T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TIBBİ VERİ KÜMELERİ ARASINDAKİ BİRLİKTELİK KURALLARININ ÇOK AMAÇLI GENETİK ALGORİTMA İLE ÇIKARILMASI

YÜKSEK LİSANS TEZİ

Buket KAYA

Anabilim Dalı: Biyomühendislik

Programı: Biyoelektronik

Tez Danışmanı: Doç. Dr. İbrahim TÜRKOĞLU

HAZİRAN-2010

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TIBBİ VERİ KÜMELERİ ARASINDAKİ BİRLİKTELİK KURALLARININ ÇOK AMAÇLI GENETİK ALGORİTMA İLE ÇIKARILMASI

YÜKSEK LİSANS TEZİ

Buket KAYA

08132101

Anabilim Dalı: Biyomühendislik

Programı: Biyoelektronik

Tez Danışmanı: Doç. Dr. İbrahim TÜRKOĞLU

HAZİRAN-2010

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TIBBİ VERİ KÜMELERİ ARASINDAKİ BİRLİKTELİK KURALLARININ ÇOK AMAÇLI GENETİK ALGORİTMA İLE ÇIKARILMASI

YÜKSEK LİSANS TEZİ

Buket KAYA

08132101

Tezin Enstitüye Verildiği Tarih: 25 Mayıs 2010
Tezin Savunulduğu Tarih: 11 Haziran 2010

Tez Danışmanı: Doç. Dr. İbrahim TÜRKOĞLU (F.Ü)
Diğer Jüri Üyeleri: Doç. Dr. Soner ALTUNDOĞAN (F.Ü)
Yrd. Doç. Dr. Ahmet ÇINAR (F.Ü)

HAZİRAN-2010

ÖNSÖZ

Tez çalışmam süresince sağladığı destek ve sabrı dolayısıyla değerli danışmanım Sayın Doç. Dr. İbrahim TÜRKOĞLU'na, şizofren hasta verilerini bize sağladığı için Doç. Dr. Mehmet TOKDEMİR ve Dr. Abdurrahim TÜRKOĞLU'na (F.Ü Tıp Fakültesi Adli Tıp Anabilim Dalı) teşekkürü bir borç bilirim.

Buket KAYA

ELAZIĞ-2010

İÇİNDEKİLER

Sayfa No

Sayfa No

ÖZET

Tıbbi verilerin hacimlerinin artması nedeniyle bu verilerden bilgi çıkarmak oldukça zorlaşır. Günümüzde çok büyük miktarda bilgi sürekli bir biçimde hastaların fizyolojik parametreleri gözlemlenerek toplanılmaktadır. Artan veri miktarları tıp uzmanları tarafından yapılan manüel analizleri bıktırıcı ve bazen imkansız birer görev haline getirmiştir. Analiz yapanlar, potansiyel olarak faydalı olabilecek bazı bilgilerin farkına varamayabilmektedir. Veri miktarında meydana gelen hızlı artış sonucu faydalı bilginin çıkarımı konusunda otomatikleştirilmiş bir yola ihtiyaç duyulur. Bu problem için kullanılan yaklaşımlardan biri veritabanları üzerinde bilgi keşfi veya veri madenciliği yöntemidir.

Bu tezin amacı, şizofreni hastalığına ait tıbbi verilerden çok amaçlı genetik algoritma kullanarak bağıntı ve sınıflandırma kurallarını çıkarmaktır. Bağıntı ve sınıflandırma kuralları özellikle tıbbi teşhis alanında uygulanan tıbbi problemler için faydalıdır. Bu tip kurallar tıbbi uzmanlar tarafından doğrulanabilir ve bu durum mevcut problemin daha iyi anlaşılmasını sağlar.

Son on yılda veri madenciliğinde kural keşfi üzerine pek çok teknik uygulanmıştır. Bu tekniklere örnek olarak uzman sistemler, yapay sinir ağları, veritabanı sistemleri ve evrimsel algoritmalar verilebilir. Evrimsel algoritmalar doğal evrim sürecinden esinlenerek geliştirilen hesapsal tekniklerin bir sınıfıdır. Doğal evrim süreci, gerçek hayatta karşılaşılan problemleri çözmek için doğal seçim ve en iyinin hayatta kalması mekanizmalarını örnek alır. Çok amaçlı evrimsel algoritmalar, kıyaslanamayan ve çoğunlukla rekabet halinde olan amaçların eş zamanlı optimizasyonu ile uğraşır.

Bu tezdeki amaç, çok amaçlı genetik algoritma kullanarak sınıflandırma ve birliktelik kurallarını etkin bir şekilde bulmaktır. Bu maksatla şizofreni hastalarının sosyodemografik ve biyokimya verilerine uygulanan yöntem, suç işleme durumları dikkate alındığında sosyodemografik verilere daha bağımlı olduğu gözlenmiştir.

Anahtar Kelimeler: Çok-amaçlı genetik algoritmalar, Veri Madenciliği, Sınıflandırma Kuralları, Birliktelik Kuralları, Optimizasyon

SUMMARY

Extraction of Association Rules in Medical Datasets via Multi-Objective Genetic Algorithms

Medical data are facing a challenge of knowledge discovery from the growing volume of data. Nowadays enormous amounts of information are continuously collected by monitoring physiological parameters of patients. The growing amounts of data has made manual analysis by medical experts a tedious task and sometimes impossible. Many hidden and potentially useful relationships may not be recognized by the analyst. The explosive growth of data requires an automated way in order to extract useful knowledge. One of the possible approaches of this problem is data mining or knowledge discovery from databases. Through data mining, interesting knowledge and regularities can be extracted and the discovered knowledge can be applied in the corresponding field to increase the work efficiency and in order to improve the quality of decision making. The objectives of the thesis are to extract association and classification rules from the schizophrenia medical data. Association and classification rules are typically useful for medical problems which have been massively applied particularly in the area of medical diagnosis. Such rules can be verified by medical experts and provide better understanding of the problem in-hand. Numerous techniques have been applied to rule discovering in data mining over the past decades, such as expert systems, artificial neural networks, linear programming, database systems and evolutionary algorithms. Among these approaches, the evolutionary algorithms have been emerged as promising techniques in dealing with the increasing challenge of data mining in medical area. The evolutionary algorithm is a class of computational techniques inspired by the natural evolution process that imitates the mechanism of the natural selection and survival-of-the-fittest in solving real life problems.

In this thesis, the aim is to discover classification and association rules via multi-objective evolutionary algorithms. Contrary to single-objective ones, multi-objective evolutionary algorithms deal with simultaneous optimization of several incommensurable and often competing objectives.

Keywords: Multi-objective genetic algorithms, Data Mining, Classification Rules, Association Rules, Optimization.

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR LİSTESİ

GA : Genetik Algoritma

VTBK : Veri Tabanı Bilgi Keşfi

1. GİRİŞ

Veri madenciliği büyük miktarlardaki veri tabanlarından saklı ve kullanıcı için değerli bilgilerin çıkarılması olarak tanımlanabilir. Veri madenciliğinin ortaya çıkmasındaki en büyük motivasyon, eldeki büyük miktarlardaki verileri karar verme mekanizması için kullanmaktır. Amaç bilinmeyen bağıntıların bulunması olduğu için, kullanılan araçlarda, kullanıcıdan alınan girdiler mümkün olduğunca az olmalıdır. Eğer kullanıcı ne aradığını biliyorsa, aranan bilgi geleneksel veri tabanı araçları (Veri Tabanı Yönetim Sistemleri) kullanılarak bulunabilir. Bu özelliğinin yanında, veri madenciliği sistemlerinin ürettiği sonuç, stratejik kararların alınmasına katkı sağlayacak yapı ve içerikte olmalıdır.

Sınıflandırma ve bağıntı kuralları çıkarma veri madenciliğinin en önemli iki araştırma konusudur. Bağıntı kuralları bir veri kümesinde nitelikler arası ilginç ilişkileri ararken, sınıflandırma nitelikler kümesinden oluşan bir kaydı önceden tanımlanmış kategorilerden hangisine atayacağını inceler.

Genelde, her bir bağıntı veya sınıflandırma kuralı kullanıcının belirlediği 2 sınırlayıcı şartı sağlar. Bunlar, destek ve güven değerleridir. $X \Rightarrow Y$ şeklindeki bir kuralın destek değeri; X ve Y niteliklerini bir arada içeren kayıtların oranını gösterirken; güven değeri X niteliğini içeren kayıtlarda Y niteliğinin de bulunma oranını tanımlar. Böylece klasik bir kural bulma işleminde hedef, kullanıcının belirlediği minimum destek ve minimum güven değerlerini aşan bütün bağıntı veya sınıflandırma kurallarını bulmaktır. Bununla birlikte, gerçek dünya uygulamalarında, farklı veritabanlarında madencilik işlemi farklı minimum destek ve minimum güven değerlerine gerek duyar. Eğer kullanıcının veri tabanındaki niteliklerin desteği hakkında bir ön bilgisi yoksa kullanıcının uygun bir minimum destek değeri belirlemesi mümkün değildir. Bundan dolayı, klasik bağıntı ve sınıflandırma kuralları çıkarma yaklaşımları bu konuda bir yetersizlik arz eder. Her ne kadar son zamanlarda yukarıda adı geçen eşik değerleri veritabanına göre uygun bir şekilde ayarlanmak için birkaç çalışma yapılsa da bu işlemler madencilik sürecine ek bir yük getirir.

Apriori benzeri algoritmalarda uygun eşik değerlerini vermek oldukça önemlidir. Öyle ki, minimum destek çok büyük seçilirse, birçok ilginç kural budanmış olur hatta kullanıcıya sunulacak kural bulunmayabilir. Bunun yanında, minimum destek değeri çok küçük seçilirse, birçok ilginç olmayan yoğun nesne kümesi ve buradan kural üretilebilir.

Bu da sistem performansını önemli ölçüde düşürür. Farklı veritabanlarının farklı minimum destek değerlerine sahip olması gerektiği aşağıdaki iki örnekle daha iyi açıklanabilir:

Örnek 1: Winconsin Göğüs Kanseri veritabanı 699 kayıt içerir. 2-yoğun nesne kümelerinin maksimum desteği 0,6366'dır. Bu veritabanının bütün yoğun nesne kümelerinin desteği [0,0014, 0,8283] aralığında dağılmıştır (http://archive.ics.uci.edu/ml/datasets.html).

Örnek 2: JASA veri arşivindeki Tümörün yeniden oluşması veri kümesi 87 kayıt içerir (http://lib.stat.cmu.edu/jasadata). 2-yoğun nesne kümelerinin maksimum desteği 0,3218'dir. Bu veritabanının bütün yoğun nesne kümelerinin desteği ise [0,0115, 0,5862] aralığında gezer (http://lib.stat.cmu.edu/jasadata).

Örnek 1 için minimum destek=0,9 olarak belirlenirse hiçbir yoğun nesne kümesi bulunmayacakken, minimum destek=0,03 olduğunda 4092 yoğun nesne kümesi elde edilecektir. Örnek 2 için, minimum destek=0,7 iken hiçbir yoğun nesne kümesi çıkarılamazken, bu değer 0,03'e düşürüldüğünde 104 yoğun nesne kümesi bulunacaktır.

Benzer bir problem tıbbi veritabanlarında bilgi keşfetmede ortaya çıkar. Örnek veritabanı hangi belirti hangi hastalıkla ilgilidir bilgisini tutarsa, birçok önemli belirti ve hastalık için tıbbi kayıtlar yoğun olmayabilir. Örnek olarak; grip şiddetli akut solunum sendromundan çok daha sık görünür ve her iki hastalık da ateş ve sürekli öksürük belirtilerini taşır. Eğer minimum destek yüksek olarak ayarlanırsa, "grip $\Rightarrow$ateş, öksürük" kuralı bulunabilse de "şiddetli akut solunum sendromu $\Rightarrow$ateş, öksürük" kuralı asla elde edilemeyecektir. Bu kuralın ortaya çıkabilmesi için minimum destek çok düşük bir değere çekilmelidir. Fakat bu durum aynı zamanda anlamsız birçok kuralın bulunmasına da neden olur. Bu demektir ki veritabanındaki nesnelerin dağılımları ancak önceden bilinirse kullanıcının belirlediği minimum destek değeri ilgili veritabanı için uygun olur. Bu da veritabanından bağımsız minimum destek değerlerinin belirlenmesi gerektiği sonucunu ortaya çıkarır.

Bir veri madenciliği sistemi geniş bir kural kümesini çıkarmasına rağmen, bu kuralların çoğu kullanıcının ilgisini çekmez. Gereksiz olanları budayabilmek için araştırmacılar bağıntı kurallarının ilginçliği için değişik kriterler önermişlerdir. Bir kuralın ilginçliği objektif veya sübjektif olarak değerlendirilebilir. Bir objektif ölçü kuralın yapısını, tahmini performansını ve istatistiksel önemini analiz ederken [1-3], sübjektif ölçüler kullanıcının doğrudan bir müdahalesini ihtiva eder. Böylece hangi kuralın ilginç yada ilginç olmadığı açık bir şekilde kullanıcı tarafından belirlenir [4, 5].

Bir kuralın anlaşılırlığı genellikle sınıflandırma kural üretimi için kullanılır [6]. Buna göre bir kuraldaki niteliklerin sayısı ne kadar az ise, kural o kadar anlaşılırdır. Ghosh ve Natt [7] bağıntı kurallarında anlaşılırlık için bir ifade tanımlamışlardır.

1.1. Tezin Amacı

Bu tez çalışmasının birinci amacı, yukarıda verilen problemlerin üstesinden gelebilmek için yeni bir yaklaşım uygulaması önermektir. Bu uygulama yaklaşımı, çok amaçlı genetik algoritma tabanlı olup 3 farklı amaç kullanılmaktadır. Sınıflandırma kurallarının çıkarılmasında amaç kural sayısı az, kuralların benzerliği düşük ve sınıflandırma doğruluğu yüksek sınıflandırıcılar elde etmektir. Çok amaçlı genetik algoritmalarda sayısal bir değere sahip uygunluk fonksiyonu yerine, kullanıcının tercihine bırakılan bir uygunluk dizisi mevcuttur.

Tezin diğer amacı ise, aynı yöntemi farklı çok amaçlar ile birliktelik kuralları çıkarmak için uygulamaktır. Bu sayede kullanıcının bir müdahalesi olmaksızın en uygun kurallar keşfedilmiş olacaktır.

1.2. Tezin Organizasyonu

Bu tez çalışmasının bundan sonraki bölümleri aşağıdaki gibi düzenlenmiştir. İkinci bölümde veri madenciliği, birliktelik kuralları ve sınıflandırma hakkında temel kavramlar anlatılmıştır. Üçüncü bölümde genetik algoritma, genetik algoritma süreci ve çok amaçlı genetik algoritmalar verilmiştir. Dördüncü bölümün birinci kısmında önerilen çok amaçlı genetik algoritma ile şizofreni hastalarının suç işleme durumu ile sosyodemografik verileri arasındaki sınıflandırma kurallarını nasıl bulduğu anlatılmıştır. Dördüncü bölümün ikinci kısmında aynı yöntem şizofreni hastalarının biyokimya verilerine uygulanmıştır. Dördüncü bölümün üçüncü kısmında ise sosyodemografik ve biyokimyasal veriler bir araya getirilerek iki veri kümesi arasında birliktelik kuralının çıkarılması için yeni çok amaçlı bir genetik algoritma önerilmiştir. Beşinci bölümde de tez çalışması ile elde edilen sonuçlar verilmiştir.

2. VERİ MADENCİLİĞİ

Veri madenciliği, organizasyonların karar aşamaları için yeni bilgiler üreten ya da gelecekle ilgili tahminler ve planlar yapılabilmesini sağlayan bir dizi teknikler ve anlayışlar bütünü olarak tanımlanmaktadır [8].

Karar aşamalarında çok kritik bazı bilgiler vardır ki, sonuçların etkileri bu bilgilerin doğruluğuyla orantılıdır. Birçok durumda cevabı tam olarak verilemeyen sorular doğrultusunda karar verilebilmektedir. Müşterilerimizin ilgi alanları, bize karşı olan bakış açıları, rakip firmalara olan ilgileri, markalarımıza olan bağlılıkları, gelir düzeyleri gibi bilgiler onlara sağladığımız mal ve hizmetlerin kalitesi üzerinde çok net etkiler yapmaktadır. Bu tür bilgiler teorik olarak her ne kadar sistemlerimizde kayıt altında olsa da, kullanılabilir bir şekilde açık ve net cevaplara ulaşabilmemiz mevcut sistemlerle imkânsız denecek kadar zordur.

Eskiden süper marketteki kasa basit bir toplama makinesi olup, müşterinin o anda satın almış olduğu malların toplamını hesaplamak için kullanılırdı. Günümüzde ise kasa yerine kullanılan satış noktası terminalleri sayesinde bu hareketin bütün detayları saklanabiliyor. Saklanan bu binlerce malın ve binlerce müşterinin hareket bilgileri sayesinde her malın zaman içindeki hareketleri ve eğer müşteriler bir müşteri numarası ile kodlanmışsa bir müşterinin zaman içindeki verilerine ulaşmak ve analiz etmek olasıdır.

Veri kendi başına değersizdir [9] ve istenilen amaç doğrultusunda bir bilgidir. Bilgi bir amaca yönelik işlenmiş veridir. Veriyi bilgiye çevirmeye veri analizi denir. Bilgi de bir soruya yanıt vermek için veriden çıkarılır. Veri sadece sayılar veya harflerden oluşmaz. Veri, sayı ve harfler ile onların anlamından oluşur, bu şekildeki veriye meta veri denir.

Çok büyük veri yığınları altında saklı olan bu bilgilere ulaşmak için uzun yıllar boyunca yapılan çalışmaların neticesinde bir dizi yöntem geliştirilmiştir. Veri madenciliği uzun yıllardır, özellikle batı ülkelerinde üzerinde çalışılan bir konu olmasına rağmen, gerçek hayatta yazılım endüstrisinin son yıllarda üretmiş olduğu ileri teknoloji ürünü yazılımlar ile kullanılmaya başlanmıştır [10].

Veri madenciliği Biyomedikal ve DNA veri analizlerinde, finans veri analizlerinde, perakende satış verilerinde, telekomünikasyon endüstrisinde, astronomi ve birçok alanda uygulanmaktadır [8, 11]. Bu uygulamaların başlıca alanları için aşağıda örnek uygulamalar verilmiştir [9].

Bağıntı: "Çocuk bezi alan müşterilerin %30'u içecek de satın alır."
Sepet analizinde müşterilerin beraber satın aldığı malların analizi yapılır. Buradaki amaç mallar arasındaki pozitif veya negatif korelasyonları bulmaktır. Çocuk bezi alan müşterilerin mama da satın alacağını veya içecek satın alanların cips de alacağını tahmin edebiliriz. Ancak otomatik bir analiz bütün olasılıkları göz önüne alır ve kolayca düşünülemeyecek olan çocuk bezi ve içecek arasındaki bağıntıları da bulur.

Sınıflandırma: "Genç kadınlar küçük araba satın alır. Yaşlı, zengin erkekler büyük, lüks araba satın alır."
Amaç bir malın özellikleri ile müşteri özelliklerini eşlemektir. Böylece bir müşteri için ideal ürün veya bir ürün için ideal müşteri profili çıkarılabilir. Örneğin bir otomobil satıcısı şirket geçmiş müşteri hareketlerinin analizi ile yukarıdaki gibi iki kural bulursa genç kadınların okuduğu bir dergiye reklâm verirken küçük modelinin reklâmını verir.

Regresyon: "Ev sahibi olan, evli, aynı iş yerinde beş yıldan fazladır çalışan, geçmiş kredilerinde geç ödemesi bir ayı geçmemiş bir erkeğin kredi notu 825'dir."
Başvuru puanlamada bir finans kurumuna kredi için başvuran kişi ile ilgili finansal güvenilirliğini notlayan örneğin 0 ile 1000 arasında bir puan hesaplanır. Bu not kişinin özellikleri ve geçmiş kredi hareketlerine dayanılarak hesaplanır.

Zaman İçinde Sıralı Örüntüler: "İlk üç taksitinden iki veya daha fazlasını geç ödemiş olan müşteriler %60 olasılıkla kanuni takibe gider."
Davranış puanı, başvuru puanından farklı olarak kredi almış ve taksitleri ödeyen bir kişinin sonraki taksitlerini ödeme/geciktirme davranışını notlamayı amaçlar.

Benzer Zaman Sıraları: "X şirketinin hisse fiyatları ile Y şirketinin hisse fiyatları benzer hareket ediyor."
Amaç zaman içindeki iki hareket serisi arasında bağıntı kurmaktır. Bunlar iki malın zaman içindeki satış miktarları olabilir. Örneğin dondurma satışları ile kola satışları arasında pozitif, dondurma satışları ile salep satışları arasında negatif bir bağıntı beklenebilir.

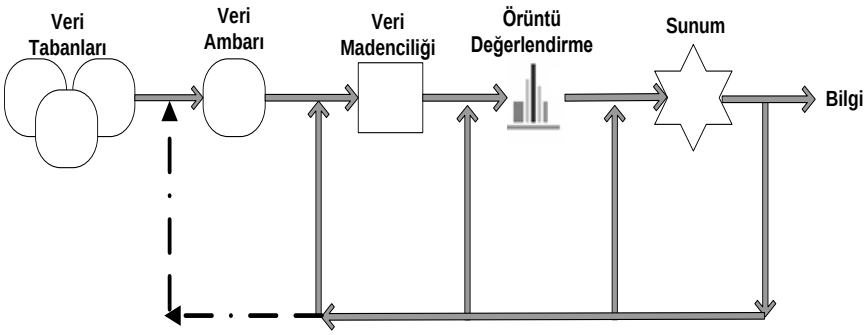
İstisnalar (Fark Saptanması): "Normalden farklı davranış gösteren müşterilerim var mı?"

Amaç önceki uygulamaların aksine kural bulmak değil, kurala uymayan istisnai hareketleri bulmaktır. Bu da olası sahtekârlıkların saptanmasını sağlar. Örneğin Visa kredi kartı için yapılan CRIS sisteminde bir yapay sinir ağı kredi kartı hareketlerini takip ederek, müşterinin normal davranışına uymayan hareketler için müşterinin bankası ile temasa geçerek müşteri onayı istenmesini sağlar.

Döküman Madenciliği: "Arşivimde (veya internet üzerinde) bu dokümana benzer hangi dokümanlar var?"
Amaç dokümanlar arasında ayrıca elle bir tasnif gerekmeden benzerlik hesaplayabilmektir. Bu genelde otomatik olarak çıkarılan anahtar sözcüklerin tekrar sayısı sayesinde yapılır.

2.1. Veritabanlarından Bilgi Keşfi Süreci

Veri madenciliği veritabanlarından bilgi keşfi (VTBK) işleminin temel bileşenlerinden birisini oluşturmakla beraber VTBK veri madenciliği dışında kendi içerisinde farklı süreçleri barındırmaktadır. Genel olarak veri tabanından bilgi keşfi işlemi Şekil 2.1'de görüldüğü gibi 5 aşamadan oluşur [12].



Şekil 2.1. Veri tabanından bilgi keşfi aşamaları

VTBK sürecini oluşturan adımlar aşağıdaki şekilde sıralanmaktadır:

a) Veri Önişlemleri :

Bu aşamada veri içerisindeki gürültüler, tutarsızlıklar ve düzensizlikler giderilir. Veri tabanında eksik veri varsa bu veriler düzenlenir. Bu işleme veri temizleme denir. İkinci

aşamada tutarsızlıklar ve gürültüden arındırılmış veriler birleştirilir. Bu işleme veri birleştirme denir. Bu aşamada farklı kaynaklardan toplanan verilerin tek bir veri ambarında toplanabilmesi için gerekli genellemeler ve normalizasyon işlemleri yapılır.

b) Veri Seçme ve Dönüştürme:

Bu aşamada, veri madenciliği sürecinde kullanılacak veriler ile ilgili bazı ön işlemler yapılır. Bu önişlemler genel olarak veri madenciliği konusu ile ilgili bilgi seçimi, madencilik yapılacak verinin türünün belirlenmesi, veriler arasında hiyerarşik yapının ve genellemelerin belirlenmesi ve veri madenciliği sonunda bulunacak veriler için görselleştirme ve sunum araçlarının belirlenmesi şeklindedir.

c) Veri Madenciliği:

Anlamlı veri örüntülerini ve veriler arasındaki saklı ilişkileri bulmak için çeşitli analiz yöntemleri ve algoritmaların kullanıldığı aşamadır.

d) Örüntü Değerlendirme:

İkinci aşamada belirlenen ölçümlere göre elde edilmiş bilgiyi temsil eden ilginç örüntüleri tanımlamak ve bu örüntülerin ilginçlik seviyelerinin tespit edilmesi aşamasıdır.

e) Bilgi Sunumu:

Çeşitli görselleştirme ve raporlama araçları kullanılarak elde edilen verilerin kullanıcılara ulaştırılması aşamasıdır.

Veritabanından Bilgi Keşfi (VTBK) süreci defalarca tekrar ve aşamalar arası atlamalar ve bu aşamalar arasında ileri geri hareketler içerebilmektedir. Günümüzde çoğunlukla veri madenciliği aşamasına odaklanılmaktaysa da, diğer tüm aşamalar VTBK işleminin bütünlüğü açısından en az veri madenciliği kadar önem taşımaktadır [13].

2.2. Veri Madenciliğinde Kullanılan Modeller

Veri madenciliği modelleri, tahmin edici ve tanımlayıcı olmak üzere iki kısımda incelenmektedir. Tahmin edici modeller, sonuçları bilinen verilerden hareket ederek bir model geliştirir ve kurulan bu modelden yararlanarak sonuçları bilinmeyen veri setleri için sonuç değerlerini tahmin etmeyi hedefler. Örneğin bir banka önceki dönemlerde vermiş olduğu kredilere ilişkin gerekli tüm verilere sahip olabilir. Bu verilerde bağımsız değişkenler kredi alan müşterinin özellikleri, bağımlı değişken ise kredinin geri ödenip ödenmediğidir. Bu verilere uygun olarak kurulan model, daha sonraki kredi taleplerinde müşteri özelliklerine göre verilecek olan kredinin geri ödenip ödenmeyeceğinin tahmininde kullanılmaktadır.

Tanımlayıcı modellerde ise karar vermeye öncülük etmede kullanılabilecek eldeki mevcut verilerdeki örüntülerin tanımlanması sağlanmaktadır. X-Y aralığında geliri, evi ve arabası olan, ayrıca çocukları okul çağında olan aileler ile, çocuğu olmayan ve geliri X-Y aralığından düşük olan ailelerin satın alma örüntülerinin birbirlerine benzerlik gösterdiğinin belirlenmesi tanımlayıcı modellere bir örnektir.

Veri Madenciliği modelleri gördükleri işlevlere göre:

- Sınıflama ve Regresyon,
- Kümeleme,
- Birliktelik Kuralları ve Ardışık Zamanlı Örüntüler

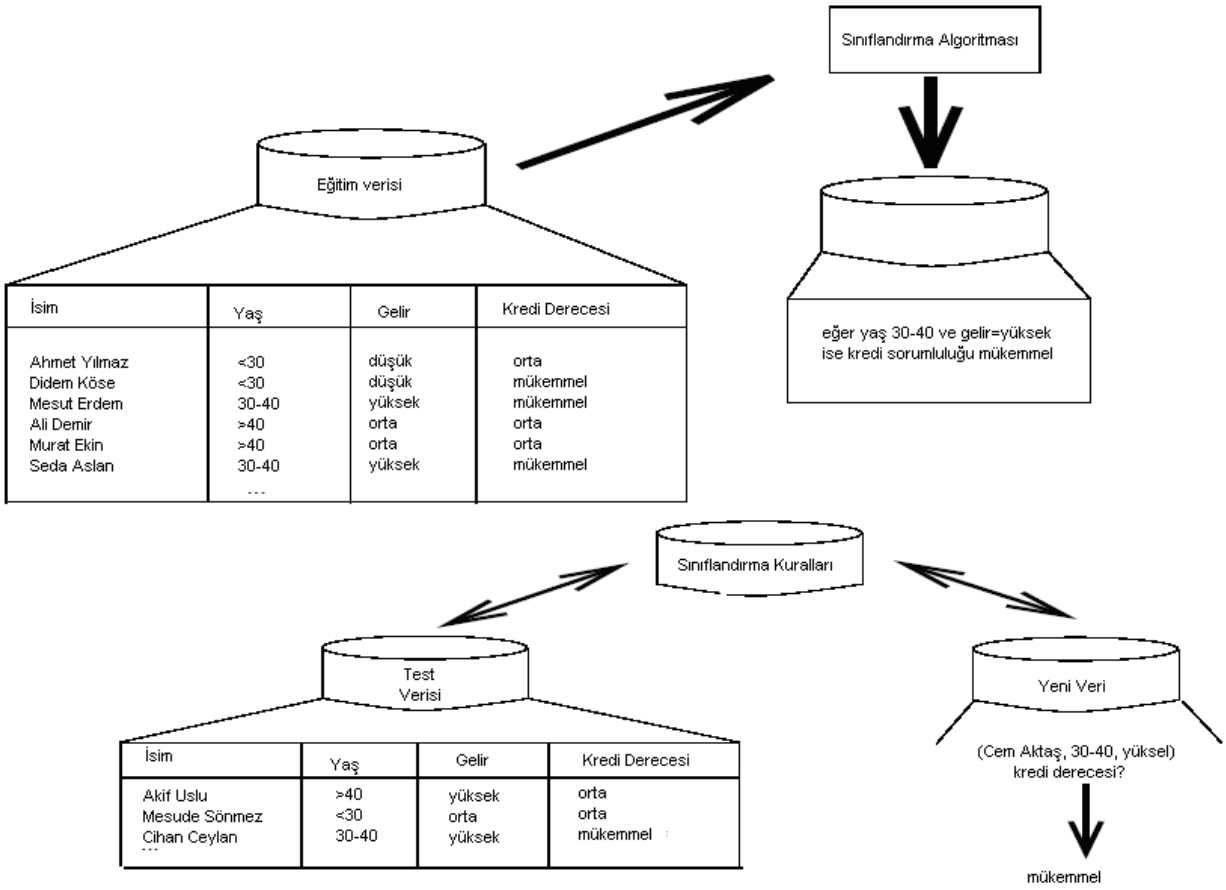
olmak üzere üç ana başlık altında incelemek mümkündür. Sınıflama ve regresyon modelleri tahmin edici, kümeleme, birliktelik kuralları ve ardışık zamanlı örüntü modelleri tanımlayıcı modellerdir.

Sınıflama ve Regresyon modelleri arasındaki temel fark, tahmin edilen bağımlı değişkenin kategorik veya süreklilik gösteren bir değere sahip olmasıdır. Bazı tekniklerde her iki modelin giderek birbirine yaklaşmasından dolayı aynı tekniklerden yararlanılması mümkün olmuştur [14].

2.3. Sınıflandırma ve Tahmin Kavramları

Veritabanları akıllı kararlar verebilmek için kullanılabilecek zengin gizli bilgiler içerirler [15]. Sınıflandırma ve tahmin, önemli veri sınıflarını tanımlamak ve gelecekteki trendi tahmin etmeye yarayacak modelleri oluşturmak için kullanılan veri analizi türleridir. Sınıflandırma kategorik etiketleri tahmin etmekte kullanılırken, tahmin modelleri ise

sürekli değer alan fonksiyonlar için kullanılırlar. Örneğin, bir sınıflandırma modeli banka kredilerinin güvenli mi yoksa riskli mi olduğunu kategorize etmek için kullanılabilir. Bir tahmin modeli de potansiyel müşterilerin gelir ve meslek bilgilerini kullanarak bilgisayar donanım harcamalarını tahmin etmek için kullanılabilir. Nörobiyoloji, istatistik, uzman sistemler ve makine öğrenmesi alanlarında birçok sınıflandırma ve tahmin algoritmaları kullanılmıştır.



Şekil 2.2. Sınıflandırma aşamaları

Veri sınıflandırması şekil 2.2.'de görüldüğü üzere iki aşamalı bir süreçtir. Birinci aşamada, önceden belirlenmiş veri sınıfları ve kavramlar kümesini ifade eden modeller oluşturulur. Model nesnelerle tanımlanan veritabanı parçacıkları analiz edilerek oluşturulurlar. Her parçacığın önceden tanımlanmış, bir nesne tarafından belirlenen bir sınıfa ait olduğu varsayılır. Sınıflandırma alanında veri parçacıkları örnekler ve nesnelerdir. Eğitim veri kümesinde yer alan parçacıklar toplu olarak analiz edilerek model

oluşturulur. Eğitim veri kümelerini oluşturan parçacıklara eğitim örnekleri denir ve örnek popülasyondan rastgele elde edilirler. Her bir eğitim örneğinin sınıf etiketi bilindiğinden bu aşama eğitimli öğrenmedir.

Tipik olarak, öğrenilmiş model sınıflandırma kuralları, karar ağaçları ve matematiksel formüllerle ifade edilir. Örneğin, müşterinin kredi bilgileri kullanılarak, kredi puanları sınıflandırma kuralları öğrenilebilir. Bu kurallar ileride karşılaşılacak veri kümelerini sınıflandırmak ve veritabanı içeriğini daha rahat anlamak için kullanılırlar.

İkinci aşamada model sınıflandırmak için kullanılır. Öncelikle sınıflandırıcının tahmin doğruluğu hesaplanır. Bu metot sınıflanarak etiketlenmiş örnekleri kullanan basit bir metottur. Bu örnekler eğitim örneklerinden bağımsız olarak ve rastgele seçilirler. Bir modelin bir test kümesindeki doğruluk oranı, model tarafından doğru olarak sınıflandırılan örnek test kümelerinin yüzdesel oranıdır. Her bir test örneği için bilinen sınıf etiketi bu örnek için öğrenilmiş modelin sınıf tahminiyle karşılaştırılır.

Sınıflandırma ve tahmin için veri hazırlanır. Aşağıda yer alan adımlar uygulanarak sınıflandırma ve tahmin sürecinin doğruluk ve etkinliği geliştirilebilir.

Veri temizliği: Veride yer alan kirliliğin giderilmesi ve eksik değerlerin giderilmesi anlamına gelir. Bütün sınıflandırma algoritmalarının veri kirliliği ve eksik veriler konusunda bazı metotlara sahip olmasına rağmen, bu adım öğrenme sürecinde karmaşıklığı azaltır.

İlişki analizi: Veri içinde yer alan nesnelerin birçoğu sınıflandırma veya tahmin ile ilgisiz olabilir. Örneğin, banka kredi başvurusunun haftanın hangi günü yapıldığı kaydı başvurunun sonucu ile ilgisizdir. Ayrıca, diğer nesneler gereksiz olabilir. Bu nedenle, gereksiz ve ilişkisi olmayan nesnelerden kurtulmak için ilişki analizi yapılabilir. Makine öğrenmesinde bu adım özellik seçilmesidir. Buna rağmen, bu nesnelerin dahil edilmesi durumunda öğrenme süreci yavaşlayabilir ve yanlış yönlendirilebilir.

Veri Dönüşümü: Veri genelleştirilerek daha yüksek seviyede kavramlarla ilişkilendirilebilir. Kavram hiyerarşileri bu amaçla kullanılabilirler. Bu sürekli değer alan nesneler için kısmen faydalıdır. Örneğin, gelir nesnesinin sayısal değerleri düşük, orta ve yüksek gibi kesikli sınıflara genelleştirilebilirler. Benzer şekilde, nominal değerler alan sokak gibi nesneler, şehir gibi daha yüksek

kavramlara genelleştirilebilirler. Genelleştirme orijinal veriyi sıkıştırdığı için daha az işlemle öğrenme süreci yürütülebilir.

Normalizasyon: Bir nesneye ait bütün değerleri belirli bir tanımlanmış aralıkta yer alacak şekilde boyutlandırılmasıdır. Uzaklık ölçümü metotları nesnelerin başlangıçta verilen küçük aralık değerlerini kullanarak çok yüksek değerler almasını engeller.

Sınıflandırma ve tahmin metotları aşağıda yer alan kıstaslara göre karşılaştırılabilirler:

Tahmin Doğruluğu: Modelin yeni veya daha önce hiç görülmemiş bir veriye ait sınıf değerini doğru olarak tahmin etme yeteneğini ifade eder.
Hız: Modelin kullanımı ve oluşturulması ile ilgili işlem maliyetlerini ifade eder.
Sağlamlık: Modelin kirli veya eksik değerler içeren verilerle doğru tahminler yapabilme yeteneğidir.
Ölçeklenebilirlik: Çok sayıda veri ile çalışırken verimli olma yeteneğidir.
Anlaşılabilirlik: Öğrenilmiş modelin anlaşılırlığı ve kavrayışını ifade eder.

2.4. Birliktelik Kuralları

Birliktelik kuralları, bir veri tabanında sık tekrarlanan nesneler arasında bir bağıntı bulan veri madenciliği yöntemlerinden biridir. Birliktelik kuralları, büyük veri kümeleri arasında birliktelik ilişkileri bulurlar. Toplanan ve depolanan verinin her geçen gün artmasından dolayı, şirketler veritabanlarındaki saklı ilişkileri ortaya çıkarmayı hedeflemektedirler. Büyük miktardaki mesleki işlem kayıtlarından ilginç birliktelik ilişkilerini keşfetmek, şirketlerin karar alma işlemlerini daha verimli hale getirebilmektedir

Birliktelik kurallarının kullanıldığı en tipik örnek Pazar sepeti analizi uygulamasıdır. Bu işlem, müşterilerin yaptıkları alışverişlerdeki ürünler arasındaki birliktelikleri bularak müşterilerin satın alma alışkanlıklarını analiz eder. Bu bilgi ışığında market yöneticileri müşterilerin satın alma alışkanlıklarına göre daha etkin satış stratejileri geliştirmesine imkan tanımış olur.

Örneğin bir müşteri süt satın alıyorsa, aynı alışverişte sütün yanında ekmek alma olasılığı nedir? Bu tip bir bilgi ışığında rafları düzenleyen market yöneticileri ürünlerindeki

satış oranını arttırabilirler [16]. Birliktelik kuralları eş zamanlı olarak gerçekleşen ilişkilerin tanımlanmasında kullanılır. Örneğin, içecek satın alan müşterilerin %75 ihtimalle patates cipsi de almaları veya ekmek ve yağ alanların %90'ının süt de satın almaları birliktelik kuralları kapsamında tespit edilebilir.

Ardışık zamanlı örüntüler ise birbiri ile ilişkisi olan ancak birbirini izleyen dönemlerde gerçekleşen ilişkilerin tanımlanmasında kullanılır. X ameliyatı yapıldığında, 15 gün içerisinde % 45 ihtimalle Y enfeksiyonu oluşması, çekiç alan bir müşterinin ilk üç ay içerisinde % 15, bu dönemi izleyen süre içerisinde ise % 10 ihtimalle çivi alması ardışık zamanlı örüntülerdir.

Sepet analizinde amaç, değişkenler arasındaki ilişkiyi bulmaktır. Bu ilişkilerin bilinmesi şirketin kârını arttırmak için kullanılabilir. Eğer X malını alanların Y malını da çok yüksek bir olasılıkla aldıklarını biliyorsanız veya bir müşteri X malını alıyor ama Y malını almıyorsa o potansiyel bir Y müşterisidir. Sepet analizi günlük işlemler sonucu elde edilen verilerden anlamlı bağıntılar çıkarmada kullanılır. "Eğer A malını alıyorlarsa % x ihtimalle B malını almaya da meyillidirler" şeklinde bir sonuç A malını satan bir mağaza için çok faydalı bir bilgi olabilir. Sepet analizi uygulamaları; çapraz satış, mağaza raflarının düzenlenmesi, katalog dizaynı ve fiyatlandırma gibi alanlarda kullanılmaktadır.

Sepet analizinde mallar arasındaki bağıntı, destek ve güven kriterleri ile hesaplanır. Destek ve güven kriterlerinin tanımları aşağıdaki gibidir.

$$Destek: P(X\ ve\ Y) = \frac{X\ ve\ Y\ mallarını\ satın\ alan\ müşterilerin\ sayısı}{Toplam\ müşteri\ sayısı} \quad \textbf{(2.1)}$$

$$Güven: P(X/Y) = \frac{P(X\ ve\ Y)\ (X\ ve\ Y\ mallarını\ satın\ alanların\ sayısı)}{P(Y)\ (Y\ malını\ satın\ alanların\ sayısı)} \quad \textbf{(2.2)}$$

Destek kriteri, veride mallar arasındaki bağıntının ne kadar sık olduğunu, güven kriteri ise Y malını almış olan bir kişinin hangi olasılıkla X malını alacağını söyler. İki ürünün satın alınmasındaki bağıntının önemli olması için hem destek, hem de güven kriterinin olabildiğince yüksek olması gerekmektedir.

Şekil 2.3'ün ilk tablosunda; bir markette farklı zamanlarda yapılan 4 farklı işlem görülmektedir. İkinci tabloda ise her ürünün ve bazı ürün birlikteliklerinin tekrar edilme yüzdeleri verilmiştir. Bu tekrar edilme oranları, bize ürünlerin destek değerini vermektedir. Şeker ve Yağ ürünlerinin destek değerlerinin referans destek değerinden düşük olmaları, bu ürünleri içeren ürün birlikteliklerinin ihmal edilebileceği anlamına gelmektedir. Son tabloda ise destek değeri referans destek değeri olarak kabul edilen % 50 olan ve en az iki üründen oluşan ürün birliktelikleri ele alınarak bunların güven değerleri sonucu görülmektedir. Burada dikkat edilirse Süt → Yumurta ilişkisi % 66 güven değerine sahip iken Yumurta → Süt ilişkisi % 100 güven değerine sahiptir. Çünkü yumurta satın alınma durumu, süt satın alınma durumlarının sadece % 66'sında gerçekleşirken süt satın alınması ise yumurta alınma durumlarının tümünde gerçekleşmiştir. Bu nedenle Yumurta → Süt birlikteliğinde % 100 güven değeri elde edilmektedir. Bu örnekten de anlaşıldığı gibi veritabanı üzerindeki bir ürün gurubunun tekrar edilme miktarı destek değerini, bir ürün grubunu oluşturan herhangi bir ürünün veritabanındaki işlem sayısı ile destek değerine sahip olan ürün birlikteliği içerisindeki yer alış sayısına oranı da güven değerini vermektedir.

Sepet analizi tekniğinin güçlü ve zayıf yönlerini sıralamak gerekirse;

Güçlü yönleri;

- Denetimsiz veri madenciliği yöntemini başarı ile uygular,
- Anlaşılabilir sonuçlar verir

Zayıf yönleri;

- İleri derecede bilgisayar performansına bağımlıdır
- Arasında bağıntı oluşturacak ürünlerin doğru olarak seçilmesi, analiz süreci için oldukça önemlidir [17].

İşlemler	Satın Alınan Ürünler
1	Süt, Ekmek, Yumurta
2	Süt, Yumurta
3	Süt, Şeker
4	Ekmek, Yağ

Tekrarlanan Ürün	Destek Değeri
Süt	% 75
Ekmek	% 50
Yumurta	% 50
Şeker	% 25
Yağ	% 25
Süt, Yumurta	% 50

Tekrarlanan Ürün	Destek Değeri	Güven Değeri
Süt → Yumurta	% 50	% 66
Yumurta → Süt	% 50	% 100

Şekil 2.3. İşlemler, satın alınan ürünler,destek değerleri ve güven değerleri

2.4.1. Apriori Algoritması

Literatürde birliktelik kuralı çıkaran değişik algoritmalar bulunmaktadır. Apriori Algoritması, birliktelik kuralı çıkarım algoritmaları içerisinde en fazla bilinen algoritmadır [18]. Bu algoritmada sık geçen öğe kümelerini bulmak için birçok kez veritabanını taramak gerekir. İlk taramada bir elemanlı minimum destek değerini sağlayan sık geçen öğe kümeleri bulunur. İzleyen taramalarda bir önceki taramada bulunan sık geçen öğe kümeleri aday kümeler adı verilen yeni potansiyel sık geçen öğe kümelerini üretmek için kullanılır. Aday kümelerin destek değerleri tarama sırasında hesaplanır ve aday kümelerinden minimum destek değerini sağlayan kümeler o geçişte üretilen sık geçen öğe kümeleri olur. Sık geçen öğe kümeleri bir sonraki geçiş için aday küme olurlar. Bu süreç yeni bir sık geçen öğe kümesi bulunmayana kadar devam eder [19].

Apriori algoritmasında önce aday öğe kümeleri oluşturulur. Bu kümeler potansiyel olarak sık geçen öğe kümeleridir. C ile gösterilir ve C[1], C[2], C[3], ….C[k] olarak k-öğe kümesini oluştururlar. Her C[k] öğe kümesi c[k-1] öğe kümesini içerir ve C[1] < C[2] < C[3]< ….< C[k] şeklinde sıralıdır. Sık geçen k-öğe kümeleri ise L ile gösterilir ve minimum destek kıstaslarını sağlarlar.

Örnek: Tablo 2.1'de bir firmada satın alınmış ürünlerle ilgili bir veritabanı görülmektedir. Bu veritabanını D olarak adlandıralım. Veritabanında 9 adet işlem

görüldüğüne göre |D| = 9 denilebilir. Bu durumda D veritabanı üzerinde Apriori algoritması kullanılarak sık kullanılan nesnelerin nasıl bulunabileceğini Şekil 2.4'te görebiliriz [20].

Tablo 2.1. Bir marketten satın alınan ürünler listesi

İşlemler	**Satın Alınan Ürün Listesi**
T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3
T8	I1, I2, I3, I5

1) Algoritmanın ilk iterasyonunda her öğe, yani 1 elemanlı öğeler kümesi C1 olarak elde edilir. Algoritma, her öğenin sayısını bulmak için veritabanını basitçe taramaktadır.
2) Bu veritabanı için minimum destek sayısı 2 olarak belirlenmiştir. Minimum destek değeri 2/9 = % 22 dir. Buradan 1 elemanlı öğeler kümesi olan L1 elde edilebilir. C1 deki tüm öğeler minimum destek değerini sağlamakta ve L1 elde edilmektedir.
3) L2'yi bulmak için önce 2 elemanlı aday öğe kümesini elde etmek gerekir ve L1xL1 den C2 elde edilir. L1'in 2'li kombinasyonları C2'nin eleman sayısını vermektedir.
4) Daha sonraki işlem D'yi taramak ve C2 deki her aday öğenin minimum destek değerini bulmaktır. Şekil 2.4'de ortadaki 2. tabloda bu görülmektedir.
5) C2'de minimum destek değerine sahip olmayan aday öğelerin çıkarılması ile L2 öğe kümesi elde edilir.
6) 3 elemanlı aday öğe kümesi bulunurken, normalde C3=L2xL2 ={{I1,I2,I3}, {I1,I2,I5}, {I1,I3,I5}, {I2,I3,I4}, {I2,I3,I5}, {I2,I4,I5}}olarak elde edilir. Bunların hepsi sık tekrarlanıyor görülebilir, ancak Apriori algoritması özelliğine göre son dört tanesi sık tekrarlanan değildir. Çünkü bunların ikili alt kümesi L2'de bulunmamaktadır. Bu nedenle son dördü C3'ten çıkarılır. Böylece D'yi taramak ve L3'ü belirlemek için bazı işlemler boşu boşuna yapılmamış olacaktır.

Tablo 2.2. Apriori algoritması ile aday öğelerin ve sık kullanılan öğelerin belirlenmesi (Minimum destek değeri 2 alınmıştır). a) 1-aday öğe ve yoğun öğe kümesi b) 2-aday öğe ve yoğun öğe kümesi c) 3-aday öğe yoğun öğe kümesi.

(a)

C1 **L1**

Her bir aday sayısı D veritabanından sayılır. →

Ürünler	Destek Sayısı
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Her adayın destek değerleri min. destek değerleriyle karşılaştırılır. →

Ürünler	Destek Sayısı
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

(b)

C2 **C2** **L2**

L1'den C2 aday kümeleri üretilir. →

Ürünler
{I1,I2}
{I1,I3}
{I1,I4}
{I1,I5}
{I2,I3}
{I2,I4}
{I2,I5}
{I3,I4}
{I3,I5}
{I3,I5}

Her bir aday sayısı D veritabanından sayılır →

Ürünler	Destek Sayısı
{I1,I2}	4
{I1,I3}	4
{I1,I4}	1
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2
{I3,I4}	0
{I3,I5}	1
{I3,I5}	0

Her adayın destek değerleri min. destek değerleriyle karşılaştırılır. →

Ürünler	Destek Sayısı
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

(c)

C3 **C3** **L3**

L1'den C2 aday kümeleri üretilir. →

Ürünler
{I1,I2,I3}
{I1,I2,I5}

Her bir aday D'den sayılır. →

Ürünler	Destek Sayısı
{I1,I2,I3}	2
{I1,I2,I5}	2

Her adayın destek değerleri min. destek değerleriyle karşılaştırılır →

Ürünler	Destek Sayısı
{I1,I2,I3}	2
{I1,I2,I5}	2

7) C3 üzerinde minimum destek değerine sahip olmayan aday öğelerin çıkarılması ile L3 öğe kümesi elde edilir.
8) Algoritmada L3xL3'ü kullanılarak 4 elemanlı aday öğe kümesi elde edilir. Bunun sonucunda {{I1,I2,I3,I5}} elde edilmesine rağmen bu kümenin alt kümesi olan {{I2,I3,I5}} L3'te bulunmamaktadır. Bu nedenle C4=0'dır ve tüm sık kullanılan birliktelikler elde edilerek algoritma sona ermektedir

3. GENETİK ALGORİTMALAR

Genetik algoritmalar (GA) doğadaki evrim yöntemlerini kullanan bir arama yöntemidir. Genetik Algoritma tekniği, Michigan Üniversitesinde yer alan John Holland [21] ve arkadaşlarının liderliğinde yapılan çalışmalar sonucu 1970'li yıllarda ortaya çıkmış ve 1975'de Holland, Doğal ve Yapay Sistemlerin Uygulanması adlı kitabını yayınlamıştır. Mekanik öğrenme konusunda çalışan Holland, Darwin'in evrim kavramından etkilenerek canlılarda yaşanan genetik süreci bilgisayar ortamında gerçekleştirmeyi düşündü. 1985'te Holland'ın öğrencisi olarak doktorasını veren David E. Goldberg [22] adlı inşaat mühendisi, 1989'da konusunda bir klasik sayılan kitabı yayınlanana karar, genetik algoritmaların pek yararı olmayan bir araştırma konusu olduğu düşünülüyordu.

Bugün bilgisayar yöntemleri biyolojik değerlendirmeden esinlenerek evrimsel hesaplama olarak adlandırılan bir şemsiye altında gruplandırılmıştır.
Evrimsel hesaplamanın ana elemanları aşağıda tanımlanmaktadır [23].

(1) Değerlendirme stratejileri
(2) Evrimsel programlama
(3) Genetik algoritmalar

Bu üç tekniğin her biri doğal değerlendirmedeki gözlemlenen süreci taklit eder ve verilen problem için aday çözümlerin değerlendirme popülasyonlarıyla etkili arama motorları sağlar. GA'lar genel olarak evrimsel hesaplama alanındaki en göze çarpan teknik olarak düşünülebilir.

GA, evrimsel programlamanın en yaygın ve en çok kullanılan dalıdır. Türkiye dâhil dünyada pek çok araştırmacı bu konuda çalışmaktadır. Son yıllarda genetik algoritmalara ilgi büyüyerek artmaktadır.

GA, hem problem çözmek hem de modelleme için kullanılmaktadır. Günümüzde genetik algoritmaların uygulama alanları genişlemektedir. Bunlardan bazıları: Atölye Çizelgeleme, Yapay Sinir Ağları Tasarımı, Görüntü Kontrolü, Elektronik Devre Tasarımı, Optimizasyon, Uzman Sistemler, Paketleme Problemleri, Makine ve Robot Öğrenmesi, Gezgin Satıcı Problemi, Ekonomik Model Çıkarma sayılabilir [24]. Canlıların yapılarında var olan bir takım özellikler sanal ortamlarda taklit edilerek modeller geliştirilmeye ve bu

modellerle de karşılaşılan problemlere çözümler bulunmaya çalışılmaktadır. GA, geleneksel sezgisel yöntemlerden daha etkili ve çözüm yaklaşımında yapılacak küçük değişikliklerle halledilebildiklerinden dolayı da daha esnektir. Bu sebeple, GA araştırıcıların ilgisini çekmektedir. Bilindiği üzere optimizasyondaki temel amaç optimal bir noktaya ulaşabilmek, daha doğrusu mümkün oldukça yaklaşmaktır. Bunu gerçekleştirmek için bilinen pek çok klasik yöntem vardır. Bu yöntemlerin başarısı optimum noktaya ulaşıp ulaşmadıkları veya ne kadar ulaşabildikleri ile ölçülür. Genetik algoritmalar, klasik optimizasyon algoritmalarından dört temel noktada ayrılır:

- GA, bir noktadan değil bir arama uzayını kullanarak aramaya başlar. Yani GA çalışmaya başladığında popülasyonda olası çözümler mevcuttur.
- GA, olasılık kurallarına göre çalışır.
- GA, her problem yapısına göre parametrelerin kodlanmış haliyle çalışır.
- GA, zor matematiksel işlemler kullanmaz. Her problemin kendi yapısına uygun olarak oluşturulan bir değerlendirme fonksiyonu kullanır.

GA'ların parametreleri; çaprazlama oranı, mutasyon oranı, popülasyon büyüklüğü, seçim, kodlama, çaprazlama ve mutasyon tipi gibi genel parametrelerdir. Çaprazlama oranı yüksek olmalıdır. Buna karşılık mutasyon oranı da çok düşük olmalıdır.

Seçim için genellikle rulet tekerleği kullanılır, bunun yanı sıra sıralı seçimi, kararlı durum ve seçkinlik gibi seçim yöntemleri de kullanılmaktadır.

GA yöntemi, yıllar boyu süregelen genetik mühendisliği ve biyoloji alanında yapılan çalışmalar sonucu ortaya atılmış bir tekniktir ve her sisteme uygulanamayabilir [25]. Standart bir GA yönteminin adımları aşağıdaki gibi verilebilir:

1. Başlangıç: n adet kromozom içeren popülâsyonun oluşturulması (problemin potansiyel çözümü),
2. Uygunluk: her x kromozomu için uygunluğun f(x) değerlendirilmesi,
3. Yeni popülasyon: Yeni popülasyon oluşuncaya kadar aşağıdaki adımların tekrar edilmesi,
    3.1. Seçim: İki ebeveyn kromozomun uygunluklarına göre seçimi (daha iyi uyum seçilme şansını artırır),

3.2. Çaprazlama: Yeni bir fert oluşturmak için ebeveynlerin bir çaprazlama olasılığına göre çaprazlanması. Eğer çaprazlama yapılmazsa, yeni fert anne veya babanın kopyası olacaktır.

3.3. Mutasyon: Yeni ferdin mutasyon olasılığına göre kromozom içindeki konumu değiştirilir.

3.4. Ekleme: Yeni bireyin yeni popülasyona eklenmesi.

4. Değiştirme: Algoritmanın yeniden çalıştırılmasında oluşan yeni popülasyonun kullanılması,

5. Test: Eğer sonuç tatmin ediyorsa, algoritmanın sona erdirilmesi ve son popülasyonun çözüm olarak sunulması.

6. Döngü: 2. adıma geri dönülmesi.

3.1. Genetik Operatörler

Kullanılan genetik operatörler, var olan popülasyon üzerine uygulanan işlemlerdir. Bu işlemlerin amacı daha iyi özelliğe sahip yeni nesiller üretmek ve arama algoritmasının alanını genişletmektir. Farklı uygulamalarda farklı operatörler kullanılmakla birlikte GA'da üç standart operatör kullanılır. Bu operatörler:

- Seçme ve yeniden üretim
- Çaprazlama
- Mutasyon
- Çaprazlama ve mutasyon olasılığı

(i) Seçme ve yeniden üretim, nesil üretimi:

Genetik algoritmalarda bir sonraki nesli oluşturmak için iyi olan bireylerin seçilmesi gerekmektedir. Popülasyondan iyi olan genlerin seçimi için birkaç yöntem vardır. Bu yöntemler içinden en çok kullanılanı, bireylerin uygunluk değerlerine bakarak en iyi uygunluk değerine sahip olan bireyleri seçmektir. Seçilen bireyler kendi aralarında çaprazlama işlemine tabi tutulur. Çaprazlama işleminden sonra elde edilen evlat bireyler daha kötü uygunluk değerine sahip bireylerin yerine yerleştirilir. Diğer seçim yöntemleri ise şunlardır:

- Rulet tekerleği seçimi
- Sıralı seçim
- Turnuva seçimi
- Kararlı durum seçimi
- Oransal seçim
- Seçkinlik

Rulet Tekerleği Seçimi: Bu seçim yönteminde tüm bireylerin uygunluk değerleri bir tekerleğe yerleştirilir. Uygunluk değeri yüksek olan bireylerin seçilme şansı daha yüksek, uygunluk değeri kötü olan bireylerin seçilme şansı ise daha düşüktür. Yalnız burada amaç kötü uygunluğa sahip bireyleri tamamen yok etmek değil, onların da çaprazlamaya az da olsa katılma ihtimalinin olmasıdır.

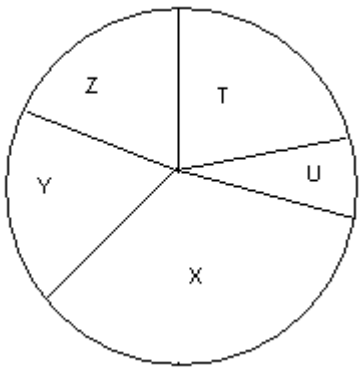
$$P_{seçilen} = f_i / \sum_{i=1}^{N} f_k \qquad \textbf{(3.1)}$$

Bağıntısıyla, bireyin seçilme olasılığı hesaplanabilir. Burada;

f_i: Bireyin popülasyondaki uygunluk değeri

f_k: popülâsyondaki toplam uygunluk değeridir.

Şekil 3.1'deki rulet tekerleği yerleşiminde, x bireyinin seçilme şansı diğerlerinden daha yüksektir. Çünkü sahip olduğu uygunluk değeri diğer bireylerden daha iyidir.



Şekil 3.1. Rulet tekerleği uygunluk yerleşimi

Sıralı Seçim: Rulet seçimi eğer uygunluk çok fazla değişiyorsa sorun çıkartabilir. Örneğin en iyi kromozomun uygunluğu %90 ise diğer kromozomların seçilme şansı azalacaktır. Bunu önlemek için sıralı seçim kullanılabilir. Sıralı seçimde en kötü uygunlukta olan kromozoma 1 değeri sonrakine 2 değeri verilir ve böylelikle seçilmede bunlara öncelik

tanınmış olur. En iyi kromozom ise N değerini alacaktır (N popülasyondaki kromozom sayısıdır). Bu şekilde kötü bireylerin de seçilme şansı artar fakat bu çözümün daha geç yakınsamasına neden olabilir.

Turnuva Seçimi: Turnuva seçiminde bir eşleştirme havuzu vardır ve rasgele seçilen genler bu eşleştirme havuzuna atılır.
Popülâsyonda bulunan bireyler içerisinden en yüksek uygunluk değerine sahip olanlar eşleştirme havuzuna alınırlar ve geriye kalan bireyler içerisinden yeni bir turnuva seçimi daha yapılır. Turnuvanın boyutu kadar bu işleme devam edilir. Sonuçta havuzda oluşan bireylerin uygunluk değeri yükseltilmiş olur.

Kararlı Hal Seçimi: Bu seçme yöntemi ise popülasyonda bulunan en iyi bireylerin kötü olan bireylerin yerine yerleştirilmesidir. Burada iyi olan bireyler herhangi bir işleme tabi tutulmaz, iyi olan birey varlığını koruyarak bir sonraki nesilde hayatını devam ettirir.

Oransal Seçim: Bu yöntemle, bir bireyden seçilecek kopya sayısı belirlenir. Bireyin popülasyondaki uygunluk değeri, popülasyonun ortalama uygunluk değerine bölünür. Elde edilen değer kadar o birey kopyalanarak yeni nesle aktarılır.

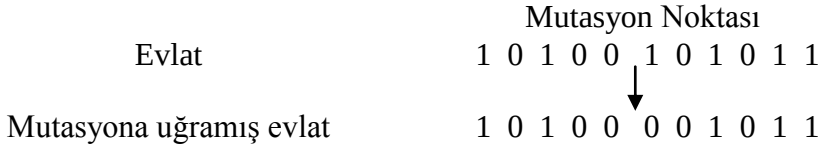
Seçkinlik : Çaprazlama ve Mutasyon yöntemleriyle yeni bir nesil oluştururken, en iyi kromozomları kaybetme olasılığımız vardır. Seçkinlik, en iyi kromozomların (ya da bir kısmının) ilk önce kopyalanıp yeni nesle aktarıldığı yöntemin adıdır. Seçkinlik GA'nın başarımını hızlı bir şekilde arttırabilir, çünkü bulunan en iyi çözümün kaybolmasını önler.

(ii) Çaprazlama: Çaprazlama operatörü GA'lardaki en önemli operatördür. Çözümün yapılar kullanılarak yeni bir çözüm oluşturulması esasına dayanır. Çaprazlama işlemi genel olarak ikili dizilerin parçalarının değiş tokuşu şeklinde gerçekleştirilir. Farklı uygulamalarda farklı kodlama yöntemleri kullanıldığı için farklı çaprazlama yöntemleri kullanılır. Bunlardan en çok kullanılanları aşağıda verilmiştir:

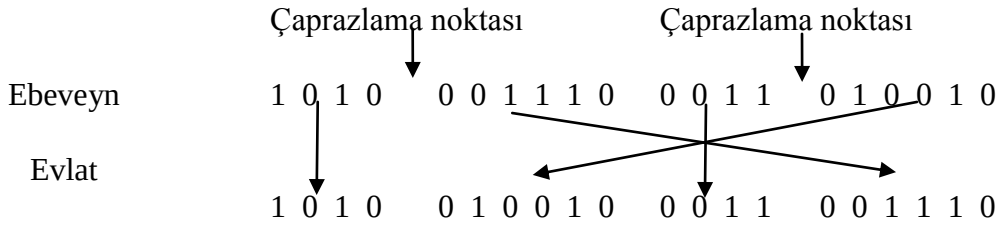
- Tek noktalı çaprazlama,
- Çok noktalı çaprazlama,
- Üniform çaprazlama,

Amaç, ebeveyn kromozom genlerinin yerini değiştirerek çocuk kromozomları üretmek ve böylece var olan uygunluk değeri daha yüksek olan kromozomlar elde etmektedir.Aşağıdaki şekil 3.2. ‘de çaprazlamanın nasıl yapıldığı gösterilmiştir.

(iii) Mutasyon: Mutasyon GA’lardaki operasyonda karar verici olarak ikinci derecede rol oynar. Amaç, var olan bir kromozomun genlerinin bir ya da bir kaçının yerlerini değiştirerek yeni kromozom oluşturmaktır. Yeniden ve sürekli yeni nesil üretimi sonucunda belirli bir süre sonra nesildeki kromozomlar birbirini tekrarlama konumuna gelebilir ve bunun sonucunda farklı kromozom üretimi durabilir veya çok azalabilir. İşte bu sebeple nesildeki kromozomların çeşitliliğini arttırmak için kromozomlardan bazıları mutasyona uğratılır. Aşağıdaki şekil 3.3. ‘te mutasyonun nasıl uygulanıldığı gösterilmiştir.



Şekil 3.2. Mutasyon



Şekil 3.3. Çaprazlama

(iii) Çaprazlama ve Mutasyon Olasılığı: Çaprazlama olasılığı ile bireyler yeni bireyleri oluşturmak için birbirleriyle eşleştirilir. Eğer çaprazlama yapılmazsa, evlat bireyler ebeveyn bireylerin tıpatıp aynısı olacaktır. Mutasyon olasılığı ile yeni evlatlar üzerinde mutasyon işlemi yapılacaktır.

3.2. Uygunluk Değerlendirmesi

Genetik algoritma süreci, başlangıç popülasyonunun oluşturulmasıyla başlar ve bundan sonra evrim geçirir. Bu evrim süresince iyi bireyler uygunluk değerlerine göre ayırt edilirler. Uygunluk, potansiyel çözümün iyilik derecesini belirleyen bir kıstastır. Bir bireyin (genotip veya kromozom) uygunluk değeri, onun iyilik derecesini belirleyen bir pozitif sayıdır. Yani bütün uygunluk değerleri pozitif olmak zorundadır, negatif olmasına izin verilmez. Kromozom, kombinasyonel optimizasyon problemleri için bir çözüm oluşturduğunda, uygunluk değeri bu çözümün maliyetini gösterir. Minimizasyon problemlerinde, düşük maliyetli çözümlerle daha uygun olan bireylerden seçilir. Uygunluğu iyi olan bireylerin seçilme olasılıkları yüksek olacaktır ve bu sayede bu bireylere genetik operatörlerin uygulanması ihtimali de yüksek olacaktır.

Genetik algoritma doğal olarak uygunluğu maksimum yaparak çalışır. Birçok optimizasyon probleminde amaç maliyeti minimum yapmaktır. Bu nedenle maliyet fonksiyonu bir uygunluk fonksiyonu ile tasarlanır.

Uygunluk fonksiyonu genelde probleme özgüdür. Bir problem için oluşturulan uygunluk fonksiyonu başka bir problemde kullanılamaz. Problem incelenerek uygunluk fonksiyonunun ne olacağına karar verilir.

3.3. Çok-Amaçlı Genetik Optimizasyon

Çok amaçlı optimizasyon tek amaçlı optimizasyondan farklı olarak, aynı anda birbirleriyle çelişen amaçların (performans ve maliyet gibi) olduğu problemlerle uğraşır. Örneğin, karmaşık bir donanım tasarımı düşünüldüğünde en yüksek performans hedeflenirken maliyetlerin de minimize edilmesi gerekmektedir. Yukarıda bahsedildiği gibi birden fazla amaç olduğunda bunlardan bazıları kısıt olarak tanımlanabilir. Örneğin, bir sistem yüksek performans ve düşük maliyet ile optimize edilirken, sistemin belirli boyutları geçmemesi ayrı bir optimizasyon kriteri olarak tanımlanabilir. Böylece, bir çok amaçlı optimizasyon problemi aşağıdaki gibi formüle edilebilir [26] .Çok amaçlı bir optimizasyon problemi genel olarak şunları içerir:

- bir parametre kümesi(karar değişkenleri),***a***
- amaç fonksiyonları kümesi, ***b***

- kısıtlar kümesi, ***c***
- ve amaç fonksiyonları ve kısıtlar karar değişkenlerinin fonksiyonlarıdır.

Matematiksel olarak bir optimizasyon problemi aşağıdaki şekilde gösterilebilir:

Min/maks y:f(x)= (f1(x), f2(x), ..., fb(x)) **(3.2)**

Kısıtlar $e(x) = (e1(x), e2(x), ..., ec(x)) \leq 0$

$x = (x1, x2, ..., xa) \in X$

$y = (y1, y2, ..., yb) \in Y,$

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

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

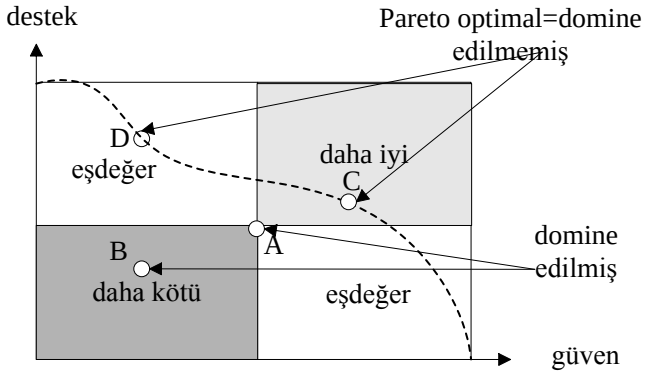
O zaman, optimum dizayn, maksimum performansı, minimum maliyet ile sağlar ve kısıtları aşmaz. Eğer böyle bir çözüm olsaydı, yani her bir amaç için optimum olan çözüm diğeri için de optimum olsaydı aslında sadece tek amaçlı optimizasyon problemi çözmemiz gerekirdi. Fakat çok-amaçlı optimizasyonu zorlaştıran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

Şekil 3.4'te amaç fonksiyonları olarak birliktelik kurallarının destek ve güven değerleri gösterilmektedir. Şekil'de gösterildiği gibi bir çözümde yer alan karar vektörü amaç değerleri açısından bir diğerinden daha iyi, daha kötü, eşit veya farksız olabilir. Daha iyi olması demek çözümün hiçbir amaç değerinin diğer çözümlerin amaç değerlerlerinden düşük olmaması ve en az bir amaç değerinde daha yüksek olması demektir. Örneğin şekilde B ile temsil edilen çözüm A çözümünden daha kötü ve C çözümü ise A çözümünden daha iyidir. Fakat C ile D çözümleri biri diğerinden daha iyi diyemeyiz. Bunun sebebi amaç değerlerinden birinin bir çözümde, diğerinin ise diğer çözümde yüksek olmasıdır.

Bu kavramı kullanarak optimum çözümü arama uzayında hiçbir diğer çözüm tarafından domine edilemeyen çözüm olarak tanımlayabiliriz. Böyle bir çözümü Pareto'nun optimum

çözümü olarak tanımlayabiliriz. Şekilde noktalı çizgiyle gösterilen çözümler ise Pareto'nun optimum çözüm kümesidir.

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



Şekil 3.4. Pareto'nun optimalite kavramı

3.4. Çok Amaçlı Genetik Algoritmalar

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

Pareto en uygun çözümü, çok amaçlı GA'nın doğasına çok uygundur [27]. Goldberg'in [28] seçme algoritması ise [29, 30]'da önerilen çok amaçlı evrimsel algoritmada daha uygundur.

Çok amaçlı genetik algoritmalarda birden fazla amaca aynı anda bakılır. Yani tek bir en uygun çözüm yoktur. Amaçlar arasında bir tercih yapılarak bir çözüm kümesi seçilebilir [30]. Bu yöntemle kullanıcı spesifik problem için olası çözümlerden birini seçebilir. Böylece kullanıcı yüksek kaliteli çözümleri inceleyerek amaçlar arasında bir tercih yaparak

bir çözüm kümesi seçme fırsatı bulacaktır. Bu yöntem kullanıcıyı tek bir çözüm kümesine zorlamaktan daha iyi bir yöntemdir.

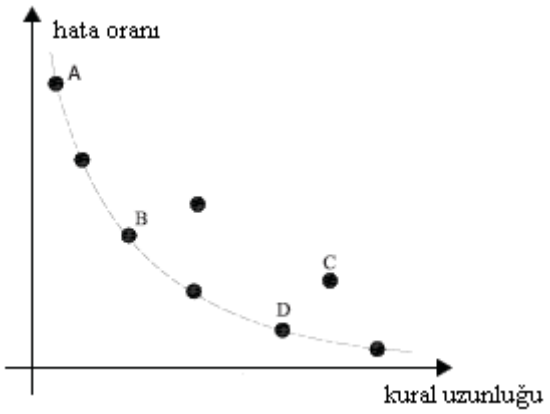
Çok amaçlı GA'ların tercih edilmesinin nedenleri şunlardır:

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

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

(iii) Genetik algoritmalar, çok amaçlı problemleri çözmek için bir aday çözüm kümesiyle çalışır [30].

Şekil 3.5'te A çözümü düşük kural uzunluğuna sahip ama hata oranı yüksek, D çözümü ise düşük hata oranına ancak yüksek kural uzunluğuna sahiptir. Her iki amaç önemli olduğundan; A çözümü D'den daha iyidir ya da tam tersi söylenemez. Diğer taraftan C çözümü D'den daha kötüdür denebilir.



Şekil 3.5. İki amaçlı problem örneği

4. ÇOK AMAÇLI GENETİK ALGORİTMA İLE VERİ KÜMELERİ İLİŞKİLERİNİN BULUNMASI

4.1. Sosyodemografik Veri Kümesinde Sınıflandırma Kurallarının Çıkarılması

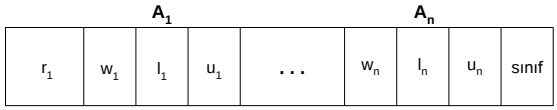
4.1.1. Şizofreni Hastalığı

Şizofreni, uzun yıllardır bilinen, toplum içinde ve akıl hastanelerinde çok sık karşılaşılan süreğen bir hastalıktır [31,33]. Kişinin alışılagelmiş algılama ve yorumlama biçimlerine yabancılaşarak, kendine özgü bir içe-kapanım dünyasına çekildiği ruhsal bir bozukluktur.

Şizofreni, birçok davranış ve düşünce bozukluğuna neden olan; beynin yapısında, fizyoloji ve kimyasında önemli değişikliklerin olduğu çok sistemli psikiyatrik bozukluklardan biridir. Erken yaşlarda başlayarak hayat boyu sürebilen bir hastalıktır. Şizofreninin tanı koydurucu bir belirtisi bulunmamaktadır. Eşlik eden özgül biyokimyasal, nöro-radyolojik, fizyolojik ve psikolojik test olmadığından bu tanı hala bir dışlama tanısı olarak kalmaktadır. Bu çalışmanın dayanak noktası, şizofreni hastaların sosyodemografik parametreleri şizofren olmayan hastalara göre farklılık gösterebileceği tezidir. Bu amaçla şizofreni hastalarının sosyodemografik verileri analiz edilerek suç işleme durumlarının bu verilere göre nasıl değişebileceği incelenecektir. Çıkarılacak kurallar sınıflandırma kuralları olup suç işleme durumunda daha öne çıkan parametreler belirlenecektir.

4.1.2. Geliştirilen Yöntemin Kromozom Yapısı

Önerilen yöntemde birey bir sınıflandırma kuralını temsil eder. Şekil 4.1'de gösterildiği gibi sınıf türünü ve kuralın mevcut olup olmadığını gösteren parçası hariç bir birey n gen ile temsil edilir. Gen sıraları niteliklerin sırasıyla uyumludur. Yani, birinci gen birinci nitelik ile ikinci gen ise ikinci nitelik ile alakalıdır.



Şekil 4.1. Kromozom yapısı

w_i ağırlık alanı [0, 1] aralığında değerler alan bir değişkendir ve muhtemel niteliğin ilgili kuralda mevcut olup olmadığını gösterir. Daha kesin bir ifadeyle w_i kullanıcının belirlediği bir eşik değerinden daha küçük ise, ilgili niteliğin o kuralda olmayacağı anlamına gelir. Bundan dolayı, bu eşik değeri ne kadar büyük olursa, ilgili niteliğin kuralda olma olasılığı o derece azalır. l_i ve u_i değişkenleri ilgili niteliğin kuraldaki alt ve üst limitini temsil eder. Sınıf geni o kuralın hangi sınıfa ait olduğunu, r_1 ise o kuralın sınıflandırıcıda olup olmayacağını belirtmek için kullanılmıştır.

4.1.3. Geliştirilen Yöntem için Kullanılan Çok Amaçlar

Bu çalışmada, çok amaçlı GA kullanılarak Şizofren Hastalarının suç işleme durumu ile Sosyodemografik parametreleri arasındaki sınıflandırma kurallarının çıkarılması hedeflenmektedir. Bir bireyin uygunluğu birbiriyle çelişen 3 farklı amaç ile bulunur.

Doğruluk Oranı: Suç işleme durumu için bulunacak kurallarda yüksek sınıflandırma doğruluk oranı istenir. Bu oran eğitim verilerinin doğruluk değeridir.

Kural Sayısı: Bir sınıflandırıcıda kural sayısı ne kadar az olursa, ilgili test verisini sınıflandırmak o kadar hızlı olur. Bu nedenle etkin bir sınıflandırıcıda kural sayısının olabildiğince az olması istenir. Fakat kural sayısının alt sınırının da veritabanının sınıf sayısı kadar olduğu unutulmamalıdır.

Benzerlik: Bir sınıflandırıcının benzerliği kurallardaki parametrelerin diğer bir deyişle nesnelerin birbirine benzeme durumlarına bağlıdır. İyi bir sınıflandırıcıdan beklenen kuralları oluşturan toplam nesne sayısının az olması ve nesnelerin olabildiğince birbirinden farklı olmasıdır. Bu şekilde parametrelerinin çoğu birbirine benzeyen kurallar sınıflandırıcıdan atılmış olur. Diğer bir yandan kural setinin az sayıda nesne içermesi, o kural setinin daha az karmaşık olduğunu gösterir. Tablo 4.1'de, sınıflandırma nesnesi hariç 5 nesneli bir sınıflandırıcıda bulunan 4 kuralın benzerliğini göstermektedir.

Tablo 4.1. Dört kurallı bir sınıflandırıcı

	A	B	C	D	E
1	1	0	0	0	1
2	1	1	0	0	1
3	0	1	0	1	0
4	1	0	1	1	1

Bir sınıfın anlaşılırlığı aşağıdaki formül ile hesaplanır.

$$Benzerlik = \sum_{i=1}^{n=5} p(i) \quad \textbf{(4.1)}$$

Burada $p(i)$, i'ninci parametrenin sınıflandırıcıda bulunma oranıdır. Örnek olarak tablo 4.1.'de, *A* parametresi dört kuralın üçünde bulunduğundan, $p(A) = 3/4 = 0{,}75$'tir.
Buna göre bir sınıflandırıcıda benzerlik değeri ne kadar düşükse o sınıflandırıcıdaki kuralların parametreleri o kadar birbirinden farklıdır. Bu da istenen bir durumdur.

4.1.4. Genetik Operatörler

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

Başlangıç pozisyonunu sağa doğru kaydır: Rastgele seçilmiş bir genin başlangıç pozisyon değerini bir artır.
Başlangıç pozisyonunu sola doğru kaydır: Rastgele seçilmiş bir genin başlangıç pozisyon değerini bir azalt.
Rastgele değiştirme: Mutasyonun bu türünde öncelikle küçük bir tamsayı üretilir. Daha sonra bu sayı bireyin her bir farklı alanının şu anki değerine eklenir ya da çıkarılır. Bu

şekilde yeni alan değerleri elde edilir. Yalnız bu yeni değerlerin, alanın alt ve üst limit değerlerini geçmemesine dikkat edilir.

4.1.5. Algoritma

Önerilen algoritma aşağıda ki şekil 4.2. 'de gösterilmiştir.

Başlangıç değerleri:

Popülâsyon boyutu: 100
Çaprazlama olasılığı: 0,8
Mutasyon olasılığı: 0,3
Durdurma kıstası: 2000 iterasyon
Minimum doğruluk: %70
Maksimum kural sayısı:7

1.....for 1 to 1000 do
2.....genetik_algoritma ile başlangıç popülasyonu üretilir
3.....üretilen başlangıç popülasyonuna göre kural setleri üretilir
4.....kural setlerinin doğruluk oranları hesaplanır
5.....kural setlerinde toplam kural sayısı ve benzlerlik hesaplanır
6.....seçme kriterlerine uygun olan kromozomlar ve kurallar seçilir
7.....iyi olan genler yoksa 2. adıma dönülür
8.....iyi genler çaprazlanarak kötü olan genlerin yerine yerleştirilir,
9.....mutasyon yapılır
10.....istenen doğruluk, kural sayısı ve benzerliğe uygun kromozomlar kural setine alınır
11.....istenen değerler bulununca veya döngü bitince işlem bitirilir.

Şekil 4.2. Önerilen algoritma

4.1.6. Şizofreni Hastalarının Sosyodemografik Verileri

Bu tez çalışmasında kullanılan şizofreni hastalarının sosyodemografik verileri Fırat Üniversitesi Tıp Fakültesi Adli Tıp Anabilim Dalından elde edilmiştir [31]. Bu

veritabanının parametreleri ve değer aralıkları aşağıdaki Tablo 4.2'de verilmiştir. Tüm verilerin 2/3'ü eğitim, 1/3'ü test için kullanılmıştır

Tablo 4.2. Şizofreni hastalarının sosyodemografik verileri

Sosyodemografik Bilgiler	**Değer Aralığı**	**Açıklama**
Yaş	[19, 59]	Yaş
Cinsiyet	{K, E}	K: Kadın, E: Erkek
Medeni durum ***(meddur)***	{1, 2, 3}	1:evli 2:bekar 3:dul
Öğrenim durumu ***(öğdur)***	{1, 2, 3}	1:okuryazar değil, 2:ilköğretim, 3:lise,yüksekokul
Aylık düzenli gelir durumu ***(aygel)***	{1, 2, 3}	1:yok 2:500 TL'nin altı, 3:1000 TL
Hastaneye yatış sayısı ***(hasyatsay)***	[1, 24]	Hastaneye yatış sayısı
Şizofreni alt tipi ***(şizaltip)***	{1, 2, 3, 4, 5}	1:paranoid tip 2:kotatonik tip 3:rezidüel tip 4:dezorgonize tip
Hastalık süresi ***(hastsür)***	[1, 4]	1:0-5 yıl 2:5-10 yıl 3:10-20 yıl 4:20 yıl ve üzeri
Hastalık başlangıç yaşı ***(hasbaşyaş)***	[13, 54]	Hastalık başlangıç yaşı
Teşhisten sonra tedaviye devam durumu ***(teddev)***	{1, 2}	1:düzenli 2.düzensiz
Suç işlediği esnada tedavi görme durumu ***(sucted)***	{1, 2, 3}	1:hayır 2:evet 3:suç işlememiş
Aile öyküsü ***(aileöyk)***	{1, 2}	1:var 2:yok
İş durumu ***(işdur)***	{1, 2, 3}	1:çalşmıyor 2:düzensiz çalışıyor 3:düzenli çalışıyor
Meslek ***(meslek)***	{1, 2, …,12}	1:yok 2:memur 3:çiftçi 4:esnaf 5:hayvancılık 6: öğrenci 7:temizlikçi 8:elektrikçi, 9:inşaatçı 10:şoför 11:boyacı 12:fırıncı
Boy ***(boy)***	[151, 182]	boy
Kilo ***(kilo)***	[52, 97]	kilo
Sigara ***(sigara)***	{1, 2}	1:sigara kullanmıyor, 2: sigara kullanıyor
Alkol ***(alkol)***	{1, 2}	1:hayır 2.evet
Uyuşturucu madde kullanımı ***(uymad)***	{1, 2}	1:hayır 2:evet
İntihar öyküsü ***(intöyk)***	{1, 2}	1:yok 2:var
İşlediği suç ***(isl.suc)***	[1, 9]	1:cinayet 2:hırsızlık 3:yaralama 4:cinsel suç 5:suç işlememiş 6-8-10: kamu malına zarar vermek 7:basit müessir fiil 9:örgüt üyesi
Önceden suç öyküsü durumu ***(öncsucöyk)***	{1, 2}	1:yok 2:var
Şizofren atak sayısı ***(şizatsay)***	[1, 2]	1:0-1 atak 2:1 ve üzeri atak
Hastanede tedavi görme süresi ***(hastedsür)***	[1, 180]	hastanede tedavi görme süresi (ay olarak)
Yaşadığı yer	{1, 2, 3}	1:köy 2:ilçe 3:şehir merkezi
Eşlik eden başka ruhsal rahatsızlık ve kullandığı ilaçlar ***(esruhrah)***	[1, 18]	1: yok 2-18: var
Sınıf ***(Suçlu veya Değil)***	[1, 2]	1: Suç işlememiş, 2: Suç işlemiş

4.1.7. Uygulama Sonuçları

Bu bölümde önerilen yöntemin etkinliğini ve kullanılabilirliğini test etmek için bazı deneyler yapıldı. Deneyler Pentium C2D 1.83GHz ve 1GB hafızaya sahip bir bilgisayar ile Windows Vista işletim sistemi altında gerçekleştirildi. Veri kümesi olarak Fırat Üniversitesi Tıp Fakültesi Hastanesinden alınan Şizofren Sosyodemografik verileri kullanıldı [31].

Deneysel testlerin tamamında çok-amaçlı genetik algoritma süreci 100 bireylik bir populasyon ile başlamıştır. Sonlandırma kriteri olarak da maksimum jenerasyon sayısı 2000 olarak belirlenmiştir. Çaprazlama olasılığı 0,8 olarak seçilirken, mutasyon olasılığı ise her bir mutasyon çeşidi için 0,3 alınmıştır.

İlk deney Tablo 4.3'te gösterildiği gibi eğitim verileri üzerinde her üç amacın değerini göstermektedir. Bu tabloda görüldüğü gibi, benzerlik arttıkça ortalama hata oranı azalmaktadır. Bu aslında beklenen bir durumdur. Çünkü benzerlik dolaylı olarak sınıflandırıcıdaki kural sayısıyla orantılıdır. Daha önce ifade edildiği gibi bir sınıflandırıcıdaki kurallar farklı niteliklerden oluşuyorsa o sınıflandırıcının anlaşılabilirliği o kadar artar. Fakat kural sayısının artması da istenmeyen bir durum olmasına rağmen ilk bakışta benzerliği azaltır. Bunun için bu iki amaç arasında bir ikilem oluşur. Zaten çok-amaçlı optimizasyon probleminin amacı da budur. Bu deneyde maksimum kural sayısı 7 olarak belirlenmiştir. Daha fazla kural sayılı sınıflandırıcılar ele alınmamıştır.

Tablo 4.4'te ise eğitim verileri ile elde edilen sınıflandırıcıların test verisi üzerinde bulunan doğruluk oranı değerleri, Ishibuchi'nin [32] yöntemiyle karşılaştırmalı olarak gösterilir. Ishibuchi'nin sezgisel kural çıkarım yönteminde, her bir sınıf için yüksek destek değerinden başlayarak aday kuralların belirli bir sayısı bulunur. Daha iyi anlaşılabilir sınıflandırma kurallarının elde edilmesi için, sadece az sayıda nesne içeren kurallar aday kural olarak göz önüne alınır. Daha sonra bu aday kurallar, çok-amaçlı bir genetik kural seçme işlemine maruz kalır. Ishibuchi'nin yönteminde amaç: *S* aday kuralların bir alt kümesi olmak üzere; *S* kural kümesi tarafından uygun bir şekilde sınıflandırılan eğitme verilerinin sayısının maksimize etmek, *S*'deki kuralların sayısını minimize etmek ve *S*'deki kuralları *n* toplam uzunluklarını minimize etmektir. Bu tabloda görüldüğü gibi geliştirilen yöntemin sonuçları diğer yöntemin sonuçlarına üstünlük sağlamaktadır. Dört farklı sınıflandırıcıda da önerilen yöntemin doğruluk oranı Ishibuchi'nin yönteminkinden daha yüksektir.

Tablo 4.3. Eğitim verileri için amaç değerler

Doğruluk Oranı (%)	Kural Sayısı	Benzerlik
84,3	4	0,47
86,2	5	0,54
89,6	6	0,58
91,1	7	0,61

Tablo 4.4. Test verileri için amaç değerler

Kural sayısı	Önerilen yöntemin doğruluk oranı (%)	Ishibuchi'nin doğruluk oranı (%)
4	81,7	79,4
5	83,0	80,3
6	87,4	84,5
7	88,3	86,8

Son olarak kural sayısı 3 olan bir sınıflandırıcının, sınıflandırma kuralları aşağıdaki gibi bulunmuştur. Bu kurallardaki her bir parametrenin varlığı ve uygun değer aralığı önerilen çok-amaçlı genetik algoritma sayesinde çıkarılmaktadır:

Kural 1: *öncsucöyk > 1 → Suç işlemiş*
Kural 2: *öncsucöyk = 1 & hastedsür > 2 → Suç işlememiş*
Kural 3: *şizaltip > 4 → Suç işlemiş*

Bu kuralların ilki şizofreni hastalığına yakalanmadan önce işlediği suç sayısı 1'den daha büyük hastaların şizofreni olduktan sonra da suç işlemeye devam ettikleri sonucunu çıkarmaktadır. İkinci kuralda eğer hastalığa yakalanmadan önce işlediği suç sayısı 1'e eşit ve hastanede tedavi görme süresi 2 aydan daha fazla olan hastalar hastalığa yakalandıktan sonra suç işlememişlerdir. Son kuralda ise şizofreni alt tipi 5 olan hastalar suç işlememişlerdir.

4.2. Biyokimyasal Veri Kümesinde Sınıflandırma Kurallarının Çıkarılması

4.2.1. Şizofreni Hastalarının Biyokimya Verileri

Bu tez çalışmasında kullanılan şizofreni hastalarının biyokimya verileri Fırat Üniversitesi Tıp Fakültesi Adli Tıp Anabilim Dalından elde edilmiştir [31]. Tablo 4.5'te biyokimya verileri verilmiştir. Bu veritabanının parametreleri ve değer aralıkları aşağıda ki tabloda verilmiştir.102 verinin 2/3'ü eğitim, 1/3'ü test için kullanılmıştır. Önerilen yöntemin bu veritabanına uygulanma nedeni şizofreni hastalarının suç işleme durumunda sosyodemografik verilerin mi yoksa biyokimya verilerinin mi daha etkili olduğunu görmektir.

4.2.2. Uygulama Sonuçları

Tezin bu bölümünde önerilen çok-amaçlı genetik algoritma yöntemi farklı bir veritabanına uygulanmıştır. Bu veritabanının özelliği tüm parametrelerinin nicel değer almasıdır. Deneysel ortam bir önceki bölümde verilenlerle aynıdır. Yine önceki deneylerde olduğu gibi bu yeni testlerin tamamında çok-amaçlı genetik algoritma süreci 100 bireylik bir populasyon ile başlamıştır. Sonlandırma kriteri olarak da maksimum jenerasyon sayısı 2000 olarak belirlenmiştir. Çaprazlama olasılığı 0,8 olarak seçilirken, mutasyon olasılığı ise her bir mutasyon çeşidi için 0,3 alınmıştır.

İlk deney Tablo 4.6'da gösterildiği gibi eğitim verileri üzerinde her üç amacın değerini göstermektedir. Bu tabloda görüldüğü gibi, biyokimya verilerinden elde edilen sınıflandırıcıların doğruluk oranları sosyodemografik verilerinkine göre düşmüştür. Bu da biyokimya parametre değerlerinin şizofreni hastanın suç işleme durumu ile daha az düzenlilik gösterdiği sonucunu çıkarır. Buna karşılık sınıflandırıcıdaki kuralların benzerlik oranı ise düşmüştür. Bu durum ise kurallardaki parametrelerin daha az olmasına ve birbirinden olabildiğince farklı olması sonucunu çıkarmaktadır. Daha önce ifade edildiği gibi bir sınıflandırıcıdaki kurallar farklı niteliklerden oluşuyorsa sınıflandırıcının benzerliği azalır. Ancak kurallardaki parametre sayısı arttıkça benzerlik artmaya başlar. Parametreleri farklı kural sayılarının artması benzerliği azaltır. Fakat kural sayısının artması da istenmeyen bir durum olduğu için bu iki amaç arasında bir ikilem oluşur. Zaten çok amaçlı optimizasyon probleminin amacı da budur.

Tablo 4.7'de ise test verisi üzerinde bulunan doğruluk oranı değerleri Ishibuchi'nin [32] yöntemiyle karşılaştırılır. Bu tabloda görüldüğü gibi geliştirilen yöntemin sonuçları bir öncekinde olduğu gibi diğer yöntemin sonuçlarına üstünlük sağlamaktadır. Kural sayısının 3 olduğu bir sınıflandırıcının sınıflandırma kuralları aşağıdaki gibi bulunmuştur:

Kural 1: *Ghrelin > 11,45 → Suç işlemiş*
Kural 2: *Stestos ≤ 109 & Ghrelin ≤ 4,28 → Suç işlememiş*
Kural 3: *Ghrelin ≤ 11,45 & Stestos > 109 → Suç işlememiş*

Bu kuralların ilki suç işleme durumunun doğrudan Ghrelin değeriyle ilgili olduğu sonucunu çıkarır. Buna Ghrelin değeri 11,45'den daha büyük hastalar suç işlemişlerdir. İkinci kuralda ise eğer bir şizofreni hastasının Stesyos değeri 109'a eşit veya küçük ise ve Ghrelin değeri 4,28'e eşit veya küçük ise bu durumda hasta suç işlemiştir. Son kural ise Stetos değerinin 109'dan daha büyük olması durumunda bu defa Ghrelin değerinin 11,45'e eşit veya küçük olması suç işleme eylemini meydana getirmektedir.

Tablo 4.5. Şizofreni hastalarının biyokimya verileri

Biyokimya Parametreleri	**Değer Aralığı**
tkol	[106, 252]
trig	[67, 203]
hdl	[21, 53]
vldl	[13, 41]
ldl	[65, 183]
shbg	[11,2; 81,8]
östriol	[20,6; 49,7]
ttestos	[196; 874,5]
stestos	[30,4; 178,11]
çinko	[81, 170]
bakır	[70, 164]
ghrelin	[1,87; 21,15]
Sınıf (Suçlu veya Değil)	1: Suç işlememiş 2: Suç işlemiş

Tablo 4.6. Eğitim verileri için amaç değerler

Doğruluk Oranı (%)	Kural Sayısı	Benzerlik
76,7	3	0,37
79,2	4	0,41
82,1	5	0,47
83,3	6	0,54

Tablo 4.7. Test verileri için amaç değerler

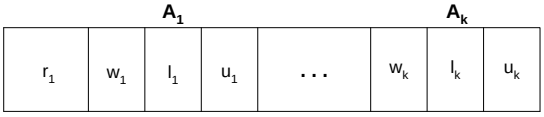
Kural sayısı	Önerilen yöntemin doğruluk oranı (%)	Ishibuchi'nin doğruluk oranı (%)
3	73,2	69,7
4	75,4	71,2
5	77,9	74,6
6	79,0	76,1

4.3. Sosyodemografik ve Biyokimyasal Veri Kümelerinde Birliktelik Kurallarının Çıkarılması

Bu bölümde Şizofreni hastalarının sosyodemografik ve biyokimyasal verileri suç işleme durumlarına göre birleştirilerek bu iki veri kümesinin parametreleri arasında birliktelik kuralları çıkarılacaktır. Böylece hem sosyodemografik ve hem de biyokimya parametrelerini ihtiva eden birliktelik kuralları görülecektir. Bu sayede her iki veri kümesinin birbiriyle etkileşimi ve analizi daha kolay yapılabilecektir.

4.3.1. Geliştirilen Yöntemin Kromozom Yapısı

Önerilen yöntemde her bir birey bir birliktelik kuralını temsil eder. Şekil 4.3'de gösterildiği gibi kuralın mevcut olup olmadığını gösteren parçası hariç bir birey *k* gen ile temsil edilir. Gen sıraları niteliklerin sırasıyla uyumludur. Yani, birinci gen birinci nitelik ile ikinci gen ise ikinci nitelik ile alakalıdır. Buradaki *k* sayısı sosyodemografik ve biyokimya verilerinin parametre sayılarının toplamıdır.



Şekil 4.3. Kromozom yapısı

w_i ağırlık alanı [0, 1] aralığında değerler alan bir değişkendir ve muhtemel niteliğin ilgili kuralda mevcut olup olmadığını gösterir. Daha kesin bir ifadeyle w_i kullanıcının

belirlediği bir eşik değerinden daha küçük ise, ilgili niteliğin o kuralda olmayacağı anlamına gelir. Bundan dolayı, bu eşik değeri ne kadar büyük olursa, ilgili niteliğin kuralda olma olasılığı o derece azalır. l_i ve u_i değişkenleri ilgili niteliğin kuraldaki alt ve üst limitini temsil eder. Sınıf geni o kuralın hangi sınıfa ait olduğunu, r_1 ise o kuralın sınıflandırıcıda olup olmayacağını belirtmek için kullanılmıştır.

4.3.2. Geliştirilen Yöntem için Kullanılan Çok Amaçlar

Bu çalışmada, çok amaçlı GA kullanılarak Şizofren Hastalarının suç işleme durumu ile Sosyodemografik ve biyokimyasal parametreleri arasındaki etkin birliktelik kurallarının çıkarılması hedeflenmektedir. Bir veritabanında bazen birbirine benzeyen yüzlerce kural çıkabilmekte ve bunların hepsi kullanıcıya aynı anda sunulmaktadır. Kullanıcı ise bunlardan sadece bir kaçıyla ilgilenmekte birbirine benzeyen kurallara önem vermemektedir. Tezin bu bölümünde de kullanıcının en güçlü bir şekilde ilgisini çekebilecek ve birbirinden farklı kuralların bulunması için çok amaçlar önerilmiştir. Bunlar;

Ortalama Destek Değeri: Çıkarılacak birliktelik kural kümesinin olabildiğince bütün veritabanını kapsaması istenir. Bir kural kümesinde ortalama destek değeri ne kadar büyükse o kurallar o kadar geniş bir aralıkta veritabanına hitap eder.

Ortalama Güven Değeri: Birliktelik kurallarının gücü doğrudan güven değeri ile ilgilidir. Bu değerin yüksek olması kuralı oluşturan parametreler arasındaki ilişkinin bir anlamda sağlamlığını belirler. Buna karşılık bazen çok büyük güven değerli kural bulunamamasına rağmen yine de o kural kullanıcının ilgisini çekebilmektedir. Ortalama güven değeri de sadece bir kurala bakmayarak kural kümesi içerisindeki bütün kuralların ortalamasını alır. Bu sayede bazen düşük güvenli kurallar bile istenen kural kümesi içine girebilmektedir.

Benzerlik: Sınıflandırıcı bölümünde anlatılana benzer şekilde bir kuralın benzerliği kuraldaki parametrelerin diğer bir deyişle nesnelerin birbirine benzeme durumlarına bağlıdır. Kullanıcıya kural kümesi sunulurken kuralları oluşturan toplam nesne sayısının az olması ve nesnelerin olabildiğince birbirinden farklı olması istenir. Bu şekilde

parametrelerinin çoğu birbirine benzeyen kurallar kümeden atılmış olur. Diğer bir yandan kural setinin az sayıda nesne içermesi, o kural setinin daha az karmaşık olduğunu gösterir.

4.3.3. Uygulama Sonuçları

Deneysel testlerin tamamında çok-amaçlı genetik algoritma süreci daha öncekilerde olduğu gibi 100 bireylik bir populasyon ile başlamıştır. Sonlandırma kriteri olarak da maksimum jenerasyon sayısı 2000 olarak belirlenmiştir. Çaprazlama olasılığı 0,8 olarak seçilirken, mutasyon olasılığı ise her bir mutasyon çeşidi için 0,3 alınmıştır.

Tablo 4.8'de görüldüğü gibi benzerlik yüksek iken birbirine benzeyen kurallar çıkmakta bu da o kural kümesinin ortalama destek değerini artırmaktadır. Buna karşılık benzerlik azaldıkça ortalama destek değeri de azalmaktadır. Ortama güven değeri ise ortalama destek değerine bağımlı değildir.

Tablo 4.8. Pareto-optimal çözümdeki amaçların değerleri

Benzerlik	Ortalama Destek Değeri (%)	Ortalama Güven Değeri (%)
0,78	21,7	82,4
0,62	19,2	81,6
0,53	16,4	84,8
0,49	12,1	80,7

Aşağıda her üç amaç değeri diğerlerine göre domime edilmiş çözümlerden biri verilmiştir. Bu çözümde bulunan kural sayısı 3'dür ve parametre değerleri birbirinden farklı olarak bulunmuştur.

Kural 1: *öncsuçöyk = 1 & Stestos [39,3, 81,34] →Uymad =1*
Kural 2: *Östriol [30,1, 46,2] → Şizaltip[3, 5] & Hastedsür [36, 180]*
Kural 3: *Ghrelin[13,95, 21,15] → Alkol=2*

Bulunan ilk birliktelik kuralında, önceki suç öyküsü ve uyuşturucu madde kullanımı parametreleri sosyodemografik verilere ait iken Stestos parametresi biyokimya verisine aittir. Böylece iki farklı veritabanından çıkarılan parametrelerle birliktelik kuralı elde

edilmiştir. Önceden işlemiş olduğu suç sayısı 1 iken Stestos sayısının 39,3 ve 81,34 arasında olması o hastanın uyuşturucu madde kullanmadığı anlamını taşımaktadır.

5. SONUÇ

Bu tez çalışmasında, şizofreni hastalarının sosyodemografik ve biyokimyasal verileri incelenerek suç işleme durumu ile ilgili öncelikle sınıflandırma kuralları elde edilmiştir. Bu kurallar elde edilirken bir sınıflandırıcı sistem tasarlanmıştır. Bu sistem çok amaçlı genetik algoritma yöntemini kullanır. Yöntemin en önemli adımlarından biri uygun çok amaçların belirlenmesidir. Bu tezde, iyi bir sınıflandırıcı sistem için öncelikle en etkili olabilecek çok amaçlar önerilmiştir. Bu amaçlar, sınıflandırıcının doğruluk oranı yüksek, sınıflandırıcıdaki kural sayısı az ve sınıflandırıcıda bulunan kuralların olabildiğince birbirine benzememesidir.

Sınıflandırıcının doğruluk oranı doğrudan sınıflandırıcının uygun sınıflandırma kabiliyetiyle ilgilidir. Bütün sınıflandırıcılarda bu değerin yüksek olması istenir. Sınıflandırıcıdaki kural sayısı sınıflandırıcının hızını etkiler. Kural sayısı ne kadar az ise yeni bir test örneği gelmesi durumunda sınıflandırıcı o kadar hızlı o örneğin sınıfını belirler. Önerdiğimiz diğer bir amaç ise kuralların birbirine benzememesidir. Benzerlikte amaç kuralda mevcut parametrelerin birbirinden farklı olma isteğidir. Bu şekilde sınıflandırıcının anlaşılabilirliği artmıştır. Bu amaçlara göre geliştirilen çok amaçlı genetik algoritma yöntemine göre bulunan sınıflandırıcılardan şizofreni hastalarının suç işleme durumları sosyodemografik verilere göre daha bağımlı olduğu gözlenmiştir. Sosyodemografik verilerdeki sınıflandırıcının doğruluk oranının yüksek olması da bunu gösterir. Önerilen uygun çok amaçlara göre geliştirilen sınıflandırıcının diğer bir avantajı ise doğruluk oranı yönünden türdeşlerine göre geri kalmaması hatta ondan daha iyi sonuçlar üretebilmesidir. Yapılan uygulamalarda, önerilen yöntemin bulduğu sınıflandırma kuralları, daha önceden önerilen [32] çok amaçlı bir genetik algoritma yöntemi olan Ishibuchi'nin yaklaşımına üstünlük sağladığı gözlemlenmiştir. Örneğin, sosyodemografik verilerde 4 kurallı bir sınıflandırıcıda önerilen yöntem test aşamasında %81,7'lik bir başarı yakalarken, Ishibuchi'nin başarı oranı %79,4'de kalmıştır. Benzer şekilde biyokimya verilerinde ise test aşamasında elde edilen doğruluk oranı %75,4 iken Ishibuchi ancak %71,2'yi yakalayabilmiştir.

Tezde ulaşılan bir diğer sonuç ise, sosyodemografik ve biyokimyasal veriler birleştirilerek her iki veri kümesi arasında birliktelik kurallarının bulunmasıdır. Bu maksatla yeni 3 tane farklı çok amaçlar önerilmiştir. Bunlar; kullanıcıya sunulacak birliktelik kurallarının ortalama destek değerleri, ortalama güven değerleri ve

parametrelerin birbirine benzeme durumlarıdır. Çoğu zaman kullanıcıya sunulan kuralların büyük bir miktarı birbirine benzemekte ya da güven değeri sadece çok yüksek olan kurallar kullanıcıya gösterilmektedir. Bu durum ise, değerli olmasına rağmen bazı kuralların göz ardı edilmesine neden olmaktadır. Bu problemi gidermek için, birbiriyle çelişen amaçlar önerilerek en uygun kuralların kullanıcıya sunulması hedeflenmektedir. Şizofreni hastalarının sosyodemografik ve biyokimyasal veri kümeleri üzerinde yapılan uygulamaların sonuçları, bu amaçlara cevap veren en etkin kuralların ortaya çıkarıldığını göstermiştir. Deneysel tecrübelerde 0,49 benzerlik oranında ve ortalama %80'nin üzerinde güven değerli kuralların çıkarıldığı görülmüştür. Kullanıcıya sunulan kuralların bu denli yüksek güvende ve az benzerlikte olması sunulan kuralların kalitesiyle doğrudan orantılıdır.

Gelecek çalışmalar için, sınıflandırıcı ya da birliktelik kuralları bulunurken keskin aralık değerleri değil de bulanık kümelerle modelleme yapılarak kural çıkarımına gidilebilir. Böylece, sınıflandırıcı veya birliktelik kural kümelerinin anlaşılabilirliği, daha da yüksek olabilecektir.

KAYNAKLAR

[1] **Bayardo, R. and Srikant, R.,** "Mining the most interesting rules", KDD-99, pp. 145–154, San Diego, CA, 1999.

[2] **Omiecinski, E.,** "Alternative interest measures for mining associations in databases", IEEE Transactions on Knowledge and Data Engineering, vol.15, no.1, pp.57-69, 2003.

[3] **Tan, P.N., Kumar, V. and Srivastava, J**., "Selecting the right interestingness measures for association patterns", SIGKDD'02, pp.33-41, Edmonton, Alberta, Canada, 2002.

[4] **Liu, B., Hsu, W. and Ma, Y.M.,** "Analyzing the subjective interestingness of association rules", IEEE Intelligent Systems, vol.15, no.5, pp.47-55, 2000.

[5] **Padmanabban, B. and Tuzhulin, A.,** "Small is Beautiful: Discovery the Minimal Set of Unexpected Patterns", KDD-2000, pp.54-63, Boston, MA, USA, 2000.

[6] **Fidelis, M.V., Lopes, H.S., and Freitas, A.A.,** "Discovering Comprehensible Classification Rules with a Genetic Algorithm", in Proceedings of Congress on Evolutionary Computation, 2000.

[7] **Ghosh, A. and Nath, B.,** "Multi-Objective Rule Mining using Genetic Algorithms", Information Science, Volume 163, Issues 1-3, 14 June 2004, Pages 123-133

[8] **Chen L, Sycara K.,** 1998, "WebMate: a personal agent for browsing and searching". The second International Conference on Autonomous Agents, ACM.

[9] **Chen MS, Park JS, Yu PS,** 1998, "Efficent data mining for path travel patterns". IEEE Trans. Knowledge and Data Eng 10:209-221.

[10] **Cohn E, Krishnamurhy B, Rexford J.,** 1999, "Efficent algoritms for predicting requests to web servers", The Eighteenth IEEE Annual Joint Conference on Computer and Communications Societies.

[11] **Ali K, Manganaris S., Srikant R,** 1997, "Partial Classification using Association Rules, In Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, pp. 115--118.

[12] **Dinçer, E.,** 2006. Veri Madenciliğinde K-Means Algoritması ve Tıp Alanında Uygulanması, *Yüksek Lisans Tezi,* Kocaeli Üniversitesi, Kocaeli.

[13] **Fayyad, U.M., Piatesky-Shapiro, G., Smyth, P., Uthurusamy, R.,** 1994. Advances in data mining and Knowledge Discovery, *AAAI Pres*, USA.

[14] **Dolgun.,M.Ö.,** 2006. Büyük Alışveriş Merkezleri İçin Veri Madenciliği Uygulamaları, *Yüksek Lisans Tezi*, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

[15] **Han J., Kamber M.,** 2000, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers,Canada,312 p

[16] **Altıntaş, T.,** 2006. Veri Madenciliği Metotlarından Olan Kümeleme Algoritmalarının Uygulamalı Etkinlik Analizi,*Yüksek Lisans Tezi,* Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Sakarya.

[17] **Karabatak, M.,** 2008. Özellik Seçimi, Sınıflama Ve Öngörü Uygulamalarına Yönelik Birliktelik Kuralı Çıkarımı Ve Yazılım Geliştirilmesi, *Doktora Tezi,* Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

[18] **Agrawal, R., Imielinski, T., Swami, A.,** 1993, Mining association rules between sets of items in large databases, *In ACM SIGMOD Conf. Management of Data*.

[19] **Çunkaş M.,** 2006. Genetik Algoritmalar ve Uygulamaları Ders Notları, Selçuk Üniversitesi, Teknik Eğitim Fakültesi, Elektronik-Bilgisayar Eğitimi, Konya.

[20] **Karabatak, M., İnce, M.C.,** 2004. Apriori Algoritması ile Öğrenci Başarısı Analizi, *Elektrik Elektronik Bilgisayar Mühendisliği Sempozyumu, ELECO 2004*, Elektronik-Bilgisayar sayfa 348-352,Bursa

[21] **J. H. Holland,** "Adaptation in Natural and Artificial Systems" Ann Arbor, MI: Univ. Mich. Press, 1975.

[22] **D. E. Goldberg,** "Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Weslay, 1989.

[23] **Karr, L. C., Freeman L. M.** (1999), Industrial Applications of Genetic Algorithms, CRC Press.350 p.

[24] **Mitchell, M. ve Forest S.** 1993, "Genetic Algorithms and Artificial Life", Artificial Life Vol. 1, pp.267-289.

[25] **Portmann M.C.,** 1996, "Genetic Algorithms and Scheduling, A State of Art and Some Propositions", Proceedings of the Workshop on Production Planning and Control, Mons, Bölüm 4, Belçika

[26] **Kaya M.,** 2006, "Multi-objective genetic algorithm based approaches for mining optimized fuzzy association rules", Soft Computing, 10(7): pp.578-586.

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

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

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

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

[31] **Türkoğlu, A.,** "Suç işlemiş ve işlememiş Şizofrenik hastalarda bazı biyokimyasal parametrelerin karşılaştırılması", Tıpta Uzmanlık Tezi, Fırat Üniversitesi, 2008.

[32] **Ishibuchi H., Yamamoto T.,** 2003, "Evolutionary Multi-objective Optimization for Generating an Ensemble of Fuzzy Rule-Based Classifiers", GECCO, 1077-1088,Japan

[33] **Türkoğlu A, Tokdemir M, Atmaca M, Namli M, Üstündağ B,** 2009, "Serum Cholesterol, Triglyceride, and Ghrelin Levels in Criminal and Non-criminal Schizophrenia Patients", Klinik Psikofarmakoloji Bulteni-Bulletin of Clinical Psychopharmacology, 19(4), 353-358.

EK

BİLGİSAYAR PROGRAMI

```
#include<stdio.h>
#include<math.h>

struct indiv
{
  unsigned *chrom;        /* Kromozom string       */
  float fitness[MAXOBJ];      /* fitness fonksionları    */
  float x[MAXVECSIZE];            /* değişkenler                */
  float dumfitness;          /* değiştirilmiş amaçlar   */
  int   flag;
  int   front;
  int   parent1;              /* iki ebevyn */
  int   parent2;
};
typedef struct indiv INDIVIDUAL ;
typedef INDIVIDUAL *POPULATION ;     /* bireylerin dizisi
*/
float       randomperc();
float       get_beta();
float get_delta();
float   get_beta_rigid();
double  noise();
float       rndreal();
int    small(float);
int    pop_size,        /* Population boyutu   */
  gen_no,              /* Mevcut jenerasyon sayısı    */
  max_gen,             /* Maximum jenerasyon sayısı    */
  no_xover,            /* Yapılan çaprazlama sayısı   */
  no_mutation,         /* Yapılan mutasyon sayısı */
  num_var,             /* Toplam değişken sayısı */
  num_bin_var,           /* İkili değişken sayısı    */
  num_int_var,           /* İnteger değişken sayısı   */
  num_enum_var,          /* Numaralandırılmış değişken sayısı
*/
  num_cont_var,          /* Sürekli değişken sayısı */
  num_obj,             /* Amaçların sayısı           */
  lchrom[MAXVECSIZE], /* Kromozom uzunluğu   */
  chromsize,
 x_strategy,
 REPORT,
 var_RIGID[MAXVECSIZE],
 BINGA,
 REALGA,
  FITNESS,
  PARAM,
  minmax[MAXOBJ],
  tourneylist[MAXPOPSIZE],
  tourneypos,
  tourneysize,
  var_type[MAXVECSIZE],
  TotalChromLength,       /* Toplam Kromozom Uzunluğu
*/
  num_enum_data[MAXVECSIZE];
float       seed,
  p_xover,             /* Çaprazlama Olasılığı  */
  p_mutation,          /* Mutasyon Olasılığı           */
  closeness,
  minx[MAXVECSIZE],
  maxx[MAXVECSIZE],
  x_lower[MAXVECSIZE],
  x_upper[MAXVECSIZE],
 afmin[MAXOBJ],
 afmax[MAXOBJ],
 dshare,
 max_spread,
 maxf[MAXOBJ],
 minf[MAXOBJ],
 avgfitness[MAXOBJ],     /* Ortalama fitness         */
 dum_avg, min_dum,
 init_dum,delta_dum,
 c1=0.0,c2=0.0,



 weightage[MAXOBJ],


  EnumData[MAXVECSIZE][MAXENUMDATA];
POPULATION oldpop, newpop;   /* Eski ve yeni populasyonlar   */
int *choices, nremain;
float *fraction;
input_parameters() {
  int   k,temp2,count;
  char  ans;
  float temp1, sumweight;
  printf("\nNumber of objective functions (2) --> ");
  scanf("%d",&num_obj);
  printf("\nSpecify minimization (1) or maximization (-1) for each function
");
 for (count=0; count<num_obj; count++)    {
     printf("\n  Objective function #%2d (1 or -1)      --> ",count+1);
     scanf("%d",&minmax[count]);    }
  printf("\nNumber of variables (Maximum %2d) --- : ",MAXVECSIZE);
  scanf("%d",&num_var);
  BINGA = FALSE;
  REALGA = FALSE;
  TotalChromLength=0;
  num_bin_var=0; num_int_var=0; num_enum_var=0; num_cont_var=0;
  for (k=0; k<= num_var-1; k++)    {
     printf("\nVariable Type for variable #%2d : \n",k+1);
     printf("\t\t 1 for Binary\n");
     printf("\t\t 2 for Integer\n");
     printf("\t\t 3 for Enumerated\n");
     printf("\t\t 4 for Continuous/Real");
     printf("\n  Give your choice (1/2/3/4) ------- : ");
     scanf("%d",&var_type[k]);
          if (var_type[k]!=ENUM)    {
        printf("\nLower and Upper bounds of x[%d] ----- : ",k+1);
        scanf("%f %f",&x_lower[k],&x_upper[k]);       }
         if (var_type[k]!=CONT)
       var_RIGID[k] = TRUE;
         if (var_type[k]==BIN) {
        num_bin_var++;
        BINGA = TRUE;
        printf("\nChromosome Length ------------------ : ");
        scanf("%d",&lchrom[k]);
        TotalChromLength += lchrom[k];       }
     else
      lchrom[k] = 0;       /* End of BIN */

     if (var_type[k]==ENUM)  {
        num_enum_var++;
        REALGA=TRUE;
        printf("\nEnter the data for variable [%d] in ASCENDING
order.\n",k+1);
        printf("End of data recognized by 999999: ");
```

```
        temp2=0;
        scanf("%f",&temp1);
        EnumData[k][0] = temp1;
        temp2++;
        while (!small(temp1-999999.0))      {
           scanf("%f",&temp1);
           EnumData[k][temp2] = temp1;
           temp2++;      }
        num_enum_data[k]=temp2-1;
        x_lower[k]=EnumData[k][0];
        x_upper[k]=EnumData[k][num_enum_data[k]-1];     }
     else
      num_enum_data[k]=0;
        if (var_type[k]==INT)       {
        num_int_var++;
        REALGA=TRUE;
        num_enum_data[k] = x_upper[k]-x_lower[k]+1;
        for (count=0;count<num_enum_data[k];count++)
         EnumData[k][count]=x_lower[k]+count; }
         if (var_type[k]==CONT)            {
        num_cont_var++;
        REALGA=TRUE;
        printf(" Are these bounds rigid ---- ? (y/n) : ");
        do { ans = getchar(); } while (ans!= 'y' && ans !='n');
        if (ans == 'y')  var_RIGID[k] = TRUE;
        else            var_RIGID[k] = FALSE;}    }
  printf("\n");
   for (count=0;count<num_var;count++)    {
     printf("Variable (#%2d) Type : ",count+1);
     switch (var_type[count])  {
       case BIN  : printf("BINARY\n");break;
       case INT  : printf("INTEGER\n");break;
       case ENUM : printf("ENUMERATED\n");break;
       case CONT : printf("CONTINUOUS\n");break;   }
     printf("Lower Bound   : %f\n",x_lower[count]);
     printf("Upper Bound   : %f\n",x_upper[count]);    }
   FITNESS=PARAM=FALSE;
  printf("\n Sharing Strategy :");
  printf("\n Sharing on Fitness Space   (f)");
  printf("\n Sharing on Parameter Space (p)");
  printf("\n Give your choice (f/p) p preferable : ");
  do { ans = getchar(); } while (ans!= 'f' && ans !='p');
  if (ans == 'f') FITNESS = TRUE;
  else               PARAM   = TRUE;
   if (FITNESS)    {
     printf("\nEqual weightage for all objectives (y or n)?: ");
     do { ans = getchar(); } while (ans!= 'y' && ans != 'n');
     if (ans == 'n')    {
        for (count=0, sumweight=0.0; count<num_obj; count++)
          {
            printf("\n  Weight for objective #%2d (0.5) ->
",count+1);
            scanf("%f", &weightage[count]);
            sumweight += weightage[count];
            printf("\n  Lower and Upper values of function #%2d
(approx.) -> ",count+1);
            scanf("%f %f", &afmin[count], &afmax[count]);       }
        for (count=0; count<num_obj; count++)
          weightage[count] /= sumweight;       }
     else
       for (count=0; count<num_obj; count++)
        weightage[count] = 1.0/(double)num_obj;   }
  printf("\nSigma share value accordingly (%8.3f) -- :
",0.5*pow(0.1,1.0/num_var));
  scanf("%f",&dshare);
   printf("\nReports to be printed ----- ? (y/n)  : ");
  do { ans = getchar(); } while (ans!= 'y' && ans !='n');
  if (ans == 'y') REPORT = TRUE;
  else           REPORT = FALSE;
  printf("\nPopulation Size ? (~ 20 times N) ---- : ");
  scanf("%d", &pop_size );
  if (pop_size > MAXPOPSIZE)     {
     printf("\n Increase the value of MAXPOPSIZE in program");
     printf("  and re-run the program");
     exit(-1);  }
  printf("\nCross Over Probability ? ( 0.7 to 1 )  : ");
  scanf("%f",&p_xover);
  printf("\nMutation Probability ? ( 0 to 0.2 ) -- : ");
  scanf("%f",&p_mutation);
  printf("\n Give the strategy for X-over");
  printf("\n 1 : Single-point crossover");
  printf("\n 2 : Uniform crossover");
  printf("\n Give your choice ----------(1/2) : ");
  scanf("%d",&x_strategy);
  if (REALGA) {
     printf("\nDistribution Index for crossover and mutation (30 50) : ");
     scanf("%f %f",&n_distribution_c, &n_distribution_m);   }
  printf("\nHow many generations (100) ? --------- : ");
  scanf("%d",&max_gen);
  printf("\nGive random seed (0 to 1.0)          : ");
  scanf("%f",&seed);
  input_app_parameters();}
initialize(){
  float u;
  int tmp,k,k1,i,j,j1,stop;
  double temp[MAXVECSIZE],coef;
  unsigned mask=1,nbytes;
   randomize();
  app_initialize();
  oldpop = (INDIVIDUAL *)malloc(pop_size*sizeof(INDIVIDUAL));
  newpop = (INDIVIDUAL *)malloc(pop_size*sizeof(INDIVIDUAL));
  if (oldpop == NULL) nomemory("oldpop in initialize()");
  if (newpop == NULL) nomemory("newpop in initialize()");
   if (BINGA)    {
     chromsize = (TotalChromLength/UINTSIZE);
     if (TotalChromLength%UINTSIZE) chromsize++;
     nbytes = chromsize*sizeof(unsigned);
      for(j = 0; j < pop_size; j++)
       {
         if((oldpop[j].chrom = (unsigned *) malloc(nbytes)) == NULL)
           nomemory("oldpop chromosomes");

         if((newpop[j].chrom = (unsigned *) malloc(nbytes)) == NULL)
           nomemory("newpop chromosomes");
       }
   }          /* End of If (BINGA) */

  for (k=0; k<= pop_size-1; k++)
   {
     oldpop[k].flag = 0;
     oldpop[k].parent1 = oldpop[k].parent2 = 0;
     oldpop[k].dumfitness = 0.0;
     for (j=0; j<=num_var-1; j++)
       {
         if (var_type[j]!=BIN)
           {
             u = randomperc();
             oldpop[k].x[j] = x_lower[j] * (1-u) + x_upper[j] * u;

             if ((var_type[j]==INT)||(var_type[j]==ENUM))
               {
                 for(k1=0;k1<num_enum_data[j]-1;k1++)
                   {
                     if (oldpop[k].x[j]<EnumData[j][0])
                       {
                         oldpop[k].x[j]=EnumData[j][0];
                         continue;
                       }
```

```
                else if
(oldpop[k].x[j]>EnumData[j][num_enum_data[j]-1])
                 {
oldpop[k].x[j]=EnumData[j][num_enum_data[j]-1];
       continue;
                 }
                else if ((EnumData[j][k1]<oldpop[k].x[j])&&
                      (EnumData[j][k1+1]>oldpop[k].x[j]))
                  oldpop[k].x[j]=( (EnumData[j][k1+1]-
oldpop[k].x[j])>
                                (oldpop[k].x[j]-EnumData[j][k1]) ) ?
                    EnumData[j][k1] : EnumData[j][k1+1];
              }             }
         }              }

    for(k1 = 0; k1 < chromsize; k1++)
     {
      oldpop[k].chrom[k1] = 0;
      if(k1 == (chromsize-1))
       stop = TotalChromLength - (k1*UINTSIZE);
      else
       stop = UINTSIZE;
      /* A fair coin toss */
      for(j1 = 1; j1 <= stop; j1++)
       {
        if (flip(0.5))
         oldpop[k].chrom[k1] = oldpop[k].chrom[k1]|mask;
        if (j1 != stop)
         oldpop[k].chrom[k1] = oldpop[k].chrom[k1]<<1;
       }              /* End of for (j1) */
     }                /* End of for (k1) */
  }                   /* End of for (k) */
  no_xover = no_mutation = 0;

 init_dum  = pop_size;          /* For Sharing */
 min_dum   = 0.0;
 delta_dum = 0.1*init_dum;
 for (k=0; k<= pop_size-1; k++)       {
    decode_string(&(oldpop[k]));
    objective(&(oldpop[k]));          } }
decode_string(ptr_indiv)
   INDIVIDUAL *ptr_indiv;{
 double *temp,coef[MAXVECSIZE];
 int j;
 if (ptr_indiv == NULL) error_ptr_null("ptr_indiv in
decode_string");
 if (BINGA)    {
    temp = (double *) malloc(num_var * sizeof(double));

    for(j=0; j<num_var; j++)
     temp[j] = 0.0;

    decodevalue(ptr_indiv->chrom,temp);

    for(j=0; j<num_var; j++)
     if (var_type[j]==BIN)      {
        coef[j] = pow(2.0,(double)(lchrom[j])) - 1.0;
        temp[j] = temp[j]/coef[j];
        ptr_indiv->x[j] = temp[j]*x_upper[j] + (1.0 -
temp[j])*x_lower[j];        }
    free(temp);    }}
nomemory(string)
   char *string; {
 printf("\nmalloc: out of memory making %s!!\n",string);
 printf("\n Program is halting .....");
 exit(-1); }
error_ptr_null(string)
   char *string;{
 printf("\n Error !! Pointer %s found Null !",string);
 printf("\n Program is halting .....");
 exit(-1); }
statistics(pop,gen)
   POPULATION pop;
   int gen;{
 int j,iobj;
 float dumsumfit = 0.0;
 float sumfitness[MAXOBJ];

 for (iobj=0; iobj<num_obj; iobj++)    {
    minf[iobj] = maxf[iobj] = 1.0e8;
    sumfitness[iobj] = 0.0;    }
  for (j=0; j<=pop_size-1; j++)    {
    dumsumfit   += pop[j].dumfitness;
    for (iobj=0; iobj<num_obj; iobj++)
     {
      sumfitness[iobj] += pop[j].fitness[iobj];
      if (pop[j].fitness[iobj] > maxf[iobj]) maxf[iobj] = pop[j].fitness[iobj];
      if (pop[j].fitness[iobj] < minf[iobj]) minf[iobj] = pop[j].fitness[iobj];
     }    }
  dum_avg = dumsumfit/(double)pop_size;
 for (iobj=0; iobj<num_obj; iobj++)
  avgfitness[iobj] = sumfitness[iobj]/(double)pop_size;
  app_statistics();}
generate_new_pop(){
 int j,k,k1,mate1,mate2;
 app_computation();
 preselect();
 for (k=0; k<= pop_size-1; k += 2)    {
    mate1 = select();
    mate2 = select();
    switch( x_strategy )                {
     case ONESITE :  cross_over_1_site(mate1,mate2,k,k+1); break;
     case UNIF    :  cross_over_unif(mate1,mate2,k,k+1)  ; break;      }
    mutation(&newpop[k]);        /* Mutation */
    decode_string(&(newpop[k]));
    update_x_BIN_ENUM(&(newpop[k]));
    objective(&(newpop[k]));
    newpop[k].parent1 = mate1+1;
    newpop[k].parent2 = mate2+1;
        mutation(&newpop[k+1]);
    decode_string(&(newpop[k+1]));
    update_x_BIN_ENUM(&(newpop[k+1]));
    objective(&(newpop[k+1]));
    newpop[k+1].parent1 = mate1+1;
    newpop[k+1].parent2 = mate2+1;    }           }
create_children(p1,p2,c1,c2,low,high,RIGID)
   float p1,p2,*c1,*c2,low,high;
   int RIGID;{
 float difference, x_mean, beta, temp;
 float u, u_l, u_u, beta_l=0.0, beta_u=0.0, beta1=0.0, beta2=0.0;
 int flag;
  if (c1 == NULL) error_ptr_null("c1 in create_children");
 if (c2 == NULL) error_ptr_null("c2 in create_children");
 flag = 0;
 if ( p1 > p2) { temp = p1; p1 = p2; p2 = temp; flag = 1; }
 x_mean = ( p1 + p2) * 0.5;
 difference = p2 - p1;
 if(difference <= 1.0e-9)    {
    *c1 = p1;
    *c2 = p2;    }
 else    {
    if (RIGID)        {
       beta_l = 1.0 + 2.0*(p1-low)/(p2-p1);
       beta_u = 1.0 + 2.0*(high-p2)/(p2-p1);
       if (MINSCHEME)           {
          if(beta_l <= beta_u) beta_u = beta_l;
          else beta_l = beta_u;
          u = rndreal(0.0,1.0);
```

```
            beta1 = beta2 = get_beta_rigid(u,beta_l);            }
          else            {
            u_l = rndreal(0.0,1.0);
            beta1 = get_beta_rigid(u_l,beta_l);

            u_u = rndreal(0.0,1.0);
            beta2 = get_beta_rigid(u_u,beta_u);            }        }
      else        {
         u = rndreal(0.0,1.0);
         beta1 = beta2 = get_beta(u);        }
      *c1 = x_mean - beta1 * 0.5 * difference;
      *c2 = x_mean + beta2 * 0.5 * difference;    }
  if (flag == 1)    {
     temp = *c1; *c1 = *c2; *c2 = temp;    }
  if (*c1 < low)  *c1 = low;
  if (*c2 > high) *c2 = high;

  if (*c1 < low || *c2 > high)
    {
      printf("Error!! p1 = %f, p2 = %f, c1 = %f, c2 =
%f\n",p1,p2,*c1,*c2);
      exit(-1);    }}
cross_over_1_site(first,second,childno1,childno2)
int first,second,childno1,childno2;{
    int j,k,site;
       if (flip(p_xover))       {
       no_xover++;
       if (BINGA)
       binary_xover(oldpop[first].chrom
,oldpop[second].chrom,
                newpop[childno1].chrom,newpop[childno2].chrom);
       site = rnd(0,num_var-1);
         for (k=0; k<=site-1; k++)
         if (var_type[site]!=BIN)        {
           newpop[childno1].x[k] = oldpop[first].x[k];
           newpop[childno2].x[k] = oldpop[second].x[k];         }
       for (k=site+1; k<=num_var-1; k++)
         if (var_type[site]!=BIN)        {
           newpop[childno2].x[k] = oldpop[first].x[k];
           newpop[childno1].x[k] = oldpop[second].x[k];         }
 if (var_type[site]!=BIN)
             create_children(oldpop[first].x[site],
oldpop[second].x[site],
                        &(newpop[childno1].x[site]),
                        &(newpop[childno2].x[site]),
 x_lower[site],x_upper[site],var_RIGID[site]);    }
    else             {
      if (BINGA)
      for (k=0; k<=chromsize; k++)      {
       newpop[childno1].chrom[k] = oldpop[first].chrom[k];
       newpop[childno2].chrom[k] = oldpop[second].chrom[k];
}
      for (k=0; k<=num_var-1; k++)      {
       newpop[childno1].x[k] = oldpop[first].x[k];
       newpop[childno2].x[k] = oldpop[second].x[k];      }    }}
float get_delta(u, delta_l, delta_u)
     float u, delta_l, delta_u;{
  float delta, aa;
    if (u >= 1.0-1.0e-9)      delta = delta_u;
  else if (u <= 0.0+1.0e-9) delta = delta_l;
  else    {
     if (u <= 0.5)       {
        aa = 2.0*u + (1.0-2.0*u)*pow((1+delta_l),
                              (n_distribution_m + 1.0));
        delta = pow(aa, (1.0 / (n_distribution_m + 1.0))) - 1.0;
}
     else       {
        aa = 2.0*(1-u) + 2.0*(u-0.5)*pow((1-delta_u),
                                  (n_distribution_m + 1.0));
          delta = 1.0 - pow(aa, (1.0 / (n_distribution_m + 1.0)));    }    }
  if (delta < -1.0 || delta > 1.0)    {
     printf("Error in mutation!! delta = %f\n",delta);
     exit(-1);    }
  return (delta);}
initreport(fp)
FILE *fp;{
   int k, iobj;
   if (fp == NULL) error_ptr_null(" File fp in initreport");
   fprintf(fp,"\n");
   fprintf(fp,"\n Number of objective functions : %2d",num_obj);
   for (iobj=0; iobj<num_obj; iobj++)      {
      fprintf(fp,"\n   Objective function #%2d : ",iobj+1);
      if (minmax[iobj] == 1) fprintf(fp,"Minimize");
      else if (minmax[iobj] == -1) fprintf(fp,"Maximize");     }
   fprintf(fp,"\n\n CROSSOVER TYPE             : ");
   if (BINGA) fprintf(fp,"Binary GA (Single-pt)");
   fprintf(fp,"\n                              ");
   switch (x_strategy)   {
    case ONESITE : fprintf(fp,"\n STRATEGY                  : 1 cross - site
with swapping"); break;
    case UNIF    : fprintf(fp,"\n STRATEGY                  : Uniformly all
variables 50 % "); break;
     default      : fprintf(fp,"\n CROSS OVER NOT SET CORRECTLY ");
break;   }
   fprintf(fp,"\n Population size           : %d",pop_size);
   fprintf(fp,"\n Total no. of generations   : %d",max_gen);
   fprintf(fp,"\n Cross over probability     : %6.4f",p_xover);
   fprintf(fp,"\n Mutation probability       : %6.4f",p_mutation);
   if (BINGA)
     fprintf(fp,"\n String length              : %d",TotalChromLength);
   fprintf(fp,"\n Number of variables");
   fprintf(fp,"\n              Binary      : %d",num_bin_var);
   fprintf(fp,"\n              Integer     : %d",num_int_var);
   fprintf(fp,"\n              Enumerated  : %d",num_enum_var);
   fprintf(fp,"\n              Continuous  : %d",num_cont_var);
   fprintf(fp,"\n                TOTAL     : %d",num_var);
   fprintf(fp,"\n Epsilon for closeness      : %g",closeness);
   if (REALGA)      {
      fprintf(fp,"\n Distribution Index (cross) : %g",n_distribution_c);
      fprintf(fp,"\n Distribution Index (mut)   : %g",n_distribution_m);     }
   fprintf(fp,"\n Sigma-share value          : %6.4f",dshare);
   fprintf(fp,"\n Sharing Strategy           : ");
   if (PARAM)
     fprintf(fp,"sharing on Parameter Space");
   else if (FITNESS)      {
      fprintf(fp,"sharing on Fitness Space");
      for (iobj=0; iobj<num_obj; iobj++)
        fprintf(fp,"\n Weightage for Obj. Fun. #%2d :
%6.4f",iobj+1,weightage[iobj]);      }
   fprintf(fp,"\n Lower and Upper bounds     :");
   for (k=0; k<=num_var-1; k++)
     fprintf(fp,"\n   %8.4f  <=  x%d  <=
%8.4f",x_lower[k],k+1,x_upper[k]);
       app_initreport();}
writechrom(chrom,fp)
unsigned *chrom;
FILE *fp; {
    int j, k, stop,bits_per_var,count=0;
    unsigned mask = 1, tmp;
    int temp[100];
   for (j=0;j<100;j++)
    temp[j]=0;
   if (fp == NULL) error_ptr_null(" File fp in initreport");
   if (BINGA == FALSE) return;
   if (chrom == NULL) error_ptr_null("chrom in writechrom");
    for(k = 0; k < chromsize; k++)     {
       tmp = chrom[k];
       if(k == (chromsize-1))
```

```
            stop = TotalChromLength - (k*UINTSIZE);
        else
            stop = UINTSIZE;
        for(j = 0; j < stop; j++)        {
            if(tmp&mask)        {
                temp[j+count*UINTSIZE]=1;          }
            else        {
                temp[j+count*UINTSIZE]=0;          }
            tmp = tmp>>1;  }
        count++;    }
    count=0;
     for(j=0;j<num_var;j++) {
        for(k=count+lchrom[j]-1;k>=count;k--) {
          fprintf(fp,"%d",temp[k]);       }
        count+=lchrom[j];
        if (j!=num_var-1) fprintf(fp,"_"); }
FILE *fp;
int num; {
  int k,j;
  char string[30];
 if (fp == NULL) error_ptr_null(" file fp in report()");
 if (BINGA)
      fprintf(fp,"\n#  No.          x         Obj. Fun. Values
(f1,f2,etc.)     String");
    else
      fprintf(fp,"\n#  No.          x         Obj. Fun. Values
(f1,f2,etc.)  ");
      for (k=0; k<= pop_size-1; k++)    {
     fprintf(fp,"\n %3d. x%d = [%10.3f ]
",k+1,1,oldpop[k].x[0]);
      for (j= 1; j<=num_var-1; j++)
       fprintf(fp,"\n      x%d = [%10.3f ]   ",j+1,oldpop[k].x[j]);
        if (BINGA)        {
 fprintf(fp,"        ");
 writechrom(oldpop[k].chrom,fp);        }
  fprintf(fp,"\n");    }
     fprintf(fp,"\n\n");
   app_report(); }
free_all() {
  int i;
   for(i = 0; i < pop_size; i++)    {
      free(oldpop[i].chrom);
      free(newpop[i].chrom);   }
  free(oldpop);
  free(newpop);
  select_free();
  app_free(); }
adjust(index)
     int index;{
 int i;
  double diff;
  diff = 2.0 * delta_dum - min_dum ;
  for(i=0; i<pop_size; i++)
    if(oldpop[i].flag == 1 || oldpop[i].flag == 0) continue;
    else oldpop[i].dumfitness += diff ;
   minimum_dum();}
phenoshare(){
    int i,j;
  float  dvalue,d,nichecount;
  double pow();
  float  distance();
    for(j=0; j<pop_size; j++)     {
      nichecount = 1.0;
      if(oldpop[j].flag == 2)      {
          for(i=0; i<pop_size; i++)         {
              if(i == j) continue;
                    if(oldpop[i].flag == 2)             {
                d = distance(&(oldpop[j]),&(oldpop[i]));
               if (d < 0.0) d = (-1.0)*d ;
               if (d <= 0.000001) nichecount++ ;
               else if(d < dshare)
                 nichecount += (1.0-(d/dshare))*(1.0-(d/dshare));             } }
}
     oldpop[j].dumfitness /= nichecount ;          } }
float  distance(critter1,critter2)
    INDIVIDUAL *critter1;
    INDIVIDUAL *critter2;{
  int i, iobj;
  double dst,sqrt();
    dst = 0.0;
  if (FITNESS) {
      for(iobj=0; iobj<num_obj; iobj++)
        dst += weightage[iobj]*square(critter1->fitness[iobj] - critter2-
>fitness[iobj])/square(afmax[iobj]-afmin[iobj]);    }
  else if (PARAM) {
      for (i=0;i<num_var;i++)
        dst += square(critter1->x[i]-critter2->x[i])/square(x_upper[i]-
x_lower[i]);    }
  dst = sqrt(dst);
  return(dst);}
minimum_dum()
{   int i;
  min_dum = 1000000000.0 ;
  for(i=0; i<pop_size; i++)
    {
      if(oldpop[i].flag == 2)
        {
          if(oldpop[i].dumfitness < min_dum) min_dum =
oldpop[i].dumfitness;     } }}
/*=================================================
====
  ANA PROGRAM ;
=================================================
===*/
main(){
  FILE *fp_out;
  FILE *fp_report;
  int j,k,k1;
  POPULATION     temp;
  input_parameters();
  fp_out = fopen("result.out","w+");
  fp_report = fopen("report","w+");
   select_memory();
  initreport(fp_report);
    gen_no=0;
  initialize();
  MakeFronts();
  statistics(oldpop,gen_no);
  if (REPORT) report(fp_report,gen_no);
  for(gen_no = 1; gen_no<=max_gen; gen_no++)
    {
      printf("."); if (gen_no%60==0) printf("\n");
      fflush(stdout);
     generate_new_pop();
      temp   = oldpop;
      oldpop = newpop;
      newpop = temp  ;
      MakeFronts();
      statistics(oldpop,gen_no);
       if (gen_no==max_gen) result(fp_out,gen_no);
      if (REPORT) report(fp_report,gen_no);   };
  free_all();
  fclose(fp_out);
  fclose(fp_report);
  app_closure();  }
/**************** Ana Programın Sonu
**************************/
```

ÖZGEÇMİŞ

1986 yılında Elazığ'da doğdu. İlk ve orta öğrenimini Elazığ'da tamamladıktan sonra 2003 yılında Fırat Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği Bölümüne girdi. 2007 yılında bu bölümden mezun oldu. Aynı yıl Fırat Üniversitesi Tunceli Meslek Yüksekokuluna Uzman olarak atandı. 2008 yılında Biyomühendislik Anabilim Dalında yüksek lisansa başladı. Halen Tunceli Üniversitesi Tunceli Meslek Yüksekokulunda Uzman olarak görev yapmaktadır.