatılmıştır.

İlk popülasyonu oluşturma: İlk adımda, popülasyon rastgele olarak üretilmiş bireylerden oluşur. Bir kromozomun olası kodlaması ikili tabanda olabilir. Şekil 2.2 de bu durumdaki bir kromozom görülebilir.



Şekil 2.2 Bir kromozomun ikili tabanda gösterimi

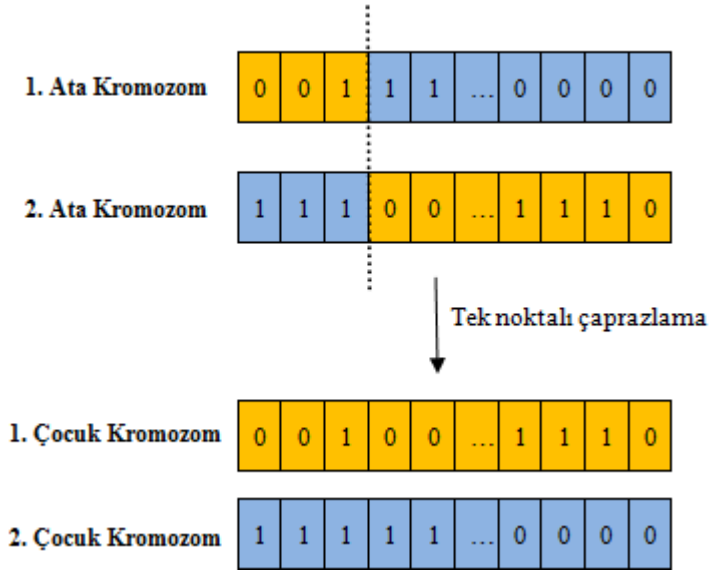
Popülasyondaki bireylerin uygunluk değerlerini hesaplama: Popülasyondaki her birey ve atalarından üretilmiş çocukları için bir hesap yapılır ve her bir bireye bir uygunluk değeri atanır. Bu değer bir bireyin çözüm kümesi içinde nasıl yer aldığını gösterir. Problemin tipi de dikkate alınarak, uygunluk değeri küçük oldukça, bir bireyin popülasyonda kalma şansı artar.

Yeniden üreme için popülasyondan ataların seçimi: Her jenerasyonda, popülasyondaki bireyler çiftleşir. Bireysel değerlendirme ve uygunluk değerleri, üreme için çiftleşmelerin seçiminde önemli bir rol oynar. Düşük bir uygunluk değeri olsa bile, bir bireyin sonraki nesil veya çiftleşme için seçilme ihtimali vardır. Ancak bunun olasılıksal olarak ihtimali düşüktür. Bu, arama uzayındaki çeşitlilik açısından olumlu bir durumdur, çünkü prematüre bireylerin popülasyon içerisindeki dominantlığı engellenmiş olur. Dairesel rulet veya turnuva türündeki seçimler bu konuda en yaygın kullanılan seçim yöntemleridir.

Dairesel rulet seçiminde, her bireye, uygunluk değerine bağlı olarak bir olasılık değeri atanır. Atalar, bu olasılık değerleri dikkate alınarak seçilir. Turnuva türündeki seçimde ise, belirli sayıdaki birey arasından çiftleşme işlemi için en uygun olanlar eş

olarak seçilir. Diğer eş de aynı şekilde seçilir ve belirlenen bu çift yeniden üretim için çiftleşirler.

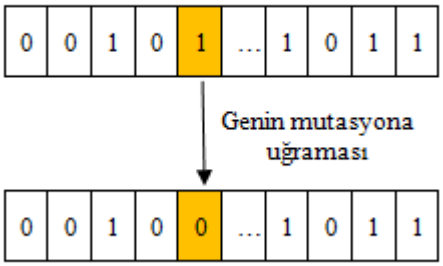
Yeniden birleştirmenin uygulanması (eşeysel çaprazlama): Çocuklar, atalar kullanılarak üretilir. Bu durum atalarının bir çaprazlama olasılığı ile çaprazlanması sonucu elde edilir. Ataların, çaprazlanmadığı durumlarda ise, çocuklar ataları ile aynı olurlar. Diğer türlü ise, kromozomların rastgele seçilen kısımlarını, birbiri ile değiştirirler. Bu çaprazlama operatörlerine pek çok örnek vardır: tek noktalı çaprazlama, çok noktalı çaprazlama ve tekdüze çaprazlama en bilindikleridir. Spears ve De Jong çaprazlama operatörleriyle ilgili çok geniş bir çalışma ortaya sunmuşlardır.[17] Çaprazlama işlemi en uygun alt çözümleri birleştirerek çözümlerin efektif olarak aranmasını sağlar. Bu tezde de kullanıldığı gibi, tek noktalı çaprazlama çok yaygın olarak kullanılan bir çaprazlama tekniğidir. Bu teknik, iki kromozomda rastgele seçilen bir ayırma noktası ile iki kromozomun ilgili kısımlarının yerini değiştirir. Bu işleme bir örnek Şekil 2.3 ‘ de yer almaktadır.



Şekil 2.3 Tek noktalı eşeysel çaprazlamaya örneği

Mutasyonun uygulanması: Çaprazlama işlemi sonrasında, üretilen çocukların mutasyona uğrama durumu olasıdır. Bir kromozomdaki genler olasılık dahilinde

mutasyona uğrar. Bu durum arama uzayında yerel veya genel olarak ilerlemeyi sağlar. Bir bireydeki bütün genlerin yine olasılık dahilinde değişme ihtimali vardır. Şekil 2.4 ‘ de de görüldüğü üzere, bir genin mutasyona uğraması sonucu, genin değeri 1’den 0’a doğru değişmiştir.



Şekil 2.4 Mutasyon örneği

2.3. Çok Amaçlı Yapı ile Optimizasyon

Bir çok amaçlı optimizasyon probleminin n adet karar değişkeni, k adet amaç fonksiyonu ve m adet kısıdı vardır. Amaç fonksiyonları ve kısıtlar, karar değişkenlerinin fonksiyonu şeklinde tanımlanır. Optimizasyon amacı aşağıdaki gibi tanımlanabilir:

$x = (x_1, x_2, \dots, x_n) \in X$ ve $y = (y_1, y_2, \dots, y_n) \in Y$ iken,

$e(x) = \left(e_1(x), e_2(x), \dots, e_m(x)\right) \leq 0$ için

$y = f(x) = \left(f_1(x), f_2(x), \dots, f_k(x)\right)$ ifadesini en aza indirge veya en yükseğe çıkar.

Burada x karar vektörü iken y amaç vektörüdür. X karar uzayını gösterirken, Y amaç uzayını temsil eder. $e(x) \leq 0$ kısıdı ise uygun çözümleri tanımlar.[18]

Çok amaçlı bir en iyileme probleminin çözümleri, baskın olmayan veya üstün noktalar türünden matematiksel olarak ifade edilir. Bir en aza indirgeme probleminde, $x^{(1)}$ şeklinde bir vektör, $x^{(2)}$ şeklindeki bir vektörden daha küçük

tanımlanır ve $x^{(1)} \prec x^{(2)}$ şeklinde gösterilir. Bunun karşılığı ise, $x^{(2)}$ ' nin hiçbir değeri $x^{(1)}$ ' den kısmen küçük değilse, $x^{(2)}$ ' nin en az bir değeri $x^{(1)}$ ' den büyük olur anlamına geldiği şeklindedir. Eğer $x^{(1)}$, $x^{(2)}$ ' den kısmen olarak küçükse, $x^{(1)}$, $x^{(2)}$ ' yi domine eder veya $x^{(2)}$, $x^{(1)}$ ' den daha değersizdir denir. Çok amaçlı bir en iyileme problemi, amaçların değişik kombinasyonlarına göre değişik çözümlerden oluşan bir çözüm kümesine sahiptir. En uygun çözümler, hiçbir çözümün kendisine üstünlük sağlayamadığı çözümler olarak tanımlanır.

Çok amaçlı en iyileme problemleriyle ilgili genel bir zorluk ise amaç fonksiyonları arasında olabilecek olası çakışmalardır. Mümkün olan çözümlerden hiç birisi tüm amaçlar için geçerli olacak en uygun çözümleri vermeyebilir. Pareto-optimal[19] çözümü ise en az amaç için çakışacağını garanti eder. Çok amaçlı en iyilemede, birden çok amaç tek bir amaç fonksiyonu oluşturmak için birleştirilir. En yaygın olarak kullanılan yöntemlerin birinde, her bir amaca ağırlık verilip, sonuç sayısallaştırılır. Böylece sonuç öznel olarak elde edilmiş olur.

2.4. Öbekleme

Öbekleme veri madenciliği tekniklerinden biri olup, literatürde üzerinde çok sayıda araştırma yapılmıştır. Verilerden oluşan bir kümeyi, öbek adını alan, anlamlı alt sınıflara ayırma işlemi olarak tanımlanabilir. Kullanıcılara, bir veri kümesinin yapısı hakkında bilgi verir. Öbekleme, açıklayıcı örüntü analizinde, gruplamada, karar verme yapılarında, makine öğrenmesini içeren ve bunlara benzer pek çok konuda çok kullanışlı olmaktadır.

Öbekleme denetlenmeyen bir sınıflandırmadır. Bunun anlamı ise, ilk başta ne kadar sınıfa ayrım yapılması gerektiği, oluşacak sınıflara ait özellikler gibi bilgiler olmamasıdır. Ayrıca öbekleme işlemi herhangi bir test örneğini ele almamaktadır. Oluşacak öbekler, istatistiksel olarak ve denetlenmeyen sembolik çıkarsama metotları kullanılarak oluşturulabilir. Bu sembolik metotlar birbirlerinden, kabul ettiği özniteliklerin türlerine (sayısal, sembolik ve yapısal nesneler), öbeklerin

gösterim şekline ve öbeklerin kurulumuna (hiyerarşik olarak veya yassı şekilde)[20] göre ayrılır. Tipik bir öbeklemenin bileşenleri şu şekilde özetlenebilir:[21]

1. Örüntü gösterimi
2. Örüntü yakınlığının tanımı
3. Öbekleme veya gruplama
4. Veri soyutlaması
5. Öbekleme değerlendirmesi

Bu bileşenler şu şekilde açıklanabilir:

Örüntü gösterimi, öbeklerin sayısıyla, mevcut örüntülerle ve öbekleme algoritmasına müsait özelliklerin tipi ve ölçeğiyle ilgilidir. Özellik çıkarımı, öbekleme işlemi için gerekli özellik kümesinin oluşturulmasıdır. Örüntü yakınlığı, öbeklemede kullanılan uzaklık fonksiyonudur. Uzaklık fonksiyonlarını tanımlayacak pek çok değişik yorum olabilir.[21] çalışmasında uzaklık fonksiyonları üzerine faydalı bir araştırma vardır. Gruplama aşaması ise değişik şekillerde tanımlanabilir. Verilerin öbeklemesi katı şekilde olacaksa, veri kümesindeki her eleman yalnızca bir öbeğe ait olmalıdır, başka öbeklerle hiçbir ilişkisi olmamalıdır. Bunun yanında dağınık öbekleme de olabilir. Bu durumda ise her elemanın herhangi bir öbekle olacak ilişkisi 0 ile 1 arasında değerler alır. Bu tezde katı şekilde öbeklendirme üzerinde durulmuştur.

Genel olarak, tüm öbekleme işlemi bazı temel adımları içerir.[14] Bunlar, özellik seçimi, belirli bir öbekleme algoritmasının çalıştırılması ve sonuçların doğrulanıp, değerlendirilmesi olarak özetlenebilir. Buna karşılık, literatürde en çok üzerinde durulan kısım hep öbekleme algoritması ve doğrulama işlemi olmuştur. Bu algoritmalar genelde 4 farklı kategori altında toplanabilir.[21]

Katı öbeklemede, N adet eleman ve k adet öbek için tüm olası öbeklemelerin sayısı aşağıdaki şekilde tanımlanabilir:[14]

$$\frac{1}{k!}\sum_{l=1}^{k}\binom{k}{k}(-1)^{k-1_{l^m}} \quad \approx k^n / k! \tag{2.1}$$

Bölümleme algoritmaları n adet nesneyi k adet öbeğe dağıtırlar. Bunun için bu tarz algoritmalar, ilk başta genel olarak rastgele şekilde olan ilk dağıtım yaparlar. Sonra, tekrarlı olarak nesneleri genel sonuç optimize edilecek şekilde öbekler arasında taşırlar. Ancak, bu tarz algoritmaların sıkıntısı ilk dağıtıma ve yerel en uygun değere çok bağımlı olmalarıdır. Sonuçta önsel bir bilgi genelde yoktur, bu sebeple gözlemlenen veri kümesi üzerinden öbek sayısını hesaplamak gerekir. Bu problem öbek doğrulama problemiyle alakalıdır ve henüz bir çözüm üretilememiştir. Bölümleme algoritmalarına örnek olarak k-means[22], PAM[23] ve CLARANS[24] verilebilir.

K-means öbeklemesini bilindik bir bölümleme öbeklemesi tekniğidir. Öbek sayısı k, algoritmaya girdi olarak verilir ve algoritma veriyi k adet altkümeye ayırır. Her elemanın ait olduğu öbeğin merkezi ile arasındaki uzaklığı hesaplanır. Ve tüm elemanlar için bu değerlerin toplamı en aza indirgenmeye çalışılır. Böylece öbekler optimize edilmeye çalışılır. Yerel en az değer bulunduğu zaman bu yöntemin tıkanma ihtimali vardır. Literatürde, k-means'in birçok çeşidi bulunmaktadır. Bu çeşitler üç farklı grup altında toplanabilir. İlk bölümlemenin seçilmesi, bölüp birleştirme işlemlerine izin verilmesi ve farklı öbekleme sonuçları için farklı ölçütlerin belirlenmesi bu 3 kategoriyi oluşturur.

Sıra düzenli öbekleme algoritması, sıra düzenli şekilde bulunan iç içe geçmiş öbekler oluşturur. Çıktı, grafiksel olarak dendogram şeklinde gösterilebilir. Sıra düzenli öbeklemenin böl-birleştir ölçütü vardır. Veri, özyinemeli olarak alt gruplara ayrılır. Bu tarz algoritmalara yığılımcı ve parçalayıcı olmak üzere iki farklı yaklaşım bulunmaktadır. Yığılımcı öbeklemede birbirine en yakın olan öbek çiftleri aşağıdan yukarıya doğru birleştirilir. Bölücü öbeklemede ise, tüm veri kümesi tek bir öbek gibi ele alınır. Daha sonra her adımda, veri kümesi tüm öbekler tekil olana kadar bölünür. Grafiksel olarak gösterim için doğal bir yol sağlasa da, yüksek hesaplama karmaşıklığına karşı dayanaklı değildir. Ayrıca açgözlü yapı, kötü kararların sonraki

aşamalarda düzeltilme imkânını ortadan kaldırır. CURE[25] ve ROCK[3] sıra düzenli öbekleme yapan iki önemli algoritmadır.

Teorik çizge öbeklemesi, G = (V, E) şeklinde bir yakınsama çizgesi ile yapılır. Çizgedeki her veri nesnesi bir köşe olarak temsil edilir. Kenarlar ise, tüm köşeler arasında yer alabilirken, köşelerin yani verilerin birbirleriyle olan uzaklıklarını gösteren ağırlık değerlerinden oluşur. CLICK[26] ve CAST[27] bu teorik çizge öbeklemesini kullanan örneklerdendir.

Yoğunluk tabanlı öbekleme, yoğunluk bölgelerini genişletmeye çalışırken, öbeklemeyi yoğunluğa dayanarak yapar. Yüksek yoğunluğu gürültüye sahip düşük yoğunluktan iyi ayırt eder. Ancak zaman karmaşıklığı açısından verimli değildir. DBSCAN[28], DENCLUE[29] ve OPTICS[30] yoğunluk tabanlı öbekleme algoritmalarına örnek olarak verilebilir.

Model tabanlı öbekleme, istatistiksel bir taslağı, öbek yapısını modellemek için kullanır. Beklenti en yükseklemesini kullanarak kümelerin yoğunluk fonksiyonlarına ait parametreleri hesaplar.

Alt uzay öbeklemesi geleneksel öbeklemenin bir uzantısıdır. Aynı veri kümesinde yer alan ancak değişik alt uzaylarda yer alan öbekleri bulmaya çalışır. Örtüşen öbeklere izin verir. PROCLUS[31], bu tarz metoda örnek olarak verilebilir.

Öz organize harita tek tabakalı sinirsel ağları girdi olarak alır ve çıktı olarak nöronları sunar.[32] İki boyutlu bir ağ üzerindeki nöronların her biri ağırlıklandırılmış bir referans vektörü ile ilişkilendirilir ve her gözlem kendisine en yakın referans vektörü yardımıyla bir nöron ile eşleştirilir. Referans vektörler başlangıçta rassal olarak ağırlıklandırılabilir.[32] Her gözlem için kendisine en yakın ağırlığa sahip bir vektör seçilir. Seçilen vektör daha sonra dağılıma en uyumlu hale gelecek şekilde güncellenir. Tüm gözlemler için bu işlemler tamamlandıktan sonra her bir gözlem, çıktı nöronları ya da kümelerle eşleştirilir. Sezgisel olarak çok boyutlu verileri eşleştirir ve ayrıca benzer kümeleri birbirine yakın yerleştirir. Gürültüye k-means yöntemine göre daha az duyarlıdır. Ancak, kullanıcının küme

sayısını ve nöron haritasındaki ağ yapısını girmesini zorunlu kılar. Ayrıca fazla sayıda ilgisiz gözlem içeren veri kümeleri için verimli çalışan bir yöntem değildir.[32]

2.4.1. Öbeklemedeki bazı anahtar noktalar

Önceden de belirtildiği gibi öbekleme literatürde çok genişçe araştırılmış ve üzerine çalışmalar yapılmaya devam edilen bir konudur. Sahip olduğu ve her geçen gün ihtiyaca yönelik genişleyen uygulama yelpazesi sebebiyle çok aktif bir araştırma alanıdır. Ayrıca yukarıda da anlatılan yaklaşımların hepsi bile bazı beklentileri karşılayamamaktadır. Yeni geliştirilmekte olan öbekleme algoritmalarının dikkate aldığı anahtar noktalardan bazıları, ayrılacak öbek sayısının otomatik olarak belirlenmesi, verilen bir veri kümesi için en doğal öbeklemenin gerçekleştirilmesi ve büyük ve yüksek boyutlu veri kümelerine karşı ölçeklenebilir olmak şeklinde sıralanabilir.

Öbek sayısının elle sabit şekilde belirlenip algoritmanın çalıştırılmasından ziyade, bu işlemi otomatik yapmanın önemi yadsınamaz. Öbekleme denetlenmeyen bir öğrenme işlemi olduğundan, öbek sayısının otomatik olarak bulunması en doğal yaklaşım olacaktır. Öbekleme ve sınıflandırma arasındaki temel fark budur. Sınıflandırma, önceden belirlenen sınıflar ile yapılan denetlenen bir öğrenme işlemidir. Örneğin, k-means algoritmasını pek çok öbek sayısı için çalıştırıp aralarından en uygun çözümü seçmek bir çözüm gibi gözükse bile bu algoritmanın ilk başta rastgele olarak belirlenen öbek merkezlerine göre işlem yapıyor olması çıkacak sonuçların her seferinde farklı çıktı vermesine sebep olabilmektedir. Araştırmacılar, k-means'in rastgele merkez belirleme sıkıntısına çözümler bulabilmek için halen çalışmaktadırlar.

Bazı araştırmacılar geleneksel genetik algoritma mantığını öbekleme ile ilişkilendirmişlerdir. Ancak, çok amaçlı genetik algoritmalar, öbekleme işlemine daha uygundur. Sonuçta öbekleme de birden çok amaca dayanarak işlem yapmaktadır. Genetik algoritmalar, küçük boyutlu veri kümeleri için tatmin edici

sonuçlar vermektedir. Ancak, pek çok çalışma da belirtildiği üzere performans ve ölçeklenebilirlik açısından sıkıntıları vardır.

Sonuç olarak, önerilmiş öbekleme algoritmaları da dikkate alarak öne sürülebilecek sorular, bir öbekleme algoritmasının değişik öbek sayıları için ne kadar verimli sonuçlar ürettiği, ölçeklenebilirlik durumu ve öbek sayısının nasıl belirlendiği olabilir.

2.4.2. Öbeklemenin doğrulanması

Önerilmiş pek çok öbekleme algoritması vardır ve her birisi verinin kümelenme işlemini ayrı şekillerde yapar. Ancak en son hepsi için ortak olan konu, işlem sonucu çıkacak en uygun öbek sayısı ve öbeklemenin gerçekten verimli yapılıp yapılmadığıdır.[33] Öbek doğrulama indislerinin kullanılması sayesinde, öbekleme sonuçları arasında sıralamalar yapmak mümkün olmaktadır.

Öbek doğrulaması, söz konusu veri kümesi için en uygun parçalamanın nasıl olması gerektiğinin anlaşılması için kullanılır. Bu konudaki amaç, en uygun öbek sayısının bulunmasıdır. Eğer veri kümesi biliniyorsa, öbekler içerisinde hangi adayın belirleyici olduğunun anlaşılması için, her bir alternatif adayı farklı parametrelerle değerlendirerek en uygun küme sayısını kontrol etmek için kullanılır. Bu da doğrulama indislerinin kullanımı ile sağlanır. Bu indisler, dışsal, içsel ve göreli olarak öbeklendirir. Dışsal ve içsel indisler, yüksek hesaplama maliyetine sahip istatistiksel testlere dayanır. İçsel doğrulama indisler, verinin miktarının ve özelliklerini kullanır.

Her doğrulama indisi, belirli bir formülden yararlanır. Öbekleme parametrelerini, alternatif çözümleri kontrol etmek ve değerlendirmek için kullanır. Henüz indisleri sınıflandıran ve hangi indisin hangi veri kümesi içi uygun olacağını belirleyen bir çalışma bulunmamaktadır. Bu durumun üstesinden gelip, doğrulama sonuçlarının güvenilirliğini arttırmak için, çoğunluk oylaması yöntemi kullanılmaktadır.[33] Bu

yöntem güvenilirdir çünkü tek bir bakış açısından ziyade değişik bakış açılarından bakarak alternatif çözümleri değerlendirme imkânı sağlar.

BÖLÜM 3

3. GENETİK ALGORİTMA İLE ÇOK AMAÇLI ÖBEKLEME

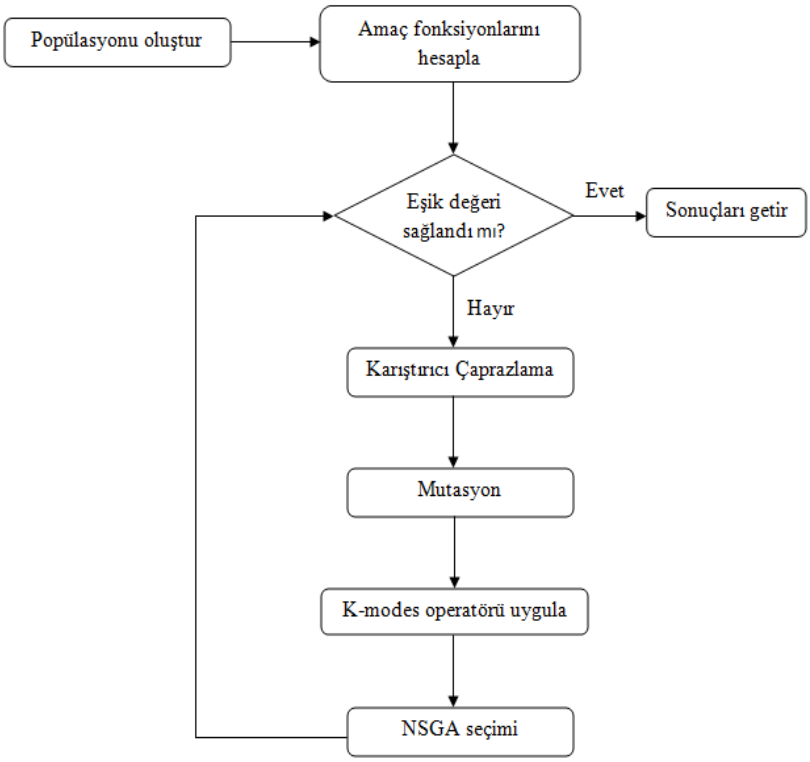
3.1. Genel Bakış

Çok amaçlı genetik algoritma ile kategorik verilerin öbeklendirilmesi, geleneksel genetik algoritma mantığı ile paralellik gösterir. Şekil 3.1 ' de bu yöntemin akış şeması basitçe gösterilmektedir. Bu tezde geliştirilen algoritmaya göre kromozom sayısı P olarak ayarlanmıştır, yani algoritma çalışmadan önce kromozom sayısı sabittir. Kromozom sayısı, kromozom ve mutasyon değerleri ile birlikte P olarak ayarlanmıştır. Geliştirilen algoritma şu şekilde çalışmaktadır: Kromozomlar öncelikle karıştırıcı çaprazlama ve mutasyon işlemlerine sokulur. Daha sonra ise aynı kromozomlar üzerine k-mod operatörü çalıştırılır. Bu operatörün her nesil için çalıştırılmasının sebebi kromozomlar için çabuk yakınsama sağlamasıdır. Bu operatör basit anlamda, tüm veri elemanlarını, uygunluk değerlerini dikkate alarak, en yakın oldukları öbek merkezine göre yeniden öbekler. Aslında k-means ile aynı mantıktadır, tek farkı kategorik veri üzerine uygulanmasıdır.

Sonuç olarak, elde 2xP adet kromozom olur ve NSGA[13] (Baskın edilemeyen sıralama) algoritması kullanılarak aralarından en iyileri seçilir. Bu algoritma temel olarak baskınlık kurallarına dayanarak çalışır. Genel olarak çok amaçlı bir en iyileme problemi [18] çalışmasındaki gibi tanımlanabilir.

Formülasyona bağlı olarak, bir y vektörünün optimizasyonu, çok amaçlı en iyileme yapısını göstermeyi amaçlayan fonksiyonların vektörüdür şeklinde açıklanabilir. e(x) vektörü, olası çözümleri belirlemek için gerekli kısıt değerlerini tutar. Burada x vektörü problemin tanım kümesini gösterirken, y kümesi ise aynı problemin menzilini tanımlar. NSGA algoritması, alternatif çözümlerin pareto olarak sıralanması için kullanılır. Örneğin, probleme çözüm olarak y_1 ve y_2 verilsin. Eğer y_1'in tüm amaç fonksiyonu sonuçları y_2'ninkilerden iyiyse, y_1'in y_2'ye göre baskın

olduğu söylenir. Ters durumda y_2, y_1'e göre baskındır denirken, diğer tüm durumlar için aralarında kıyas yapılamaz denir. Bu durumda iki çözüm de aynı pareto seviyesinde yer alır.



Şekil 3.1 Kategorik verilerin öbeklenme işlemini sağlayan çok amaçlı genetik algoritmanın akış şeması

NSGA sıralaması sonuçları, katman katman verir. Her katmanda, birbirine üstünlük sağlayamayan sonuçlar yer almaktadır. Bir sonraki katmanda bulunan herhangi bir sonuç, önceki katmandaki tüm sonuçlara göre daha değersizdir.

Genetik algoritmanın sonlanma koşulu olarak pek çok durum kullanılabilir. Mesela, sabit bir nesil sayısı girmek bunlardan birisi olabilir. Ancak bu durumda, genetik algoritmanın, kromozomlar yakınsamadan sonlanma riski bulunur. Başka bir seçenek, amaç fonksiyonları daha fazla optimize edilemediği zaman algoritmayı sonlandırmak olabilir. Bu tezde, kromozom sonuçlarını yakınsadığı zaman,

algoritmanın otomatik olarak sonlamasının sağlayacak ikinci yöntem tercih edilmiştir.

Çok amaçlı genetik algoritmanın sonuçları alından sonra, NxN boyutlu bir M matrisi oluşturulmaktadır. Burada N veri kümesindeki eleman sayısını göstermektedir.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix}$$

Şekil 3.2 Genetik algoritma sonucu ortaya çıkan örnek bir matris

Şekil 3.2 ‘ dek matrisde her bir x_i değeri, i ve j elemanlarının birlikte bulunma sayılarını verir. Çok amaçlı genetik algoritmanın sonuçları, veri içerisindeki eleman ikililerinin kaç öbek içerisinde birlikte yer aldığı bilgilerinin bir özetini verir. Alternatif sonuçları kullanmakta olan, öbekleme sonuçlarının bir kümesi, herhangi bir amaç fonksiyonundan ödün verilmeden elde edilir. Kısacası bu matrisin gösterdiği şey, düğümleri arasındaki linklerin, bu düğümlerin birbiriyle olan güçlülük ilişkisini gösterdiği tam bir çizgedir.

3.2. Algoritmanın Açıklanması

3.2.1. Kromozom kodlaması

Bu çalışmada kromozomlar, bit gösterimi ile tutulmaktadır. Her bir bit rastgele olarak tutulmaktadır. Bir D veri kümesi için, n adet eleman olsun, bu durumda her kromozom, k biti n kere içerir.

1	**0**	**0**	1	1	1	…	…	…	**1**	**0**	**1**

Şekil 3.3 Kromozom gösterimi

Bu çalışmada, öbek sayısının değeri otomatik olarak ayarlanmaktadır. Normalde, $\sqrt{n}$ adet öbek oluşturulması gerektiği düşünülmüştür ve bir kromozomdaki bit sayısı bu değeri ortaya çıkarmak için yardımcı olur. Geliştirilen yaklaşımda, öbek sayısının iterasyonlar sırasında değişikliğe uğraması söz konusudur. Öbek sayısı için sabit bir değer atanmamaktadır. Verilen bir $D = x_1, x_2, ..., x_n$ veri kümesi için, $\sqrt{n}$ değeri hesaplanır ve log $|\sqrt{n}|$ adet bit ayrılır. Bu sonuç öbek sayısını gösteren k değerini vermektedir.

3.2.2. Karıştırıcı Çaprazlama

Bit gösterimi üzerine karıştırıcı şu şekilde çalışmaktadır: iki farklı kromozomdan yeni bir çocuk kromozom üretilmektedir. Bu rastgele seçilen bir çaprazlama noktası üzerinden yapılır. Daha sonra her nesilde, kromozomlar bu nokta üzerinden birleştirilir.

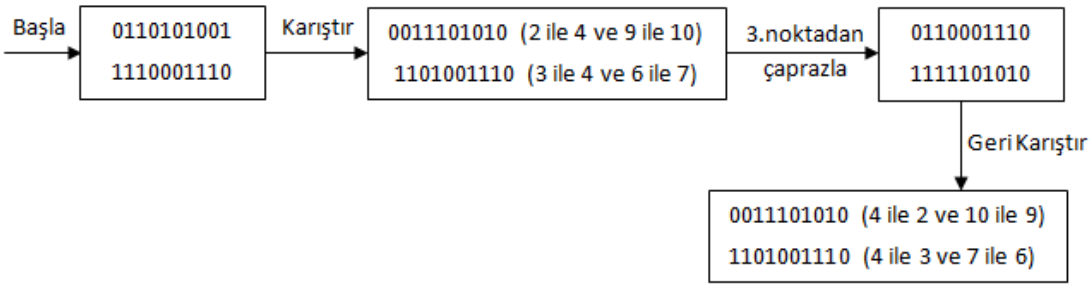
Karıştırıcı çaprazlama işlemi, ata kromozomlar çiftleştirilmeden önce bir karıştırma aşamasına sahiptir. Bu aşama, bu yöntemi klasik çaprazlama aşamasından farklı kılar.

Önceden de değinildiği gibi, her kromozom, veri kümesindeki elemanların atanacağı öbek sayısını belirlemek için bazı bitlere sahiptir. Her bir kromozom için karıştırma aşamasında, iki bit rastgele olarak seçilir ve yerleri değiştirilir. Karıştırma aşaması veri kümesindeki elemanların sayısı kadar tekrar edilir. Daha sonra kullanılmak üzere, her bir yer değiştirme işlemi (hangi kromozom için hangi bitlerin yer değiştirdiği) bir dizide saklanır.

Karıştırma aşaması tamamlandığında, kromozomlar çiftleştirilir ve bu çiftleşme sonucu yeni kromozomlar üretilmiş olur.

Tüm kromozomlar üretildikten sonra, geri karıştırma isimli bir aşama oluşan tüm bu kromozomlar üzerine uygulanır. Bu, önceden yapılan karıştırma işleminin tam tersidir. Önceden dizide saklandığı söylenen yer değiştirme bilgileri bu aşamada

kullanılır. Her kromozom için yeri değiştirilmiş olan bitler, bu aşamada tekrar eski yerlerine getirilir. Şekil 3.4 ‘ deki görsel anlatımla bu kısım daha anlaşılır olacaktır.



Şekil 3.4 Örnek bir karıştırıcı çaprazlama işlemi

Klasik çaprazlama işleminde, belli sayıda uygulanan iterasyonlar sonucu bazı nesillerin sürekli üretilme ihtimali olmaktadır. Ancak uygulanan karıştırma işleminde, her seferinde yeni nesiller üretilme ihtimali çok artmaktadır.

Öbek sayısının bu şekilde belirlenmesindeki ve öbek değerlerinin ikili tabanda gösterilmesinin temel amacı, genetik algoritmanın karıştırma kısmının daha verimli çalışmasını sağlamaktır.

3.2.3. Amaç Fonksiyonları

Veri kümelerinin öbeklendirilmesi için 3 farklı amaç fonksiyonu kullanılmıştır. Bunlar, k-mod içsel uzaklık[1], k-mod dışsal uzaklık[2] ve EWCD[9] fonksiyonlarıdır.

3.2.3.1. EWCD

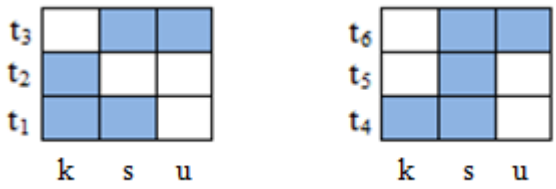
Bu amaç fonksiyonu, işlemsel bir veri kümesi için, bölümleme tabanlı bir öbekleme yapan bir algoritma kullanır. Bu algoritma, koyabildiği kadar sık elemanı öbeklere atamaya çalışır. Veri kümesindeki elemanların öznitelik değerlerinin, farklı öbekler için çakışması olası bir durumdur. Bu algoritma, amaç fonksiyonu değerinin yardımı

ile tüm bu durumları dikkate alır. Bu fonksiyonun çalışma mantığı aşağıdaki örnek ile daha iyi anlaşılacaktır;

Bir işlemsel veri kümesine örnek olarak, bir maketteki farklı alışveriş sepetleri verilebilir. Aşağıdaki kümelerin (t_1, t_2, t_3 – t_4, t_5, t_6) iki farklı marketteki, değişik müşterilerin sepetleri olduğunu varsayılırsa, Şekil 3.5'deki gibi bir görselleştirme yapılabilir;

t_1 = {kola, süt} t_2 = {kola} t_3 = {süt, su} 1. market

t_4 = {kola, süt} t_5 = {süt} t_6 = {süt, su} 2. market



Şekil 3.5 EWCD ile iki farklı öbek

Yukarıdaki şekilde yer alan iki farklı matrisde, 6 farklı alışveriş sepeti bulunmaktadır. Her bir nesne bir öznitelik olarak düşünülürse, 3 farklı öznitelik vardır(kola, süt, su). Ayrıca her market bir öbek olarak düşünülürse, iki farklı öbekleme mevcuttur. Hangi öbeklemenin daha verimli olduğu ise WCD metriği ile ölçülebilir;

$$\mathbf{WCD}(\mathbf{C_k}) = \frac{\sum_{j=1}^{\mathbf{M_k}} \mathbf{frekans}(\mathbf{I_{kj}})^2}{\mathbf{S_k \, x \, N_k}} \qquad (3.1)$$

Bu formülde C_k değeri k numaralı öbeği, I_{kj} , k numaralı öbekteki j. özniteliği, S_k değeri ise k numaralı öbekte yer alan tüm özniteliklerin, toplam bulunma sayılarını verirken, N_k, k numaralı öbeğe atanmış toplam eleman sayısını gösterir.

Bu bilgiler dahilinde şekildeki iki öbek için, WCD değerlerini şu şekilde hesaplanabilir:

İlk öbekde (soldaki matris), kola ve süt 2şer kere, su ise 1 kez geçmektedir. Bu öbek için $S_k = 5$ olur. Formüldeki payda değeri

$\sum_{j=1}^{M_k} \text{frekans}(I_{kj})^2 = 2^2 + 2^2 + 1^2 = 9$ olur.

3 farklı sepet olduğu için, $N_k = 3$ olur. Payda değeri = 5 x 3 = 15 olur. Bu durumda 1. cluster için WCD değeri 9 / 15 çıkar.

İkinci öbek için ise, kola ve su 1 kez yer alırken, süt 3 kere bulunmaktadır. Bu cluster için de S_k değeri 5 olur. Paydadaki toplam kısmı ise ilk öbek için yapılan hesaba benzer şekilde, bu kez, 11 değerini almaktadır. Yine 3 farklı sepet olduğu için, N_k , 5 değerini almaktadır. Tüm bu hesaplarla sonuç yani ağırlıklı kapsam yoğunluğu değeri 11 / 15 çıkar.

İkinci öbek için daha iyi bir sonuç almak beklenen bir durumdur. Şekle bakılırsa, ikinci öbekte süt özniteliği 3 sepet için de ortaktır. 3 farklı elemanı, üçünde birden ortak bir öznitelik olmadan öbeklendirmektense, bu 3 elemanı ortak bir öznitelik ile (süt) öbeklemek daha mantıklı olmaktadır. Bu daha öncede değinilen, ağırlıklı kapsam yoğunlu fonksiyonun, koyabildiği kadar sık elemanı bir öbeğe atama özelliğinin bir sonucudur.

Yukarıda anlatılan, ağırlıklı kapsam yoğunluğu tabanlı öbekleme ölçütüdür. $C^K = C_1, C_2, \ldots\ldots, C_K$ gibi bir öbekleme sonucunu ölçmek ve değerlendirmek için beklenen ağırlıklı kapsam yoğunluğu isimli amaç fonksiyonu kullanılmaktadır. Genetik algoritma bu fonksiyonun değerini en yükseğe çekmeye çalışır.

Genetik algoritmanın başında ilk popülasyon oluşturulurken, beklenen ağırlıklı kapsam yoğunluğu mantığı kullanılarak ilk kromozomların oluşturulması sağlanmıştır. Bu işlemin asıl amacı, genetik algoritmaya girecek ilk kromozom dizilerinin rastgele değil, daha güvenilir bir şekilde oluşturulmasıdır. Bu ilk dizi

oluşturma işlemi, klasik yöntemlerde, rastgele olarak yapılmaktadır. Ancak bu çalışma sırasında, rastgele olan yöntem ile diziler oluşturduğu zaman, amaç fonksiyonunun skorlarının düşük olduğunu gözlemlenmiştir. Bu durumun önüne geçmek için, beklenen ağırlıklı kapsam yoğunluğu fonksiyonu yardımı ile ilk kromozomlar oluşturulmuştur. Bu yöntemde izlenen işlem aşamaları şu şekilde özetlenebilir:

Girdi:

C: n adet öbek için öbek kümesi (C_1,C_2,…,C_n) (bu kümede 1' den n' e kadar sayılar bulunur. Mesela 1 değeri, 1. öbeği temsil eder)

S: Veri kümesinde bulunan her eleman için değerler içeren kromozom dizini. Bu dizinde, ilk olarak tüm değerlere boş öbek değeri olan 0 atanmıştır. Daha sonra bu 0 değerleri, ilgili eleman hangi öbeğe atanmışsa o öbeğin numarasıyla değiştirilir. Örneğin 2 öbeğe ayrılmış ve 3 farklı eleman içeren bir veri kümesi için bu dizi {1,1,2} şeklinde olabilir. Burada ilk iki eleman 1. öbeğe, 3. eleman ise 2. öbeğe atanmış demektir.

Algoritma:

S ' deki her t_i elemanı için, C ' deki tüm öbek değerlerini dene. Bu öbek değerlerinden S ' in o anki beklenen ağırlıklı kapsam yoğunluğu değerini en yüksek yapan c_j öbek değerini, t_i nin öbek değeri olacak şekilde ata.

Her bir elemanın, uygun öbek numarasını bulmak için, BAKY değeri her seferinde en baştan hesaplanmaz. Sonuçta ilk başta, boş öbek numarası değerleri atanmışken oluşmuş bir BAKY değeri vardır. Bu bilgi kullanılarak, her bir yeni öbek değeri denendiğinde, bu yeni değerin, BAKY değerini ne kadar azaltıp, arttırdığı anlaşılabilir. Bu mantıkla, BAKY değerini en fazla yükseltmiş olan öbek değeri, ilgili elemana atanabilir. Bu metoda, ekleme arttırması adı verilir.

$C^K = C_1, C_2, \ldots\ldots, C_K$ gibi bir öbekleme sonucu için (K < N), BAKY aşağıdaki formüldeki gibi hesaplanabilir:

$$\mathrm{EWCD}(C^K) = \frac{1}{N} \sum_{k=1}^{K} \frac{\sum_{j=1}^{M_k} frekans(I_{kj})^2}{S_k}$$

(3.2)

Burada M_k, k numaralı öbekteki birbirinden farklı öznitelik sayısını gösterirken, I_{kj}, k numaralı clusterdaki j. özniteliği belirtir. S_k ise, k öbeğinde bulunan tüm özniteliklerin sayılarının toplamıdır.

Kullanılan diğer iki amaç fonksiyonu ise, k-mod algoritmasının iki varyasyonudur. Bu iki fonksiyon anlatılmadan önce şöyle bir tanım yapılması konunun anlaşılması adına yararlı olacaktır;

n adet elemandan oluşan bir veri kümesi D olsun. D = $\{x_1, x_2 \ldots x_n\}$ için, x_i, D ' deki i. elemanı gösterirken, A= $\{A_1, A_2 \ldots A_m\}$ şeklinde, her x_i için m adet öznitelik vardır. A_j'nin j. attribute olduğu düşünülürse, her bir A_j birbirinden farklı değerler alabilmektedir. V_i ise i. attribute'un alabileceği bu değerleri tanımlayan değer kümesi iken, V_i={ $V_{i1}, V_{i2}, \ldots, V_{is}$ } şeklinde tanımlanır. Bu kümede V_{jk} ,j. attribute'un k. değeridir.

$f(V_{ij})/|D|$ oranı, V_{ij} değerinin D veri kümesinde kaç kere geçtiğini gösterirken, $f(V_{ij}/|C_i|)$ oranı ise V_{ij} değerinin C_j veri kümesinde kaç kere geçtiğini belirtir. $f(|C_i|)$ ise j. clusterda, toplam eleman sayısını verir.

3.2.3.2. K-mod içsel uzaklık

x_i ve x_j , D'deki i. ve j. elemanlar olsun.Veri kümesindeki, her x_i, x_j ikilisi için, Aj özniteliği üzerinden bir mod değeri hesaplanır. Eğer x_i ve x_j için , A_j özniteliğinin değeri aynı ise, iki eleman arasındaki uzaklık değeri değişmez. Ancak, farklı ise iki eleman arasındaki uzaklık değeri 1 arttırılır.

Bu tanımlar aşağıdaki şekilde formülize edilebilir;

$$\mathbf{d}(\mathbf{X},\mathbf{Y}) = \sum_{\mathbf{j=1}}^{\mathbf{m}} \boldsymbol{\delta}(\mathbf{x_j},\mathbf{y_j}) \tag{3.3.1}$$

$$\boldsymbol{\delta}(\mathbf{x_j},\mathbf{y_j}) = \begin{cases} \mathbf{0} & \mathbf{if}\ \mathbf{x_j} = \mathbf{y_j} \\ \mathbf{1} & \mathbf{if}\ \mathbf{x_j} \neq \mathbf{y_j} \end{cases} \tag{3.3.2}$$

C' nin, her j özniteliği için, d_j uzaklık değeri elde edilir. Tüm bu d_j değerlerinin toplamı ise bu D veri kümesinde, C öbeği için uzaklık değerini vermektedir. Daha sonra tüm C öbekleri için uzaklık toplamı, tüm veri kümesi için öbekleme sonucundaki toplam uzaklık değerini verir.

Anlaşılacağı üzere, genetik algoritmanın amacı bu değeri en aza indirgemektir. Sonuçta bu değer en aza indirgendiği zaman, D 'deki tüm elemanlar, bulundukları öbeklerin merkezine doğru sıkışmış olmaktadır. Bu diğer bir ifadeyle, her bir elemanın, diğer öbeklerin merkezlerinden uzaklaştığı anlamına gelmektedir.

3.2.3.3. K-mod dışsal uzaklık

Bu fonksiyon da, k-mod içsel uzaklık ile benzer mantıkta çalışmaktadır. Tek fark ise uzaklık hesabının, elemanlar üzerinden değil, öbekler üzerinden yapılmasıdır. Bir öbeğin, diğer öbeklerle olan uzaklığı, iki öbeğin modları arasındaki fark ile belirlenmektedir.

İki öbeğin, mod değerleri aynı ise uzaklık değeri değiştirilmezken, farklı olması durumunda uzaklık 1 arttırılır.

Bu sefer ise, bu değerin en yükseğe çıkarılması amaçlanmaktadır. Sonuçta bu değerin artması demek, öbeklerin birbirinden daha ayrık olacağı anlamına gelir. Bu şekilde bir öbekleme sonucunun ideal olduğu açıktır.

3.2.3.4. Sonuçların h-confidence ile birleştirilmesi

Çok amaçlı öbekleme sonuçları aslında, bir sonuç kümesi topluluğunu temsil eder. Verilen D = {x_1, x_2 ,...,x_n} şeklinde bir veri kümesi için, frekans(x_1, x_2 ,...,x_n) ' in, x_1'den, x_n'e kadar olan elemanların birlikte aynı öbekte yer alma sayıları olduğu düşünülürse, h-confidence değeri aşağıdaki formül ile hesaplanabilir;[34]

$$\boldsymbol{h(x_1, x_2, \dots, x_n)} = \frac{frekans(x_1,x_2,\dots,x_n)}{\text{maksimum } (frekans(x_1),frekans(x_2),\dots frekans(x_n))} \quad (3.4)$$

Bu çalışmada, h-confidence bilgisi, veri gruplarını, tüm veri kümesi tek bir öbeğe atanana kadar, alttan yukarıya doğru birleştirmek için kullanılmıştır.

H-confidence yöntemi sadece küçük boyuttaki çok karmaşık olmayan veri kümeleri sınıflandırılırken kullanılmıştır. Sonuçta, yüksek ölçekli veri kümeleri için, veri ağacının boyutu ve derinliği çok fazla artmaktadır.

Bu yöntemdeki temel mantık, her seviyesi belli öbek numaralarını temsil edecek bir ağacı kökten, dallara doğru oluşturmak ve sonuçları buna göre yorumlamaktır.

3.3. NSGA

Bu çalışmada, çok amaçlı öbekleme yapan genetik algoritmanın geliştirilmesi için, son yıllarda bu konuda oldukça popüler olan bir yöntemden de faydalanılmıştır. Baskın olmayan sıralama genetik algoritması, içerisine aldığı kromozom dizinindeki kromozomların birden çok amaç fonksiyonu değerini kullanarak, bir sonraki nesle aktarılacak kromozomların belirlenmesini sağlar. Aşağıda bu çalışmada kullanılmış olan NSGA ' nın sözde kodu verilmektedir.

NSGA Sözde Kodu

1: **Girdi**: sonuçKümesi (Sonuç kromozomları)

2: **Çıktı**: Her bir elemanın katman numarasını tutan L dizisi. Örn. L[0] = 1 ise, 1. eleman 1. katmandadır anlamını taşır.

Her bir katmanda kaç adet eleman olduğunu tutan R dizisi. Örn. R[0] = 10 ise, 1. katmanda 10 eleman vardır anlamını taşır.

enYüksekKatman değeri.

3: anlıkElemanNo = 0;

4: anlıkKatmanNo = 0;

5: enYüksekKatmanDegeri = 0;

6: Her i elemanı için, sonuçKümesi[i].durum = işlenmedi.

7: **while (**işlenmemiş en az bir eleman olduğu sürece**) do**

8: anlıkKatmanNo++;

9: **for** (i = 0; i <= elemanSayısı; i++) **do**

10: **if** (sonuçKümesi[i].durum == işlenmiş) **then**

11: continue;

12: **endif**

13: **for** (j = 0; j <= elemanSayısı; j++) **do**

14: **if** ((i == j) veya (sonuçKümesi[j].durum == işlenmiş)) **then**

15: continue;

16: **endif**

17: karşılaştırmaDegeri = Karşılaştır(sonuçKümesi[i], sonuçKümesi[j]);

18: **if** (karşılaştırmaDeğeri == 0) **then**

19: sonuçKümesi[j].hal = domine ediliyor;

19: **else if** (karşılaştırmaDeğeri == 1) **then**

20: **if** (sonuçKümesi[i].hal ≠ domine ediliyor) **then**

21: sonuçKümesi[j].hal = belirsiz;

22: **endif**

23: **else if** (karşılaştırmaDeğeri == 2) **then**

```
24:                 sonuçKümesi[i].hal = domine ediliyor;
25:                   break;
26:               endif
27:          endfor
28:          if (sonuçKümesi[i].hal ≠ domine ediliyor) then
29:               sonuçKümesi[i].durum = işlendi;
30:               L[i] = anlikKatmanNo;
31:               anlikElemanNo++;
32:           endif
33:   endfor
34: endwhile
```

Algoritmadan da anlaşılacağı üzere, öbeklendirilmek istenen kromozomlar bu algoritmaya girdi olarak yollanır. Her bir kromozomun üzerinde önceden hesaplanmış amaç fonksiyonu değerleri vardır. Bu algoritmanın amacı ise, kendisine yollanan bir dizi kromozomdan, en iyilerini bir sonraki iterasyona sokmak için ayıklamaktır. Bu mantıkla algoritma sonucunda, yollanan kromozomlar, katmanlara ayrılmış olurlar. Birbirine üstünlük kuramayan kromozomlar ise aynı katmanda yer alırlar.

Ana döngü, tüm kromozomlar içerisinde, en az bir tanesi bile işlenmemişse sürekli devam eder. İlk başta işlenmemiş bir kromozom bulunur ve geri kalan kromozomlarla karşılaştırılır. Eğer ana kromozom, karşılaştırıldığı tüm kromozomlara göre baskınsa veya eşit durumdaysa, o anki katmana bu ana kromozom atanır. Ancak en az bir kromozom ondan daha baskınsa, bu kromozom daha sonra tekrar ele alınmak üzere, döngüden o an çıkılır ve başka bir kromozom ele alınır.

Bu çalışmada, genetik algoritmanın iterasyonlarında, bir sonraki nesle, eldeki kromozomların yarısı aktarılır. Bu sebeple oluşan katmanlardan, belli sayıdaki kromozomları seçmek gerekmektedir. Bu amaçla algoritma tamamlanınca başka bir

aşama daha yer alır. Bu kısımda önemli olan ise, aynı katmandaki veriler arasında seçim yapmam gerekirken nasıl bir yol izleneceğidir. Sonuçta, katman numarası küçük olan elemanlar, kendinden büyük katmanda yer alan bir elemandan daha üstündür. Ancak aynı katmandaki veriler için bu durum söz konusu değildir.

Bu durumda, yine kromozomların amaç fonksiyonu değerleri dikkate alınır. Her bir kromozom için, bu değerlerine bağlı olarak, kendisinden bir önceki ve bir sonraki kromozomlar dikkate alınarak hesaplanan bir uzaklık metriği atanır. Bu kromozomlar, bu metriğe göre sıralanır ve aralarından istenen sayıdakileri bu değere göre seçilmiş olur.

BÖLÜM 4

4. BÜYÜK ÖLÇEKLİ VERİ KÜMELERİNİN ÖBEKLENDİRİLMESİ

Önceki bölümlerde de değinildiği üzere, çok amaçlı öbekleme yaparken karşılaşılan en önemli sorunlardan birisi, geliştirilen yöntemlerin büyük boyutlu ve karmaşık veri kümeleri için ölçeklenebilir olmamasıdır. Bu çalışmada, bu konuya bir çözüm getirmek üzere var olan bazı metotların yardımıyla da, verimli bir şekilde çalışan bir yöntem öne sürülmüştür. Yöntemin detaylarına girmeden önce, kullanılan bazı veri madenciliği konuları ile ilgili bilgiler vermek faydalı olacaktır.

4.1. Kullanılan Kavramlar

4.1.1. Birliktelik Kuralları

Veri madenciliğinde, birliktelik kurallarının çıkarımı son yıllarda üzerine çalışılan ve çok revaçta olan bir konudur. Bir veri kümesinde yer alan öznitelikler için, bu öznitelikleri birbirine bağlı kılacak bazı kuralların üretilmesine birliktelik kuralı çıkarımı adı verilir. Başka bir deyişle, büyük bir veri kümesi içinde saklı duran ve ilginç olabilecek bazı ilişkileri ortaya çıkarmaya denir. Çıkarılan bu ilişkiler ilişki kuralları veya sıklık elemanları adını alır. Çizelge 4.1 ‘ de bir markette bulunan örnek 3 adet alışveriş listesi olduğu düşünülsün;

Çizelge 4.1 Örnek alışveriş listesi

İşlem No	**Elemanlar**
1	{Ekmek, Süt}
2	{Ekmek}
3	{Süt, Gazete, Ekmek}

Bu çizelgeden {Ekmek} ⟶ {Süt} şeklinde bir kural çıkartılabilir. Bunun anlamı, Ekmek ve Süt satışları arasında bir ilişki olduğudur. Yani bir markette bir kişinin sepetinde ekmek varsa sütün de belirli ihtimaller altında olacağı belirtilmektedir.

Bu tarz bilgiler pazarlama faaliyetlerinde karar vermek açısından çok yarar sağlamaktadır. Zaten birliktelik kurallarının çıkarımı, ilk pazarlama aktiviteleri için öne sürülmüştür. Tabi ki bu konunun hitap ettiği alan sadece pazarlama alanıyla sınırlı değildir. Günümüzde biyoenformatikten, web madenciliğine kadar pek çok alanda kullanılmaktadır.

4.1.1.1. Matematiksel Tanım

$I = \{i_1, i_2, \dots, i_n\}$, n adet elemandan oluşan ve her bir elemanın değer kümesi {0,1} şeklinde olan bir eleman kümesi olsun. $D = \{t_1, t_2, \dots, t_n\}$ ise içerisinde n adet işlem barındıran bir veri kümesini göstersin. D‘ deki her işlemin tekil bir numarası olurken, her işlem içerisinde I‘ daki elemanların bir alt kümesini barındırsın. Bir kural, bir çıkarım olarak $\mathbf{X \Rightarrow Y}$ şeklinde tanımlanır. Burada X ve Y, I kümesinin elemanları iken, ortak hiçbir eleman içermezler.

Tanımların daha anlaşılır olması için şu şekilde bir örnek verilebilir:

I = {Ekmek, Süt, Gazete} olsun. Örnek bir kural ise {Ekmek, Süt}⇒{Gazete} olsun. Bu örneğin bir marketteki elemanları temsil ettiği düşünülürse, kuralın anlamı, bir müşteri Ekmek ve Süt aldığı zaman Gazete de alacaktır şeklinde olabilir.

Tüm olası kurallardan, işe yarayanları seçebilmek için bazı kısıtlar kurallar üzerine uygulanır. En çok bilinen ve bu tezde de kullanılan kısıtlar, destek[4] ve güven[5] kavramlarıdır.

[4] ing: support
[5] ing: confidence

Destek kavramı, bir kural için, veri kümesindeki tüm işlemler üzerinden hesaplanır. Örneğin bir X kuralı için, destek(X)‘ in değeri, veri kümesinde X‘ i içeren işlemlerin, tüm işlemlere oranı ile bulunur. Çizelge 4.1 dikkate alınırsa, destek({Süt}) = 2 / 3 olacaktır.

Bir kuralın güveni ise,

$$\text{güven}\left(X \Rightarrow Y\right) = \frac{\text{destek}\left(X \Rightarrow Y\right)}{\text{destek}(X)} \tag{4.1}$$

formülü ile bulunur.

Örneğin, güven ({Ekmek} ⇒ {Süt}) = 2 / 2 = 1 çıkar.

Bunun anlamı ise veri kümesinde, {Ekmek} ⇒ {Süt} kurallarının %100 ‘ ü doğrudur. Bir işlem kümesinde, Ekmek varsa mutlaka Süt de olmalıdır anlamı taşır.

4.1.1.2. Kuralların Üretimi

İlişki kuralları genel olarak, kullanıcı tarafından belirlenen en az destek ve en az güven kısıtlarını aşmak zorundadır. Kullanıcının belirlediği sınırın altında kalan kurallar elenir. Kural üretimi genelde iki aşamada yapılır. İlk aşamada veri kümesindeki tüm sıklık eleman kümeleri üzerine en az destek kısıdı uygulanır ve elde kalanlar ile ikinci aşamaya geçilir. İkinci aşamada ise bu sefer, elde kalan sıklık elemanlarına, en az güven kısıdı uygulanır. Belirlenen değerin altında kalan kurallar elenir ve elde kalan tüm kurallar sıklık eleman kümesini oluşturur.

Tahmin edileceği üzere, veri kümesindeki tüm sıklık eleman kümelerini bulmak çok masraflı bir iştir. Tüm olası sıklık kümelerinin sayısı, I kümesinin kuvvet kümesi şeklindedir. Bu sebeple bu aşamada performansı arttıracak pek çok algoritma önerilmiştir. Apriori isimli algoritma, bu tezde de tercih edilen ve yaygın olarak kullanılan bir algoritmadır.[35]

4.1.2. Apriori Algoritması

Apriori algoritması, bilgisayar bilimleri ve veri madenciliği alanında birliktelik kurallarının çıkarımı için kullanılan çok yaygın bir yöntemdir. İçerisinde işlemler içeren veritabanları üzerinde çalışmak için tasarlanmıştır.

Verilen bir eleman kümesi[6] için, algoritma, eleman kümesi içinde en az 'c' adedinde geçen alt kümeleri bulmaya çalışır. Bu yöntem aşağıdan yukarıya bir çalışma yapısı izler. İlk aşamada her sıklık alt kümesi birer eleman içerir ve her aşama sonunda bu alt kümeler birer elemanla genişletilir. Her aşamada elde olan alt küme ise asıl veri kümesi kullanılarak test edilir. Algoritma oluşturduğu alt kümeleri genişletemediği zaman durur. Algoritmanın amacı, verinin değişik alt kümeleri arasında ilişkiler bulmaktır. Verinin her kümesi belli sayıda eleman içerir ve bu kümeler işlem[7] adını alır.

Algoritma, genişlik öncelikli arama yapar.[36] k uzunluğundaki aday eleman kümeleri, k-1 uzunluğundaki eleman kümelerinden üretilir. Daha sonra içerisinde sık olmayan eleman alt kümesi içeren adayları eler.

Bu algoritmanın çalışma mantığını göstermek için şu şekilde bir örnek verilebilir:

Bir T işlemsel veri kümesinin elemanlarının aşağıdaki gibi olduğu farz edilsin.
1){Ekmek, Süt, Gazete, Su} 2){Ekmek, Süt} 3){Süt, Gazete, Su} 4){Süt, Gazete} 5){Ekmek, Süt, Su} 6){Gazete, Su} 7){Süt, Su}

Apriori'nin yapacağı ilk şey, veritabanındaki tüm elemanlar için sıklıkları, yani destek değerlerini hesaplamaktır. Bu durumda aşağıdaki destek değerleri oluşur.

destek({Ekmek}) = 3 / 7 destek({Süt}) = 6 / 7 destek({Gazete}) = 4/7 destek({Su}) = 5 / 7

[6] ing: *itemset*
[7] ing: *transaction*

Daha sonra algoritma kullanıcının gireceği bir en az destek değerini dikkate alır. Bu örnek için en az destek değeri 3 / 7 şeklinde ayarlanmış olsun. Bu değerden dolayı yukarıdaki kümelerden elenecek bir küme olmayacaktır. Bir sonraki aşamada ise, eldeki tüm sıklık elemanlarından ikililer oluşturulur ve bu ikililerin sıklık değerleri hesaplanır:

destek({Ekmek, Süt}) = 3 / 7, destek({Ekmek, Gazete}) = 1 / 7
destek({Ekmek, Su}) = 2 / 7
destek({Süt, Gazete}) = 3 / 7, destek({Süt, Su}) = 4 / 7
destek({Gazete, Su }) = 3 / 7

Aynı mantıklı bu oluşan kümelerden 2. si ve 3. sü elenirler. Bundan sonraki aşamada üçlü eleman kümeleri oluştururken, artık tüm olasılıklar ele alınmaz. İçerisinde sadece, {Ekmek, Gazete} ve {Ekmek, Su} ikililerini içermeyen 3lüler oluşturulur.

Bu durumda oluşabilecek tek üçlü {Süt, Gazete, Su} olur. Bu aşamada, 4 lü bir yapı oluşturmak mümkün olmadığı için algoritma burada durur. Son kalan {Süt, Gazete, Su} kümesinin destek değeri 2 / 7 olurken, 3 / 7 lik en az destek değeri şartından dolayı, bu en az destek değeri ile algoritma çalıştırılınca seçilecek bir sıklık kümesi oluşmaz.

4.1.3. Önemlilik[8]

Bir sıklık kümesinin öneminden bahsederken, o kümenin asıl veri kümesi içerisinde kapsadığı veri sayısı dikkate alınır. Daha önceden önemlilik kavramı üzerine pek çok çalışma önerilmiştir. Bunlardan bazıları örüntü[9] kural önemliliğine, bazıları ise sıklık eleman kümelerine veya ilişki paternlerinin önemliliğini konu alır.[37, 38, 39, 40] [38] çalışmasına göre önemlilik ölçütü nesnel ve öznel olmak üzere ikiye ayrılmaktadır. Destek, güven, tf-idf kavramları nesnellere örnek olarak verilebilir.

[8] ing: *significance*
[9] ing: *pattern*

Öznel olarak tanımlananlar ise genelde önbilgi veya bir arka plan modeline bağlı olarak elde edilen sonuçlardır.

Bir sıklık eleman kümesinin önemliliği ise [41] çalışmasında, S fonksiyonu olarak ifade edilir ve bir s sıklık kümesini S(s) şeklinde gerçek bir değer ile eşleştirir.

4.1.4. Entropi

n adet elemandan oluşan bir veri kümesinin S olduğu düşünülsün. S' nin d kolondan ve N adet satırdan oluştuğu durumda A_j , S' deki j. kolondaki tüm değerleri göstersin. S' deki i. satır ise d boyutlu bir vektör olarak $<a_{i1}, a_{i2}, \dots , a_{id}>$ şeklinde gösterilebilir. Bu vektörde a_{ij} alabileceği değerler tanımKümesi(A_j) ile gösterilir. Diğer bir deyişle, a_{ij} sadece A_j kolonundaki değerlerden birine sahip olabilir. $P(a_{ij})$ ise a_{ij} kategorik değerinin j. kolonda bulunma ihtimali gösterir. Bu ise j. kolondaki a_{ij} sayılarının geçme sıklığı hesaplanarak kolayca bulunabilir. D_j ise A_j ' nin değer kümesindeki farklı elemanların sayısını gösterir.

Her kolonun entropisi Denklem 4.2.1' deki hesaplanabilir. Bir veri kümesinin entropisi ise, kümede bulunan tüm kolonların entropilerinin toplamı olarak Denklem 4.2.2 ' deki gibi gösterilebilir.

$$E(A_j) = -\frac{1}{\log_2 D_j} \sum_{a_{ij} \epsilon A_j} P(a_{ij}) \log_2 P(a_{ij}) \quad (4.2.1)$$

$$E(S) = \frac{1}{d} \sum_{i=1}^{d} E(A_j) \quad (4.2.2)$$

Yukarıdaki entropi anlatımı aşağıdaki örnek ile daha anlaşılır olacaktır. Çizelge 4.2 ' de 4 elemandan oluşan bir veri kümesinin verildiği düşünülsün:

Çizelge 4.2 Örnek bir veri kümesi

Veri No	A_1	A_2	A_3
1	a	c	g
2	a	d	h
3	b	e	h
4	a	f	i

Yukarıdaki tanımlara uygun olarak, d = 3 ve N = 3 olur. A = {A_1, A_2, A_3} ve D_1 = 2, D_2 = 4, D_3 = 3 olur. Bu durumda her A kolonunun entropi değerlerini aşağıdaki gibi hesaplayabiliriz:

A_1 kolonu için, olasılığı hesaplanacak değerler a ve b değerleridir.

$P(a) = \frac{3}{4}$ ve $P(b) = \frac{1}{4}$ olur.

$$E(A_1) = -\frac{1}{\log_2 2}\left(\frac{3}{4} * \log_2 \frac{3}{4} + \frac{1}{4} * \log_2 \frac{1}{4}\right) = -(0{,}75 * -0{,}415 + 0{,}25 * -2)$$

$E(A_1) = 0{,}81125$ olur.

A_2 kolonu için, olasılığı hesaplanacak değerler c, d, e, f değerleridir. Ve her birinin olasılığı $\frac{1}{4}$ olur.

$$E(A_2) = -\frac{1}{\log_2 4}\left(\frac{1}{4} * \log_2 \frac{1}{4}\right) * 4 = 1 \text{ olur.}$$

A_3 kolonu için ise olasılığı hesaplanacak değerler, g, h ve i değerleridir.

$P(g) = \frac{1}{4}$ $P(h) = \frac{2}{4}$ $P(i) = \frac{1}{4}$ olur.

$$E(A_3) = -\frac{1}{\log_2 3}\left(\frac{1}{4} * \log_2 \frac{1}{4} + \frac{2}{4} * \log_2 \frac{2}{4} + \frac{1}{4} * \log_2 \frac{1}{4}\right)$$

$$E(A_3) = -0{,}631 * (0{,}25 * -2 + 0{,}5 * -1 + 0{,}25 * -2) = 0{,}9465 \text{ çıkar.}$$

Bu durumda E(S) = 0,81125 + 1 + 0,9465 = 2,75775 olur.

Öbekleme uygulamalarında entropinin kullanımı ise, öbekleme sonucunda ortaya çıkan tüm öbeklerin kendi içlerindeki öbeklenme kalitesinin entropilerine dikkat ederek ölçülmesi sonucu ortaya çıkar. Buradan, [42] çalışmasında önerilmiş beklenen entropi kavramı ortaya çıkmaktadır. Bir veri kümesi k adet öbeğe ayrılmış olsun ve C^k , k adet öbeğin bölümlenme yapısı olsun. C_i , i. öbeği gösterirken entropi değeri $E(C_i)$ şeklinde tanımlansın. Bu şekilde, her öbek için hesaplanacak entropi değerlerinin, herhangi bir kısıt eklenmezse içerisinde çok fazla veri de, az sayı da veri de barındıran öbeklerde aynı çıkma olasılığı olmaktadır. Daha büyük boyuttaki bölümlenmelerin genel öbekleme üzerinde daha fazla etkisi olacağından, büyük öbeklere öncelik verecek şekilde ağırlıklar eklemek mantıklı olacaktır. Bu sebeple bir öbeklemenin beklenen entropi tanımı, tüm öbeklerin boyutuyla orantılı olarak entropi değerlerinin ağırlıklandırılması ve bu ağırlıklandırılmış entropilerin toplamı ile ortaya çıkmaktadır. Bu ifadeler aşağıdaki formülde de rahatlıkla görülebilir.

$$BE(C^k) = \sum_{i=1}^{k} \frac{N_i}{N} E(C_i) = \frac{1}{N}\sum_{i=1}^{k} N_i E(C_i) \quad (4.3)$$

Burada N, veri kümesindeki toplam eleman sayısı olurken, N_i ise i. öbekteki eleman sayısını belirtir. Bu denklemde her C_i öbeğinin entropi değerinin, C_i öbeğinin içerdiği eleman sayısı ile çarpıldığı görülmektedir. Bunun sebebi yukarıda bahsedilen ağırlıklandırmadan dolayıdır.

Yukarıdaki tanımlar ile bir kategorik veri kümesi üzerinde çalışan bir öbeklemenin başarısı şu şekilde ölçülebilir: Beklenen entropi küçük oldukça, her öbeğin olası entropi değerinin de küçük olduğu anlamına gelir. Entropi değerinin küçük olması demek, öbekte birbirine yakın elemanlar olduğunu belirtir. Her öbekte birbirine yakın elemanlar varsa da o öbek daha sıkıdır. Bu mantıkla, aynı sayıda öbeğe

ayrılmış birden çok öbekleme sonucu varsa, beklenen entropi mantığıyla, entropisi küçük olanın daha güvenilir olduğu söylenebilir.

4.1.5. Fazlalık[10]

Fazlalık, adından da anlaşılacağı üzere, basitçe bilgi tekrarı olarak düşünülebilir. Bu tezdeki mantıkla ise, iki adet sıklık eleman kümesi[11] ele alındığı zaman, bu kümelerin asıl veri kümesi içerisinde yer alan elemanlardan kaçını aslında gereksiz yere içerdiğidir.
[41] çalışmasında p ve q şeklinde iki patern arasındaki fazlalık değeri, bu paternlerin, önemlilik değerleri dikkate alınarak şu şekilde hesaplanmaktadır:

$$R(p,q) = S(p) + S(q) - S(p,q) \tag{4.4}$$

Bu formülde, iki paternin önemlilik değerleri ayrı ayrı toplandıktan sonra, ikisinin birlikte sahip oldukları önemlilik değerleri bu toplamdan çıkarılıp, bu iki kümenin yarattığı fazlalık anlaşılmaktadır.

Bu çalışmada ise, iki adet sıklık eleman kümelerinin arasındaki fazlalık miktarı, bu iki kümenin entropi değerleri dikkate alınarak hesaplanmaktadır. Oluşan formül, önceki bölümde açıklanan entropi tanımı dikkate alınarak aşağıdaki şekilde verilebilir:

$$R(p,q) = E(p) + E(q) - E(p,q) \tag{4.5}$$

4.1.6. Kazanım[12]

Bu çalışmada kazanım kavramı, eldeki sıklık eleman kümeleri içinden, en iyilerini seçerken kullanılmaktadır. Algoritmanın başında elde bir sıklık elemanları havuzu vardır. Algoritma her adımda bu havuzdan, bir sıklık eleman kümesini alır ve

[10] ing: *redundancy*
[11] ing: *frequent itemset*
[12] ing: *gain*

seçilecek havuza atar. İşte bu yapılırken, dikkat edilen şey o kümenin seçiminden ne kadar kazanım aslında diğer bir deyişle yeni bilgi kazanılacağıdır. Sonuçta, en son elde kalan sıklık eleman kümesi topluluğun, asıl veri kümesini en iyi şekilde temsil etmesi istenir. Bu durumda seçilen sıklık eleman kümeleri arasındaki çakışmalar ne kadar az olursa ve aynı zamanda bu kümeler veri kümesindeki elemanları ne kadar geniş şekilde kapsarsa, seçilen sonuç kümesi o kadar başarılı temsilciler içerecektir. Bu mantıkla, yeni bir sıklık eleman kümesi ele alındığı zaman, önceden seçilmiş elemanlarla arasındaki fazlalık miktarları kontrol edilir. Önceden seçilen kümelerle hangisi ile arasındaki fazlalık değeri en yüksek ise bu en yüksek değer dikkate alınır. Daha sonra bu maksimum değer, kendi önem değerinden çıkartılır. Tüm sıklık eleman kümeleri için bu değer hesaplanır ve kazanım adını alır. İşte bu kazanım miktarı en yüksek olan sıklık eleman kümesi, veri kümesini temsil etmeye aday sıklık kümeleri arasına eklenir. Anlatılan bu yapı aşağıdaki şekilde formülize edilebilir:

$$g(f) = E(f) - max_{q \in F^k} R(f, q) \tag{4.6}$$

Eldeki tüm sıklık eleman kümeleri F olsun. F^k ise, F ' den o ana kadar seçilmiş tüm sıklık eleman kümelerinin tutulduğu küme olsun. Yeni seçilmek için kazanımı hesaplanacak aday küme $f \in F - F^k$ olsun. Bu durumda yukarıdaki formül ile f kümesinin kazanım değeri hesaplanmış olur.

4.1.7. Örtüşme[13]

Bu çalışmada sıklık eleman kümelerinin diğer kümelerle bir araya geldiğinde ortaya çıkardıkları örtüşme başka bir deyişle eleman çalışması isimli bir kavram oluşturulmuştur. Bu değer de aslında kümelerin entropi değerleri dikkate alınarak hesaplanır. Sıklık eleman kümelerinden oluşan bir yapının örtüşme değeri, içerisindeki tüm kümelerin aralarındaki fazlalık değerlerinin toplamının normalize

[13]ing: *overlap*

edilmesi şeklinde hesaplanır. Aşağıdaki formülde F, içerisinde sıklık kümeleri içeren bir küme topluluğu olsun. O(F) değeri bu kümenin örtüşme değerini vermektedir.

$$O(F) = \frac{1}{d*\frac{(d-1)}{2}} \sum_{i=1}^{d} \sum_{j=2\ ,j>i}^{d} R(p_i, q_j) \quad (4.7)$$

Bir f sıklık kümesinin örtüşme değerine katkısı ise, daha sonra açıklanacak algoritmada f kümesinin, aday sıklık eleman kümelerinin arasına konulup konulmayacağında rol oynamaktadır. Bu sebeple bu kavramın da tanımını vermek faydalı olacaktır. Aşağıdaki formülde F içerisinde f de dahil olmak üzere sıklık eleman kümeleri barındıran bir küme topluluğu olduğu düşünülürse;

$$OK(f) = O(F) - O(F - \{f\}) \quad (4.8)$$

Geliştirilen algoritmada, bir sıklık eleman kümesinin seçiminde, o kümenin örtüşme değerine yaptığı katkının önemli olduğu yukarıda belirtilmişti. Bu durumu daha belirgin kılmak için aşağıdaki örnek verilmiştir.

a, b, c seçilen üç adet sıklık eleman kümesi olsun. Bu kümelere eklenecek bir küme daha olsun. Ve bu küme d ve e şeklinde iki sıklık eleman kümesi içerisinden seçilsin.

Aşağıda ise tüm bu kümelerin aralarındaki fazlalık değerleri şu şekilde verilmiş olsun:

$R(a,b) = 0{,}5\ R(a,c) = 0{,}2\ R(a,d) = 1{,}6\ R(a,e) = 0{,}8$
$R(b,c) = -0{,}2\ R(b,d) = 3{,}2\ R(b,e) = -0{,}8$
$R(c,d) = 1{,}8\ R(c,e) = 0{,}8$
$R(d,e) = -0{,}8$

F kümesinin sadece a, b, c sıklık eleman kümesi olduğu düşünülürse;

$O(F) = \frac{(0{,}5+\ 0{,}2-0{,}2)}{3} \approx 1{,}67$

olarak F kümesinin örtüşme değeri hesaplanır.

F_d = F ∪ {d} olsun. Bu durumda $O(F) = \frac{(0{,}5+\ 0{,}2+1{,}6-0{,}2+3{,}2+1{,}8)}{6} \approx 1{,}18$ olur.

F_e = F ∪ {e} olsun. Bu durumda $O(F) = \frac{(0{,}5+\ 0{,}2+0{,}8-0{,}2-0{,}8+0{,}8)}{6} \approx 0{,}22$ olur.

Bu bilgilerle iki sıklık eleman kümesinin de örtüşme değerine katkıları bulunabilir.

$OK(d) = 1{,}67 - 1{,}18 = 0{,}49 \quad OK(e) = 1{,}67 - 0{,}22 = 1{,}45$

Görüldüğü üzere d, e den daha küçük bir OK sonucuna sahiptir. Bu sebeple d ve e arasından seçim yapılacaksa, tercih edilecek küme e olmalıdır. Çünkü e kümesi seçilen kümeler arasına eklendiğinde, tüm sıklık eleman kümelerinin çakışma değeri azalmaktadır.

Burada kafa karıştıran bir kısım olabilir. Sonuçta önceden belirli bir sayıda kümenin belli bir örtüşme değeri vardır. Yani veri kümesindeki bazı elemanlar tekrarlı olarak, değişik sıklık eleman kümeleri tarafından temsil edilmektedir. Normalde varolan bu kümelere yeni bir sıklık eleman kümesi eklendiğinde, örtüşme değerinin artması beklenir. Yani daha çok sıklık eleman kümesinin, daha çok benzer elemanı temsil etmesi gerekir. Ancak yukarıdaki örnekte, yeni bir küme eklendiğinde, eskisine göre değerin azaldığı görülmektedir. Bunun sebebi ise, formüldeki pay kısmında bulunan sonucun (fazlalık değerlerinin toplamının), payda tarafından normalize edilmesidir.

4.2. Geliştirilen Yöntem

Bu kısımda, büyük ölçekli veri kümelerinin öbeklendirilmesini verimli bir şekilde sağlamak için, bu çalışmada öne sürülmüş yöntem anlatılacaktır. Yöntemde kullanılan kavramlar Bölüm 4.1 ‘ de anlatılmıştır. Yöntemin sözde kodu aşağıdaki gibi görülebilir. Bu yöntemde her elemanın entropi değerleri -1 ile çarpılmıştır bu sebeple, entropisi yüksek olan küme daha kullanışlıdır.

Geliştirilen yöntemin sözde kodu

Girdi: İçinde n adet sıklık eleman kümesi bulunduran F
Tüm kümelerin entropi değerleri

Output: Seçilen sıklık eleman kümeleri, kesinSeçilenListe

1: s , entropisi en küçük küme olsun

2: $F^k = \{s\}$

3: F^k içerisindeki tüm sıklık eleman kümelerinin, asıl veri kümesinde yer alan elemanlardan, kapsamadıklarının oranı, m olarak tanımlansın.

4: m > 0 olduğu sürece

5: Öyle bir sıklık kümesi f bul ki, bu f ' in, F – F^k ' daki elemanlar dikkate alındığı zaman kazanım değeri maksimum olsun.

6: $F^k = F^k \cup f$ ve F = F - f

7: m ' i F^k üzerinden hesapla

8: 4. adıma atla

9: F^k kümesindeki elemanları kesinSeçilen listeye aktar.

10: F ' deki elemanlar bitene kadar

11: Öyle bir sıklık kümesi f bul ki, bu f ' in, F^k ' daki elemanlar dikkate alındığı zaman kazanım değeri maksimum olsun.

12: f ' in F^k üzerindeki örtüşme değeri katkısını hesapla ve sakla

13: $F^k = F^k \cup f$ ve F = F – f

14: nsgaListesi isimli listeye f ' i ekle

15: nsgaListesi'ndeki elemanları 1. amaç fonksiyonu entropi değerleri, 2. amaç fonksiyonu örtüşme değeri katkı değerleri olmak üzere NSGA ' ya sok.

16: NSGA sonucu 1. katmanda yer alan tüm sıklık eleman kümelerinden en yüksek numaralı olanı bul ve nsgaListesindeki elemanlardan, bulunan elemana kadar olan tüm sıklık eleman kümelerini kesinSeçilenListeyeAktar.

17: kesinSeçilenListe'deki tüm elemanları öbekleme işlemi için çok amaçlı genetik algoritmaya gönder.

Yöntemin ilk kısmında, eldeki veri kümesini temsil edecek sıklık eleman kümeleri üretilmektedir. Bu kümelerin elde edilmesi için Bölüm 4.1.2' de anlatılan Apriori algoritması kullanılmıştır. Bu algoritma, tüm sıklık kümelerini bulduktan sonra en az destek ve en az güven kısıtlarını uygulayarak, algoritmaya sokulacak sıklık eleman kümelerini belirler.

Tüm sıklık eleman kümelerinin entropi değerleri hesaplanır. Aralarında en düşük entropiye sahip olan küme, ilk küme olarak seçilenler arasına konulur. Daha sonra

kaçırma oranı isimli bir metrik tutulur. Bunun için seçilenler arasındaki sıklık eleman kümeleri dikkate alınır. Elemanların bir aradayken, asıl veri kümesindeki elemanlardan kapsayamadıklarının oranı bu kaçırma oranını verir. Kaçırma oranı 0 olana veya sabitleşene kadar, seçilenler dışındaki elemanlardan her birinin, seçilenler ile arasındaki kazanım değerleri hesaplanır. Bunlar en yüksek değeri veren sıklık eleman kümesi seçilenler arasına eklenir. Kaçırma oranı 0 olduktan sonra ise, geri kalan ve henüz seçilmemiş elemanlar artık birer aday olurlar. Geri kalan tüm elemanlar içerisinden, önceki kazanım değeri mantığı ile, en yüksek değeri verenler, nsgaya sokulmak üzere bir listeye aktarılır. Tek fark ise listeye aktarılırken, seçilen kümedeki elemanlar üzerindeki örtüşme değeri katkıları da hesaplanır. Nsga listesi oluştuktan sonra ise, bu listedeki tüm elemanlar nsga algoritmasına sokulur. Amaç fonksiyonu olarak ise her elemanın entropi değeri ve listeye aktarılırken hesaplanan örtüşme değeri katkı değerleri kullanılır. Nsga‘ dan geri dönen sonuçlardan 1. katman dikkate alınır. Bu katmandaki elemanlar içerisinden en yüksek numaraya sahip sıklık eleman kümesi bulunur ve bu elemana kadar olan tüm sıklık eleman kümeleri seçilen listeye aktarılır.

Bu mantıkla çalışan algoritma sonucu, çok amaçlı genetik algoritma öbeklemesine sokulacak sıklık eleman kümeleri belirlenmiş olur.

4.3. Yeni Öbek Sayıları için Öbeklemenin Hızlandırılması

Önceden değinildiği üzere, geleneksel olarak kullanılan kategorik verilerin öbeklenmesi işleminde, algoritmanın öbekleme yapacağı öbek sayısını ön bilgi olarak bilmesi gereklidir. Bunun anlamı da, sonucu incelenmek istenen her yeni öbek sayısı için öbekleme işleminin baştan çalıştırılmasına gerek olduğudur. Bu tatmin edici bir yöntem değildir. Bu şekilde belki küçük ölçekli veriler için çok büyük bir zaman sorunu ortaya çıkmasa bile, büyük ölçekli bir veri kümesi işin içine girdiğinde, o öbekleme yönteminin verimli bir şekilde sonuçlara ulaşamayacağı açıktır.

Sonuç olarak k adet öbek için bir sonuç elde edilmişse, örneğin ortama yeni bir öbek eklendiğinde, eldeki öbeklerde yer alan verilerin bazılarının bu yeni öbeğe kaydırılması, eldeki bilgilerle mümkün olabileceği görüşü [43] çalışması ile öne sürülmüştür. Bu çalışmada, bu yöntem nümerik veri kümeleri için ortaya atılmıştır. Bu tez ise kategorik veriler üzerine olduğundan, belirtilen çalışmadaki yöntem temel alınarak, kategorik verileri aynı mantıkla artırımlı olarak öbekleyecek bir yöntem ortaya çıkartılmıştır.

En basit mantıkla bu yöntemin izlediği mantık şu şekilde açıklanabilir:

k adet öbek için öbekleme işleminin yapıldığı bir durumda, öbeklendirilen veri içerisinden k adet eleman öbek merkezleri olarak seçilmektedir. Geri kalan elemanlar için ise daha önceden açıklanan yöntemler yardımı ile bu öbek merkezleri ile aralarında bir uzaklık değeri tanımlanmaktadır. Ve bir eleman hangi merkeze en yakınsa, o öbeğin bir elemanı olarak atanmaktadır. Geliştirilen yöntemle ise, tüm elemanlar arasından bir tanesi (k+1). öbeğin merkezi olarak seçilir. Geri kalan her eleman için ise, kendisinin bulunduğu öbeğin merkezi ve yeni öbek merkezi olarak belirlenen eleman dikkate alınır. Bu aşamada izlenebilecek iki yol, ya ele alınan elemanı kendi öbeğinde bırakmak ya da yeni belirlenen merkezin etrafında öbeklendirmek şeklindedir.

Merkez olarak atanmamış tüm veri elemanları arasından yeni merkez olarak seçilecek eleman aşağıdaki formül ile tanımlanan h_n değerini maksimize eden eleman olarak seçilir:

$$h_n = \sum_{j=1}^{N} \text{maksimum}\left(u_j - \left\|e_n - e_j\right\|^2, 0\right) \quad (4.9)$$

Bu formülde h_n değeri her n elemanı için hesaplanmaktadır. Bir n elemanı ele alındıktan sonra, geri kalan her j elemanı teker teker dolaşılır. Formüldeki u_j değeri, ele alınan j. elemanın, kendi atanmış olduğu öbeğin merkezi ile arasındaki uzaklığı verir. e_n ve e_j ise veri kümesindeki n. ve j. elemanları gösterir. Mutlak değer içinde hesaplanan ise bu iki elemanın birbirlerine olan uzaklıklarıdır. h değeri maksimum

olan eleman, geri kalanlardan en iyi ayrılmış eleman olacaktır. Bu sebeple yeni merkez olarak seçilmesi mantıklı olmaktadır.

Bir eleman yeni öbek merkezi olarak atandıktan sonra, geri kalan elemanlar içinden kendi öbek merkezi ile arasındaki uzaklık, yeni seçilen öbek merkezi ile arasındaki uzaklıktan büyük olanlar, bu yeni elemanın temsil ettiği öbeğe atanırlar.

Bu tezde iki eleman arasındaki uzaklığın nasıl belirlendiği ise aşağıdaki çizelgedeki örnek veri kümesi kullanılarak anlaşılabilir. Bu örnek için de entropi tanımı yapılırken kullanılan veri satırları kullanılmaktadır:

Çizelge 4.3 Örnek bir veri kümesi

Veri No	A_1	A_2	A_3
1	a	c	g
2	a	d	h
3	b	e	h
4	a	f	i

Burada 1. ve 2. veri arasındaki uzaklık şu şekilde hesaplanabilir:

Tüm özniteliklerin iki elemanda da aldığı değerler karşılaştırılır. Bu iki değer aynı ise, elemanlar arasındaki uzaklık değeri artmaz. Ancak farklı olduğu taktirde uzaklık değeri bir arttırılır. Bu mantıkla 1. ve 2. verilerin 1. öznitelik değerleri aynı iken, 2. ve 3. öznitelikler için farklıdır. Bu sebeple bu iki veri arasındaki uzaklığın en basit ifadeyle 2 olduğu söylenebilir.

BÖLÜM 5

5. DENEYLER

Bu bölüm küçük ölçekli veriler ve büyük ölçekli veriler olmak üzere iki başlık altında incelenecektir. Önceden de değinildiği üzere, büyük ölçekli verilerin öbeklenmesinde, veriyi sıklık eleman kümeleri ile test edip, o şekilde öbeklendirme işlemi yapılmıştır. Bu yöntemin, diğer çalışmalarda kullanılan yöntemlere göre en büyük avantajı öbekleme sonuçlarını çok daha verimli bir şekilde ortaya çıkartmasıdır. Ayrıca çalışmalarda yaygın olarak kullanılan veri kümeleri için, belirli öbek sayıları altında elde edilen sonuçların diğer çalışmalarla karşılaştırıldığında çok tatmin edici olduğu görülmektedir.

5.1. Saflık Metriği

Veri kümeleri üzerinde öbekleme yaparken, öbeklemenin ne kadar tutarlı ve doğruya yakın olduğunun ölçülmesi amacıyla saflık[14] isimli metrik kullanılmıştır. Bu metrik, öbekleme sonucu ortaya çıkan tüm öbekleri dikkate alır ve verilerin bu öbekler içerisinde ne kadar homojen dağıldığını kontrol eder. Bu doğrultuda, her öbekte en bulunan eleman belirlenir ve bu elemanın toplam sayısı, veri kümesindeki tüm elemanların toplam sayısına oranlanır. Aşağıda ise bu metriğin kullandığı formülüzasyon yer almaktadır. Bu formülde n, toplam eleman sayısını, k, sonuçta elde edilen öbeklerin miktarını, t_i ise veri kümesindeki i. eleman türünü (sınıfını) gösterir. $t_i \cap Ö_k$ ifadesi ise i. sınıfa ait elemanlardan k. öbekte kaç adet bulunduğunu belirtir.

$$purity = \frac{1}{n} \sum_{k=1}^{|Ö|} \max\left((t_1 \cap Ö_1), (t_2 \cap Ö_2), \dots, (t_i \cap Ö_k)\right) \qquad (5.1)$$

[14] ing: *purity*

5.2. Küçük Ölçekli Veri Kümeleri

Bu tezde üzerine deneyler yapılan küçük ölçekli veri kümelerine Çizelge 5.1‘ den ulaşılabilir. Bu veri kümeleri, UCI veri ambarından[15] alınmışlardır.

Çizelge 5.1 Kullanılan küçük ölçekli veri kümeler

Veri Kümesi	Öbek Sayısı	Eleman Sayısı	Öznitelik Sayısı
Zoo	7	101	17
Soybean (Small)	4	47	35
Hayes – Roth	3	160	4
Peptide	2	203	10
Congressional Voting	2	435	16

5.2.1. Çıkan Sonuçların Ölçümü

Bu 5 adet veri kümesi için, geleneksel k-mod algoritması ve bu tezde geliştirilen çok amaçlı öbeklendirme yapan genetik algoritma çalıştırılmış ve aşağıdaki tablolarda yer alan sonuçlar elde edilmiştir. Öbekleme yönteminin çıkardığı sonuçları gözlemlemek için ise purity isimli metrik kullanılmıştır. Bu tezde geliştirilen yöntem kullanılırken, önceden de belirtildiği üzere 3 farklı amaç fonksiyonu(k-mod içsel, k-mod dışsal ve ewcd) kullanılmıştır. Algoritma çalıştırılırken bu fonksiyonlar ikili ikili kullanılmışlardır.

Çizelge 5.2 Zoo veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
7	0,7129	0,7822	0,6832	0,9010
8	0,7228	0,8218	0,7624	0,9208
9	0,7624	0,8416	0,8020	0,9307
10	0,7921	0,8812	0,8515	0,9604
11	0,8020	0,8812	0,8812	0,9703
12	0,8317	0,9208	0,9208	0,9703
13	0,8416	0,9406	0,9505	0,9802
14	0,8515	0,9505	0,9604	0,9802

[15] http://archive.ics.uci.edu/ml/

Çizelge 5.3 Soybean (small) veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
4	0,7447	1,0000	1,0000	0,9787
5	0,8298	1,0000	1,0000	0,9787
6	0,8085	1,0000	1,0000	0,9787
7	0,8511	1,0000	1,0000	1,0000
8	0,9362	1,0000	1,0000	1,0000

Çizelge 5.4 Hayes-Roth veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
3	0,4318	0,4621	0,4091	0,4545
4	0,4318	0,4621	0,4470	0,4545
5	0,4242	0,4621	0,4470	0,4545
6	0,4545	0,4924	0,4848	0,4773
7	0,4697	0,4924	0,4924	0,4924
8	0,4773	0,5303	0,4924	0,5455
9	0,4924	0,5682	0,5303	0,5455
10	0,4773	0,5909	0,5530	0,5455

Çizelge 5.5 Peptide veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
2	0,7567	0,8227	0,8227	0,8227
3	0,7621	0,8227	0,8374	0,8424
4	0,7780	0,8227	0,8374	0,8473
5	0,7780	0,8227	0,8424	0,8424
6	0,7812	0,8227	0,8473	0,8473
7	0,7812	0,8227	0,8424	0,8424
8	0,7812	0,8227	0,8473	0,8473
9	0,7883	0,8325	0,8522	0,8473
10	0,7904	0,8374	0,8522	0,8621
11	0,7919	0,8276	0,8571	0,8571
12	0,7960	0,8424	0,8522	0,8473

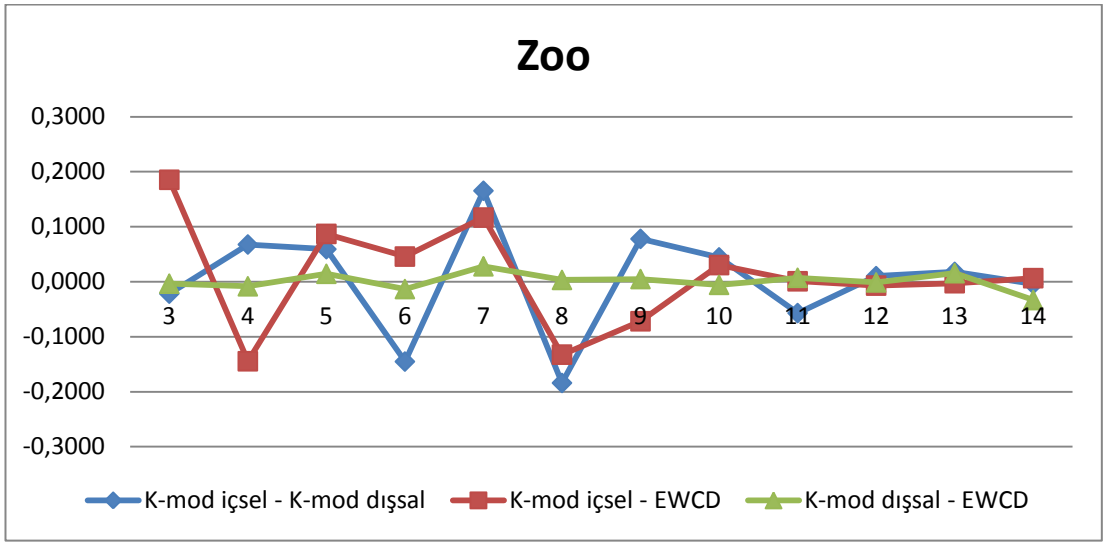
Çizelge 5.6 Congressional Voting veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
2	0,6138	0,6667	0,6138	0,7011
3	0,6897	0,8483	0,6805	0,8230
4	0,7793	0,8483	0,6805	0,8230
5	0,7770	0,8483	0,6943	0,8345
6	0,8069	0,8483	0,7011	0,8345
7	0,8230	0,8483	0,7080	0,8345
8	0,8069	0,8483	0,7678	0,8345

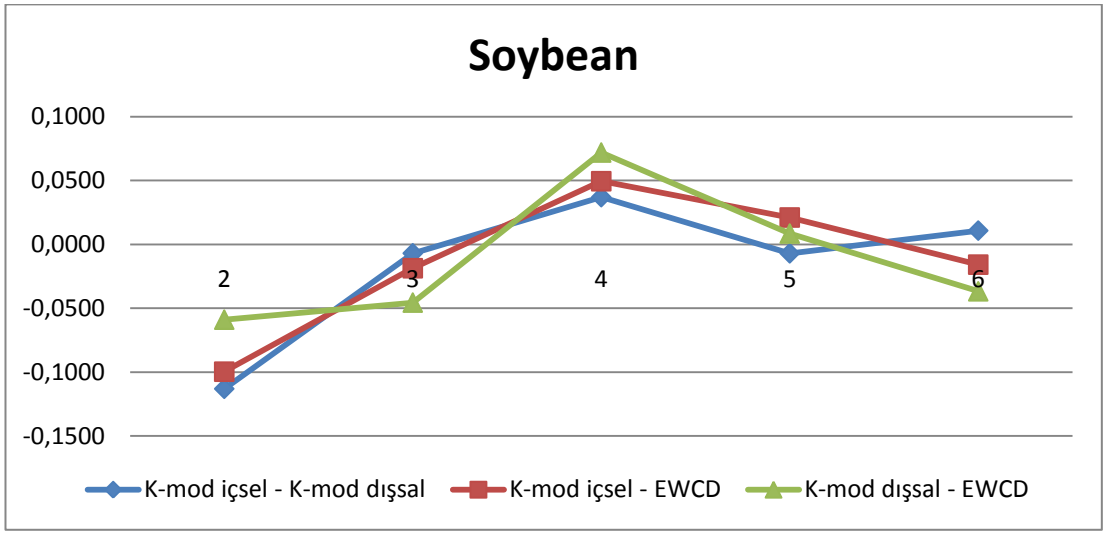
Çizelge 5.2 – 5.6 arasından da görüleceği üzere, purity sonuçları için, geliştirilen ve çok amaçlı öbekleme yapan genetik algoritma, her durumda, tek bir amaç fonksiyonu ile çalışan geleneksel k-mod algoritmasının sonuçlarına üstün gelmektedir.

5.2.2. Δ^2IEE Metriği ile Öbekleme Sonuçlarının Doğrulanması

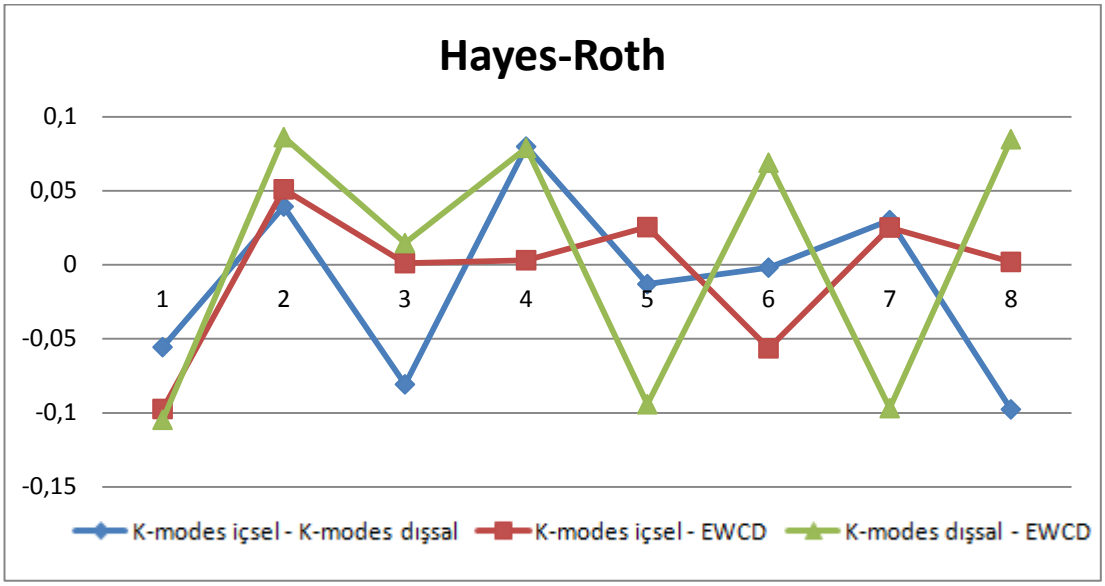
Test edilen ve sonuçları yukarıda gösterilen 5 adet veri kümesi için elde edilen öbekleme sonuçlarının doğrulanması için Δ^2IEE metriği kullanılmıştır. Bu metrik yardımı ile bir veri kümesinin aslında kaç sayıda öbeğe ayrılması gerektiği veya hangi öbek sayıları için o veriyi öbeklemenin mantıklı olduğu gibi bilgilere ulaşılabilir. Bu bilgiye ulaşabilmek için ise önceki aşamada elde edilen verinin öbeklenmiş sonuçlarına ihtiyaç vardır. Bu doğrultuda yukarıdaki veri kümelerinden 'Zoo', 'Soybean' ve 'Hayes-Roth' için tüm amaç fonksiyonlarının ikili kombinasyonları için Δ^2IEE ölçümleri yapılmış olup, bilgilerine aşağıda Şekil 5.1' den Şekil 5.3' e kadar ulaşılabilir.



Şekil 5.1 Zoo veri kümesi için öbek sayısı tahmin sonuçları



Şekil 5.2 Soybean veri kümesi için öbek sayısı tahmin sonuçları



Şekil 5.3 Hayes-Roth veri kümesi için öbek sayısı tahmin sonuçları

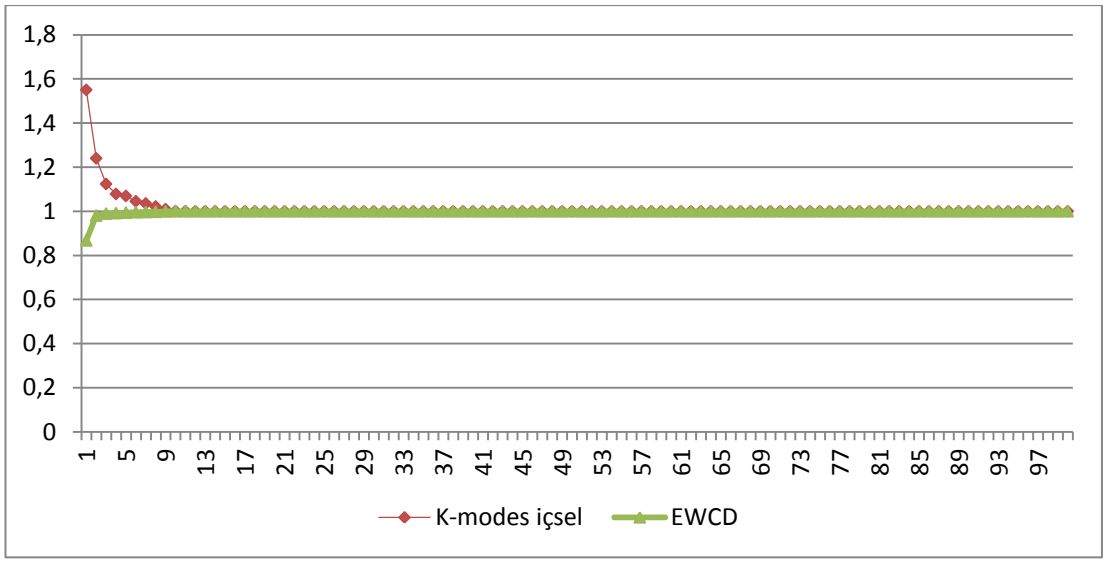
Yukarıdaki grafiklerde, her amaç fonksiyonu ikilisi için tepe noktaları, veri kümelerinin öbeklendirilmesi için en iyi öbek sayısını vermektedir. Zoo veri kümesi için, $\Delta^2 IEE$ değeri 7 öbek için en yüksek değerine ulaşırken, bunu 5 ve 9 öbek sayıları takip eder. Bu sonuçlardan, Zoo veri kümesi için öbeklendirme yaparken, 7, 5 ve 9 öbek sayılarını seçmenin uygun olabileceği anlaşılır. Zoo veri kümesinin gerçekte de 7 öbekten oluşması bu sonucu doğrulamaktadır. Aynı mantıkla, soybean veri kümesi için, $\Delta^2 IEE$ değerinin 4 öbek sayısını gösterdiği görülmektedir. Bu sayı soybean veri kümesinin gerçek öbek sayısı ile aynıdır. Hayes-Roth veri kümesi ise gerçekte 3 öbekten oluşmaktadır. Ve $\Delta^2 IEE$ değeri bu veri kümesi için en yüksek değerini 5 daha sonra ise 3 öbek sayısında almaktadır. Bu sonuçlardan, bu yaklaşıma dayanarak öbekleme işleminin yapılacağı öbek sayılarının belirlenmesi, doğru bir tercih yapıldığını göstermektedir.

5.2.3. Genetik Algoritmadaki İterasyon Sayısının Belirlenmesi

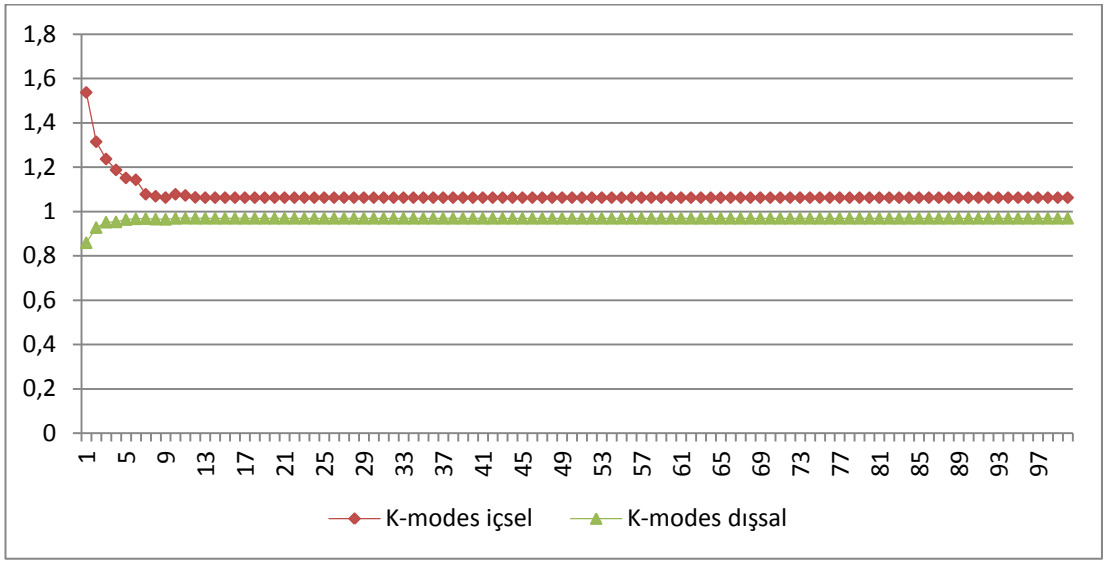
Bu tezde yapılan çalışmanın deneyleri i5 işlemciye ve 4 MB RAM ‘ e sahip bir makinede ve python programlama dili kullanılarak yapılmıştır. Genetik algoritmadaki kromozomların sayısı 100 olarak belirlenmiştir. Her iterasyonda

kromozomların çabuk yakınsaması için k-mod operatörü, kromozomlar üzerine uygulanmıştır. Genetik algoritmanın sonlanma kriteri olarak ise 100 iterasyon sonucu genetik algoritmanın bitirilmesi uygun görülmüştür. Bu sayının yeterli olduğu yine yapılan deneyler sonucu görülmüştür. Aşağıda şekillerde, Zoo ve Hayes-Roth veri kümeleri için, kromozomların ne şekilde yakınsadığı takip edilebilir. Bu şekillerde yer alan ölçümlerde iterasyon sayısını 100 olarak ayarlanmış olup, k-mod operatörü dolayısıyla kromozomların çok daha önce yakınsadığı görülebilir.

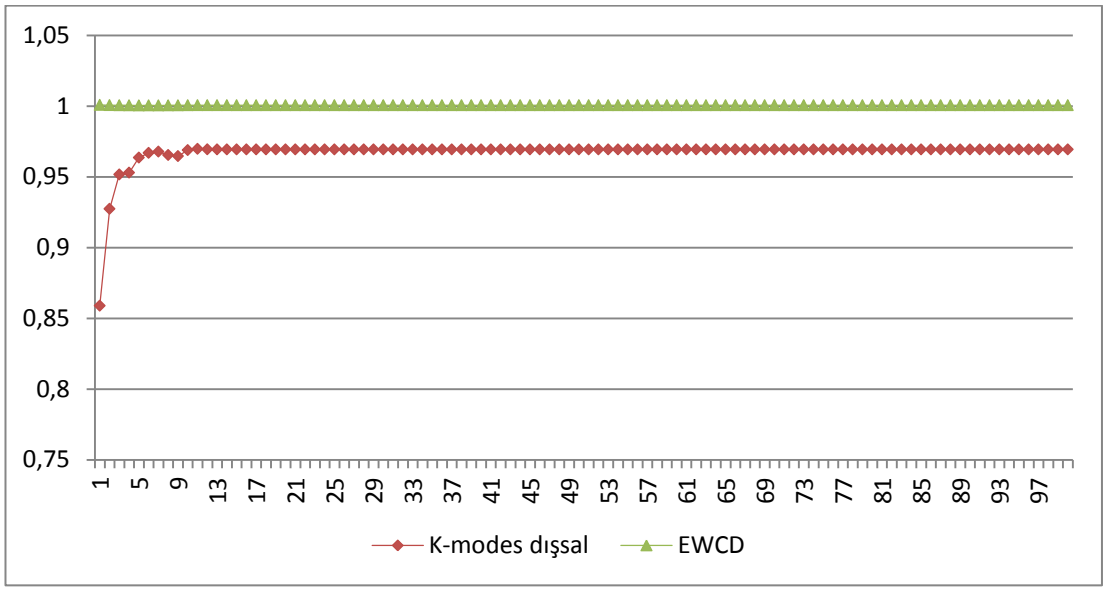
Zoo veri kümesi, önceden de bahsedildiği üzere 7 farklı hayvanın bilgilerini içerirken, 101 farklı hayvanı 17 farklı özniteliklerini gösterecek şekilde içerir. Şekil 5.4‘ den Şekil 5.6’ya kadar bu veri kümesi için, kromozomların nasıl yakınsadığı gözlemlenebilir.



Şekil 5.4 K-mod içsel ve EWCD fonksiyonlarının Zoo veri kümesi için yakınsamaları

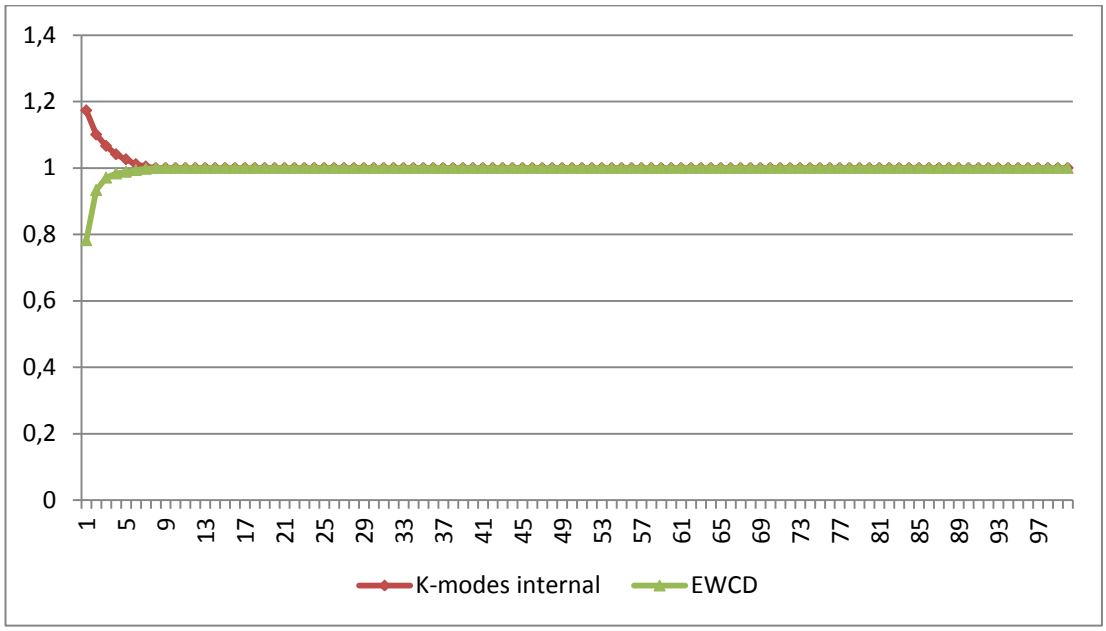


Şekil 5.5 K-mod içsel ve K-mod dışsal fonksiyonlarının Zoo veri kümesi için yakınsamaları

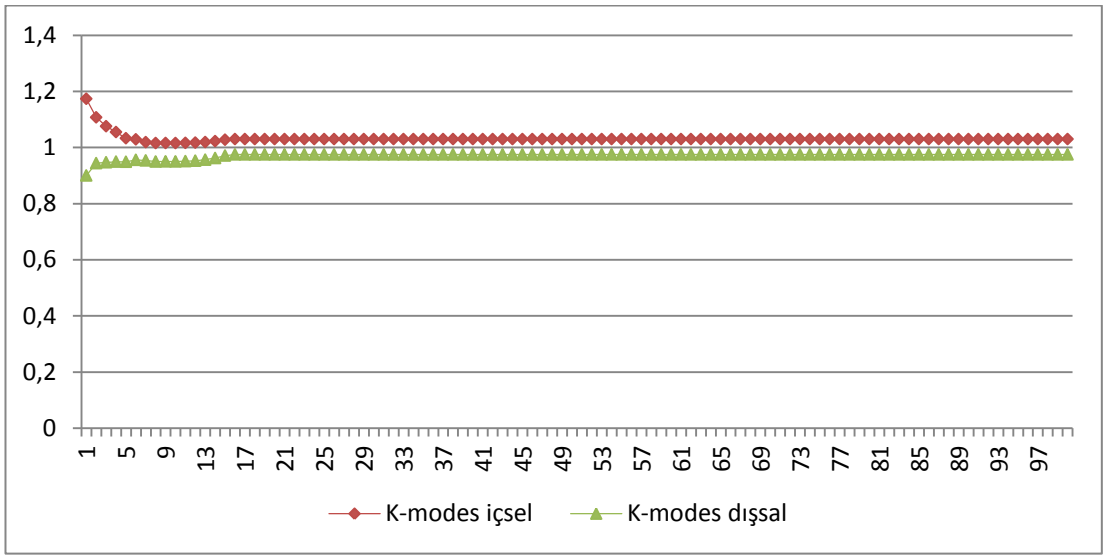


Şekil 5.6 K-mod dışsal ve EWCD fonksiyonlarının Zoo veri kümesi için yakınsamaları

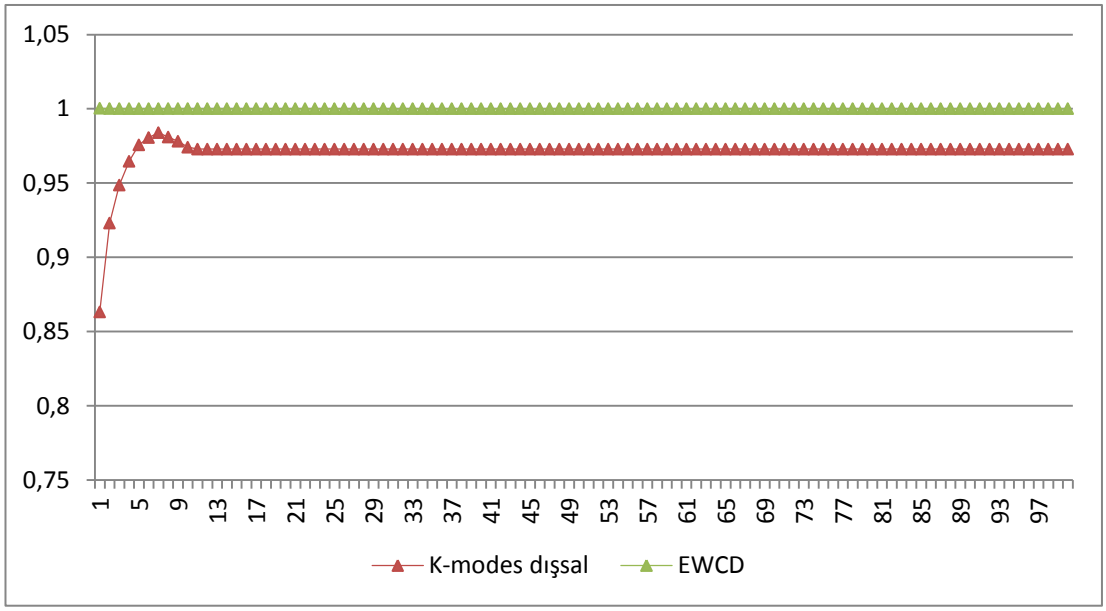
Hayes-Roth veri kümesi 160 adet elemanı 5 öznitelik altında içerirken bu elemanları 3 farklı öbekte kümeler. Şekil 5.7‘ den Şekil 5.9’ a kadar Hayes-Roth veri kümesi için tüm amaç fonksiyonu ikililerinin yakınsama sonuçları görülebilir.



Şekil 5.7 K-mod içsel ve EWCD fonksiyonlarının Hayes-Roth veri kümesi için yakınsamaları



Şekil 5.8 K-mod içsel ve K-mod dışsal fonksiyonlarının Hayes-Roth veri kümesi için yakınsamaları



Şekil 5.9 K-mod dışsal ve EWCD fonksiyonlarının Hayes-Roth veri kümesi için yakınsamaları

5.3. Büyük Ölçekli Veri Kümeleri

Daha önceden de değinildiği üzere verilerin öbeklenmesi işleminde, eğer ele alınan veri kümesi büyük ölçekli ise uygulanan yöntemin verimliliği açısından bazı sıkıntılar ortaya çıkabilmektedir. Bu amaçla bu çalışmada büyük ölçekli verilerin öbeklendirilmesinde kullanılan yöntemden daha önce bahsedilmiştir. Bu doğrultuda büyük ölçekli olarak sayılan 'Car', 'Mushroom' ve 'Nursery' kümeleri üzerine deneyler yapılmıştır. Bu veri kümelerinin özellikleri Çizelge 5.6'da gösterilmektedir.

Çizelge 5.7 Kullanılan büyük ölçekli veri kümeleri

Veri Kümesi	Öbek Sayısı	Eleman Sayısı	Öznitelik Sayısı
Car	4	1728	5
Nursery	5	12960	8
Mushroom	2	8124	22

5.3.1. Çıkan Sonuçların Ölçümü

Yukarıda tanımı yapılan veri kümeleri için, küçük ölçekli verilerin öbeklenmesinde kullanılan yöntemden farklı bir yöntem kullanıldığına önceden de değinilmişti. Bu bağlamda tekrar özetlemek gerekirse, veri kümelerindeki elemanlar doğrudan öbekleme işlemine sokulmayıp, onları temsil edecek sıklık eleman kümeleri çıkartılmıştır. Daha sonra ise bu sıklık eleman kümeleri öbeklenmiş ve asıl veride yer alan elemanlar, bu ortaya çıkan öbeklerden hangisine en yakınsa ona atanmıştır ve sonuçta elemanlar öbeklere ayrılmışlardır.

Aşağıda bu üç veri kümesinin öbeklenmesi sonucu tüm amaç fonksiyon ikililerinin ortaya çıkardığı purity sonuçları görülebilir. Tablolardan da görüleceği üzere, geliştirilen yöntem her durumda geleneksel k-mod algoritmasının sonuçlarına üstün gelmektedir.

Çizelge 5.8 Car veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
2	0,7010	0,7002	0,7002	0,7002
3	0,7000	0,7002	0,7002	0,7002
4	0,6999	0,7101	0,7222	0,7292
5	0,7000	0,7101	0,7002	0,7002
6	0,7000	0,7274	0,7280	0,7257
7	0,7000	0,7222	0,7222	0,7240
8	0,6980	0,7326	0,7332	0,7431
9	0,6990	0,7280	0,7286	0,7228
10	0,7180	0,7338	0,7274	0,7216

Çizelge 5.9 Nursery veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
2	0,3010	0,5869	0,6938	0,5915
3	0,5250	0,7478	0,6902	0,7279
4	0,4264	0,6046	0,6505	0,6156
5	0,3860	0,6133	0,6361	0,6017
6	0,4290	0,6269	0,6245	0,6080
7	0,4308	0,6057	0,6160	0,6031
8	0,4781	0,6184	0,6150	0,6076
9	0,4959	0,6172	0,6131	0,6299
10	0,5110	0,6147	0,6106	0,6222
11	0,5298	0,6165	0,6007	0,6113
12	0,5300	0,6112	0,5998	0,6155

Çizelge 5.10 Mushroom veri kümesi için purity sonuçları

Öbek Sayısı	K–Mod	K-Mod içsel – K-Mod dışsal	K-Mod içsel – EWCD	K-Mod dışsal – EWCD
2	0,4159	0,8807	0,8806	0,8884
3	0,5262	0,8613	0,8615	0,8699
4	0,7515	0,8811	0,8805	0,8844
5	0,8874	0,8794	0,8801	0,8840
6	0,8193	0,8796	0,8796	0,8799
7	0,8823	0,8784	0,8786	0,8810
8	0,8898	0,8759	0,8859	0,8769
9	0,8912	0,9081	0,9087	0,9087
10	0,8900	0,9097	0,9094	0,9095
11	0,8617	0,9100	0,9100	0,9101
12	0,8666	0,9088	0,9099	0,9088

5.3.2. Sıklık eleman kümeleri ile öbeklemenin tüm veri elemanları kullanılarak öbeklemeyle karşılaştırılması

Uygulanan bu iki yöntem arasındaki farkları gözlemleyebilmek için, diğer kullanılan büyük ölçekli veri kümelerine göre nispeten daha küçük boyutlu olan Car veri kümesi, her iki yöntem altında da çalıştırılmıştır. Aşağıdaki çizelgede, Car veri

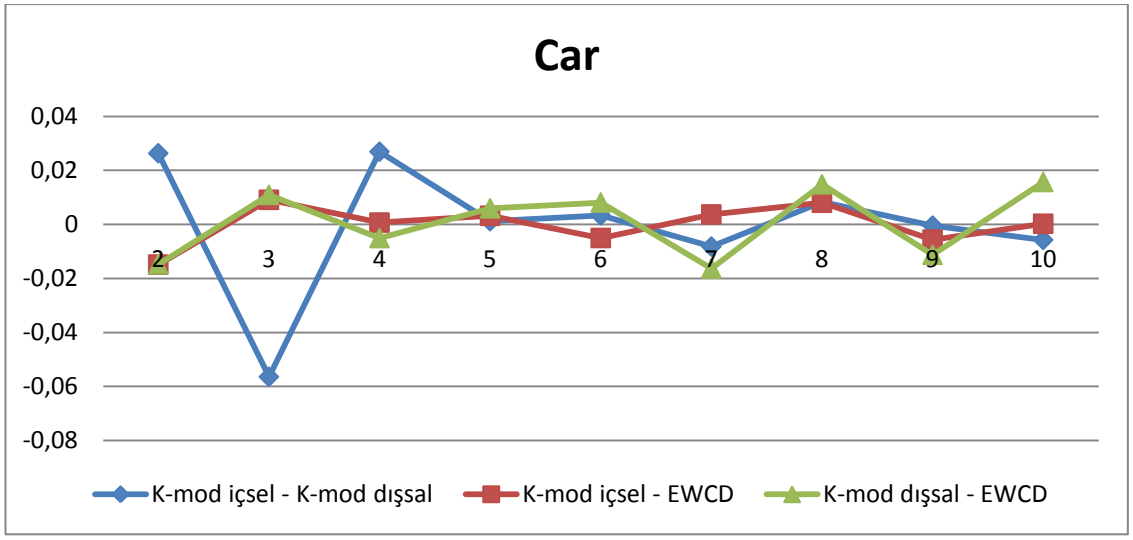
kümesinin hem sıklık eleman kümeleri kullanılarak öbeklendirilmiş hem de bütün veri elemanları kullanılarak öbeklendirilmiş sonuçlarına ulaşılabilir.

Çizelge 5.11 Car veri kümesi üzerinden yöntemlerin kıyası

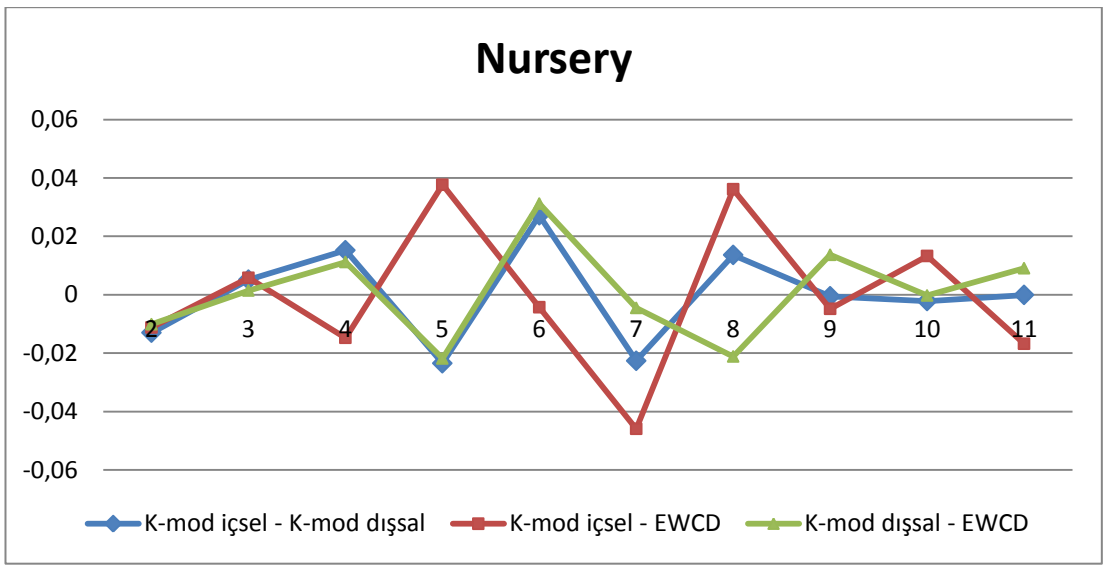
	K-Mod içsel – K-Mod dışsal		**K-Mod içsel – EWCD**		**K-Mod dışsal – EWCD**	
Öbek Sayısı	**Sıklık eleman kümeleri ile**	**Tüm elemanlar ile**	**Sıklık eleman kümeleri ile**	**Tüm elemanlar ile**	**Sıklık eleman kümeleri ile**	**Tüm elemanlar ile**
2	0,7002	0,7002	0,7002	0,7002	0,7002	0,7002
3	0,7002	0,7014	0,7002	0,7031	0,7002	0,7066
4	0,7101	0,7095	0,7222	0,7037	0,7292	0,7078
5	0,7101	0,7170	0,7002	0,7049	0,7002	0,7124
6	0,7274	0,7182	0,7280	0,7054	0,7257	0,7193
7	0,7222	0,7182	0,7222	0,7066	0,7240	0,7205
8	0,7326	0,7222	0,7332	0,7106	0,7431	0,7222
9	0,7280	0,7222	0,7286	0,7141	0,7228	0,7297

Yukarıdaki sonuçlardan da, çok daha verimli çalışan sıklık eleman kümeleri ile öbekleme işleminin, tüm elemanlar dikkate alınarak yapılan öbekleme yönteminin sonuçlarıyla çok yakın sonuçlara ulaştığı, hatta bazı durumlarda az da olsa üstün geldiği görülmektedir. Bu durum, bu tezde üzerine yoğunlaşılan iki yöntemin de başarılı sonuçlar ürettiğini ortaya çıkarmaktadır.

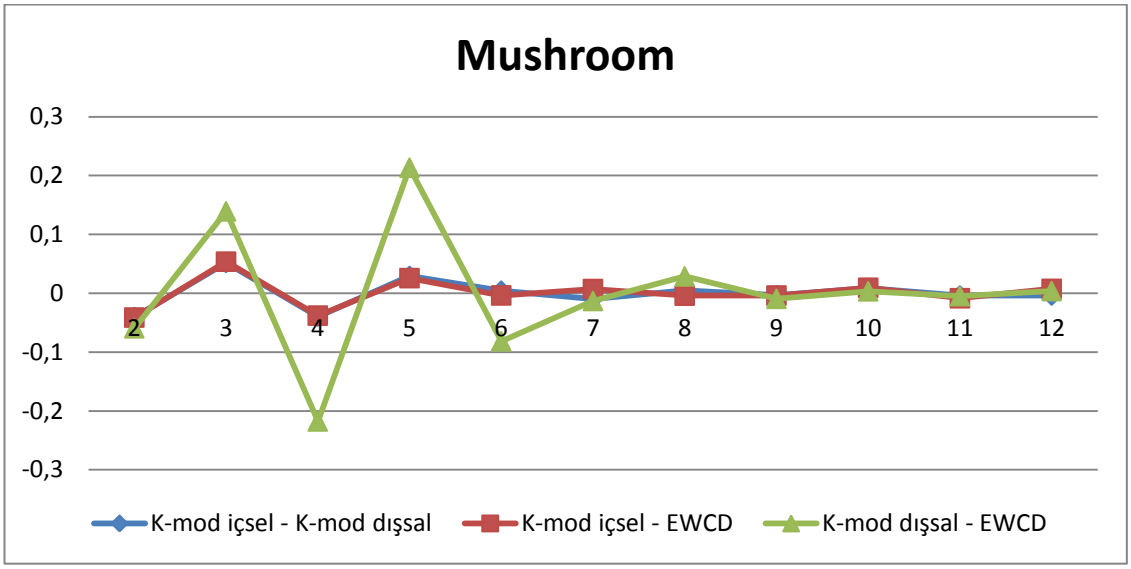
5.3.3. Δ^2IEE Metriği ile Öbekleme Sonuçlarının Doğrulanması



Şekil 5.10 Car veri kümesi için öbek sayısı tahmin sonuçları



Şekil 5.11 Nursery veri kümesi için öbek sayısı tahmin sonuçları



Şekil 5.12 Mushroom veri kümesi için öbek sayısı tahmin sonuçları

Yukarıdaki sonuçlar incelenecek olursa, 'Car' veri kümesi için k-mod içsel ve k-mode dışsal ikilileri için doğru öbek sayısı olan 4'ün yakalandığı görülmektedir. Diğer amaç fonksiyon ikilileri için ise, tahmin edilen öbek sayısının 3 olduğu görülmektedir, bu da gerçek öbek sayısına çok yakın bir sonuç olmaktadır.

'Nursery' veri kümesi için gerçek öbek sayısı olan 5'in k-mod içsel ve ewcd ikilisi tarafından bulunduğu ortaya konulmaktadır. Diğer ikililer için ise 6 öbeğin ideal olduğu belirtilmektedir. Yine bu da asıl sonuca çok yakın bir değerdir.

Mushroom için, asıl öbek sayısı olan 2 değeri için sonuçlar düşük gözükse bile, 3 öbek sayısı amaç fonksiyonlarının ideal bulduğu sonuçlardır. Diğer veri kümeleri gibi, 'Mushroom' veri kümesi içinde tüm amaç fonksiyonları ideale çok yakın sonuçlar bulmuşlardır.

Bu kısımdaki sonuçlardan da, eldeki amaç fonksiyonlarının sıklık eleman kümeleri kullanarak çok verimli ve tutarlı öbekleme işlemi yaptıkları görülmektedir.

5.3.4. Öznitelik sayısı fazla olan veri kümeleri ve sorunlar

Önceki bölümlerde büyük ölçekli veri kümeleri için uygulanmış olan sıklık eleman kümeleri kullanarak öbekleme yönteminin başarılı sonuçlar ürettiği gösterilmiştir. Ancak büyük ölçekli veri kümeleri kullanılırken, kümelerde yer alan eleman sayıları fazla olurken, her elemanın sahip olduğu öznitelik sayısının en fazla 22 olduğu görülmektedir. Bu doğrultuda, yöntemde açık bir kapı bırakmamak adına, yüksek öznitelik sayısına sahip veri kümeleri de incelenmek istenmiştir.

Bu doğrultuda [44] çalışmasında bahsi geçen 'Reuters' veri kümesi kullanılmıştır. Bu veri kümesi UCI veri ambarında yer almaktadır. İçerisinde 'Reuters Haber Ajansı'nda yer alan 21578 adet makale yer almaktadır. Bu makaleler çeşitli konularla ilgilidir. Öznitelikler olarak ise tüm haberlerde geçen kelimeler yer almaktadır. Daha sonra bir eleman (makale) için, eğer bir kelime bu makalede geçiyorsa 1, geçmiyorsa 0 değerini alacak şekilde veri kümesi düzenlenmiştir. Başka bir deyişle, her elemanın öznitelik değerleri 0 veya 1 değerlerini alabilmektedir. Ancak özniteliklerin tüm haberlerdeki kelimeler olarak tutulması, aynı köke sahip kelimelerin sanki farklı öznitelikleri gösteriyormuş gibi bir sorunun ortaya çıkmasına sebep olmuştur. Bu amaçla, yapılan çalışmada, 'Reuters' veri kümesi içerisindeki özniteliklerden kökleri barındıranlar bazı morfolojik yöntemler ile elenmiştir. Ve en son veri kümesini temsil eden 193 adet özniteliğin olmasına karar verilmiştir. Tüm makaleler içerisinde ise 8293 tanesi seçilmiştir. Özetle, elde kalan veri kümesi 8293 elemana sahip olurken, her eleman 193 adet öznitelikten oluşmuştur.

Oluşan bu karmaşık veri kümesi, sıklık eleman kümeleri ile öbeklenmek istendiğinde ise algoritmaya girecek sıklık eleman kümelerinin üretiminde zaman karmaşıklığı sorunu ortaya çıkmıştır. Sonuçta, önceden detayları da anlatılan eleman kümeleri üretme yönteminde kullanılan ağaç yapısı çok fazla dallanmaktadır. Bu da çalışılan tek makineyle sonuç elde edilmesini imkânsız kılmıştır.

Bu sebeple, sıklık eleman kümelerini üretmek için işlemi pek çok makineye dağıtabilecek bir yapı araştırılmıştır. Sonuç olarak 'Apache' firması[45] tarafından

geliştirilmiş ve map-reduce yöntemi kullanarak veri madenciliği, makine öğrenmesi konusundaki işleri öbek adı verilen pek çok makineye dağıtarak çok hızlı şekilde sonuç alınması sağlayan ve yine 'Apache' firması tarafından 'Hadoop[46]' isimli sistem üzerinde çalışan 'Mahout[47]' kütüphanesi kullanılmıştır.

Bu çalışmada bu sistem, laboratuar ortamında her biri 3GB RAM'e sahip 10 adet makine üzerine kurulmuştur. 'Reuters' veri kümesinden sıklık eleman kümeleri bu sistem sayesinde ve vaat edildiği gibi çok verimli bir şekilde üretilip büyük ölçekli kategorik verilerin öbeklenmesini sağlayan yöntemde çalıştırılmıştır. Ayrıca bu sıklık eleman kümeleri ile birlikte k-mod algoritması da çalıştırılmıştır. Aşağıdaki çizelgede ise, bu tezde geliştirilen ve k-mod algoritmasının ürettiği sonuçları karşılaştırmalı olarak yer almaktadır.

Çizelge 5.12 Reuters veri kümesi için purity sonuçları

Öbek Sayısı	**K–Mod**	**K-Mod içsel – K-Mod dışsal**	**K-Mod içsel – EWCD**	**K-Mod dışsal – EWCD**
2	0,6103	0,7010	0,6996	0,7090
3	0,6287	0,7095	0,7001	0,7090
4	0,6310	0,7110	0,7176	0,7117
5	0,6310	0,7110	0,7243	0,7220
6	0,6497	0,7110	0,7345	0,7298
7	0,6667	0,7229	0,7399	0,7315
8	0,6854	0,7304	0,7445	0,7397
9	0,6991	0,7400	0,7445	0,7490
10	0,7009	0,7447	0,7512	0,7490

Bu sonuçlardan da görüleceği üzere, geliştirilen yöntem her koşulda genel k-mod algoritmasına üstün gelmektedir. Ayrıca kullanılan 'Reuters' veri kümesi gerçekte 10 adet öbeğe ayrılmış durumdadır. 10 öbek için çıkan sonuçlarda ise öne sürülen metot yaklaşık %75 gibi yüksek bir öbekleme başarısına ulaşmıştır. Bu da yöntemin tutarlılığını ortaya koymaktadır.

BÖLÜM 6

6. SONUÇ

Veri madenciliğinin en önemli alanlarından biri olan öbekleme konusunda son dönemlerde yapılmış çalışmaların miktarı yadsınamaz boyuttadır. Bu çalışmaların çoğundaki amaç, öbekleme işlemini efektif ve güvenilir şekilde yapmaktır. Bu tezde yapılan çalışmanın amacını da bu temel oluşturmaktadır.

Bu tezde en basit ifadeyle kategorik verilerin öbeklenme problemi ele alınmıştır. Genel yöntemlerden farklı olarak bir veri kümesinin çok amaçlı yapı ile nasıl öbeklendirilebileceği incelenmiştir. Ve bu yapı genetik algoritma mantığı ile birleştirilerek başarılı bir yöntem ortaya çıkartılmıştır. Yöntemin tutarlılığının testi ise öncelikle küçük ölçekli veri kümeleri kullanılarak yapılmıştır. Bu doğrultuda test için belirlenen veri kümeleri kullanılarak ortaya çıkan öbekleme sonuçları belirlenen metrik üzerinden ölçülmüş ve başarılı sonuçlar elde edildiği ortaya konulmuştur.

Çoğu öbekleme algoritmasının verimli şekilde sonuç almasını engelleyen büyük ölçekli veri kümeleri için ise, küçük ölçekli veri kümelerinde kullanılan yönteme eklemeler yapılmıştır. Bu durumda sıklık eleman kümeleri kavramı işin içerisine sokulmuştur ve öbekleme işlemi veri kümesi elemanlarından, bu elemanlar kullanılarak ortaya çıkartılan sıklık eleman kümeleri üzerine kaydırılmıştır. Ortaya çıkan sıklık eleman kümelerinin öbekleri kullanılarak ise, artık veri elemanlarını hangi öbeğe atanacağına karar verilmiştir. Bu yöntemin çok verimli bir şekilde çalıştığı ve yine tutarlı sonuçlar ortaya çıkardığı benzer algoritmaların sonuçları ile de karşılaştırılarak doğrulanmıştır.

Çalışmanın son kısmında ise çok boyutlu ve çok fazla öznitelik içeren veri kümelerinde sıklık eleman kümeleri çıkartılırken sıkıntılar olacağı düşünülerek bu konu incelenmiştir. Bu durumda bazı kütüphaneler yardımı, pek çok makine eş zamanlı kullanılarak sıklık eleman kümelerinin çıkartım işlemi yapılmış ve büyük

ölçekli veri kümeleri için geliştirilen yöntem kullanılabilmiştir. Bu kısım için de elde edilen sonuçlar tatmin edici olmuştur.

Kısaca bu tez çalışmasının amacına ulaştığı ve kategorik verilerin öbeklenmesi ile ilgili konuda literatüre önemli katkılar yapıldığı söylenebilir. Zaten bu tez ile alakalı saygın bir dergide[16] ve uluslar arası bir konferansta[17] yayınlanmış olan çalışmalar bu durumun en güzel örneğidir.

[16] Journal of Universal Computer Science
[17] ASONAM 2011

KAYNAKLAR

[1] Huang, Z.: A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining, In Research Issues on Data Mining and Knowledge Discovery, 1997, 1-8.

[2] Huang, Z.: Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values, Data Mining Knowledge Discovery, 2 (3), 1998, 283-304.

[3] Guha, S., Rastogi, R., Shim, K.: ROCK: A Robust Clustering Algorithm for Categorical Attributes, Inf. Syst. 25(5), 2000, 345-366.

[4] Andritsos, P., Tsaparas, P., Miller, R. J., Sevcik, K.C.: LIMBO: Scalable Clustering of Categorical Data, Conference on Extending Database Technology (EDBT'04), 2004, 123-146.

[5] Ganti, V., Gehrke, J., Ramakrishnan, R.: CACTUS - Clustering Categorical Data Using Summaries, Conference on Knowledge Discovery and Data Mining (KDD'99), ACM, San Diego, 1999, 73-83.

[6] Barbará, D., Li, Y., Couto, J.: COOLCAT: An Entropy - Based Algorithm for Categorical Clustering. Conference on Information and Knowledge Management (CIKM'02), ACM, McLean, 2002, 582-589.

[7] Gibson, D., Kleiberg, J., Raghavan, P.: Clustering Categorical Data: An Approach based on Dynamical Systems, In Proceedings of 24th International Conference on Very Large Databases (VLDB'98), New York City, 1998, 311-323

[8] Yang, Y., Guan, X., You, J.: CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data, In Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02), ACM, New York City, 2002, 682-687.

[9] Hua Yan, Keke Chen, Ling Liu : Efficiently Clustering Transactional Data with Weighted Coverage Density, CIKM '06 Proceedings of the 15th ACM international conference on Information and knowledge management Pages 367 - 376

[10] Coello, C.C.: An Updated Survey of ga - based Multi - Objective Techniques,

Technical Report Lania-RD-98-08, Laboratorio Nacional de Inform´atica Avanzada, 1998.

[11] Ozyer, T., Alhajj, R.: Deciding on Number of Clusters by Multi-Objective Optimization and Validity Analysis, Journal of Multi-Valued Logic and Soft Computing 14(3), 2008, 457-474.

[12] Ozyer, T., Alhajj, R.: Parallel Clustering of High Dimensional Data by Integrating Multi-Objective Genetic Algorithm with Divide and Conquer, Appl. Intell. 31(3), 2009, 318-331.

[13] Srinivas, N., Deb, K.: Multi-Objective Function Optimization Using NonDominated Sorting Genetic Algorithms, Evolutionary Computation, 2(3), 1995, 221–248.

[14] Usaman Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth : The KDD process for extracting useful knowledge from volumes of data : Magazine Communications of the ACM Volume 39 Issue 11, Ekim 1996 Sayfa 27 – 34

[15] Eiben, A. E. et al (1994). "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87. ISBN 3-540-58484-6.

[16] Koza, John (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press. ISBN 0-262-11170-5

[17] Kenneth A. De Jong, William M. Spears: An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. PPSN 1990: 38-47

[18] Zitzler, E.: Evolutionary Algorithms for Multi-Objective Optimization: Methods and Applications, PhD thesis, Zurich: Swiss Federal Institute of Technology (ETH), Aachen, Germany, 1999.

[19] Fudenberg, D. and Tirole, J. (1983). Game Theory. MIT Press. Chapter 1, Section 2.4. ISBN 0-262-06141-4.

[20] Jiang, D., Tang, C., Zhang, A., Cluster analysis for gene expression data: IEEE Transactions on Knowledge and Data Engineering, 16(11), 1370-1386, Kasım 2004.

[21] Jain, A.K., Murty, M.N., Flynn., P.J., Data clustering: A review. CM Computing Surveys (CSUR), 31(3), 264-323, Temmuz 1999.

[22] Hartigan, J., Clustering Algorithms. *John Wiley and Sons Inc.*, New York, NY, 1975.

[23] Sergios Theodoridis & Konstantinos Koutroumbas (2006). *Pattern Recognition 3rd ed.*. pp. 635.

[24] Raymond T. Ng and Jiawei Han : CLARANS: A Method for Clustering Objects for Spatial Data Mining, Member, IEEE Computer Society Vol. 14, No. 5, Eylül / Ekim 2002

[25] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases, Proceedings of ACM SIGMOD Conference on Management of Data, 73-84, Ocak 1998.

[26] R. Sharan, R. Shamir : CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis, ISMB-00 Proceedings 2000

[27] Ben-Dor, A., Shamir, R., Yahkini. Z., Clustering gene expression patterns. Journal of Computational Biology, 6(3), 281-297, Nisan 1999.

[28] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu (1996-). "A density-based algorithm for discovering clusters in large spatial databases with noise". In Evangelos Simoudis, Jiawei Han, Usama M. Fayyad. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. ISBN 1-57735-004-9.

[29] A.Hinneburg, H.H. Gabriel : Denclue 2.0: Fast Clustering Based on Kernel Density Estimation (2007) Proceedings of the 7th International Symposium on intelligent data analysis

[30] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander (1999). "OPTICS: Ordering Points To Identify the Clustering Structure". ACM SIGMOD international conference on Management of data. ACM Press. pp. 49–60.

[31] Aggarwal, Charu C.; Wolf, Joel L.; Yu, Philip S.; Procopiuc, Cecilia; Park, Jong Soo (1999), "Fast algorithms for projected clustering", ACM SIGMOD Record (New York, NY: ACM) 28 (2): 61–72, DOI:10.1145/304181.304188

[32] Kaski, S., Nikkilä, J. and T. Kohonen. Methods for interpreting a self-organized map in data analysis. Proceedings of the European Symposium on Artificial Neural Networks, 185-190, Brussels, Belgium, 1998.

[33] Kim, Y. Lee. S., A clustering validity assessment index. Proceedings of Asian Conference on Knowledge Discovery and Data Mining, pages 602-608, Nisan 2003.

[34] Xiong, H., Tan, P. N., Kumar, V.: Hyperclique Pattern Discovery. Data Mining Knowledge Discovery 13 (2), 2006 , 219-242.

[35] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.

[36] http://en.wikipedia.org/wiki/Breadth-first_search

[37] S. Jaroszewicz and D.A. Simovici. A general measure of rule interestingness. PKDD'01.

[38] S. Jaroszewicz and D.A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. KDD'04.

[39] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. IEEE Trans. Knowledge & Data Engineering, 8:970–974, 1996.

[40] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns, KDD'02.

[41] D. Xin, H. Cheng, X. Yan, J. Han : Extracting Redundancy Aware TopK Patterns

[42] K. Chen, L. Liu : Towards Finding Optimal Partitions of Categorical Datasets, Ekim, 2003

[43] A. Likas, N. Vlassis, J. J. Verbeek : The global k-means clustering algorithm, Pattern Recognition 36 (2003) 451 – 461

[44] M. Peters, M. J. Zaki CLICK: Clustering Categorical Data using K-partite Maximal Cliques 2004

[45] http://www.apache.org/

[46] http://wiki.apache.org/hadoop/ProjectDescription

[47] http://www.ibm.com/developerworks/java/library/j-mahout/

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	:	DURSUN, Kayhan
Uyruğu	:	T.C.
Doğum tarihi ve yeri	:	26.04.1986 Trabzon
Medeni hali	:	Bekâr
Telefon	:	0 (535) 569 97 07
Email	:	kdursun@etu.edu.tr kayhandursun@gmail.com

Eğitim

Derece	**Eğitim Yeri**	**Mezuniyet Tarihi**
Lisans	TOBB ETÜ | Bilgisayar Müh.	2010

İş Deneyimi		**Görev**
2010-2012	TOBB ETÜ	Ders ve Araştırma Asistanlığı
2010-2010	Kasırga Bilişim Elektronik Ltd.	Yazılım Programlama
2009-2009	Kasırga Bilişim Elektronik Ltd.	Yazılım Programlama
2009-2009	Portakal Teknoloji	Yazılım Programlama
2008-2008	SIEMENS E.C.	Yazılım Programlama

Yabancı Dil

İngilizce

İspanyolca

Yayınlar

1. Onur Can Sert, Kayhan Dursun, Tansel Özyer, Jamal Jida, Reda Alhajj: The Unification and Assessment of Multi-Objective Clustering Results of Categorical Datasets with H-Confidence Metric. J. UCS 18(4): 507-531 (2012)

2. Onur C. Sert, Kayhan Dursun, Tansel Özyer: Ensemble of Multi-Objective Clustering Unified With H-Confidence Metric as Validity Metric, IEEE ASONAM 2011, Advances in Social Network Analysis and Mining

3. Kayhan Dursun, Ahmet Tunç Bilgin, Cumhur Kılıç, Osman Abul: An AI Approach to the Mastermind Game, Fırat University Electrical and Electronics Engineering and Computer Symposium 2011