T.C.
TUNCELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÜNCEL SÜRÜ ZEKÂSI ALGORİTMALARIYLA SINIFLANDIRMA KURALLARININ KEŞFİ

YÜKSEK LİSANS TEZİ

Sinem AKYOL

Anabilim Dalı: Elektrik-Elektronik Mühendisliği

DANIŞMAN

Yrd. Doç. Dr. Bilal ALATAŞ

NİSAN-2013

T.C.
TUNCELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÜNCEL SÜRÜ ZEKÂSI ALGORİTMALARIYLA SINIFLANDIRMA KURALLARININ KEŞFİ

YÜKSEK LİSANS TEZİ

Sinem AKYOL

0921031017

Tezin Enstitüye Verildiği Tarih : 29.04.2013
Tezin Savunulduğu Tarih : 17.04.2013

Tez Danışmanı : **Yrd. Doç. Dr. Bilal ALATAŞ (T.Ü.)**

Diğer Jüri Üyeleri : **Yrd. Doç. Dr. Eyyüp ÖKSÜZTEPE (T.Ü.)**
Yrd. Doç. Dr. Ömer ÇELİK (T.Ü.)

NİSAN-2013

Sinem AKYOL tarafından hazırlanan GÜNCEL SÜRÜ ZEKÂSI ALGORİTMALARIYLA SINIFLANDIRMA KURALLARININ KEŞFİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Bilal ALATAŞ
Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği/~~oy çokluğu~~ ile Elektrik Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. Bu tez, Tunceli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

Başkan : Yrd. Doç. Dr. Bilal ALATAŞ (T.Ü.)

Üye : Yrd. Doç. Dr. Eyyüp ÖKSÜZTEPE (T.Ü.)

Üye : Yrd. Doç. Dr. Ömer ÇELİK (T.Ü.)

Tarih : 17.04.2013

ÖNSÖZ

Çalışmalarım boyunca, değerli görüş ve katkılarıyla beni yönlendiren, her konuda desteğini esirgemeyen, kıymetli tecrübelerinden faydalandığım tez danışmanım Sayın Yrd. Doç. Dr. Bilal ALATAŞ'a teşekkürü borç bilirim. Ayrıca YLTUB011-15 kodlu proje ile çalışmalarıma maddi destek sağlayan Tunceli Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne teşekkür ederim.

Sinem AKYOL

TUNCELİ-2013

İÇİNDEKİLER

Sayfa

ÖZET

Optimizasyon, bir problemde belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçme işlemidir. Optimizasyon problemleri için birçok algoritma önerilmiştir. Sezgisel algoritmalar, büyük boyutlu optimizasyon problemleri için, kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalardır. Genel amaçlı sezgisel optimizasyon algoritmaları, biyoloji tabanlı, fizik tabanlı, sürü tabanlı, sosyal tabanlı, müzik tabanlı ve kimya tabanlı olmak üzere altı farklı grupta değerlendirilmektedir. Sürü zekâsı tabanlı optimizasyon algoritmaları kuş, balık, kedi ve arı gibi canlı sürülerinin hareketlerinin incelenmesiyle geliştirilmiştir. Veri madenciliği, büyük ölçekli verilerden anlamlı ve faydalı bilginin keşfedilmesi işlemidir. Sınıflandırma kural madenciliği en çok kullanılan veri madenciliği yöntemlerinden biridir ve bu yöntemle veri kümelerinden kullanıcıların rahatça anlayabileceği kurallar çıkarılmaktadır.

Bu çalışmada, sürü zekâsı optimizasyon algoritmaları (Ateşböceği Algoritması, Ateşböceği Sürü Optimizasyonu, Karınca Koloni Optimizasyonu, Parçacık Sürü Optimizasyonu, Yapay Balık Sürüsü Algoritması, Bakteriyel Besin Arama Optimizasyon Algoritması, Kurt Koloni Algoritması) tanıtılmış ve bu optimizasyonlardan Kedi Sürüsü Optimizasyon Algoritması ile Yapay Arı Koloni Algoritması ayrıntılı olarak incelenmiştir. Visual C# dilinde Kedi Sürüsü Optimizasyon Algoritmasına uygun bir program yazılmıştır ve hazırlanan bu programla UCI veri ambarından alınan 4 adet veritabanında sınıflandırma kuralları keşfedilmiştir. Bulunan sonuçlarla Weka programından elde edilen sonuçlar karşılaştırılmıştır.

Kedi Sürüsü Optimizasyon Algoritmasının, sınıflandırma kural madenciliğinde kullanılması ilk kez bu çalışmada yapılmıştır. Herhangi bir optimizasyon yapılmadığı halde sınıflandırma kural keşfinde etkili bir yöntem olmaktadır. Önerilen bu yöntemle istenilen özelliklere göre uygunluk fonksiyonuna ilaveler esnek şekilde yapılabilmektedir. Geliştirilen programda, bulunan kurallardaki niteliklerin değer aralıkları kurallarla eş zamanlı olarak bulunmaktadır, ayrıca bir ön işlem yapılmamaktadır.

Anahtar Kelimeler: Sürü tabanlı optimizasyon, Kedi sürüsü optimizasyonu, Yapay arı koloni algoritması, Veri madenciliği, Sınıflandırma kural madenciliği

ABSTRACT

Classification Rule Mining with Current Swarm Intelligence Algorithms

Optimization is the process of finding the best solution of a problem. There are many optimization algorithms proposed for optimization problems. The heuristic algorithm can give solutions close to optimum in an acceptable period of time for large-scaled optimization problems. The metaheuristic optimization algorithms are evaluated in six different groups which are biology-based, physics-based, swarm-based, social-based, music-based and chemistry-based. The swarm-based optimization algorithms have been developed by observing the behaviors of creatures e.g. birds, fishes, cats, bees etc. Classification rule mining is one of the most commonly used data mining methods and with this method users can easily understand the rules extracted form data sets.

In this study, swarm-based optimization algorithms (Firefly Algorithm, Glowworm Swarm Optimization, Ant Colony Optimization, Particle Swarm Optimization, Artificial Fish-Swarm Algorithm, Bacterial Foraging Optimization Algorithm, and Wolf Colony Algorithm) are described and Cat Swarm Optimization and Artificial Bee Colony Algorithms are studied in detail. In Visual C# programming language, a program is written in accordance with Cat Swarm Optimization Algorithm and classification rules are discovered within 4 databases obtained from the UCI data warehouse.

In this study, Cat Swarm Optimization Algorithm has been firstly used for classification rule mining. Although there is no much more optimization, Cat Swarm Optimization is an effective method for classification rule mining. Additions could be flexibly made to fitness function in accordance with desired properties. In the developed program, the ranges of values of attributes in the rules are synchronously adjusted with the rules obtained from the proposed method, any preprocess is not applied for the ranges of attributes.

Keywords: Swarm-based optimization, Cat swarm optimization, Artificial bee colony algorithm, Data mining, Classification rule mining

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR

AA : Arılar Algoritması

ABC : Yapay Arı Koloni Algoritması

AHH : Arama Hafızası Havuzu

AKO : Arı Koloni Optimizasyonu

AOE : Bal- Arısı Optimizasyonunda Evlilik

AS : Arı Sistemi

BAS : Bulanık Arı Sistemi

DBS : Değişen Boyutların Sayısı

DO : Değişim Oranı

GA : Genetik Algoritma

GSP : Gezgin Satıcı Problemi

HBMO : Bal-Arısı Eşleşme Optimizasyonu

KGA : Kraliçe-Arı Gelişim Algoritması

KPD : Kendi Pozisyonunu Değerlendirme

KSO : Kedi Sürüsü Optimizasyonu

MÇA : Maksimum Çevrim Sayısı

MR : Karışım Oranı

PSO : Parçacık Sürü Optimizasyonu

SAA : Sanal Arı Algoritması

SBA : Seçilen Boyutun Arama Aralığı

SEMBOLLER LİSTESİ

a_i	: *i.* niteliğin alt sınırı
c_1	: Bir sabit
DN	: Doğru negatifler
DP	: Doğru pozitifler
$f(v_i)$	: Kaynağın maliyet değeri
$hata_i$	: Geliştirilememe sayacı
j	: Bir kediden elde edilen kopya sayısı
$kedi_k$	: Mevcut kedinin pozisyonu
M	: Optimize edilecek parametre sayısı
N	: Yiyecek kaynağı sayısı
n	: Veri kümesindeki karar niteliklerinin sayısı
$Nitelik_i maks$	: *i*. niteliğin veri kümesinde aldığı maksimum değer
$Nitelik_i min$	: *i.* niteliğin veri kümesinde aldığı minimum değer
$\omega_1, \omega_2, \omega_3, \omega_4$	: Ağırlıklar
P_i	: *i.* kopyanın seçilme oranı
r_1	: Gelişigüzel bir değer
UD	: Uygunluk değeri
UD_i	: Kedinin *i.* kopyasının uygunluk değeri
UD_{maks}	: Maksimum uygunluk değeri
UD_{min}	: Minimum uygunluk değeri
$\ddot{u}_i$	: *i.* niteliğin üst sınırı
$v_{k,d}$	: *k.* kedinin *d.* pozisyonunun hızı
$x_{best,d}$	: En iyi uygunluk değerine sahip kedinin pozisyonu
x_{ij}	: *i.* yiyecek kaynağının *j.* parametresi
$x_{k,d}$	: *k.* kedinin *d.* pozisyonu
YN	: Yanlış negatifler
YP	: Yanlış pozitifler

1. GİRİŞ

Optimizasyon, en iyileme anlamına gelmektedir. Bir problem için, verilen şartlar altında tüm çözümler arasından en iyi çözümü elde etme işidir. Belirli sınırlamaları sağlayacak şekilde, bilinmeyen parametre değerlerinin bulunmasını içeren herhangi bir problem, optimizasyon problemi olarak adlandırılabilmektedir (Murty, 2003).

Bazen tek başlarına hiçbir iş yapamayan varlıklar, toplu hareket ettiklerinde çok zekice davranışlar sergileyebilmektedir. Bir topluluğa ait bireyler, en iyi bireyin davranışından ya da diğer bireylerin davranışlarından ve kendi deneyimlerinden yararlanarak yorum yapmakta ve bu bilgileri ileride karşılaşacakları problemlerin çözümleri için bir araç olarak kullanmaktadırlar. Örneğin, bir canlı sürüsünü oluşturan bireylerden birisi bir tehlike sezdiğinde bu tehlikeye karşı tepki verir ve bu tepki sürü içinde ilerleyip tüm bireylerin tehlikeye karşı ortak bir davranış sergilemesini sağlar. Canlıların sürü içerisindeki bu hareketleri gözlemlenerek sürü zekâsı tabanlı optimizasyon algoritmaları geliştirilmiştir.

Veri madenciliği, büyük ölçekli veriler arasından anlamlı ve faydalı bilginin çıkarılması işlemidir. Bu konu için çok sayıda yöntem ve algoritma geliştirilmiştir. Sınıflandırma kural madenciliği en çok kullanılan veri madenciliği yöntemlerinden biridir. Bu yöntemle veri kümelerinden kullanıcıların rahatça anlayabileceği kurallar çıkarılmaktadır. Sınıflandırma kural madenciliği yöntemlerine örnek olarak karar ağaçları, yapay sinir ağları ve genetik algoritmalar verilebilir.

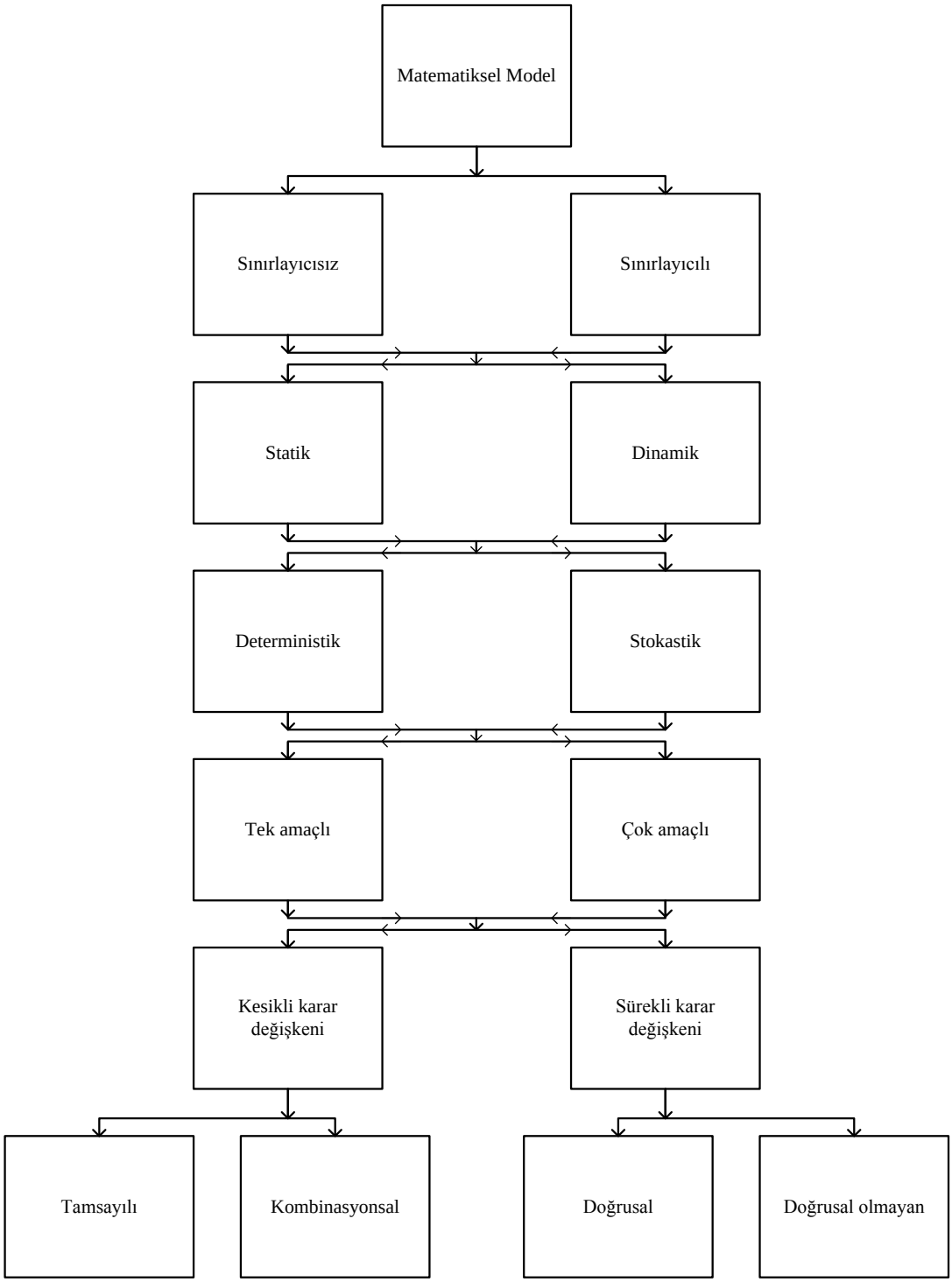
Bu tez çalışmasında sürü optimizasyon algoritmaları anlatılmış ve bu algoritmalardan Kedi Sürüsü Optimizasyonu (KSO) ve Yapay Arı Koloni Algoritması (YAKA) ayrıntılı olarak incelenmiştir. İkinci bölümde sezgisel optimizasyon ve sezgisel optimizasyonun grupları hakkında bilgi verilmiştir. Üçüncü bölümde sürü zekâsı optimizasyon algoritmalarından bahsedilmiştir. Dördüncü bölümde KSO hakkında bilgi verilmiştir. Beşinci bölümde YAKA ayrıntılı olarak incelenmiştir. Altıncı bölümde, veri madenciliği ve yöntemlerinden bahsedilmiştir. Yedinci bölümde, Kedi Sürüsü Optimizasyon Algoritması kullanılarak sınıflandırma kurallarının keşfi anlatılmıştır ve Weka programı hakkında bilgi verilmiştir. Son olarak sonuçlar sekizince bölümde yazılarak çalışma sonlandırılmıştır.

2. OPTİMİZASYON

Optimizasyon, en iyileme anlamına gelmektedir. Bir problem için, verilen şartlar altında tüm çözümler arasından en iyi çözümü elde etme işidir. Belirli sınırlamaları sağlayacak şekilde, bilinmeyen parametre değerlerinin bulunmasını içeren herhangi bir problem, optimizasyon problemi olarak adlandırılabilmektedir. Optimizasyon işleminde ilk adım olarak karar parametreleri veya karar değişkenleri ya da tasarım parametreleri olarak da adlandırılan parametreler kümesinin tanımlanması gerekmektedir. Karar değişkenlerinin amaç üzerindeki etkilerinin analitik olarak gösterilmesiyle amaç fonksiyonu oluşturulur. Çoğu durumda, karar değişkenlerinin sadece belirli değerleri kullanılmalıdır. Karar değişkenlerinin değerleri üzerindeki bu sınırlandırmalara sınırlayıcılar denir. Farklı bir ifadeyle optimizasyon, karar değişkenlerinin mümkün olan tüm kombinasyonları arasından verilen tüm sınırlayıcıları sağlayan ve amaç fonksiyonunu en iyi hale getiren (maksimizasyon ya da minimizasyon) kombinasyonun bulunması işidir.

Bu amaçla literatürde birçok optimizasyon algoritması önerilmiştir. Optimizasyon probleminin kolayca çözülebilecek bir yapıya oturtulması için çoğu zaman problemin davranışlarıyla ilgili kurallar ve elemanları arasındaki bağlantılar için, ilgilenilen karar probleminin yapısına göre şekillenen matematiksel modeller kurulur (Murty, 2003).

Model, eğer karar değişkenleri üzerinde hiçbir sınırlama yoksa sınırlayıcısız, en azından bir sınırlama olması durumunda sınırlayıcılı olur. Gerçek hayatta genellikle sınırlayıcılı problemler karşımıza çıkar. Eğer problem tek bir dönem için çözülecekse statik model, birden fazla dönem göz önüne alınarak çözülecekse dinamik model kullanılır. Modelin algoritmada işletilmesi esnasında belirli, kesin parametre veya girdiler kullanılıyorsa model deterministik, olasılık özelliği varsa model stokastiktir. Eğer birden fazla amaç varsa, çok amaçlı problemler ortaya çıkar. Eğer tüm karar değişkenleri pozitif reel (gerçel) değerler alıyorsa sürekli optimizasyon problemi söz konusudur. Tüm karar değişkenlerinin tamsayı değerler alması gerekiyorsa kesikli optimizasyon problemi ortaya çıkar. Bazı karar değişkenlerinin reel, bazılarının tamsayı değer alması durumunda ise karışık kesikli optimizasyon problemi ile karşılaşılır. Eğer karar değişkenlerinin kombinasyonsal seçenekleri söz konusuysa kombinasyonsal optimizasyon problemleri ortaya çıkar (Murty, 2003). Şekil 2.1'de bu model türleri grafiksel olarak görülmektedir. Burada yukarıdan aşağıya ve soldan sağa gidildikçe modelin kurulması ve işletilmesi zorlaşır.



Şekil 2.1. Optimizasyon için matematiksel modeller

Optimizasyon algoritmalarının çoğu, sistemin modeli ve amaç fonksiyonu için matematiksel modellere ihtiyaç duymaktadır. Karmaşık sistemler için matematiksel modelin kurulması çoğu zaman zordur. Model kurulsa bile, çözüm zamanı maliyeti çok yüksek olduğundan kullanılamamaktadır (Güden vd., 2005). Klasik optimizasyon

algoritmaları, büyük ölçekli kombinasyonsal ve doğrusal olmayan problemlerde yetersizdir. Aynı durum, tamsayı ya da ayrık karar değişkenlerinin kullanıldığı çoğu doğrusal optimizasyon modelleri için de geçerlidir. Bu tür algoritmalar, verilen bir probleme bir çözüm algoritması uyarlamada etkin değildir. Bu da çoğu durumda, geçerliliğinin onaylanması zor olabilen bazı varsayımları gerektirir. Genellikle klasik algoritmaların doğal çözüm mekanizmalarından dolayı, ilgilenilen problem algoritmanın onu idare edeceği şekilde modellenir.

Klasik optimizasyon algoritmalarının çözüm stratejisi genellikle amaç ve sınırlayıcıların tipine (doğrusal, doğrusal olmayan vb.) ve problemi modellemede kullanılan değişkenlerin tipine (tamsayı, reel) bağlıdır. Bunların etkinliliği aynı zamanda problem modellemede çözüm uzayı (konveks, konveks olmayan vb.), karar değişken sayısı ve sınırlayıcı sayısına oldukça bağlıdır. Diğer önemli bir eksiklik ise farklı tipte karar değişkenleri, amaç ve sınırlayıcıların olması durumunda problem formülasyonlarına uygulanabilecek genel çözüm stratejileri sunmamalarıdır. Yani çoğu algoritma belirli tipteki amaç fonksiyonu ya da sınırlayıcıların olduğu modelleri çözmektedir. Ancak çoğu yönetim bilimi, bilgisayar, mühendislik gibi bir çok farklı alandaki optimizasyon problemleri eşzamanlı olarak formülasyonlarında farklı tipteki karar değişkenleri, amaç fonksiyonu ve sınırlayıcıları gerektirir. Bu yüzden klasik sezgisel ve genel amaçlı sezgisel optimizasyon algoritmaları önerilmiştir. Bunlar son yıllarda oldukça popüler yöntemler haline gelmiştir çünkü, bunların hesaplama gücü iyidir ve dönüşümleri kolaydır. Yani tek amaç fonksiyonlu bir problem için yazılmış bir sezgisel program, kolaylıkla çok amaçlı bir probleme ya da farklı bir probleme uyarlanabilmektedir (Alataş, 2007).

2.1. Sezgisel Optimizasyon

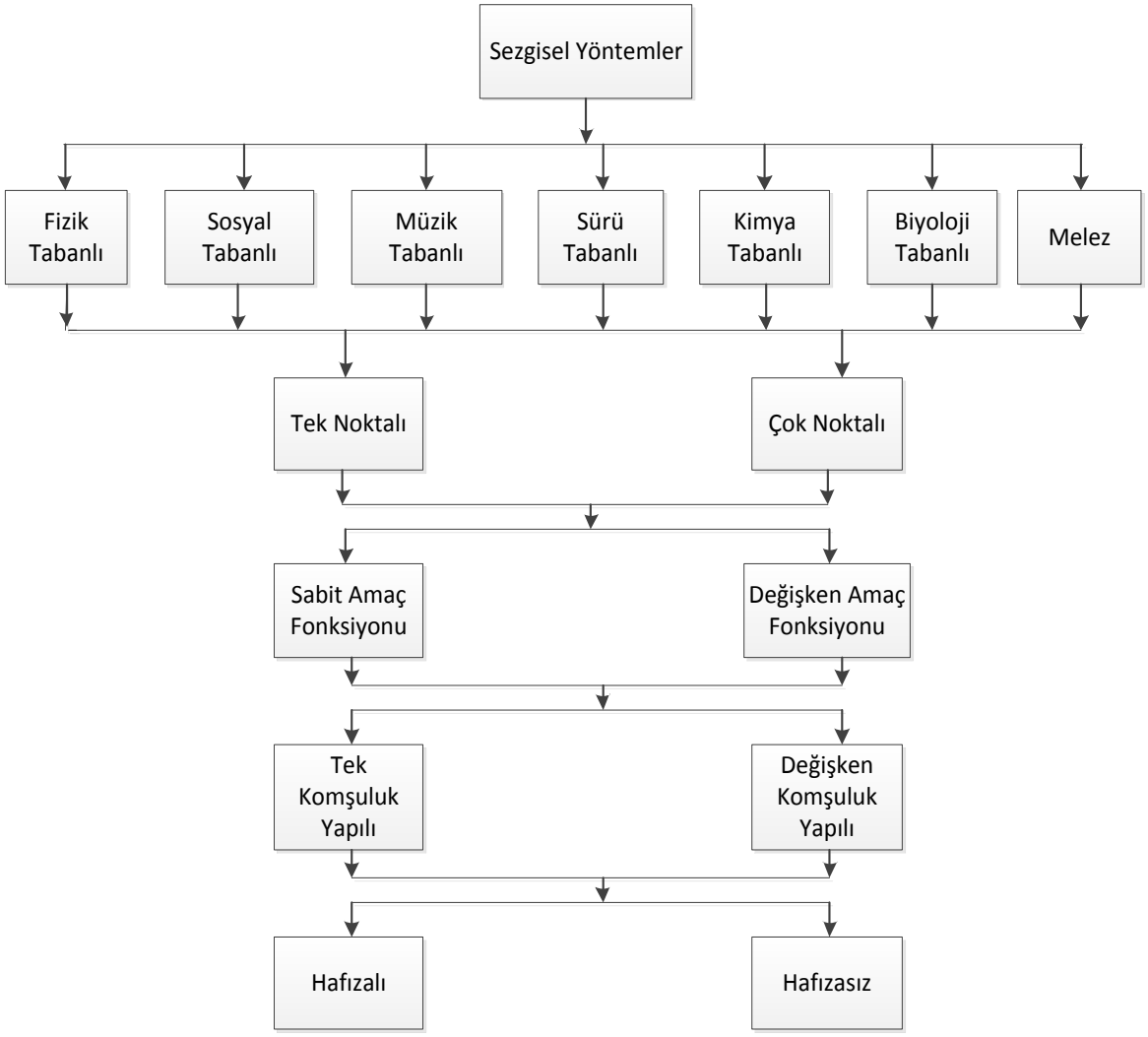
Gerçek yaşam problemlerinin çoğunda problemin çözüm uzayı sonsuz veya tüm çözümlerin değerlendirilemeyeceği kadar büyük olur. Bunun için kabul edilebilir bir sürede çözümlerin değerlendirilerek iyi bir çözümün bulunması gerekmektedir. Böyle problemler için kabul edilebilir bir sürede çözümlerin değerlendirilmesiyle aslında tüm çözüm uzayında "bazı çözümlerin" değerlendirilmesi aynı anlama gelmektedir. Bazı çözümlerin neye göre ve nasıl seçileceği sezgisel tekniğe göre değişir. Değerlendirmeye dahil olan çözümlerin içerisinde optimal çözümün yer alması garanti edilememektedir. Bu sebeple de sezgisel tekniklerin bir optimizasyon problemine önerdiği çözüm, optimal değil iyi çözüm olarak algılanmalıdır (Cura, 2008).

Herhangi bir amacı gerçekleştirmek veya hedefe varmak için çeşitli alternatif hareketlerden etkili olanlara karar vermek amacı ile tanımlanan kriterler veya bilgisayar metotlarıdır. Bu algoritmalar, çözüm uzayında optimum çözüme yakınsaması ispat edilemeyen algoritmalar olarak da adlandırılır. Bu tür algoritmalar yakınsama özelliğine sahiptir, ama kesin çözümü garanti edemezler ve sadece kesin çözüm yakınındaki bir çözümü garanti edebilirler.

Sezgisel algoritmalara gerek duyulmasının sebepleri şunlardır: a) Optimizasyon problemi kesin çözümü bulma işleminin tanımlanamadığı bir yapıya sahip olabilir. b) Anlaşılırlık açısından sezgisel algoritmalar karar verici açısından çok daha basit olabilir. c) Sezgisel algoritmalar, öğrenme amaçlı ve kesin çözümü bulma işleminin bir parçası olarak kullanılabilir. d) Matematik formülleriyle yapılan tanımlamalarda genellikle gerçek dünya problemlerinin en zor tarafları (hangi amaçlar ve hangi sınırlamalar kullanılmalı, hangi alternatifler test edilmeli, problem verisi nasıl toplanmalı) ihmal edilir. Model parametrelerini belirleme aşamasında kullanılan verinin hatalı olması, sezgisel yaklaşımın üretebileceği alt optimal çözümden daha büyük hatalara sebep olabilir (Karaboğa, 2011).

Genel amaçlı sezgisel yöntemler; biyoloji tabanlı, fizik tabanlı, sürü tabanlı, sosyal tabanlı, müzik tabanlı ve kimya tabanlı olmak üzere altı farklı grupta değerlendirilmektedir. Ayrıca bunların birleşimi olan melez yöntemler de vardır. Bahsedilen bu yöntemler Şekil 2.2'de sunulmaktadır. Genetik algoritma (GA) (Holland, 1975), diferansiyel gelişim algoritması (Storn ve Price, 1995) ve karınca koloni algoritması (Dorigo vd., 1991) biyolojik tabanlı; emperyalist yarışmacı algoritma (Atashpaz-Gargari ve Lucas, 2007) ve parlamenter optimizasyon algoritması (Borji, 2007) sosyal tabanlı;

yapay kimyasal reaksiyon algoritması (Alataş, 2011) kimya tabanlı; armoni arama algoritması (Geem vd., 2001) müzik tabanlı; yerçekimsel arama algoritması (Rashedi vd., 2009) ve zeki su damlacıkları algoritması (Shah-Hosseini, 2009) fizik tabanlı ve Parçacık Sürü Optimizasyonu (PSO) (Kennedy ve Eberhart, 1995), KSO (Chu vd., 2006) sürü tabanlı algoritma ve modellerdir. Kültürel algoritma da hem biyoloji hem de sosyal tabanlı algoritma olarak sınıflandırılabilir (Alataş, 2007).



Şekil 2.2. Sezgisel yöntemler

3. SÜRÜ ZEKÂSI OPTİMİZASYON ALGORİTMALARI

Sürü, birbirleriyle etkilesen dağınık yapılı bireyler yığını anlamında kullanılır. Bireyler insan veya karınca olarak ifade edilebilir. Sürülerde N adet temsilci bir amaca yönelik davranışı gerçekleştirmek ve hedefe ulaşmak için birlikte çalışmaktadır. Kolaylıkla gözlenebilen bu "kollektif zekâ" temsilciler arasında sık tekrarlanan davranışlardan doğmaktadır. Temsilciler faaliyetlerini idare etmek için basit bireysel kurallar kullanmakta ve grubun kalan kısmıyla etkileşim yolu ile sürü amaçlarına ulaşmaktadır. Grup faaliyetlerinin toplamından bir çeşit kendini örgütleme doğmaktadır.

Aşağıda şimdiye kadar farklı araştırmacıların önerdiği sürü zekâsı optimizasyon algoritmaları alt başlıklar halinde açıklanmıştır.

3.1. Ateş Böceği Algoritması (Firefly Algorithm)

Ateş böceği algoritması, Dr. Xin-She Yang (Cambridge Üniversitesi-2007) tarafından geliştirilen ve tropikal iklim bölgelerindeki ateşböceklerinin sosyal davranışlarını baz alan bir metasezgisel optimizasyon algoritmasıdır (Yang, 2009). Bu algoritma diğer sürü zekâsı tabanlı algoritmalarla bir çok benzerliği bulunmasına rağmen kavram ve uygulamada daha basittir. Bir ateş böceğinin ışıklarını yakıp söndürmesinin birincil amacı, diğer ateş böceklerini çekmek için bir sinyal sistemi olarak hareket etmektir. Yanıp sönen ışıkların üretimindeki karmaşık biyokimyasal sürecin detayları ve gerçek amacı bilim dünyasında hâlâ bir tartışma konusu olmasına rağmen, birçok araştırmacı yanıp sönen ışıkların ateşböceğine, arkadaşlarını bulmada, olası avlarını çekmede ve avcılarından kendilerini korumada yardımcı olduğuna inanmaktadır.

Ateşböceği algoritmasında, verimli optimal çözümler elde etmek için, verilen bir optimizasyon probleminin amaç fonksiyonu, ateşböceği sürüsüne parlak ve daha çekici yerlere gitmede yardım eden yanıp sönen ışık ya da ışık şiddeti ile ilişkili olmaktadır. Bütün ateş böcekleri tek cins olarak kabul edilmektedir ve birbirilerini çekmeleri bu algoritmanın temelini oluşturmaktadır. Bir ateş böceği ne kadar parlak olursa diğer ateş böcekleri için o kadar çekici hale gelmektedir. Kendisinden daha parlak bir ateşböceği gördüğünde ona doğru gidecektir (Apostolopoulos ve Vlachos, 2011; Yang, 2010).

3.2. Ateş Böceği Sürü Optimizasyonu (Glowworm Swarm Optimization)

Ateş böceği sürü optimizasyonu, K. N. Krishnanand ve D. Ghose tarafından 2005 yılında geliştirilmiştir (Krishnanand ve Ghose, 2005). Çok modelli fonksiyonları optimize etmek için önerilen sürü zekâsı tabanlı bir algoritmadır. Bu yöntemi kullanmanın temel amacı tüm yerel maksimumları yakalamayı sağlamaktır. Çok modelli fonksiyon optimizasyon problemlerinde, ateş böceği sürü optimizasyonu ve önceki yaklaşımlar arasındaki en önemli fark, birden çok zirveyi verimli bir şekilde yerleştiren sürüdeki bireylerin kullandığı dinamik karar alanıdır. Sürüdeki her bir birey komşularını seçmek için karar alanını kullanmaktadır ve komşularından aldığı sinyal gücüyle hareketlerini belirlemektedir. Bu biraz, bir ateş böceğinin eşlerini veya avlarını çekmek için kullandığı ışık yakıp söndürmeye neden olan lusiferine (bir enzim ile birleşerek ışın üreten bir madde) benzer olmaktadır. Daha parlak ışık daha çekici olmaktadır.

Bu algoritmanın ateş böceği algoritmasından (firefly algorithm) farkı "komşuların yeterlilik sayısı" sınırı olmaması ve mesafeye dayalı herhangi bir algı sınırı olmamasıdır (Krishnanand ve Ghose, 2009).

3.3. Karınca Koloni Optimizasyonu (Ant Colony Optimization)

Gerçek karınca koloni davranışlarının matematiksel modelleri üzerine dayalı bir algoritmadır. İlk çalışma Dorigo ve arkadaşları tarafından yapılmıştır (Dorigo vd., 1991). Yaptıkları çalışmada kendi sistemlerine “karınca sistemi”, elde ettikleri algoritmaya ise “karınca algoritması” adını vermişlerdir. Karınca çevre şartlarına göre besin kaynağı ile evi arasında gidebileceği yolları belirlemektedir. Belirlenen yollardan birinden ilk geçen karınca yola feremon adında bir koku bırakmaktadır. Eğer yol kısa ise bu koku daha yoğun olmaktadır ve diğer karıncalar da aynı şekilde yolda devam etmektedirler. İki yolun kesiştiği noktada karınca hangi yola gideceğini belirlemektedir. Hangi yolu seçeceğine ilk önce koku miktarının yoğunluğuna göre ikinci olarak ise gelişigüzel bir ölçüte göre karar vermektedir. Bu gelişigüzel seçimin nedeni ise bütün karıncaların aynı yolda gitmesini engelleyerek yeni ve daha kısa yolları keşfetmektir (Karaboğa, 2011).

3.4. Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)

Sezgisel yöntemlerden biri olan PSO tekniği ilk olarak kuş ve balık sürülerinin hareketlerinden esinlenerek doğrusal olmayan nümerik problemlere optimal sonuçlar bulmak için 1995–1996 yıllarında sosyolog-psikolog James Kennedy ve elektrik mühendisi Russel Eberhart tarafından ortaya atılmıştır (Kennedy ve Eberhart, 1995). PSO popülasyon tabanlı olasılıksal bir optimizasyon yöntemi olup çok parametreli ve çok değişkenli optimizasyon problemlerine çözümler üretmek için kullanılmaktadır (Alataş, 2007).

Parçacık sürü kavramı basitleştirilmiş sosyal sistemin bir simülasyonu olarak ortaya çıkmıştır. Başlangıçtaki amaç, kuş ya da balık sürü koreografisinin grafiksel olarak simülasyonlarını yapmaktır. Ancak grafiksel simülasyondan sonra, parçacık sürü modelinin bir optimizasyon yöntemi olarak kullanılabileceği keşfedilmiştir.

Kuş toplulukları gerçek yiyecek kaynağını bilmemelerine rağmen, yiyecek kaynağından ne kadar uzakta olduklarını öğrenmeye çalışırlar. Öğrenmek için izlenen yöntem yiyecek kaynağına en yakın olan kuşu izlemektir. PSO'da her bir kuş parçacık olarak, kuş topluluğu da sürü olarak temsil edilir. Parçacık hareket ettiğinde, kendi koordinatlarının uygunluk değeri yani yiyeceğe ne kadar uzaklıkta olduğu hesaplanır. Bir parçacık, koordinatlarını, hızını yani çözüm uzayındaki her boyutta ne kadar hızla ilerlediği bilgisini, şimdiye kadar elde ettiği en iyi uygunluk değerini ve bu değeri elde ettiği koordinatları hatırlamalıdır. Çözüm uzayında her boyuttaki hızının ve yönünün her seferinde nasıl değişeceği, komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının birleşiminden elde edilecektir (Alataş, 2007).

3.5. Yapay Balık Sürüsü Algoritması (Artificial Fish-Swarm Algorithm)

Yapay balık sürüsü optimizasyonu, yiyecek aramada balık sürüsünün sosyal davranışlarının benzetimi tabanlı bir zeki optimizasyon algoritmasıdır. Doğada balık, yiyeceğini besin değeri yüksek alanları tek tek arayarak ya da diğer balıkları izleyerek bulabilmektedir. Çok balıklı bölgenin besin değeri genellikle daha yüksek olmaktadır. Bu optimizasyonun temel fikri, küresel optimuma ulaşmak için balık bireyinin yerel aramasıyla toplanma ve izleme gibi balık davranışlarını taklit etmektir. Bir yapay balığın yaşadığı çevre başlıca çözüm uzayıdır ve bu çevre diğer yapay balıkların da konumudur. Bir sonraki davranışı mevcut durumuna ve yerel çevresel durumuna bağlı olmaktadır. Bir

yapay balık kendi faaliyetleri ve arkadaşlarının faaliyetleri yolu ile çevresini etkileyecektir (Jiang vd., 2009).

3.6. Bakteriyel Besin Arama Optimizasyon Algoritması (Bacterial Foraging Optimization Algorithm)

Bakteriyel Besin Arama Algoritması, E. coli bakterisinin beslenme davranışından esinlenerek karmaşık mühendislik problemlerini çözmek için geliştirilmiş bir hesaplama tekniğidir. Bakteriler, karmaşık yaşam formlarındaki diğer canlılara göre çok daha basit yapıdadırlar. Sınırlı algı ve hareket kabiliyetlerini kullanarak optimum düzeyde enerji harcayıp beslenme faaliyetlerini gerçekleştirmeleri gerekmektedir. Diğer yaşam formlarına nazaran modellenebilmeleri daha kolaydır. Bu türden canlılardan biri olan E. coli bakterisi, yapısı ve çalışma sekli en iyi anlaşılan mikroorganizmalardan birisidir. E. coli bakterisi besin maddesine ulaştığında diğer bakterileri uyarıcı etkiye sahip kimyasal bir madde salgılamaktadır. Bu madde, diğer E. coli bakterilerinin besini bulan bakterinin bulunduğu yere doğru hareket etmesini sağlamaktadır. Eğer gıda yoğunluğu çok fazla ise bakteriler kenetlenerek grup halinde hareket edebilmektedirler (Başbuğ, 2008).

3.7. Kurt Kolonisi Algoritması (The Wolf Colony Algorithm)

Bu algoritma, kurt kolonisinin sıkı bir organize sisteme sahip olmasından esinlenilerek geliştirilmiştir. Kurtlar görevleri diğerleriyle bölüşmektedirler ve avlandıkları zaman tutarlı adımlar atmaktadırlar. Az miktarda yapay kurt aktif olduğu av aralığında aramaya atanmaktadır. Arama kurtları avı keşfettiği zaman, avın konumunu diğer kurtlara ulumayla bildirmektedir. Diğer yapay kurtlar ava yaklaşmakta ve avı kuşatmaktadırlar. Kurt kolonisinin atanma kuralı, yiyeceğin ilk olarak güçlü kurda atanması ve daha sonra zayıf olana atanmasıdır. Kurt koloni algoritması bu davranışların taklit edilmesiyle geliştirilmiştir (Liu vd., 2011).

4. KEDİ SÜRÜSÜ OPTİMİZASYONU

KSO, kedigillerin hareketlerinin incelenmesiyle ortaya çıkarılmıştır. Kedilerin davranışlarına benzetim yapılarak iki alt model oluşturulmuştur. Bu optimizasyondaki matematiksel modeller kedilerin hareketlerinin çözümlenmesiyle meydana gelmiştir (Chu vd., 2006). KSO kullanılarak yapılan bazı çalışmalar şunlardır: Santaso ve arkadaşları kümeleme problemi için KSO'yu kullanmışlardır (Santosa ve Ningrum, 2009). Wang ve arkadaşları en az önemli bitin yerine en iyisini getirmek için KSO stratejisi kullanmışlardır (Wang vd., 2010). Hwang ve arkadaşları müşterilere göre en uygun sözleşme kapasitesi problemini çözmek için KSO ve PSO'yu birlikte kullanmışlardır (Hwang vd., 2009). Destek vektör makinesi için özellik seçimi ve parametre optimizasyonu probleminde KSO, Lin ve Chen tarafından önerilmiştir (Lin ve Chien, 2009). Kalaiselvan ve arkadaşları filigran performansının arttırılması amacıyla KSO uygulamışlardır (Kalaiselvan vd., 2011). Wang ve Wu e-öğrenmede duygu tanıma problemi için KSO'yu destek vektör makinesiyle birlikte kullanmışlardır (Wang ve Wu, 2011). KSO algoritmasının paralel versiyonu da Tsai ve arkadaşları tarafından önerilmiştir (Tsai vd., 2008).

4.1. Kedigillerin Hareketlerini İnceleme

Biyolojik sınıflandırmaya göre, yaklaşık 30 tane farklı örneğin aslan, leopar, kaplan, kedi vb. kedi cinsinden yaratık uzayı vardır. Çoğunun farklı yaşam alanı olmasına rağmen, kedigiller benzer davranış modellerini sergilemektedirler. Kedilerin avlanma becerisi, kediler için kalıtsal değildir, alıştırmalar aracılığıyla kazanılmaktadır. Bu avlanma becerisi ile, yaban kedileri yiyeceklerini temin etmeyi sağlamaktadırlar ve türlerinin hayatta kalması garanti altına alınmaktadır. Ayrıca evcil kedilerde benzer doğal avlanma becerisi ve hareketli nesnelere güçlü bir merak sergilemektedirler. Bütün kedilerin, bu güçlü merakı paylaşmasına rağmen, zamanlarının çoğunu hareketsiz (durağan) geçirmektedirler. Kediler çok yüksek seviyede atikliğe sahiptirler. Bu atiklik dinlenme zamanlarında bile kendilerini bırakmamakta büyük geniş gözler sürekli olarak etrafını gözetlemektedir. Kediler, çok zeki ve bilinçli (planlı) yaratıklar oldukları halde tembel gibi görünmektedirler. Kedilerin dokuz canlı olduğu söylenerek, kedilerin güçlü canlılığına gönderme yapılmaktadır. Ev içinde olan kedi sık sık alçak frekansta ses çıkarmaktadır. Kediler hoşnut oldukları, tehlikede veya hasta oldukları zaman mırlarlar. Mırlamanın alçak frekansının, hücre onarımına

yardım ettiğine inanılmaktadır. Bu kedilerin canlılığı için bir neden olabilir (Chu ve Tsai, 2007).

4.2. Önerilen Algoritmalar

Önerilen KSO için kedilerin başlıca iki tane davranışsal özelliği modellenmiştir. Bunlar "arama modu" ve "izleme modu" olarak isimlendirilmiştir. Bu iki modun birleşimi KSO'ya daha iyi bir performans için izin vermektedir (Chu ve Tsai, 2007).

4.2.1. Çözüm Kümesinin Sunumu

KSO'da önerilen algoritmada, optimizasyon problemini çözmek için kedileri ve kedilerin davranışlarının modeli kullanılmaktadır, örneğin çözüm kümesini tasvir etmek için kediler kullanılmaktadır.

KSO'da ilk önce iterasyonda kaç kedinin kullanılacağına karar verilmektedir, daha sonra problemi çözmek için kediler KSO'nun içine uygulanmaktadır. Bütün kediler *M* boyuttan oluşan kendi pozisyonlarına, her bir boyut için hızlara, kedinin uyumunu uygunluk fonksiyonuna yansıtan bir uygunluk değerine ve kedinin izleme modunda mı yoksa arama modunda mı olduğunu belirlemek için bir bayrağa (flag'e) sahiptir. Final çözüm kedilerden bir tanesinin en iyi pozisyonu olacaktır. KSO en iyi çözümü iterasyon sonuna ulaşıncaya kadar saklayacaktır (Chu ve Tsai, 2007).

4.2.2. Dinlenme ve Tetikte Olma - Arama Modu

Bu alt mod dinlenmede fakat tetikte olmanın bir periyodu boyunca kedinin modellenmesi için kullanılmaktadır (sonraki hareketi için çevresine bakınma). Arama modu aşağıda belirtildiği gibi dört gerekli faktöre sahiptir: arama hafızası havuzu (*AHH*), seçilen boyutun arama aralığı (*SBA*), değişen boyutların sayısı (*DBS*) ve kendi pozisyonunu değerlendirme (*KPD*). *AHH* her bir kedinin arama hafızasının boyutunu tanımlamak için kullanılır, bazı noktaları kediye göre sıralayarak belirtir. Daha sonra anlatılacak kurallara göre kedi, hafıza havuzundan bir nokta ayıracaktır. *SBA* seçilen boyutlar için mutasyon oranını bildirir. Arama modu süresince, eğer bir boyut mutasyon için seçilmişse, yeni ve eski değerler arasındaki farklılık aralık dışında olmamalıdır, aralık

SBA tarafından tanımlanır. *DBS* boyutlardan kaç tanesinin değişime uğrayacağını ifade eder. Bütün bu faktörler arama modunda önemli rol oynar. *KPD* bilinen bir ikili değerdir, ve kedilerin bulunduğu noktanın hareket için aday noktalardan biri olup olmayacağını bildirir. *KPD*, *AHH* değerini etkilemez (Chu ve Tsai, 2007).

Arama modunun adımları:

Adım 1: *j=AHH* olduğunda $kedi_k$'nin bulunduğu pozisyonda *j* tane kopya yap. Eğer *KPD* değeri doğruysa, *j=(AHH-1)* olur, sonra mevcut pozisyonu, adaylardan biri olarak tut.

Adım 2: *DBS*'ye göre, her bir kopya için mevcut değerin *SBA* yüzdesini gelişigüzel olarak artır veya azalt ve eskisiyle yerini değiştir.

Adım 3: Bütün aday noktaların uygunluk değerini hesapla.

Adım 4: Eğer bütün uygunluk değerleri (*UD*) tam olarak aynı değilse, eşitliğe göre her bir aday noktanın seçilen olasılığını Eşitlik (4.1)'e göre hesapla, aksi takdirde her bir aday noktanın seçilen olasılıklarının tümüne 1 ata.

Adım 5: Aday noktalardan taşınmak için noktayı gelişigüzel çıkart (ayır), ve $kedi_k$'nin pozisyonuyla değiştir.

$$P_i = \frac{|UD_i - UD_b|}{|UD_{maks} - UD_{min}|}, \; 0 < i < j \; olduğunda \tag{4.1}$$

Eğer uygunluk fonksiyonunun amacı minimum çözümü bulmaksa, $UD_b=UD_{maks}$, aksi takdirde $UD_b=UD_{min}$ (Chu ve Tsai, 2007; Santosa ve Ningrum, 2009; Wang vd., 2010).

4.2.3. Hareket- İzleme Modu

İzleme modu, kedinin hedefi izlemedeki durumunu modellemek için bir alt modeldir. Bir kere kedi izleme moduna gittiğinde, her bir boyutu için kendi hızlarına göre hareket etmektedir. İzleme modundaki çalışma aşağıdaki gibi açıklanabilir (Chu ve Tsai, 2007).

Adım 1: Bütün boyutlar için hızları ($v_{k,d}$) Eşitlik (4.2)'yi kullanarak güncelle.

Adım 2: Hızların, maksimum hızın aralığında olduğunu kontrol et. Yeni hız aralığının üzerinde olduğu takdirde limite eşitliği at (*limit=sınır*).

Adım 3: $kedi_k$'nin pozisyonunu Eşitlik (4.3)'ü kullanarak güncelle.

$$v_{k,d} = v_{k,d} + r_1 * c_1\left(x_{best,d} - x_{k,d}\right), \qquad d = 1,2,\dots,M \tag{4.2}$$

$x_{best,d}$, en iyi uygunluk değerine sahip kedinin pozisyonu; $x_{k,d}$, $kedi_k$'nin pozisyonu, c_1 bir sabit ve r_1 gelişigüzel bir değerdir (Chu ve Tsai, 2007; Santosa ve Ningrum, 2009; Wang vd., 2010).

$$x_{k,d} = x_{k,d} + v_{k,d} \tag{4.3}$$

4.3. Kedi Sürüsü Optimizasyonunun Temel Tanımlaması

KSO'nun, arama modu ve izleme modu adında iki alt modu vardır. Bu iki modu algoritma şeklinde birleştirmek için, arama moduyla izleme modunu birleştirmeyi gerektiren bir karışım oranı (*KO*) tanımlanmaktadır. Kediler dinlenme zamanında hareket etmeye karar verdiyseler, hareket çok dikkatli ve yavaşça yapılmaktadır. Bu hareket, arama moduna yansıtılmaktadır. İzleme modu kedi tarafından bir hedefin takip edilmesini modellemektedir. Kediler, enerji kaynaklarını fazla kullanmalarına yol açan objeleri takip etmeye çok az zaman harcamaktadırlar. Kedilerin zamanlarının çoğunu dinlenmeye ve gözetlemeye (mesela zamanlarının çoğu arama modunda geçmektedir) harcadığını garantilemek için *KO'*ya çok küçük bir değer atanmaktadır. KSO'nun süreci aşağıda anlatılmaktadır (Chu ve Tsai, 2007; Santosa ve Ningrum, 2009; Wang vd., 2010).

Adım 1: Süreçte *N* tane kedi yarat.

Adım 2: *M* boyutlu çözüm uzayına gelişigüzel kediler serpiştir ve her kedinin hızına maksimum hız aralığında olan gelişigüzel değerler ver. Sonra kedilerin numaralarını gelişigüzel ayır ve onları *KO*'ya göre izleme modunun içine ata ve diğerlerini arama modunun içine ata.

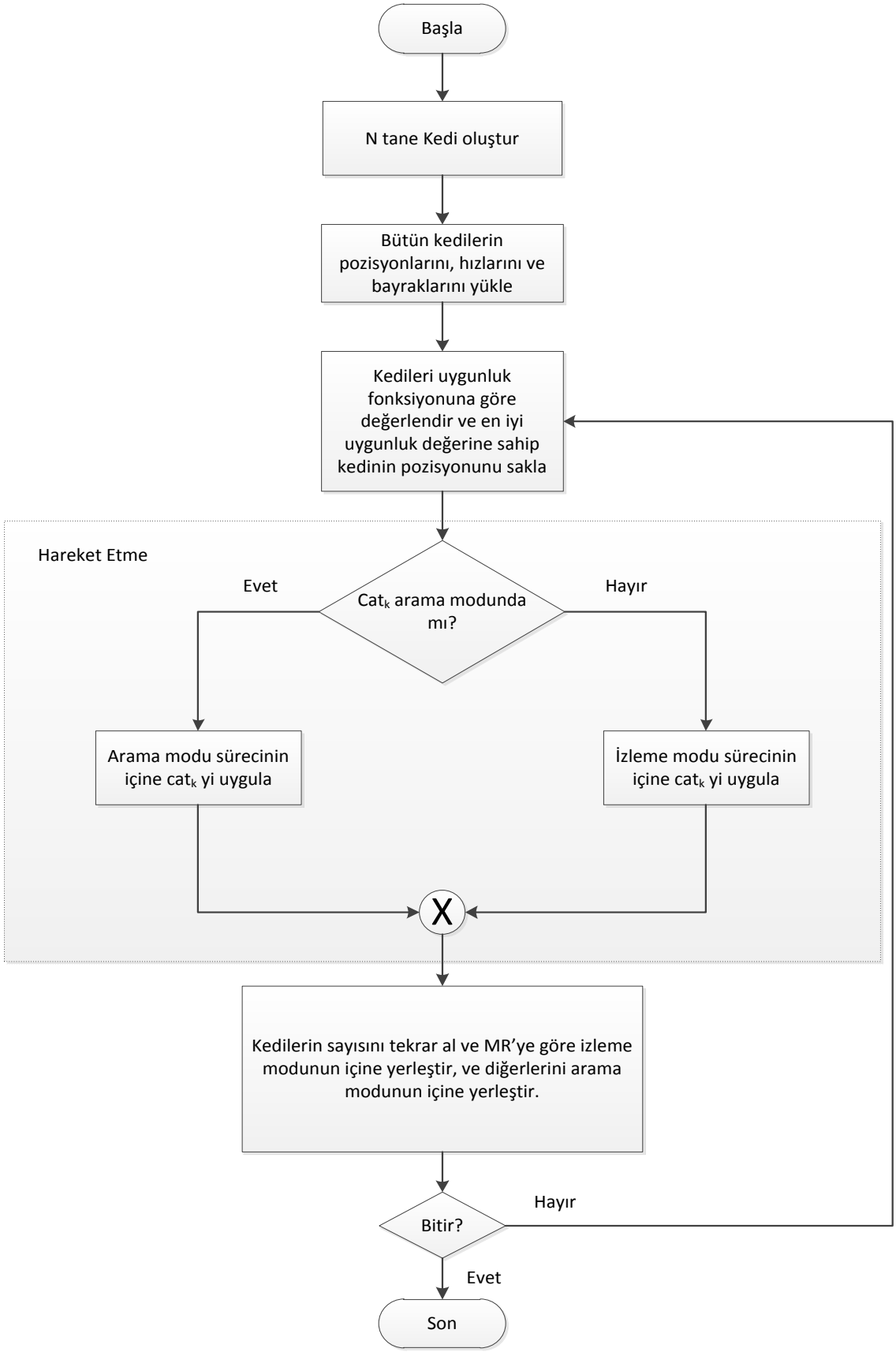
Adım 3: Amaç kriterini yansıtan uygunluk fonksiyonunu kedilerin pozisyona göre belirle ve en iyi kediyi hafızada sakla. Şimdiye kadarki en iyi çözümü yansıtmasından dolayı sadece en iyi kedinin pozisyonu saklanır.

Adım 4: Bayraklarına (flag'lerine) göre kedileri hareket ettir, eğer $kedi_k$ arama modundaysa, kediyi arama modu sürecine uygula, aksi takdirde izleme modu sürecine uygula.

Adım 5: Kedilerin numarasını yeniden ayır ve *KO*'ya göre izleme modunun içine ata, sonra diğer kedileri arama modunun içine ata.

Adım 6: Sonlandırma (bitirme) koşullarını kontrol et, eğer doyuma ulaşmışsa programı sonlandır, aksi durumda Adım 3'ten Adım 5'e kadar tekrar et.

KSO sürecinin diyagramı Şekil 4.1'de sunulmaktadır.



Şekil 4.1. KSO'nun süreç diyagramı

5. YAPAY ARI KOLONİ ALGORİTMASI (YAKA)

Doğal bir arı kolonisinde arılar arasında yapılacak işlere göre bir görev paylaşımı vardır. Arılar bu iş paylaşımını merkezi bir birim olmadan, kendi kendilerine gerçekleştirmektedirler. Bu iş paylaşımı ve kendi kendine organize olabilme sürü zekâsının iki önemli özelliğidir. Tereshko'nun reaktif difüzyon denklemine dayalı olarak ortaklaşa zekânın ortaya çıkmasını sağlayan minimal yiyecek arama modelinde üç temel bileşen vardır. Bunlar yiyecek kaynakları, görevli işçi arılar ve görevsiz işçi arılardır. Ayrıca bu minimal model bir yiyecek kaynağına yönelme ve yiyecek kaynağını bırakma olmak üzere iki modda çalışmaktadır (Tereshko, 2000).

Arılar bal, polen veya nektar bulmak için yiyecek kaynaklarına gitmektedirler. Yiyecek kaynağının değeri, yuvaya yakınlığı, çeşidi, nektar yoğunluğu, nektarın çıkarılmasının kolaylığı gibi birçok etkene bağlıdır. Ayrıca basit olması açısından sadece yiyecek kaynağının zenginliği gibi tek bir özellik de ele alınabilir. Görevli işçi arılar, nektarın, önceden keşfedilmiş olan belli kaynaklardan kovana getirilmesinden sorumludurlar ve gittikleri kaynağın kalitesi ve yeriyle ilgili bilgileri kovandaki diğer arılarla paylaşmaktadırlar. Görevsiz işçi arılar ise nektarı toplanabilecek yeni yiyecek kaynaklarını aramaktadırlar. İçsel bir dürtüye veya dış bir etmene bağlı olarak gelişigüzel kaynak arayışında olan kaşif arılar ve kovanda bekleyip görevli arıları izleyerek bu arılar tarafından paylaşılan bilgiyi kullanıp yeni bir kaynağa yönelen gözcü arılar olmak üzere görevi belirsiz iki tür arı vardır. Kaşif arıların sayısının kovandaki diğer arılara oranı %5-10 arasındadır (Karaboğa, 2011).

Ortaklaşa bilginin oluşumundaki en önemli etmen arılar arasındaki bilgi paylaşımıdır. Yiyecek kaynağının yeri ve kalitesi hakkındaki bilgi paylaşımı kovandaki dans alanında olmaktadır. Dans eden arıya diğer arılar antenleri aracılığıyla dokunarak kaynağın tadı ve kokusu hakkında da bilgi alırlar. Arıların yiyecek arama davranışı modellenerek geliştirilen en güncel algoritma YAKA'dır. Bu algoritmada temel alınan modelde basit olması nedeniyle bazı kabuller yapılmaktadır. Buna göre: (kaynak sayısı = görevli arı sayısı = işçi arıların sayısı) olarak belirlenmektedir ve nektarı tükenmiş kaynağın görevli arısı kaşif arıya dönüşmektedir. Yiyecek kaynaklarındaki nektar miktarı, kaynaklarla ilgili çözümlerin uygunluğuna ve bu kaynakların yerleri ise optimizasyon problemine ait olası çözümlere karşılık gelmektedir. YAKA en fazla nektara sahip yiyecek kaynağının yerini

bulmaya çalışarak uzaydaki çözümlerden problemin minimumunu veya maksimumunu veren çözümü bulmaya çalışmaktadır (Karaboğa, 2011).

YAKA kullanılarak yapılan bazı çalışmalar şunlardır: Hedayatzadeh, Hasanizadeh, Akbari ve Ziarati çok amaçlı problemlerin optimizasyonunda YAKA'yı kullanmışlardır (Hedayatzadeh vd., 2010). Banharnsakun, Achalakul ve Sirinaovakul, büyük boyutlu problemler için YAKA'nın dağıtık bir sürümünü önermişlerdir (Banharnsakun vd., 2010). Alam, Kabir ve Islam sürekli fonksiyon optimizasyonu için YAKA'yı kullanmışlardır (Alam vd., 2010). Karaboğa ve Görkemli gezgin satıcı probleminde YAKA'yı uygulamışlardır (Karaboğa ve Görkemli, 2011). Guo, Cheng ve Liang sayısal optimizasyon algoritmasında YAKA'yı kullanmışlardır (Guo vd., 2011). Ning ve Zhang bulanık sinir ağına dayalı bir konuşma tanıma sisteminde YAKA'yı uygulamışlardır (Ning ve Zhang, 2011). Jadhav, Patel, Sharma ve Roy YAKA'yı rüzgar enerjisi geçişimli kombine emisyon görev dağıtımı problemi için kullanmışlardır (Jadhav vd., 2011). Çelik, Karaboğa ve Köylü kural tabanlı sınıflandırma modelinin optimizasyonu için YAKA'yı önermişlerdir (Çelik vd., 2011).

5.1. Önerilen Algoritma

YAKA'da arama uzayı, yiyecek kaynaklarını içeren kovan çevresidir. Algoritma ilk olarak arama uzayındaki çözümlere karşılık gelen gelişigüzel yiyecek kaynakları yerleri üretmektedir ve bu yerler her bir parametrenin alt ve üst sınırları arasında gelişigüzel değer üretilerek oluşturulmaktadır. Aynı zamanda oluşturulan her bir kaynağın geliştirilememe sayaçları da sıfırlanmaktadır. Bu aşamadan sonra durdurma kriteri sağlanıncaya kadar yiyecek kaynakları görevli arı, kaşif arı ve gözcü arı süreçlerinden geçirilerek daha iyisi bulunmaya çalışılmaktadır. Her bir yiyecek kaynağının bir görevli arısı olmaktadır, bu şekilde yiyecek kaynağı sayısı ile görevli arı sayısı eşitlenmektedir. Yiyecek kaynağının komşuluğunda yeni bir yiyecek kaynağı belirlenmektedir. Eğer bu kaynak daha iyi ise işçi arı bu kaynağı hafızasına almaktadır ve geliştirilememe sayacı sıfırlanmaktadır. Aksi durumda ise geliştirilememe sayacı bir arttırılmaktadır (Karaboğa, 2011; Banharnsakun vd., 2010).

Gözcü arılar danslardan öğrendikleri bilgilerle nektar miktarlarına göre bir kaynak seçmektedir. Burada nektar miktarı uygunluk değerine karşılık gelmektedir ve yiyecek kaynağının bir gözcü arı tarafından seçilme olasılığı rulet tekerleği yöntemi kullanılarak

hesaplanmaktadır. Bir kaynağın uygunluk değeri arttıkça, bu kaynak bölgesini seçecek gözcü arı sayısı da artmaktadır. Gözcü arılar daha iyi kaynağı bulabilmek için bir seçim yapmaktadırlar. Eski kaynak ile yeni kaynak arasında karşılaştırma yapmaktadırlar. Eğer eski kaynak daha iyi ise bu çözüm saklanmaktadır ve geliştirilememe sayacı bir arttırılmaktadır. Aksi halde yeni çözüm saklanmaktadır ve geliştirilememe sayacı sıfırlanmaktadır. Bütün gözcü arılar yiyecek kaynaklarına dağılıncaya kadar bu işlemler devam etmektedir (Karaboğa, 2011; Ning ve Zhang, 2011).

Geliştirilememe sayacı kontrol edilerek, bir kaynağın tükenip tükenmediğine bakılmaktadır. Eğer yiyecek kaynağı tükenmişse kaynağın görevli arısının bu kaynağı bırakıp yeni kaynak araması gerekmektedir. Bu şekilde görevli arı kaşif arı olmaktadır ve bu arı için gelişigüzel çözüm arama süreci başlamaktadır. Temel YAKA'nın adımları aşağıdaki gibidir

Adım 1: Eşitlik (5.1) ile x_{ij}, $i=1,\ldots,N$, $j=1..M$, çözümlerine başlangıç değerlerini ata ve geliştirilememe sayaçlarını ($hata_i$) sıfırla. Uygunluk değerlerini hesapla. N yiyecek kaynağı sayısı ve M optimize edilecek parametre sayısıdır.

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{maks} - x_j^{min}\right) \tag{5.1}$$

Adım 2: $i=1$ den N'ye kadar Eşitlik (5.2)'yi kullanarak x_i çözümünün görevli arısı için yeni bir kaynak üret.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{5.2}$$

Bu işlemde üretilen v_{ij} değerinin daha önceden belirtilmiş olan alt ve üst sınırları aşması durumunda Eşitlik (5.3)'ü kullanarak j. parametreye ait alt veya üst sınır değerlerine ötele.

$$v_{ij} = \begin{cases} x_j^{min} , & v_{ij} < x_j^{min} \\ v_{ij} , & x_j^{min} \le v_{ij} \le x_j^{maks} \\ x_j^{maks}, & v_{ij} > x_j^{maks} \end{cases} \tag{5.3}$$

Bu kaynağın maliyet değerini $f(v_i)$ Eşitlik (5.4)'te yerine koyarak bu çözümün uygunluk değerini hesapla. v_i ve x_i arasında seçim işlemini uygula ve daha iyi olanı seç. x_i çözümü gelişememiş ise $hata_i$ bir arttır.

$$uygunluk_i = \begin{cases} 1/(1+f_i) , & f_i \ge 0 \\ 1 + abs(f_i) , & f_i < 0 \end{cases} \tag{5.4}$$

$f_i, \overrightarrow{v_1}$ kaynağının maliyet değeridir.

Adım 3 : Gözcü arıların seçim işlemi yaparken kullanacakları uygunluk değerine dayalı olasılık değerlerini Eşitlik (5.5)'i kullanarak hesapla.

$$P_i = \frac{uygunluk_i}{\sum_{j=1}^{N} uygunluk_j} \quad (5.5)$$

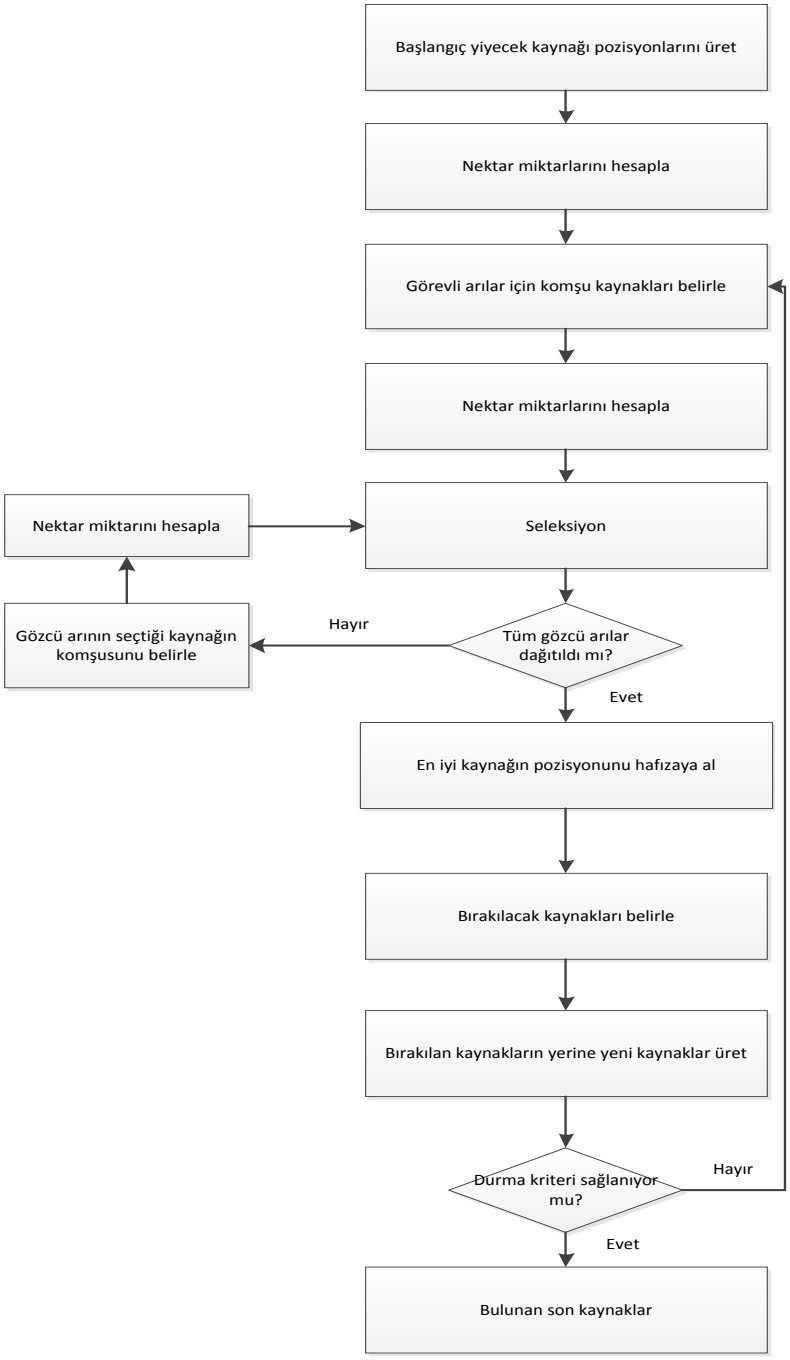
Adım 4 : Rulet tekerleğine göre seçim işleminde her bir kaynak için [0, 1] aralığında üretilen p_i değeri gelişigüzel üretilen bir değerden büyükse gözcü arı için Eşitlik (5.2)'yi kullanarak yeni bir kaynak üret ve üretilen v_i ile x_i arasında seçim işlemi uygula, daha iyi olanı seç. x_i çözümü gelişememişse $hata_i = hata_i + 1$, gelişmişse $hata_i = 0$ yap. Bu adımı tüm gözcü arılar yiyecek kaynağı bölgelerine dağılıncaya kadar tekrar et.

Adım 5 : Kaynağın nektarının tükenip tükenmediğini kontrol et. Eğer tükenmişse Eşitlik (5.1)'i kullanarak gelişigüzel üretilen yeni bir çözümle değiştir.

Adım 6 : En iyi çözümü hafızada tut.

Adım 7 : Sonlandırma koşullarını kontrol et. Eğer koşullar sağlanmıyorsa Adım 2'den Adım 6'ya kadar tekrar et.

YAKA'nın temel adımları Şekil 5.1'de sunulmaktadır.



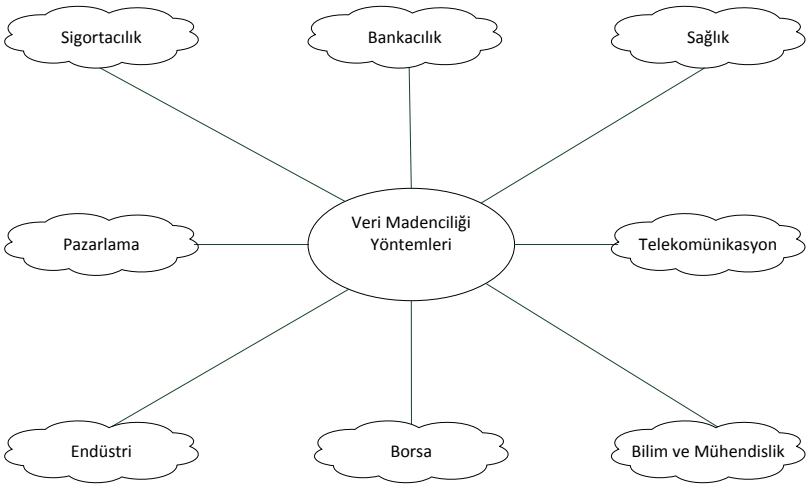
Şekil 5.1. YAKA'nın akış diyagramı

6. VERİ MADENCİLİĞİ

Bilişim alanındaki gelişmelerle bütün veriler sayısal ortama kaydedilmeye başlanmıştır. Verilerin uzun süre depolanması büyük kapasiteli veritabanlarının oluşmasına neden olmuştur. Bu nedenle büyük veritabanlarındaki verileri yönetmek için veri ambarı ve bu verilerden istenilen yararlı, anlamlı ve ilginç bilgiye ulaşılmasını sağlayan veri madenciliği kavramları ortaya çıkmıştır (Özkan, 2008).

Büyük ölçekli veriler arasından daha önce keşfedilmemiş faydalı ve anlaşılır bilgilerin çıkarılması işlemine veri madenciliği, diğer bir deyimle de veritabanlarından bilgi keşfi denilmektedir. Veri madenciliğinin amacı depolanan verilerin belirli yöntemler kullanılarak ilgili kurumlar için var olan veya gelecekte ortaya çıkabilecek gizli bilgileri açığa çıkarmaktır.

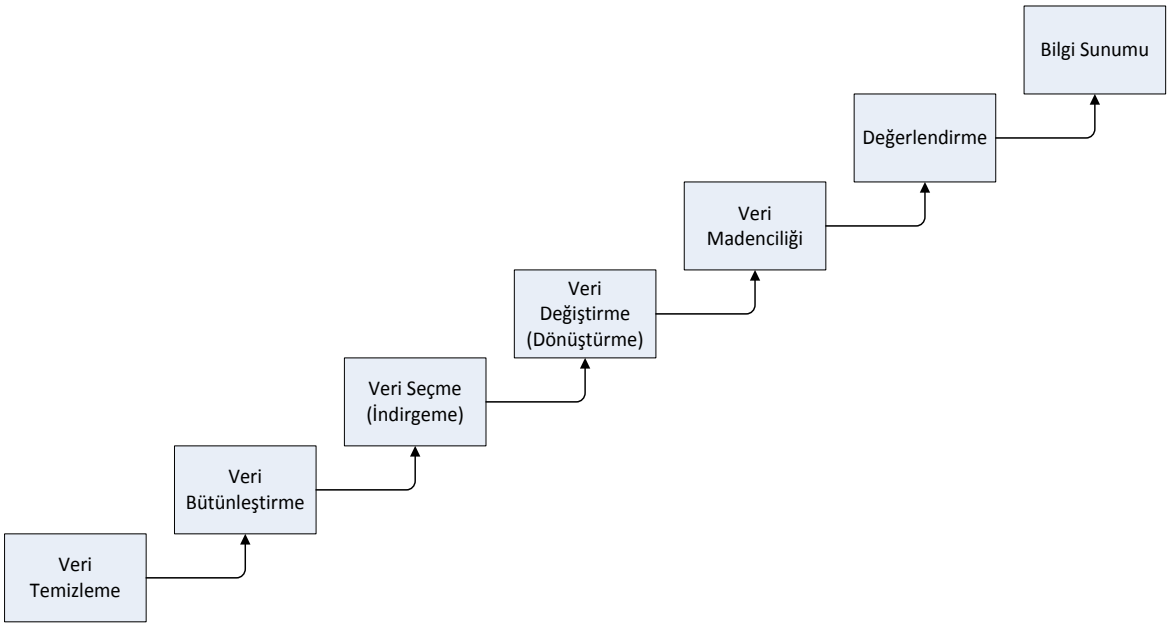
Veri madenciliği günümüzde, Şekil 6.1'de de gösterildiği gibi sigortacılık, bankacılık, sağlık, pazarlama, telekomünikasyon, endüstri, borsa, bilim ve mühendislik gibi pek çok alanda yaygın olarak kullanılmaktadır. Örneğin; bankacılıkta kredi kartı harcamalarına göre müşteri gruplarının değerlendirilmesinde, sigortacılıkta riskli müşteri gruplarının belirlenmesinde, sağlıkta tıbbi teşhis ve uygun tedavi sürecinin belirlenmesinde, pazarlamada mevcut müşterilerin elde tutulması ve yeni müşterilerin kazanılmasında, telekomünikasyonda hatların yoğunluk tahminlerinde, endüstride kalite kontrolünde, borsada hisse senetlerinin fiyat tahmininde, veri madenciliği yöntemleri kullanılmaktadır (Özkan, 2008; Tekerek, 2011).



Şekil 6.1. Veri madenciliğinin kullanıldığı alanlar

Veri madenciliği, verilerden doğru, yeni, faydalı ve anlaşılır bilgiler elde etmek için kullanılan özel bir süreçtir. Bu süreçler Şekil 6.2'de de gösterildiği gibi; veri temizleme,

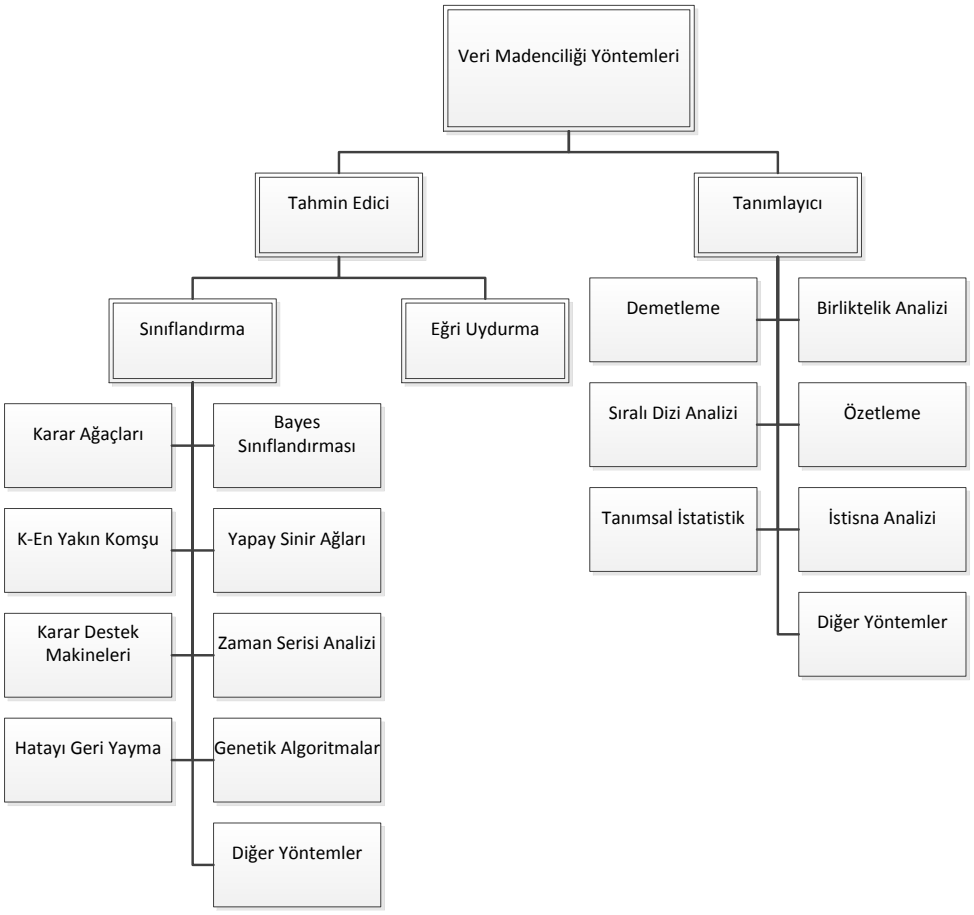
veri bütünleştirme, veri seçme (indirgeme), veri değiştirme (dönüştürme), veri madenciliği, değerlendirme ve bilgi sunumudur.



Şekil 6.2. Veri madenciliğinin süreçleri

Veri temizleme süreci, gürültü olarak adlandırılan, uygun olmayan ve hatalı girilmiş verilerin ayıklanması işlemidir. Eksik veriler yerine, uygun yeni değerler belirlenerek konulmaktadır. Veri bütünleştirme sürecinde, farklı veri tabanları veya farklı veri kaynaklarından alınan verilerin birlikte değerlendirmeye alınabilmesi için tek türe dönüştürülmesi veya bütünleştirmesi yapılmaktadır. Veri indirgeme, veri madenciliği uygulamalarında çözümleme işleminin daha kısa sürmesi için, sonucun değişmeme koşuluyla veri sayısının ya da değişkenlerin sayısının azaltıldığı süreçtir. Veri dönüştürme sürecinde ise, bazen uygulamaya kaynak veriyi aynen katmak uygun olmayabilir, bu yüzden kaynak veri farklı formatlara veya değerlere dönüştürülmektedir. Veri madenciliği süreci, veri örüntülerini yakalayabilmek için sınıflandırma, kümeleme, birliktelik kuralları, vb. yöntemlerinden birinin kullanılması işlemidir. Değerlendirme süreci, keşfedilen bilginin geçerlilik, yenilik, yararlılık ve basitlik kıstaslarına göre değerlendirilmesidir. Bilgi sunumu süreci ise elde edilmiş bilginin kullanıcıya sunulmasıdır (Özkan, 2008; Tekerek, 2011).

Veri madenciliği konusu için çok sayıda yöntem ve algoritma geliştirilmiştir. Bu yöntemler Şekil 6.3'te gösterilmektedir. Geliştirilen yöntemlerin çoğu istatistiksel tabanlıdır. Bu tez çalışmasında sınıflandırma kural madenciliği yöntemi kullanılmıştır.



Şekil 6.3. Veri madenciliği yöntemleri

6.1. Sınıflandırma Kural Madenciliği

Sınıflandırma kural madenciliği en çok kullanılan veri madenciliği yöntemlerinden biridir. Sınıflandırma yöntemiyle veri kümelerinden kullanıcıların rahatça anlayabileceği kurallar çıkarılmaktadır. Daha sonra bu kurallar yardımıyla yeni gelen örnek önceden belirlenmiş bir sınıfa atanmaktadır. Tahmini doğruluk ve anlaşılabilirlik iki önemli değerlendirme ölçütüdür. Tahmini doğruluk, oluşturulan modelin daha önceden görülmemiş örnekleri sınıflandırmadaki performansını, anlaşılabilirlik ise kullanıcılar

tarafından anlaşılabilirliğini ölçmektedir. Sınıflandırma kural madenciliği için çeşitli yöntem ve algoritmalar bulunmaktadır (Alataş ve Akın, 2004).

Karar ağaçları, C4.5, ID3 ve CART gibi yöntemlerle oluşturulan, akış şemalarına benzeyen ve "böl-ve-yönet" stratejisini kullanan bir veri sınıflandırma modelidir. Her bir özellik düğüm tarafından temsil edilmektedir. En üst yapı "kök", en alt yapı "yaprak" ve bu ikisinin arasındaki yapılarda "dal" olarak adlandırılmaktadır. Karar verme işlemine açık bir şekilde hakim olması ve çok fazla sayıda işlem yapmaya gerek duymadan sınıflandırma işlemini yapması karar ağaçlarının avantajıdır. Ancak yanlış veya tutarsız örneklere aşırı uymaya neden olabileceğinden, yapılandırma öncesinde analize özen gösterilmelidir. Sürekli değişken yönetimi zorludur ve değişken önceliğinin belirlenmesini gerektirmektedir (Özkan, 2008; Alataş ve Akın, 2004).

Karar listeleri, eğitim verisinden çıkarılan bilginin açık bir şekilde gösterilmesi yönünden karar ağaçlarına benzemektedir. Karar ağaçlarından farklı olarak "ayır-ve-yönet" stratejisini kullanmaktadırlar. Eğitim verisinin bir alt kümesini kapsaması için bir kural oluşturulmaktadır, sonra kalan örnekleri tekrarlı olarak kapsamak için daha fazla kural üretilmektedir. Karar listelerinin sonunda "Eğer-O-Zaman" kuralları bulunur ve yeni bir örnek sınıflandırılırken sırayla bu kurallara uygulanmaktadır. Örnek sınıflandırılıncaya kadar, ilk kuraldan başlayarak bütün kurallarda denenmektedir. Eğer örnek sınıflandırılmamışsa listenin en altındaki varsayılan bir kural işletilmektedir. Dezavantajı bireysel kuralların anlaşılmasının zor olabilmesidir, listedeki bir kural, önceki tüm kurallar göz önünde bulunarak ele alınmalıdır. Ayrıca gürültülü eğitim verisine uyabileceğinden kural budama işlemi uygulanmaktadır.

Evrimsel hesaplama, sınıflandırma kural madenciliğinde önemli yöntemlerden biridir. Genetik algoritma ve genetik programlama en bilinenleridir. Bu yöntemde genel bir arama yapılır ve diğer kaba seçim algoritmalarına göre nitelik etkileşimiyle daha iyi baş edebilmektedir. Çok değişik tiplerdeki verileri işlenebilmektedir ve sonuçları açıklanabilirdir. Bu yaklaşımın dezavantajı, en iyi sonucun üretildiği garanti edilememektedir ve işlem yükü ağır olabilmektedir (Alataş ve Akın, 2004).

Yapay bağışıklık sistemlerinden klonal seçim algoritması, sürü zekâsı tekniklerinden karınca koloni optimizasyon optimizasyon algoritması temelli algoritmalar, parçacık sürü optimizasyonu algoritması gibi sezgisel algoritmalarda sınıflandırma kural madenciliğinde kullanılmaktadır. Klasik algoritmaların çalışmadığı farklı veri tiplerinin olduğu veritabanlarında, bu tür sezgisel algoritmalar iyi sonuçlar verebilmektedir.

Bunlara ek olarak sınıflandırmada örnek-tabanlı öğrenme, yapay sinir ağları, lojistik gerileme ve Bayesian ağları yaklaşımları da bulunmaktadır. Genel olarak dezavantajları, tahmini doğruluklarının bazı durumlarda iyi olmasına karşın, açıklayıcı güçleri eksiktir. Bu metotlara bazen bulanık mantık eklenerek bulanık kurallar bulunmaktadır (Alataş ve Akın, 2004).

Kural budama, karar listelerinde, gürültülü eğitim verisinden kaçınmak için kullanılan bir işlemdir. Bu işlem için iki temel strateji vardır. İlki, komple bir kural kümesi oluşturulmaktadır, sonra ise nitelikleri kurallardan eleyerek ya da bireysel kurallar silinerek kural kümesi basitleştirilmektedir. Bu, önceden tanımlı bazı budama kriterlerine bağlı olarak optimize edilmesiyle yapılmaktadır. Diğer strateji ise artımsal budamadır. Her kural, algoritmayla oluşturulduktan hemen sonra basitleştirilmektedir (Alataş ve Akın, 2004).

7. KEDİ SÜRÜSÜ OPTİMİZASYON ALGORİTMASI KULLANILARAK SINIFLANDIRMA KURALLARININ KEŞFİ

7.1. Weka

Weka (Waikato Environment for Knowledge Analysis), Waikato Üniversitesi tarafından Java programlama dili ile gerçekleştirilmiş, açık kaynak kodlu bir programdır. Weka veri madenciliği görevleri için makine öğrenme algoritmalarının bir koleksiyonudur. Algoritmalar doğrudan bir veri kümesine uygulanabilmektedir ya da Java kodundan da çağrılabilinmektedir. Veri ön işleme, sınıflandırma, regresyon, kümeleme, birliktelik kuralları ve görselleştirme için araçlar içermektedir. Bu tez çalışmasında elde edilen sonuçlar Weka'da bulunan One-R, Ridor, JRip ve Part algoritmalarından elde edilen sonuçlarla karşılaştırılmıştır (URL-1, 2012).

7.1.1. One-R

One-R veya "One Rule (Bir Kural)" R. C. Holt tarafından önerilen basit bir algoritmadır (Holt, 1993). Bu algoritma eğitim verilerinde her bir özellik için bir kural üretmektedir ve sonra One-R'sine göre en küçük hata oranına sahip kural seçilmektedir (Novakovic vd., 2010).

7.1.2. Part

I. H. Witten ve E. Frank tarafından önerilen, böl ve yönet kuralından oluşan bir algoritmadır (Witten ve Frank, 2005). Part Algoritması, kurallar kümesini sıralayan ve karar listeleri olarak adlandırılan kümeler üretmektedir. Bu algoritma her bir çalıştırmada kısmi bir C4.5 karar ağacı oluşturmaktadır ve en iyi kuralı bir yaprağın içinde olmaktadır (Novakovic vd., 2010).

7.1.3. Ridor

Ridor, ilk olarak varsayılan kuralı, sonra ise varsayılan kural için (ağırlıklı) hata oranı en az olan istisnayı oluşturmaktadır. Daha sonra her bir istisna için en iyi istisnaları üretmektedir ve saf (kusursuz) olana kadar bu işlemi tekrarlamaktadır. Böylece istisnaların genişlemesi gibi bir ağaç sunmaktadır ve yaprakta sadece varsayılan kural olmaktadır,

istisnalar olmamaktadır. İstisnalar varsayılan kuraldaki sınıftan başka bir sınıf öngören bir kurallar kümesidir (Novakovic vd., 2010).

7.1.4. Jrip

W. W. Cohen tarafından hata azalımı yapan artan budama tekrarlaması olarak önerilmiştir (Cohen, 1995). Bu algoritma bütün pozitif örnekler kapsanıncaya kadar boş bir kural kümesine kuralları art arda ekleyerek bir kural kümesini oluşturmaktadır. Kurallar hiçbir negatif örnek kapsanmayıncaya kadar bir kuralın öncülüne açgözlülükle koşulları ekleyerek oluşturulmaktadır. Bir kural kümesi inşa edildikten sonra, optimizasyon mesajları kural kümesine gönderilmektedir, böylece boyutunu azaltmaktadır ve eğitim verilerinde uygunluğunu geliştirmektedir (Novakovic vd., 2010).

7.2. Kullanılan Veritabanları

Bu tezde, KSO algoritmasının çalışma mantığına göre Visual C# dili kullanılarak bir program kodlanmıştır. Kodlanan bu programda UCI veri ambarından alınan Ecoli (URL-2, 2012), Diabetes (URL-3, 2012), Liver Disorders (BUPA) (URL-4, 2012) ve New Thyroid (URL-5, 2012) veritabanları kullanılarak kurallar üretilmiştir.

7.2.1. Ecoli Veritabanı

Ecoli veritabanında 336 tane örnek ve 8 adet sınıf bulunmaktadır. Bu 8 adet sınıf ve sınıflardaki örnek sayısı Tablo 7.1'de gösterilmektedir.

Tablo 7.1. Ecoli veritabanındaki sınıflar

Sınıflar	Bu Sınıftaki Örnek Sayısı
cp	143
im	77
pp	52
imU	35
om	20
omL	5
imL	2
imS	2

Bu veritabanında 7 tane nitelik bulunmaktadır. Bu nitelikler ile aldıkları değer aralıkları ise Tablo 7.2'de gösterilmektedir. Bütün nitelikler 0-1 arasında reel değerler almaktadır.

Tablo 7.2. Ecoli veritabanındaki nitelikler ve aldıkları değer aralıkları

Nitelikler	En Düşük Değer	En Yüksek Değer
mcg	0	0.89
gvh	0.16	1
lip	0.48	1
chg	0.5	1
aac	0	0.88
alm1	0.03	1
alm2	0	0.99

Ecoli veritabanının Weka'daki örnek bir görüntüsü Şekil 7.1'de gösterilmektedir.

1: mcg Numeric	2: gvh Numeric	3: lip Numeric	4: chg Numeric	5: aac Numeric	6: alm1 Numeric	7: alm2 Numeric	8: **class** Nominal
0.67	0.81	0.48	0.5	0.25	0.42	0.25	pp
0.64	0.72	0.48	0.5	0.49	0.42	0.19	pp
0.68	0.82	0.48	0.5	0.38	0.65	0.56	pp
0.32	0.39	0.48	0.5	0.53	0.28	0.38	pp
0.7	0.64	0.48	0.5	0.47	0.51	0.47	pp
0.63	0.57	0.48	0.5	0.49	0.7	0.2	pp
0.74	0.82	0.48	0.5	0.49	0.49	0.41	pp
0.63	0.86	0.48	0.5	0.39	0.47	0.34	pp
0.63	0.83	0.48	0.5	0.4	0.39	0.19	pp
0.63	0.71	0.48	0.5	0.6	0.4	0.39	pp
0.71	0.86	0.48	0.5	0.4	0.54	0.32	pp
0.68	0.78	0.48	0.5	0.43	0.44	0.42	pp
0.64	0.84	0.48	0.5	0.37	0.45	0.4	pp
0.74	0.47	0.48	0.5	0.5	0.57	0.42	pp
0.75	0.84	0.48	0.5	0.35	0.52	0.33	pp
0.63	0.65	0.48	0.5	0.39	0.44	0.35	pp
0.69	0.67	0.48	0.5	0.3	0.39	0.24	pp
0.7	0.71	0.48	0.5	0.42	0.84	0.85	pp
0.69	0.8	0.48	0.5	0.46	0.57	0.26	pp
0.64	0.66	0.48	0.5	0.41	0.39	0.2	pp
0.63	0.8	0.48	0.5	0.46	0.31	0.29	pp
0.66	0.71	0.48	0.5	0.41	0.5	0.35	pp
0.69	0.59	0.48	0.5	0.46	0.44	0.52	pp
0.68	0.67	0.48	0.5	0.49	0.4	0.34	pp

Şekil 7.1. Ecoli veritabanından bir görüntü

Ecoli veritabanında cp, im, pp, imU, om ve omL sınıfları için kurallar bulunmuştur. imL ve imS sınıflarında örnek sayısı az olduğu için kural bulunmamıştır.

7.2.2. Pima Indians Diabetes Veritabanı

Diabetes veritabanında 768 tane örnek ve 2 adet sınıf bulunmaktadır. Bu 2 adet sınıf ve sınıflardaki örnek sayısı Tablo 7.3'te gösterilmektedir.

Tablo 7.3. Diabetes veritabanındaki sınıflar

Sınıflar	**Bu Sınıftaki Örnek Sayısı**
tested_negative	500
tested_positive	268

Bu veritabanında 8 tane nitelik bulunmaktadır. Bu nitelikler ile aldıkları değer aralıkları ise Tablo 7.4'te gösterilmektedir. Bütün nitelikler tamsayı veya reel değerler almaktadır.

Tablo 7.4. Diabetes veritabanındaki nitelikler ve aldıkları değer aralıkları

Nitelikler	En Düşük Değer	En Yüksek Değer
preg	0	17
plas	0	199
pres	0	122
skin	0	99
insu	0	846
mass	0	67.1
pedi	0.078	2.42
age	21	81

Pima Indians Diabetes veritabanının Weka'daki örnek bir görüntüsü Şekil 7.2'de gösterilmektedir.

1: preg Numeric	2: plas Numeric	3: pres Numeric	4: skin Numeric	5: insu Numeric	6: mass Numeric	7: pedi Numeric	8: age Numeric	9: class Nominal
4.0	154.0	62.0	31.0	284.0	32.8	0.237	23.0	tested_negative
0.0	102.0	75.0	23.0	0.0	0.0	0.572	21.0	tested_negative
9.0	57.0	80.0	37.0	0.0	32.8	0.096	41.0	tested_negative
2.0	106.0	64.0	35.0	119.0	30.5	1.4	34.0	tested_negative
5.0	147.0	78.0	0.0	0.0	33.7	0.218	65.0	tested_negative
2.0	90.0	70.0	17.0	0.0	27.3	0.085	22.0	tested_negative
1.0	136.0	74.0	50.0	204.0	37.4	0.399	24.0	tested_negative
4.0	114.0	65.0	0.0	0.0	21.9	0.432	37.0	tested_negative
9.0	156.0	86.0	28.0	155.0	34.3	1.189	42.0	tested_positive
1.0	153.0	82.0	42.0	485.0	40.6	0.687	23.0	tested_negative
8.0	188.0	78.0	0.0	0.0	47.9	0.137	43.0	tested_positive
7.0	152.0	88.0	44.0	0.0	50.0	0.337	36.0	tested_positive
2.0	99.0	52.0	15.0	94.0	24.6	0.637	21.0	tested_negative
1.0	109.0	56.0	21.0	135.0	25.2	0.833	23.0	tested_negative
2.0	88.0	74.0	19.0	53.0	29.0	0.229	22.0	tested_negative
17.0	163.0	72.0	41.0	114.0	40.9	0.817	47.0	tested_positive
4.0	151.0	90.0	38.0	0.0	29.7	0.294	36.0	tested_negative
7.0	102.0	74.0	40.0	105.0	37.2	0.204	45.0	tested_negative
0.0	114.0	80.0	34.0	285.0	44.2	0.167	27.0	tested_negative
2.0	100.0	64.0	23.0	0.0	29.7	0.368	21.0	tested_negative
0.0	131.0	88.0	0.0	0.0	31.6	0.743	32.0	tested_positive
6.0	104.0	74.0	18.0	156.0	29.9	0.722	41.0	tested_positive
3.0	148.0	66.0	25.0	0.0	32.5	0.256	22.0	tested_negative
4.0	120.0	68.0	0.0	0.0	29.6	0.709	34.0	tested_negative

Şekil 7.2. Pima Indians Diabetes veritabanından bir görüntü

Pima Indians Diabetes veritabanında sadece “tested-negative” sınıfı için kural bulunmuştur. Bulunan kuralların dışında kalan örnekler “tested-positive” sınıfı olarak düşünülmüştür.

7.2.3. Liver Disorders (BUPA) Veritabanı

Liver Disorders (BUPA) veritabanında 345 tane örnek ve 2 adet sınıf bulunmaktadır. Bu 2 adet sınıf ve sınıflardaki örnek sayısı Tablo 7.5’te gösterilmektedir.

Tablo 7.5. BUPA veritabanındaki sınıflar

Sınıflar	Bu sınıftaki örnek sayısı
1	145
2	200

Bu veritabanında 6 tane nitelik bulunmaktadır. Bu nitelikler ile aldıkları değer aralıkları ise Tablo 7.6’da gösterilmektedir. Bütün nitelikler tamsayı veya reel değerler almaktadır.

Tablo 7.6. BUPA veritabanındaki nitelikler ve aldıkları değer aralıkları

Nitelikler	En Düşük Değer	En Yüksek Değer
mcv	65	103
alkphos	23	138
sgpt	4	155
sgot	5	82
gammagt	5	297
drinks	0	20

Liver Disorders (BUPA) veritabanının Weka’daki örnek bir görüntüsü Şekil 7.3’te gösterilmektedir.

1: mcv Numeric	2: alkphos Numeric	3: sgpt Numeric	4: sgot Numeric	5: gammagt Numeric	6: drinks Numeric	7: **class** Nominal
90.0	51.0	23.0	17.0	27.0	3.0	1
81.0	61.0	32.0	37.0	53.0	3.0	2
89.0	89.0	23.0	18.0	104.0	3.0	2
89.0	65.0	26.0	18.0	36.0	3.0	2
85.0	59.0	25.0	20.0	25.0	3.0	2
92.0	61.0	18.0	13.0	81.0	3.0	2
89.0	63.0	22.0	27.0	10.0	4.0	1
90.0	84.0	18.0	23.0	13.0	4.0	1
88.0	95.0	25.0	19.0	14.0	4.0	1
89.0	35.0	27.0	29.0	17.0	4.0	1
91.0	80.0	37.0	23.0	27.0	4.0	1
91.0	109.0	33.0	15.0	18.0	4.0	1
91.0	65.0	17.0	5.0	7.0	4.0	1
88.0	107.0	29.0	20.0	50.0	4.0	2
87.0	76.0	22.0	55.0	9.0	4.0	2
87.0	86.0	28.0	23.0	21.0	4.0	2
87.0	42.0	26.0	23.0	17.0	4.0	2
88.0	80.0	24.0	25.0	17.0	4.0	2
86.0	67.0	11.0	15.0	8.0	4.0	2
92.0	40.0	19.0	20.0	21.0	4.0	2
85.0	60.0	17.0	21.0	14.0	4.0	2
91.0	57.0	15.0	16.0	16.0	4.0	2
96.0	55.0	48.0	39.0	42.0	4.0	2
79.0	101.0	17.0	27.0	23.0	4.0	2

Şekil 7.3. Liver Disorders (BUPA) veritabanından bir görüntü

7.2.4. Thyroid Disease (New Thyroid)Veritabanı

New Thyroid veritabanında 215 tane örnek ve 3 adet sınıf bulunmaktadır. Bu 3 adet sınıf ve sınıflardaki örnek sayısı Tablo 7.7'de gösterilmektedir.

Tablo 7.7. New Thyroid veritabanındaki sınıflar

Sınıflar	Bu sınıftaki örnek sayısı
1	150
2	35
3	30

Bu veritabanında 5 tane nitelik bulunmaktadır. Bu nitelikler ile aldıkları değer aralıkları ise Tablo 7.8'de gösterilmektedir. Bütün nitelikler tamsayı veya reel değerler almaktadır.

Tablo 7.8. New Thyroid veritabanındaki nitelikler ve aldıkları değer aralıkları

Nitelikler	En Düşük Değer	En Yüksek Değer
T3resin	65	144
Thyroxin	0.5	25.3
triiodothyronine	0.2	10
Thyroidstimulating	0.1	56.4
Tsh_value	-0.7	56.3

New Thyroid veritabanının Weka'daki örnek bir görüntüsü Şekil 7.4'te gösterilmektedir.

1: T3resin Numeric	2: Thyroxin Numeric	3: Triiodothyronine Numeric	4: Thyroidstimulating Numeric	5: TSH_value Numeric	6: **class** Nominal
91.0	8.0	1.7	2.1	4.6	1
103.0	8.5	1.8	1.9	1.1	1
98.0	9.1	1.4	1.9	-0.3	1
111.0	7.8	2.0	1.8	4.1	1
107.0	13.0	1.5	2.8	1.7	1
119.0	11.4	2.3	2.2	1.6	1
122.0	11.8	2.7	1.7	2.3	1
105.0	8.1	2.0	1.9	-0.5	1
109.0	7.6	1.3	2.2	1.9	1
105.0	9.5	1.8	1.6	3.6	1
112.0	5.9	1.7	2.0	1.3	1
112.0	9.5	2.0	1.2	0.7	1
98.0	8.6	1.6	1.6	6.0	1
109.0	12.4	2.3	1.7	0.8	1
114.0	9.1	2.6	1.5	1.5	1
114.0	11.1	2.4	2.0	-0.3	1
110.0	8.4	1.4	1.0	1.9	1
120.0	7.1	1.2	1.5	4.3	1
108.0	10.9	1.2	1.9	1.0	1
108.0	8.7	1.2	2.2	2.5	1
116.0	11.9	1.8	1.9	1.5	1
113.0	11.5	1.5	1.9	2.9	1
105.0	7.0	1.5	2.7	4.3	1
114.0	8.4	1.6	1.6	-0.2	1
114.0	8.1	1.6	1.6	0.5	1
105.0	11.1	1.1	0.8	1.2	1
107.0	13.8	1.5	1.0	1.9	1

Şekil 7.4. New Thyroid veritabanından bir görüntü

7.3. Geliştirilen Uygulama

Bu tez çalışmasında, Visual C# programlama dili kullanılarak KSO Algoritmasının çalışma mantığına göre, sınıflandırma kural keşfi yapan bir program kodlanmıştır.

7.3.1. Kedilerin Oluşturulması ve İlk Değerlerin Atanması

Kullanılan dört veritabanı için de 200 adet kedi yaratılmıştır. Her kedi için pozisyon ve hız değerleri gelişigüzel verilmiştir. Her kedinin ayrıca bayrakları (flag) bulunmaktadır ve bu değere göre arama ya da izleme modu süreçlerine uygulanmaktadır.

Programda her bir kedi bir aday kuralı temsil etmektedir ve program çalıştırıldıktan sonra en iyi uygunluk değerine sahip kediler elde edilen sınıflandırma kurallarıdır. Kedinin her bir pozisyonu ise uygulanan veritabanındaki niteliklerini belirtmek için kullanılmaktadır. Her bir nitelik için Şekil 7.5'te de gösterildiği gibi üç pozisyon tutulmaktadır. İlki olan *b*, o niteliğin aday kuralda olup olmadığını göstermektedir. Bu değer [0,1] aralığında değerler almaktadır, eğer 0.5 in altındaysa o nitelik kuralda yer almamaktadır, 0.5'in üstündeyse de nitelik kuraldadır. İkincisi *a,* nitelik kuralda varsa o niteliğin alabileceği alt sınır, üçüncüsü olan *ü ise*, alabileceği üst sınır olmaktadır. *a* ve *ü* değer aralıkları programda kedilere gelişigüzel bir şekilde atanmaktadır.

$Nitelik_1$			$Nitelik_2$			...			$Nitelik_n$		
b_1	a_1	$ü_1$	b_2	a_2	$ü_2$	...	...	...	b_n	a_n	$ü_n$

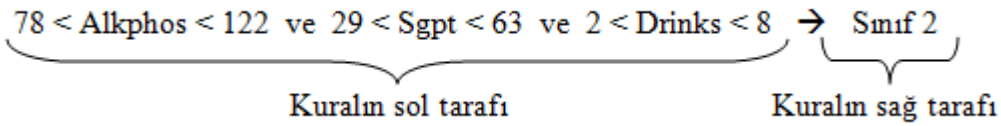
Şekil 7.5. Bir kedinin yapısı

Bir veritabanında n tane nitelik varsa kedilerin her birinde n*3 tane pozisyon bulunmaktadır. Örnek olarak Liver Disorders (BUPA) veritabanında 6 tane nitelik bulunmaktadır. Bu veritabanı için oluşturulan her kedinin, Şekil 7.6'da da gösterildiği gibi 18 tane pozisyonu olmaktadır.

Mcv			Alkphos			Sgpt			Spot			Gammagt			Drinks		
0.36	65	93	0.8	78	122	0.75	29	63	0.15	70	81	0.04	201	214	0.8	2	8

Şekil 7.6. Liver Disorders (BUPA) veritabanı için oluşturulan bir kedinin pozisyonları

Şekil 7.6'da, *Alkphos, Sgpt* ve *Drinks* nitelikleri *b* değerleri 0,5'ten büyük olduğu için kuralda yer almaktadır yani aynı zamanda kuralın sol tarafında yer almaktadır. Kuralın sağ tarafı ise sınıfını temsil etmektedir. Bu aday kuralın "Sınıf 2" için çalıştırıldığı düşünülürse, bu kuralın ifade edilişi Şekil 7.7'deki gibi olmaktadır.



Şekil 7.7. Bir aday kuralın ifade edilişi

7.3.2. Uygunluk Fonksiyonunun Hesaplanması

Kedilerin uygunluk değerleri Eşitlik (7.1)'deki uygunluk fonksiyonu ile hesaplanmaktadır ve sadece uygunluk değeri en iyi olan kedinin pozisyonu hafızada saklanmaktadır.

$$uygunluk = \omega_1 * \frac{DP}{DP+YN} * \frac{DN}{DN+YP} - \omega_2 * anlaşılırlık - \omega_3 * aralıkoranı + \omega_4 * \frac{DP}{DP+YP} \quad (7.1)$$

ω_1, ω_2, ω_3 ve ω_4 değerleri kullanıcı tarafından tanımlanan ağırlıklar olmaktadır ve en iyi sonucu verecek şekilde belirlenmelidir. Bu dört değerin toplamı 1'i vermektedir.

- *DP*, doğru pozitiflerdir, yani kuralla aynı sınıf etiketine sahip olup, kural tarafından kapsanan örneklerin sayısıdır.
- *YP*, yanlış pozitiflerdir, bunlar kuraldan farklı sınıf etiketine sahip olup, kural tarafından kapsanan örneklerin sayısıdır.
- *YN*, yanlış negatiflerdir, kural tarafından kapsanmayan ama kuralla aynı sınıf etiketine sahip örneklerin sayısıdır.
- *DN*, doğru negatiflerdir, bunlar ise kural tarafından kapsanmayıp, kuralla da aynı sınıf etiketine sahip olmayan örneklerin sayısıdır.

anlaşılırlık kriteri daha basit kuralların keşfi için kullanılmaktadır. Bu kriterin hesaplanması Eşitlik (7.2)'de gösterilmektedir. Burada *n* veri kümesindeki karar niteliklerinin sayısını vermektedir.

$$anlaşılırlık = 1 - \frac{kuralın\ solundaki\ nitelik\ sayısı - 1}{n} \quad (7.2)$$

aralıkoranı, ise kuraldaki niteliklerin alt sınır ve üst sınır aralığının dar tutulması için konulan bir kriterdir. Bu kriterin hesaplanması Eşitlik (7.3)'te gösterildiği gibidir. Burada

n, veri kümesindeki karar niteliklerinin sayısını, *ü$_i$, i.* niteliğin üst sınırı, *a$_i$, i.* niteliğin alt sınırı, *Nitelik$_i$max, i.* niteliğin veri kümesinde aldığı maksimum değeri, *Nitelik$_i$min, i.* niteliğin veri kümesinde aldığı minimum değeri ifade etmektedir.

$$aralıkoranı = \frac{\sum_{i=1}^{i=n} \frac{ü_i - a_i}{Nitelik_i max - Nitelik_i min}}{n} \quad (7.3)$$

Daha farklı kurallar elde edilmek istenirse, yukarıdaki kriterlere ek olarak ilginçlik gibi başka kriterler de eklenebilir.

7.3.3. Bitim Şartı

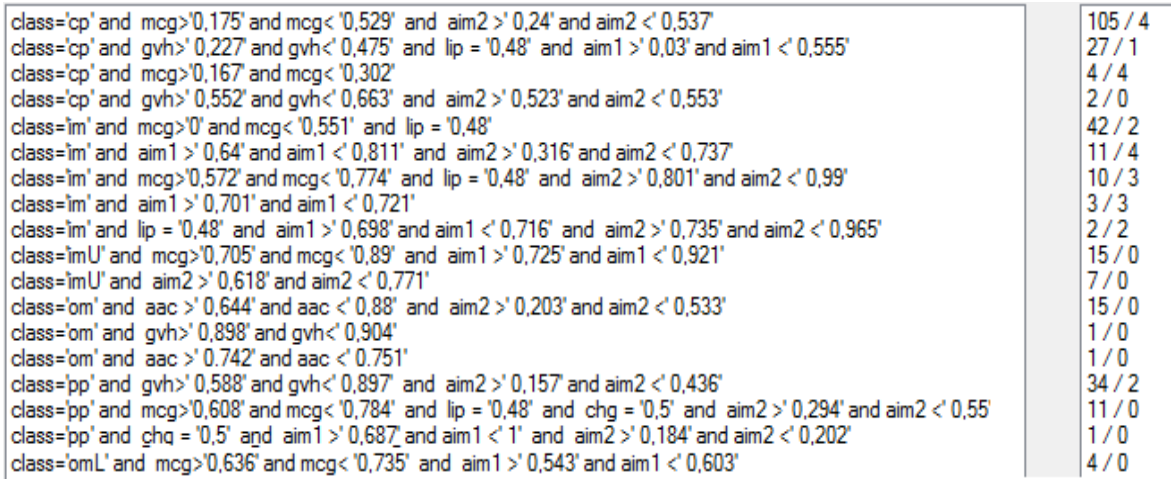
Uygulamada bitim şartı her bir kural için 100 iterasyon olarak alınmıştır. Bu 100 iterasyon boyunca en iyi uygunluk değerine sahip kedi keşfedilen kuraldır. Her bir sınıf içinde o sınıftaki örneklerin %80-%100'ü keşfedilen kurallar tarafından kapsanıncaya kural keşfi yapılmaktadır.

7.3.4. Uygulama Sonuçları

Her veritabanı için 3 ayrı çalıştırmada elde edilen sonuçlar ekran çıktısı olarak aşağıda verilmiştir. Şekillerde her satırda ilk başta kuralın ait olduğu sınıf, daha sonra kuralın alabileceği nitelik değerleri bulunmaktadır. Her kuralın karşısında ise o kural için DP/YP oranı verilmektedir. Bir çalıştırmadaki başarı oranı ise Toplam DP/Veritabanındaki toplam veri sayısı olarak hesaplanmaktadır. Bunlara ek olarak her veritabanı için Weka programında One-R, Ridor, JRip ve Part algoritmalarından elde edilen sonuçlar da tablolar halinde gösterilmiştir. Sınıflandırma kural keşfi yapılırken veritabanlarındaki verilerin hepsi eğitim verisi olarak kullanılmıştır ve bulunan kurallar verilerin tümü üzerinde test edilmiştir.

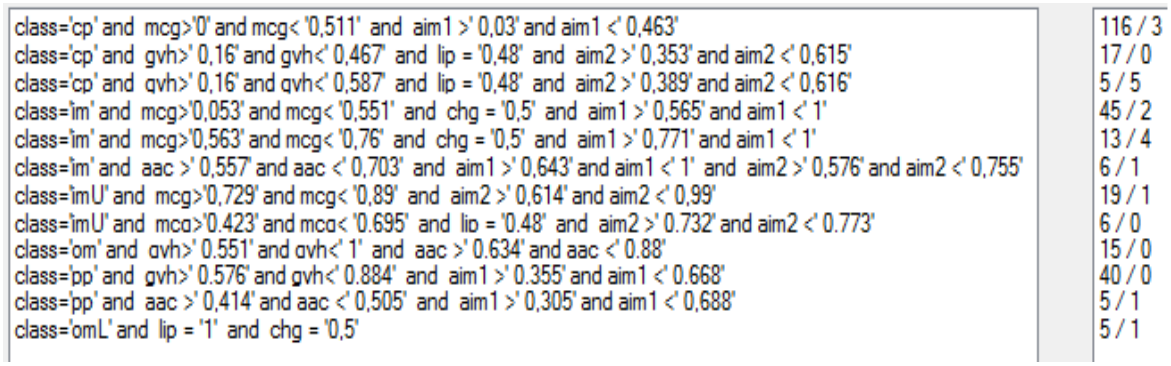
7.3.4.1. Ecoli Veritabanı Sonuçları

Ecoli veritabanı için elde edilen sonuçlar Şekil 7.8, Şekil 7.9 ve Şekil 7.10'da gösterildiği gibidir.



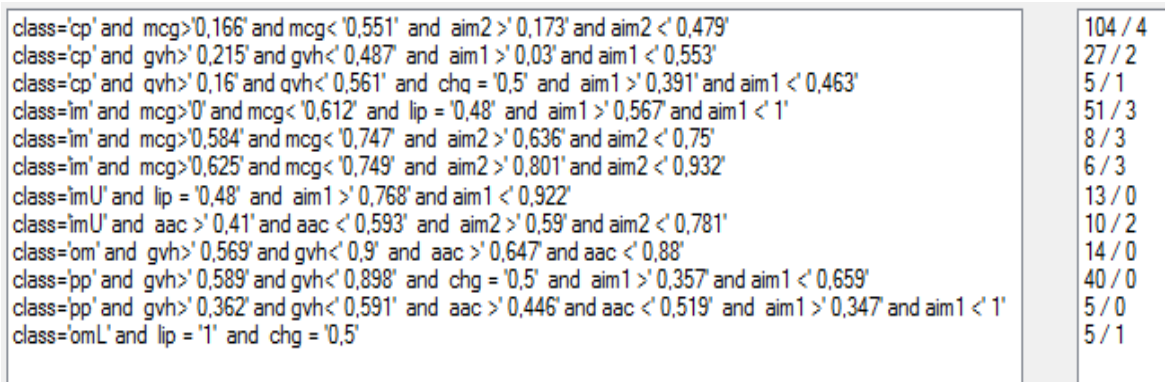
Şekil 7.8. Ecoli veritabanı için birinci çalıştırmada elde edilen kurallar

Şekil 7.8'de 4 tane "cp", 5 tane "im", 2 tane "imU", 3 tane "om", 3 tane "pp" ve 1 tanede "omL" sınıfından olmak üzere toplam 18 tane kural bulunmuştur. Bu çalıştırmada ki verilerin doğru sınıflandırma oranı 295/336=0,878'dir. Yani %87,8 oranla doğru çalışmaktadır.



Şekil 7.9. Ecoli veritabanı için ikinci çalıştırmada elde edilen kurallar

Şekil 7.9'da Ecoli veritabanı için program ikinci bir kez çalıştırıldığında elde edilen sonuçlar yer almaktadır. Buna göre "cp" sınıfından 3 tane, "im" sınıfından 3 tane, "imU" sınıfından 2 tane, "om" sınıfından 1 tane, "pp" sınıfından 2 tane ve "omL" sınıfından 1 tane olmak üzere toplam 12 tane kural bulunmuştur. Bu çalıştırmada verilerin doğru sınıflandırma oranı 292/336= 0,869'dur. Yani % 86,9 doğru sonuç vermektedir.



Şekil 7.10. Ecoli veritabanı için üçüncü çalıştırmada elde edilen kurallar

Şekil 7.10'da Ecoli veritabanı için program üçüncü kez çalıştırıldığında elde edilen sonuçlar yer almaktadır. Bu çalıştırmada "cp" sınıfı için 3 tane, "im" sınıfı için 3 tane, "imU" sınıfı için 2 tane, "om" sınıfı 1 tane, "pp" sınıfı için 2 tane ve "omL" sınıfı için 1 tane olmak üzere toplam 12 tane kural bulunmuştur. Bu çalıştırmada verilerin doğru sınıflandırma oranı 288/336=0,857'dir. Yani 3. çalıştırmada program % 85,7 doğru sonuç vermektedir.

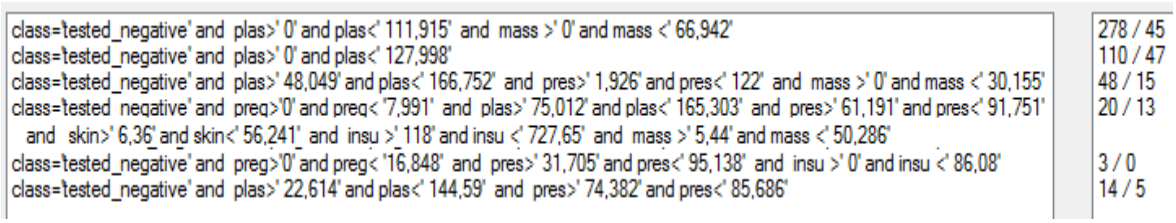
Ecoli veritabanı için yukarıda verilen 3 ayrı sonuca bakılarak, ortalama doğru kurallar bulma yüzdesi %86,8 olmaktadır. Weka programından elde edilen sonuçlar ise Tablo 7.9'da gösterilmektedir.

Tablo 7.9. Weka programında Ecoli veritabanı için elde edilen sonuçlar

Kullanılan Algoritma	Bulunan Kural Sayısı	Toplam DP/ Toplam Veri Sayısı	Doğruluk Yüzdesi
One-R	7	232/336	%69,04
Part	13	308/336	%91,66
Ridor	46	297/336	%88,39
Jrip	10	305/336	%90,77

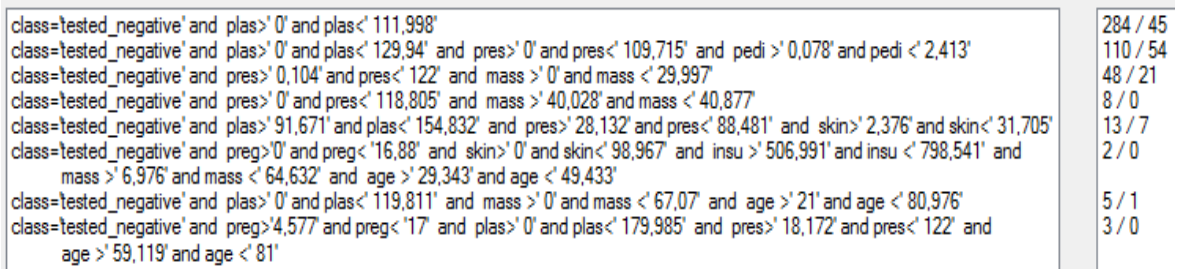
7.3.4.2. Pima Indians Diabetes Veritabanı Sonuçları

Pima Indians Diabetes veritabanı için elde edilen sonuçlar Şekil 7.11, Şekil 7.12 ve Şekil 7.13'te gösterildiği gibidir. Bu veritabanında sadece 2 tane sınıf olduğundan yalnızca "tested-negative" sınıfı için kurallar bulunmuştur. Bu kuralların dışında kalanlar ise "tested-positive" sınıfında olmaktadır.



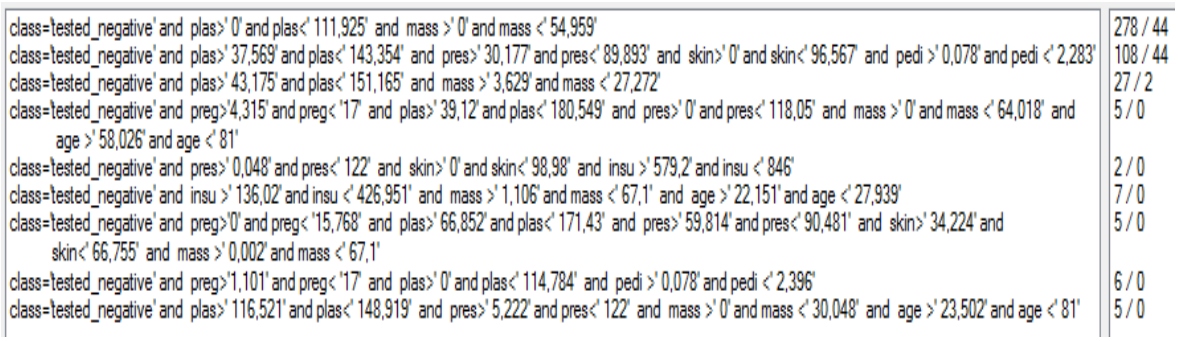
Şekil 7.11. Diabetes veritabanı için birinci çalıştırmada elde edilen kurallar

Şekil 7.11'de Pima Indians Diabetes veritabanı için program çalıştırıldığında elde edilen sonuçlar yer almaktadır. Buna göre "tested-negative" sınıfı için toplam 6 tane kural bulunmuştur. Bu çalıştırmada verilerin doğru sınıflandırma oranı 616/768=0,802'dir. Yani program %80,2 doğru çalışmaktadır.



Şekil 7.12. Diabetes veritabanı için ikinci çalıştırmada elde edilen kurallar

Şekil 7.12'de Pima Indians Diabetes veritabanı için program ikinci defa çalıştırıldığında elde edilen sonuçlar gösterilmektedir. Buna göre "tested-negative" sınıfı için 8 tane kural bulunmuştur. İkinci çalıştırmada verileri doğru sınıflandırma oranı 613/768=0,798'dir. Yani %79,8 doğru sonuçlar bulmaktadır.



Şekil 7.13. Diabetes veritabanı için üçüncü çalıştırmada elde edilen kurallar

Şekil 7.13'te Pima Indians Diabetes veritabanı için program üçüncü kez çalıştırıldığında elde edilen sonuçlar yer almaktadır. Buna göre "tested-negative" sınıfı için 9 tane kural

bulunmuştur. Bu çalıştırmada verileri doğru sınıflandırma oranı 621/768=0,809'dur. Yani %80,9 doğru sonuçlar vermektedir.

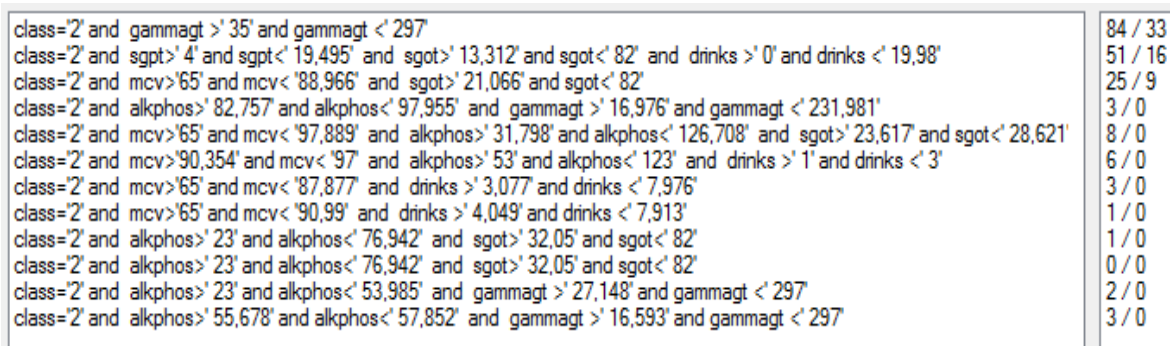
Pima Indians Diabetes veritabanı için yukarıda verilen 3 ayrı sonuca bakılırsa, ortalama olarak programın doğru bulma yüzdesi %80,3 olmaktadır. Bu veritabanı için Weka programından elde edilen sonuçlar ise Tablo 7.10'da gösterilmektedir.

Tablo 7.10. Weka programında Diabetes veritabanı için elde edilen sonuçlar

Kullanılan Algoritma	Bulunan Kural Sayısı	Toplam DP/ Toplam Veri Sayısı	Doğruluk Yüzdesi
One-R	10	586/768	%76,30
Part	13	624/768	%81,25
Ridor	4	605/768	%78,77
Jrip	4	609/768	%79,29

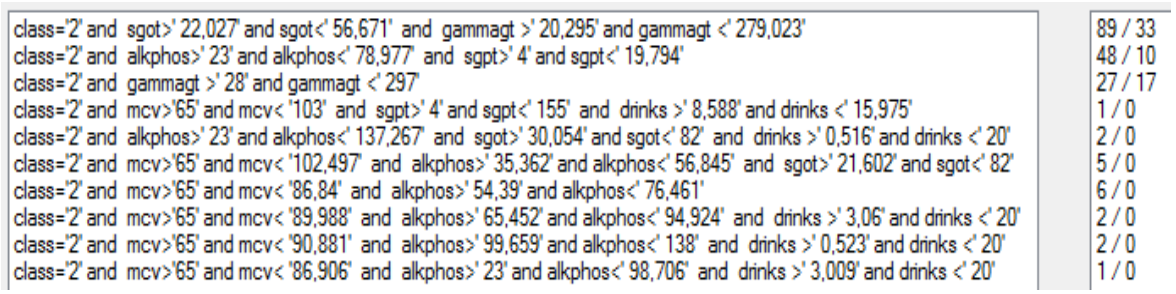
7.3.4.3. Liver Disorders (BUPA) Veritabanı Sonuçları

Liver Disorders (BUPA) veritabanı için elde edilen sonuçlar Şekil 7.14, Şekil 7.15 ve Şekil 7.16'da gösterildiği gibidir. BUPA veritabanında da sadece 2 tane sınıf olduğundan "2" sınıfı için kurallar bulunmuştur. Bu kuralların dışında kalanlar ise "1" sınıfına ait olmaktadır.



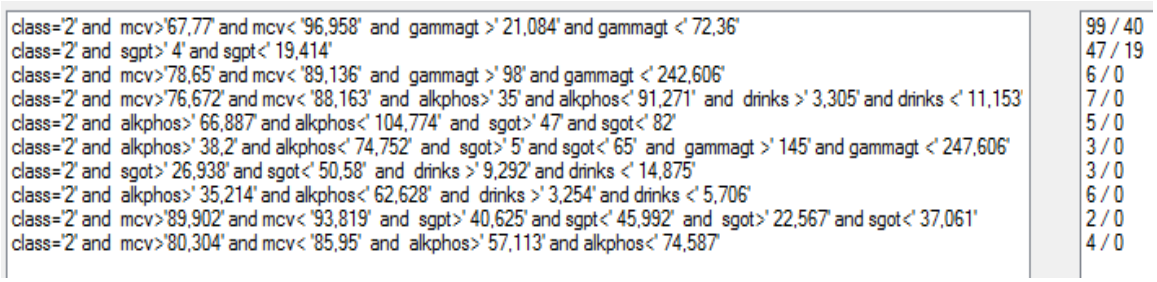
Şekil 7.14. BUPA veritabanı için birinci çalıştırmada elde edilen kurallar

Şekil 7.14'te BUPA veritabanı için birinci çalıştırıldığında elde edilen kurallar yer almaktadır. "2" sınıfına ait toplam 11 tane kural bulunmuştur. Bu çalıştırmada verilerin doğru sınıflandırılma oranı 274/345=0,794'tür. Yani %79,4 doğru kurallar bulmaktadır.



Şekil 7.15. BUPA veritabanı için ikinci çalıştırmada elde edilen kurallar

Şekil 7.15'te BUPA veritabanı için ikinci çalıştırmada elde edilen kurallar yer almaktadır. "2" sınıfına ait toplam 10 tane kural bulunmuştur. Buna göre verilerin doğru sınıflandırma oranı 268/345=0,777'dir. Yani program ikinci çalıştırmada %77,7 doğru kurallar üretmektedir.



Şekil 7.16. BUPA veritabanı için üçüncü çalıştırmada elde edilen kurallar

Şekil 7.16'da BUPA veritabanı için üçüncü çalıştırmada elde edilen kurallar yer almaktadır. "2" sınıfına ait toplam 10 tane kural bulunmuştur. Buna göre verilerin doğru sınıflandırma oranı 268/345=0,777'dir. Yani program üçüncü çalıştırmada %77,7 doğru kurallar üretmektedir.

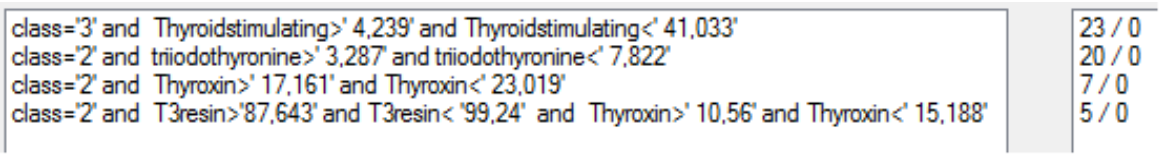
Liver Disorders (BUPA) veritabanı için yukarıda verilen 3 ayrı sonuca bakılırsa, ortalama olarak programın doğru bulma yüzdesi %78,3 olmaktadır. Bu veritabanı için Weka programından elde edilen sonuçlar ise Tablo 7.11'de gösterilmektedir.

Tablo 7.11. Weka programında BUPA veritabanı için elde edilen sonuçlar

Kullanılan Algoritma	Bulunan Kural Sayısı	Toplam DP/ Toplam Veri Sayısı	Doğruluk Yüzdesi
One-R	14	235/345	%68,11
Part	15	297/345	%86,08
Ridor	3	246/345	%71,30
Jrip	5	270/345	%78,26

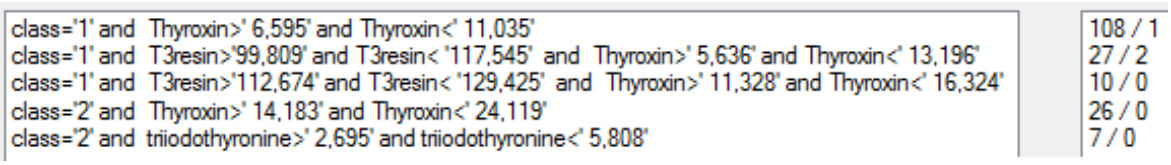
7.3.4.4. Thyroid Disease (New Thyroid) Veritabanı Sonuçları

Thyroid Disease (New Thyroid) veritabanı için elde edilen sonuçlar Şekil 7.17, Şekil 7.18 ve Şekil 7.19'da gösterildiği gibidir. Bu veritabanında 3 tane sınıf bulunmaktadır. Sınıflandırma kuralları bulunurken sadece iki sınıf için kural bulunmaktadır. Bu kuralların dışında kalanlar ise diğer sınıfta olmaktadır.



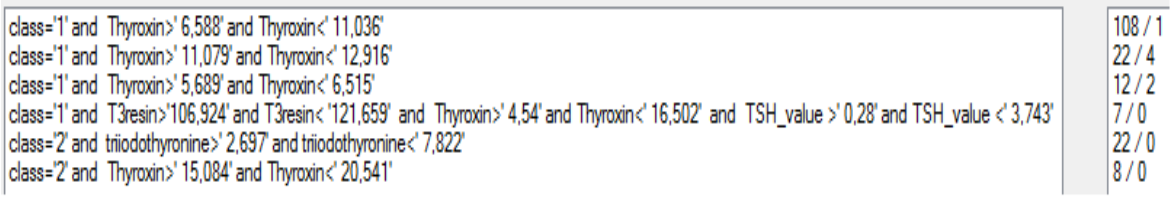
Şekil 7.17. New Thyroid veritabanı için birinci çalıştırmada elde edilen kurallar

Şekil 7.17'de Thyroid Disease (New Thyroid) veritabanı bir kere çalıştırıldığında elde edilen sonuçlar yer almaktadır. Program "3" ve "2" sınıfları için çalıştırılmıştır ve "3" sınıfından 1 tane, "2" sınıfından da 3 tane olmak üzere toplamda 4 tane sınıflandırma kuralı üretilmiştir. Buna göre verilerin doğru sınıflandırılma oranı 205/215=0,953'tür. Yani bu çalıştırmada program %95,3 doğru kurallar bulmaktadır.



Şekil 7.18. New Thyroid veritabanı için ikinci çalıştırmada elde edilen kurallar

Şekil 7.18'de Thyroid Disease (New Thyroid) veritabanı ikinci kez çalıştırıldığında elde edilen sonuçlar yer almaktadır. Bu çalıştırmada "1" ve "2" sınıfları için kurallar bulunmuştur. "1" sınıfından 3 tane ve "2" sınıfından 2 tane olmak üzere toplamda 5 tane kural bulunmuştur. Buna göre verilerin doğru sınıflandırma oranı 206/215=0,958'dir. Yani bu çalıştırmada program %95,8 doğru kurallar üretmektedir.



Şekil 7.19. New Thyroid veritabanı için üçüncü çalıştırmada elde edilen kurallar

Şekil 7.19'da Thyroid Disease (New Thyroid) veritabanı üçüncü kez çalıştırıldığında elde edilen sonuçlar yer almaktadır. Bu çalıştırmada da "1" ve "2" sınıfları için kurallar bulunmuştur. "1" sınıfı için 4 tane ve "2" sınıfı için 2 tane olmak üzere toplamda 6 tane kural bulunmuştur. Buna göre verilerin doğru sınıflandırma oranı 206/215=0,958'dir. Yani program %95,8 doğru kurallar üretmektedir.

Thyroid Disease (New Thyroid) veritabanı için yukarıda verilen 3 ayrı sonuca bakılırsa, ortalama olarak programın doğru bulma yüzdesi %95,6 olmaktadır. Bu veritabanı için Weka programından elde edilen sonuçlar ise Tablo 7.12'de gösterilmektedir.

Tablo 7.12. Weka programında New Thyroid veritabanı için elde edilen sonuçlar

Kullanılan Algoritma	**Bulunan Kural Sayısı**	**Toplam DP/ Toplam Veri Sayısı**	**Doğruluk Yüzdesi**
One-R	3	198/215	%92,09
Part	4	213/215	%99,06
Ridor	7	206/215	%95,81
Jrip	4	209/215	%97,20

8. SONUÇ

Optimizasyon problemleri için birçok algoritma önerilmiştir. Sezgisel algoritmalar, büyük boyutlu optimizasyon problemleri için, kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalardır. Bu tezde, sezgisel optimizasyon algoritmalarından sürü zekâsı tabanlı olanlar incelenmiş ve bu algoritmalardan KSO ile Visual C# dilinde, sınıflandırma kural keşfi yapacak bir program geliştirilmiştir. UCI veri ambarından alınan 4 adet veritabanı bu programda uygulanmış ve elde edilen sonuçlar Weka programından elde edilen sonuçlar ile karşılaştırılmıştır. Buna göre;

Ecoli veritabanı için, bu tez çalışmasında elde edilen kuralların doğruluk yüzdesi ortalama olarak %86,8 olmaktadır. Weka programında ise One-R algoritması için %69,04, Part algoritması için %91,66, Ridor algoritması için %88,39 ve Jrip algoritması içinde %90,77 olmaktadır. Karşılaştırma yapıldığında One-R algoritmasına göre daha iyi sonuçlar elde edilirken, diğer 3 algoritmaya göre daha kötü sonuçlar elde edilmiştir.

Pima Indians Diabetes veritabanı için elde edilen kuralların doğruluk yüzdesi ortalama olarak %80,3 olmaktadır. Weka programında ise One-R algoritması için %76,30, Part algoritması için %81,25, Ridor algoritması için %78,77 ve Jrip algoritması için%79,29 olmaktadır. Bulunan değerler karşılaştırıldığında Part algoritmasına göre daha kötü sonuçlar elde edilirken diğer 3 algoritmadan ise daha iyi sonuçlar bulunmuştur.

Liver Disorders (BUPA) veritabanı için elde edilen sonuçların kuralların yüzdesi %78,3 olmaktadır. Weka programında ise One-R algoritması için %68,11, Part algoritması için %86,08, Ridor algoritması için %71,30 ve Jrip algoritması için %78,26 olmaktadır. Değerler karşılaştırıldığında, Part algoritmasına göre daha kötü sonuçlar elde edilirken, diğer üç algoritmaya göre ise daha iyi sonuçlar elde edilmiştir.

Thyroid Disease (New Thyroid) veritabanı için elde edilen kuralların doğruluk yüzdesi ise ortalama olarak %95,6 olmaktadır. Weka programında ise One-R algoritması için %92,09, Part algoritması için %99,06, Ridor algoritması için %95,81 ve Jrip algoritması için %97,20 olmaktadır. Karşılaştırma yapıldığında Part ve Jrip algoritmalarının göre daha kötü sonuçlar elde edilirken, One-R algoritmasından daha iyi sonuçlar bulunmuştur ve Ridor algoritmasına ise yakın sonuçlar bulunmuştur.

KSO algoritmasının, sınıflandırma kural madenciliğinde kullanılması ilk kez bu çalışmada yapılmıştır. Karşılaştırılan bu sonuçlara göre, KSO algoritmasında herhangi bir optimizasyon yapılmadığı halde sınıflandırma kural keşfinde etkili bir yöntem olmaktadır. Önerilen bu yöntemle istenilen özelliklere göre uygunluk fonksiyonuna ilaveler esnek şekilde yapılabilmektedir. Ayrıca geliştirilen programda, bulunan kurallardaki niteliklerin değer aralıkları kurallarla eş zamanlı olarak bulunmaktadır. Yani niteliklerin değer aralıkları için ayrıca bir ön işlem yapılmamaktadır.

KAYNAKLAR

Alam, M. S., Ul Kabir, M. W., Islam, M. M., 2010. Self-Adaptation of Mutation Step Size in Artificial Bee Colony Algorithm for Continuous Function Optimization, 13th International Conference on Computer and Information Technology, 69-74.

Alataş, B., 2007. Kaotik Haritalı Parçacık Sürü optimizasyon algoritmaları geliştirme, Doktora tezi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

Alataş, B., 2011. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization, Expert Systems with Applications, **38**, 13170-13180.

Alataş, B., Akın, E., 2004, Sınıflandırma Kurallarının Karınca Koloni Algoritmasıyla Keşfi, ELECO'2004, Bursa, Aralık, 357-361.

Apostolopoulos, T., Vlachos, A., 2011. Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem, Hindawi Publishing Corporation International Conference Journal of Combinatorics, **2011**.

Atashpaz-Gargari, E. ve Lucas, C., 2007. Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition, IEEE Congress on Evolutionary Computation, CEC 2007, 4661-4667.

Banharnsakun, A., Achalakul, T., Sirinaovakul, B., 2010. Artificial Bee Colony Algorithm on Distributed Environments, Second World Congress on Nature and Biologically Inspired Computing, 13-18.

Başbuğ, S., 2008. Bakteriyel Besin Arama Algoritması ile Lineer Anten Dizilerinin Diyagram Sıfırlaması, Yüksek Lisans Tezi, Erciyes Üniversitesi, Fen Bilimleri Enstitüsü.

Borji, A., 2007. A New Global Optimization Algorithm Inspired by Parliamentart Political Competitions, Lecture Notes in Computer Science, 4827/2007, 61-71.

Chu, S. C., P. W. Tsai, J. S. Pan, 2006. Cat Swarm Optimization, 9th Pacific Rim International Conference on Artificial Intelligence, LNAI, **4099**, 854-858.

Chu, S. C., Tsai, P. W., 2007. Computational Intelligence Based on The Behavior of Cats, International Journal of Innovative Computing, Information and Control, **3**, 163-173.

Cohen, W. W., 1995. Fast Effective rule indication, Machine Learning: Proceedings of the Twelfth International Conference, Lake Tahoe, California.

Cura, T., 2008. Modern Sezgisel Teknikler ve Uygulamaları, Papatya Yayıncılık Eğitim.

Çelik, M., Karaboğa, D., Köylü, F., 2011. Artificial Bee Colony Data Miner (ABC-Miner), Innovations in Intelligent Systems and Applications (INISTA), 96-100.

Dorigo M., Maniezzo, V. ve Colorni, A., 1991. The Ant System: An Autocatalytic Optimizing Process. Tech. Rep. No. 91- 016, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo M., Maniezzo, V., Colorni, A., 1991. The Ant System: An Autocatalytic Optimizing Process, Tech. Rep., Dipartimento di Elettronica, Politecnico di Milano, Italy.

Geem, Z. W., Kim, J. H., Loganathan, G. V., 2001. A New Heuristic Optimization Algorithm: Harmony Search, Simulation, **76**, 60-68.

Guo, P., Cheng, W., Liang, J., 2011. Global Artificial Bee Colony Search Algorithm for Numerical Function Optimization, Seventh International Conference on Natural Computation, **3**, 1280-1283.

Güden, H., Vakvak, B., Özkan, B. E., Altıparmak, F., Dengiz, B., 2005, Genel Amaçlı Arama Algoritmaları ile Benzetim Eniyilemesi: En İyi Kanban Sayısının Bulunması, Endüstri Mühendisliği Dergisi, **16**, 2-15.

Hedayatzadeh, R., Hasanizadeh, B., Akbari, R., Ziarati, K., 2010. A Multi-Objective Artificial Bee Colony for Optimizing Multi-Objective Problems, 3rd International Conference on Advanced Computer Theory and Engineering, **5**, 277-281.

Holland, J. H., 1975. Adaption in Natural and Artificial Systems, University of Michigan Pres, Ann Arbor, MI.

Holte, R. C., 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Dataset, Machine Learning, **11**, April, 63-90.

Hwang, J. C., Chen, J. C., Pan, J. S., Huang, Y.C., 2009. CSO and PSO to Solve Optimal Contract Capacity for High Tension Customers, International Conference on Power Electronics and Drive Systems, 246-251.

Jadhav, H. T., Patel, J., Sharma, U., Roy, R., 2011. An Elitist Artificial Bee Colony Algorithm for Combined Economic Emission Dispatch Incorporating Wind

Power, 2nd International Conference on Computer and Communication Technology, 640-645.

Jiang, M., Yuan, D., Cheng, Y., 2009. Improved Artificial Fish Swarm Algorithm, Fifth International Conference on Natural Computation, 281-285.

Kalaiselvan, G., Lavanya, A., Natrajan, V., 2011. Enhancing the Performance of Watermarking Based on Cat Swarm Optimization Method, International Conference on Recent Trends in Information Technology, 1081-1086.

Karaboga, D., Gorkemli, B., 2011. A Combinatorial Artificial Bee Colony Algorithm for Traveling Salesman Problem, International Symposium on Innovations in Intelligent Systems and Applications, 50-53.

Karaboğa, D., 2011, Yapay Zekâ Optimizasyon Algoritmaları, Nobel Yayın Dağıtım.

Kennedy, J., Eberhart, R. C, 1995. Particle Swarm Optimization, IEEE International Conference on Neural Networks, Piscataway, NJ, 1942-1948.

Krishnanand, K.N., Ghose, D., 2005. Detection of Multiple Source Locations Using a Glowworm Metaphor with Applications to Collective Robotics, IEEE Swarm Intelligence Symposium, 84-91.

Krishnanand, K.N., Ghose, D., 2009. Glowworm Swarm Optimisation: a New Method for Optimisin Multi-Modal Functions, International Journal of Computational Intelligence Studies, India, **1**, 93-119.

Lin, K. C., Chien, H. Y., 2009. CSO-Based Feature Selection and Parameter Optimization for Support Vector Machine, Joint Conferences on Pervasive Computing, 783-788.

Liu, C., Yan, X., Liu, C., Wu, H., 2011. The Wolf Colony Algorithm and Its Application, Chinese Journal of Electronics, **20**, 212-216.

Murty, K. G., 2003, Optimization Models For Decision Making, Internet Edition, Chapter 1: Models for Decision Making,**1,** 1-18.

Ning, A., Zhang, X., A, 2011. Speech Recognition System Based on Fuzzy Neural Network Trained by Artificial Bee Colony Algorithm, International Conference on Electronics, Communications and Control, 9-11, 2488-2491.

Novakovic, J., Minic, M., Veljovic, A., 2010. Genetic Search for Feature Selection in Rule Induction Algorithms, 18th Telecommunications forum TELFOR, Serbia, Belgrade, November 23-25.

Özkan, Y., 2008. Veri Madenciliği Yöntemleri, Papatya Yayıncılık Eğitim.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. GSA: A Gravitational Search Algorithm, Information Sciences, **179**, 2232-2248.

Santosa, B.; Ningrum, M.K., 2009. Cat Swarm Optimization for Clustering, International Conference of Soft Computing and Pattern Recognition, 54-59.

Shah-Hosseini, H., 2009. The Intelligent Water Drops Algorithm: A Nature-Inspired Swarm-Based Optimization Algorithm, International Journal of Bio-Inspired Computation, **1**, 71-79.

Storn, R., Price, K., (1995), Differential Evolution: A Simple andEfficient Adaptive Scheme for Global Optimization overContinuous Spaces, Technical Report TR-95-012, International Computer Science Institute, Berkeley.

Tekerek, A., 2011. Veri Madenciliği Süreçleri ve Açık Kaynak Kodlu Veri Madenciliği Araçları, Akademik Bilişim'11 – XIII. Akademik Bilişim Konferansı Bildirileri, İnönü Üniversitesi, Malatya, Şubat 2-4.

Tereshko, V., 2000. Reaction-Diffusion Model of a Honey Bee Colony's Foraging Behaviour, 6th International Conference on Parallel Problem Solving from Nature, 3-540-41056-2, Springer-Verlag, London, UK, 807-816.

Tsai, P. W., Pan, J. S., Chen, S.M., Liao, B.Y., Hao, S.P., 2008. Parallel Cat Swarm Optimization, International Conference on Machine Learning and Cybernetics, **6**, 3328-3333.

Wang, W., Wu, J., 2011. Emotion Recognition Based on CSO&SVM in E-Learning, Seventh International Conference on Natural Computation, **1**, 566-570.

Wang, Z. H., Chang, C. C., Li, M. C., 2010. Optimizing Least-Significant-Bit Substitution Using Cat Swarm Optimization Strategy, Information Sciences, 10.1016/j.ins.2010.07.011.

Witten, I. H., Frank, E., 2005. Data Mining: Partical Machine Learning Tools and Techniques, Morgan Kaufmann, San Francisco.

Yang, X. S., 2009, Firefly Algorithms Formultimodal Optimization, Proceedings of the Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computing Sciences, Springer, Sapporo, Japan, **5792**, 178-178.

Yang, X. S., 2010. Firefly Algorithm, Levy Flights and Global Optimization, Research and Development in Intelligent Systems, Springer, London, UK, **XXVI**, 209-218.

URL-1, http://www.cs.waikato.ac.nz/~ml/weka/ Weka veri madenciliği programı, 20 Haziran 2012.

URL-2, http://archive.ics.uci.edu/ml/datasets/Ecoli Ecoli veritabanı. 20 Haziran 2012

URL-3, http://archive.ics.uci.edu/ml/datasets/Diabetes Diabet veritabanı.1 Eylül 2012.

URL-4, http://archive.ics.uci.edu/ml/datasets/Liver+Disorders BUPA Liver Disorders veritabanı, 1 Kasım 2012.

URL-5, http://sci2s.ugr.es/keel/dataset.php?cod=66 New Thyroid veritabanı, 20 Kasım 2012.

ÖZGEÇMİŞ

1987 yılında Tunceli'de doğdu. İlk ve orta dereceli öğrenimini Tunceli'de tamamladı. 2005 yılında Ege Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde öğrenim görmeye hak kazandı. 4 yıllık lisans öğrenimini tamamladıktan sonra 22 Haziran 2009 yılında "Bilgisayar Mühendisi" unvanıyla mezun oldu. Şubat 2010 tarihinde Tunceli Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik Anabilim Dalında Yüksek Lisans öğrenimine başladı. Şubat 2010 tarihinde, Kuramsal Temeller anabilim dalında Tunceli Üniversitesi'ne Araştırma Görevlisi olarak atandı ve hala görevine devam etmektedir.