T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KAOTİK HARİTALI PARÇACIK SÜRÜ OPTİMİZASYONU ALGORİTMALARI GELİŞTİRME

Bilal ALATAŞ

Tez Yöneticisi:
Prof. Dr. Erhan AKIN

DOKTORA TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ, 2007

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KAOTİK HARİTALI PARÇACIK SÜRÜ OPTİMİZASYONU ALGORİTMALARI GELİŞTİRME

Bilal ALATAŞ

DOKTORA TEZİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez, 26/ 10 / 2007 tarihinde aşağıda belirtilen jüri tarafından oybirliği /~~oyçokluğu~~ ile başarılı / ~~başarısız~~ olarak değerlendirilmiştir.

Danışman:	Prof Dr. Erhan AKIN	
Üye:	Prof. Dr. Yakup DEMİR	
Üye:	Prof. Dr. Z. Hakan AKPOLAT	
Üye:	Doç. Dr. Şeref SAĞIROĞLU	
Üye:	Yrd. Doç. Dr. Ali KARCI	

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../..... tarih ve ..... sayılı kararıyla onaylanmıştır.

TEŞEKKÜR

Bu çalışmanın her aşamasında büyük desteğini gördüğüm doktora tez danışmanım Prof. Dr. Erhan AKIN' a sonsuz teşekkürlerimi sunarım.

Çalışmalarım süresince özellikle verdiği doktora bursuyla maddi destek sağlayan Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) Bilim İnsanı Destekleme Daire Başkanlığına (BİDEB); verdiği doktora projesi desteğiyle tezin hazırlanmasındaki maddi katkılarından dolayı Fırat Üniversitesi Bilimsel Araştırma Projeleri (FÜBAP) birimine; evrimsel hesaplama konusundaki yardımlarından dolayı Yrd. Doç. Dr. Ali KARCI' ya; özellikle kaos konusundaki fikirlerinden dolayı Yrd. Doç. Dr. A. Bedri ÖZER' e ve beni her durumda desteklediği için eşime ayrıca teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

SİMGELER LİSTESİ

x_i:	Parçacığın mevcut pozisyonu
v_i:	Parçacığın mevcut hızı
y_i:	Parçacığın kişisel en iyi pozisyonu
f:	Optimize edilen amaç fonksiyonu
$\hat{y}$:	Parçacığın direkt olarak etkileşim içinde olduğu parçacıkların kümesi
l:	Komşuluk büyüklüğü
c_1:	Kişisel en iyiye doğru hızlanma katsayısı
c_2:	Global en iyiye doğru hızlanma katsayısı
n:	Boyut
l_n:	n. boyut için alt sınır
u_n:	n. boyut için üst sınır
w:	Atalet ağırlığı
iter:	İterasyon
maksiter:	İzin verilen maksimum iterasyon sayısı
τ :	Global en iyi parçacığın indeksi
ρ :	Ölçekleme faktörü
d_{maks}:	İki parçacık arasındaki en büyük uzaklık
c_{1i}:	Kişisel en iyiye doğru hızlanma katsayısının başlangıç değeri
c_{1f}:	Kişisel en iyiye doğru hızlanma katsayısının son değeri
c_{2i}:	Global en iyiye doğru hızlanma katsayısının başlangıç değeri
c_{2f}:	Global en iyiye doğru hızlanma katsayısının son değeri
$peniyi_t$:	Parçacığın lokal en iyi değeri
$(peniyi_t)_{\text{ortalama}}$:	Parçacıkların lokal en iyi değerlerinin ortalaması
$geniyi_t$:	Sürüdeki tüm lokal en iyi değerin en iyisi
$N_{hatalar}$:	Ardışık hata sayısı
$N_{basarilar}$:	Ardışık başarı sayısı
X_n:	n. kaotik sayı
Q_{seviye}:	Algoritmayı belli toleransa yakınsadığında sonlandırma şartı
$N_{tüm}$:	Tüm deneme sayısı
$N_{basarili}$:	Maksimum iterasyon sayısında Q_{seviye} üzerinde sonuç bulan deneme sayısı
$\underline{x}$:	x'in alt sınırı

$\bar{x}$:	x'in üst sınırı
r_i:	Kaba parametre
$Sinir_{maks}(r_i)$:	Kaba parametre r_i'nin değeri için izin verilen maksimum sınır
$N_{TumPozitif}$:	Pozitif kayıtların sayısı
$N_{Pozitif}$:	Keşfedilen kurallar tarafından kapsanan kayıtların sayısı
$N_{TumNegatif}$:	Negatif kayıtların sayısı
$N_{Negatif}$:	Kural tarafından yanlışlıkla kapsanan negatif kayıt sayısı
I:	Amaç fonksiyon sayısı
$\uparrow$:	Maksimizasyon
$\downarrow$:	Minimizasyon
Ort:	Ortalama
Min:	En iyi
Maks:	En kötü
Med:	Medyan
SS:	Standart sapma
Ort İter:	Optimum değere yakınsamak için gereken ortalama iterasyon sayısı

KISALTMALAR LİSTESİ

PSO:	Parçacık Sürü Optimizasyonu
GA:	Genetik Algoritma
ZDHK:	Zamanla Değişen Hızlanma Katsayıları
YGPSO:	Yakınsama Garantili Parçacık Sürü Optimizasyonu
ÇBPSO:	Çoklu Başlatmalı Parçacık Sürü Optimizasyonu
ÇİPSO:	Çekici ve İtici Parçacık Sürü Optimizasyonu
DPSO:	Dağıtan Parçacık Sürü Optimizasyonu
DGPSO:	Diferansiyel Gelişimli Parçacık Sürü Optimizasyonu
U-UOTPSO:	Uygunluk-Uzaklık Oran Tabanlı Parçacık Sürü Optimizasyonu
KHPSO:	Kaotik Haritalı Parçacık Sürü Optimizasyonu
ZDAA:	Zamanla Değişen Atalet Ağırlığı
StZDAA-PSO:	Stokastik Zamanla Değişen Atalet Ağırlıklı Parçacık Sürü Optimizasyonu
KPSOA:	Kaba Parçacık Sürü Optimizasyon Algoritması
ÖÖTPSO:	Öz Örgütlenmiş Tehlikelilikli Parçacık Sürü Optimizasyonu
A-S:	Ata-Sonuç
AS:	Alt Sınır
ÜS:	Üst Sınır
KKPSO:	Kaba Kaotik Parçacık Sürü Optimizasyonu
HD:	Harici Depo
ÇAKKPSO:	Çok Amaçlı Kaba Kaotik Parçacık Sürü Optimizasyonu
A:	Anlaşılabilirlik
TD:	Tahmini Doğruluk
S:	Sınıf

ÖZET

Doktora Tezi

KAOTİK HARİTALI PARÇACIK SÜRÜ OPTİMİZASYONU ALGORİTMALARI GELİŞTİRME

Bilal ALATAŞ

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği
2007, Sayfa: 140

Bu tezde kuş sürülerinin hareketlerinden esinlenerek geliştirilmiş optimizasyon yöntemi olan Parçacık Sürü Optimizasyonu (PSO) algoritmasına, performansının arttırılması amacıyla çeşitli eklentiler ve düzenlemeler yapılmıştır. Özellikle PSO'nun yerel optimum noktalara takılıp kalmasını engelleyerek global yakınsama hızını arttırmak amacıyla yumuşak hesaplama tekniklerinden biri sayılan kaos, PSO ile birleştirilmiş ve "kaotik haritalı PSO algoritmaları" adı altında on iki farklı PSO algoritması önerilmiştir. Önerilen bu algoritmaların literatürde iyi sonuç verdiği belirlenmiş diğer PSO algoritmalarıyla performans karşılaştırmaları yapılmıştır.

Ayrıca sürekli karar değişkenlerinin de kullanılması gereken problemlerde ve aralıkların temsil olarak kullanılması gereken durumlarda PSO ile birlikte yumuşak hesaplama tekniklerinden sayılan kaba kümelerin alt dalı olan aralık cebrinin etkili olarak kullanılabileceği gösterilmiş, bununla ilgili PSO hesaplamaları açıklanmıştır. Bu amaçla "kaba PSO" algoritması önerilmiş ve bunun ilk uygulaması olarak özellikle veri madenciliğinde sürekli değerli verilerde kural keşfi için yeni ve etkili yöntem olarak kullanılabileceği gösterilmiştir.

PSO, kaos ve kaba kümelerin üçünün birleşimiyle "kaba kaotik PSO algoritmaları" adı altında genel amaçlı PSO algoritmaları önerilmiş ve bunlar, etkin çözüm yöntemi bulunmayan sürekli değerli verilerde nicel birliktelik kurallarının otomatik keşfi alanında ilk kez uygulanmış ve etkili sonuçlar elde dilmiştir.

PSO algoritmasının çok amaçlı optimizasyon problemleri için de çalışabilmesi için çeşitli düzenlemeler yapılmış ve ilk uygulaması yine veri madenciliğinin alt dalı olan sınıflandırma kurallarının madenciliğinde yapılmıştır ve istenen amaçlar doğrultusunda etkili sonuçlar elde edilmiştir.

Son olarak, çok amaçlı PSO ile kaos ve kaba kümelerin birleşimiyle yeni PSO algoritmaları, "çok amaçlı kaba kaotik PSO algoritmaları", önerilmiş ve bunlar da ilk olarak etkin çözümler bulmak amacıyla veri madenciliği alanında uygulanmıştır.

Anahtar Kelimeler: Parçacık sürü optimizasyonu, kaotik haritalar, aralık cebri, çok amaçlı optimizasyon, nicel birliktelik kural madenciliği, sınıflandırma kural madenciliği.

ABSTRACT

PhD Thesis

DEVELOPMENT OF CHAOTIC MAPS EMBEDDED PARTICLE SWARM OPTIMIZATION ALGORITHMS

Bilal ALATAŞ

Firat University
Graduate School of Natural and Applied Sciences
Department of Electronics and Electrical Engineering
2007, Page: 140

In this thesis, addition and modifications to particle swarm optimization (PSO) algorithm which is an optimization technique developed inspiring by movement of flock of birds have been performed. Especially, chaos which has been regarded as one of the soft computing techniques has been embedded to PSO in order to increase the convergence speed by escaping from the local optimum points; and twelve PSO algorithms with the name "chaos embedded PSO algorithms" have been proposed. Performance comparisons of these algorithms with the other PSO algorithms which have been reported to have good performance in the literature have been performed.

Furthermore, it has been shown that interval algebra which is a branch of rough set, regarded as one of the soft computing techniques, can be effectively used with PSO for problems in which continuous decision variables and intervals should be used as representation, and PSO computation related to this representation have been described. "Rough PSO" has been proposed for this purpose and has been shown to be effectively used in rule mining within continuous valued variables as a first application.

PSO, chaos, and rough set have been combined and general purposed PSO algorithms with the name "rough chaotic PSO algorithms" have been proposed. These algorithms have been firstly used in numeric association rules mining in which there is not an efficient and automatic technique. Promising results have been obtained.

Various modifications have been performed for the PSO to let it work for multi-objective optimization problems and first application has been performed in classification rule mining task of data mining. Efficient results according to the objectives have been obtained.

Lastly, new PSO algorithms, "multi-objective rough chaotic PSO algorithms", which include multi-objective PSO, chaos, and rough sets combination, have been proposed and have been applied in data mining in order to find efficient solutions.

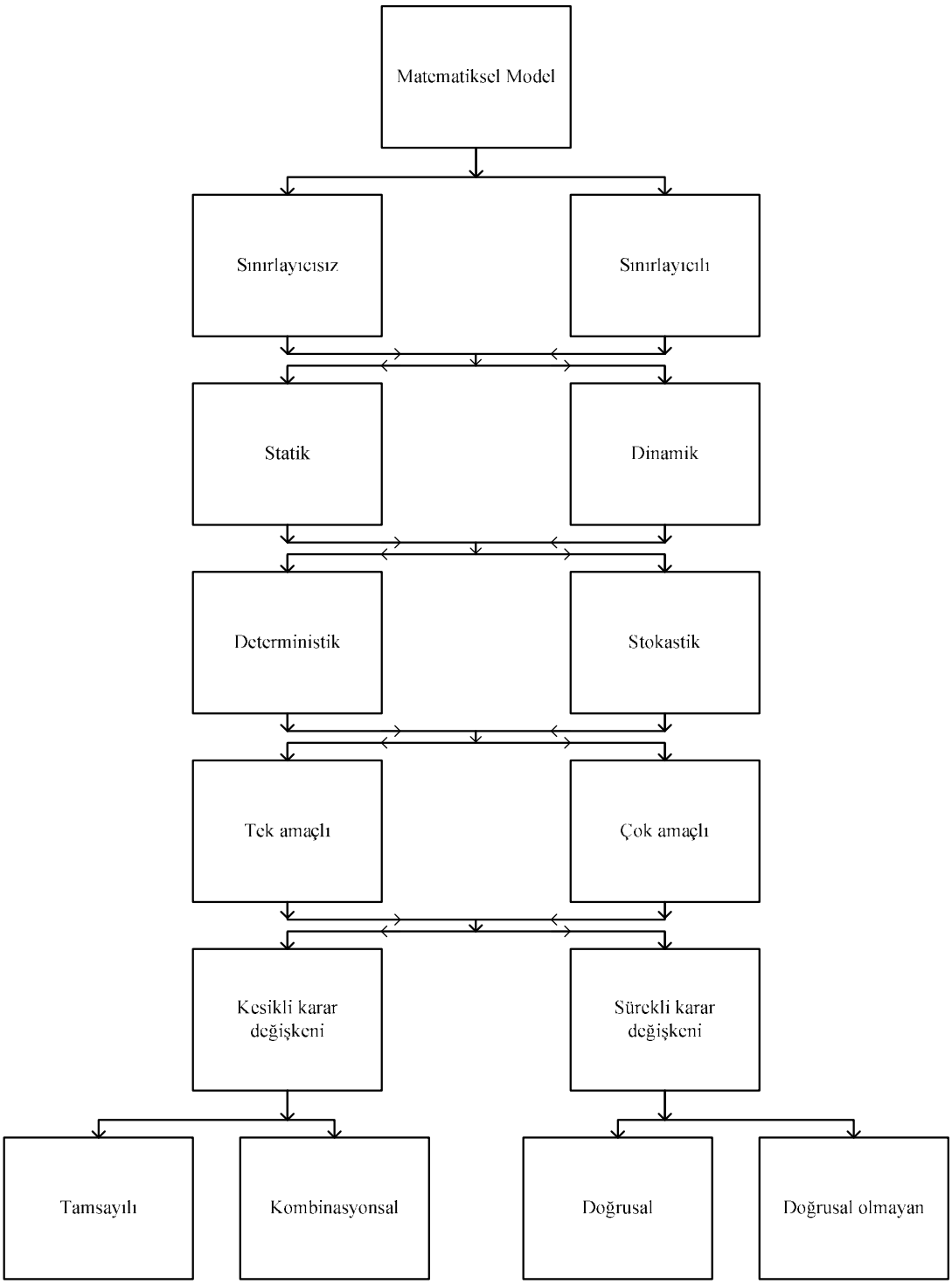
Keywords: Particle swarm optimization, chaotic maps, interval algebra, multi-objective optimization, numeric association rule mining, classification rule mining.

1. GİRİŞ

Optimizasyon, bir problem için, verilen şartlar altında tüm çözümler arasından en iyi çözümü elde etme işidir. Optimizasyonun performansını etkileyen ve kontrolümüz altında değerleri olan değişkenlere *karar değişkenleri* denir. 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. O halde 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 [1].

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 [1]. Şekil 1.1'de bu model türleri grafiksel olarak görülmektedir. Burada yukarıdan aşağıya ve soldan sağa gidildikçe modelin kurulması ve işletilmesi zorlaşır.



Şekil 1.1. 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 [2]. 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.

Genel amaçlı sezgisel yöntemler; biyolojik tabanlı, fizik tabanlı ve sosyal tabanlı olmak üzere üç farklı grupta değerlendirilmektedir [3]. Ayrıca bunların birleşimi olan melez yöntemler de vardır. Genetik algoritma (GA) [4–6], diferansiyel gelişim algoritması [7, 8], karınca koloni algoritmaları [9, 10], yapay sinir ağları [11], arı koloni algoritmaları [12] ve yapay bağışıklık sistemleri [13, 14] biyolojik tabanlı; ısıl işlem [15] ve elektromanyetizma algoritması [16] fizik tabanlı ve tabu arama [17] sosyal tabanlı algoritma ve modellerdir. Kültürel algoritma [18] da hem biyolojik hem de sosyal tabanlı algoritma olarak sınıflandırılabilir. Bu tezde üzerine çalışılan, optimizasyon ve veri madenciliğinde uygulamaları yapılan Parçacık Sürü Optimizasyonu (PSO) [19] da hem biyolojik hem de sosyal tabanlı algoritma sınıfına girmektedir.

Algoritma tek bir çözümden başlayıp bunu operatörlerle ilerletiyorsa bunlara tek noktalı yöntemler denir ve tabu arama, ısıl işlem gibi bütün yerel arama tabanlı sezgisel algoritmalar bu gruba girer. Çok noktadan yani bir popülasyon üzerinden çözüme başlanıp bu farklı noktalarla optimizasyon yapılıyorsa bunlar çok noktalı ya da popülasyon tabanlı yöntemlerdir. Bunlara

GA, PSO, karınca koloni algoritmaları, arı koloni algoritmaları, yapay bağışıklık sistemleri ve elektromanyetizma algoritması örnek olarak verilebilir.

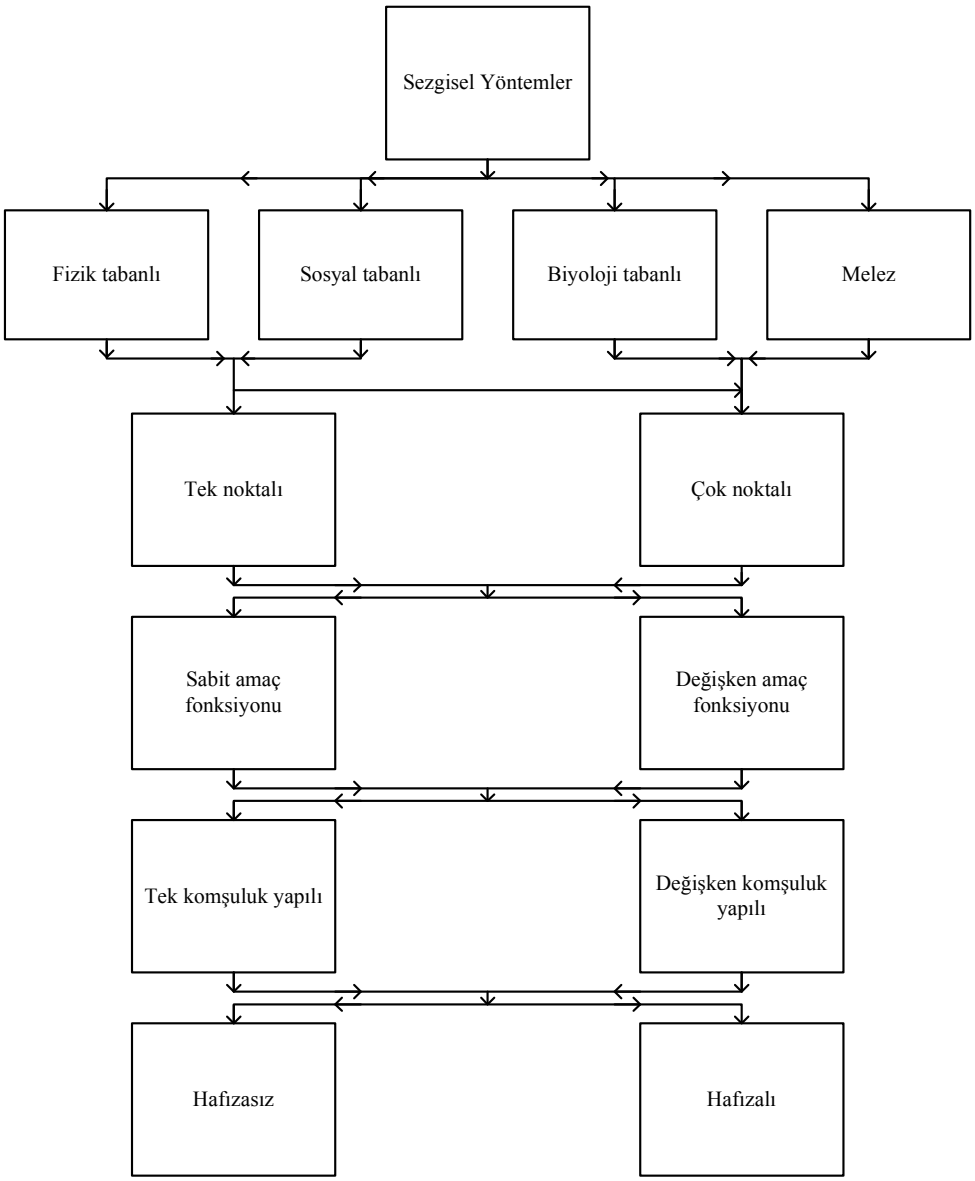
Bazı sezgisel algoritmalar problemin gösterimini gerçekleştirirken amaç fonksiyonunu sabit tutar ve sabit amaç fonksiyonlu olarak adlandırılırlar. Bazıları da örneğin rehberli yerel arama algoritmasındaki gibi değiştirir ve değişen amaç fonksiyonlu olarak adlandırılırlar. Değiştirmekteki amaç yerel minimumdan kurtulmaktır. Bu mantık, yerel minimumdan kaçmak için bazen diğer sezgisel algoritmalara da uygulanabilmektedir.

Çoğunlukla sezgisel yöntemler tek bir komşuluk yapısında çalışır ve tek komşuluk yapılı olarak sınıflandırılabilir. Bazıları da değişken komşuluk arama algoritmasında olduğu gibi arama işlemini sistematik bir şekilde değiştirerek birden fazla yerel arama yöntemiyle diğer çözüm alanlarına ulaşmaya çalışır ve değişken komşuluk yapılı şeklinde sınıflandırılabilir. PSO'nun iki versiyonu da bulunabilmektedir.

Algoritmalar çalışırken daha önceki durumlar ya da en iyi durumlar hatırlanıyorsa hafızalı, hatırlanmıyorsa hafızasız şeklinde sınıflandırılabilir. Örneğin PSO ve tabu arama hafızalı, GA hafızasızdır.

Tüm bu sınıflandırma türleri Şekil 1.2'de görülmektedir. Ancak yapılan bazı çalışmalarla bu sınıflandırma türlerine ait algoritmalara eklentiler ve modifikasyonlar yapılmakta ve bunlarla algoritma ilk önerildiği haliyle ait olduğu sınıftan başka sınıfa ait özellikleri de barındırabilmektedir. Bu gösterimde de, yukarıdan aşağıya ve soldan sağa gidildikçe modelin kurulması ve işletilmesi biraz daha zorlaşmaktadır. Ancak bu zorluk yukarıda bahsedilen matematiksel modellerdeki zorluklar kadar fazla değildir. Küçük modifikasyonlarla dönüşümler ayarlanabilmektedir.

Bu bölümde çok noktalı, sosyal, fizik ve biyolojik tabanlı stokastik yöntemler biraz daha incelenecektir.



Şekil 1.2. Sezgisel yöntemler

1.1. Çok Noktalı (Popülasyon Tabanlı) Algoritmalar

1.1.1. Evrimsel Hesaplama

Evrimsel hesaplama Rechenberg'in Evrim Stratejileri (Evolutionsstrategie) adlı çalışmasında tanıtılmıştır [20]. Tasarımda ve uygulamada evrim kavramı bulunan hesaplama-tabanlı problem çözme yöntemlerinin temel mantığını içerir. En çok kullanılan yöntemler

- Genetik Algoritmalar
- Evrimsel Algoritmalar
- Evrimsel Programlama

- Evrim Stratejileri
- Sınıflayıcı Sistemleri
- Genetik Programlama

olarak bilinir. Bunlar arasında en çok kullanılan ise GA adı verilen yöntemdir. GA'lar doğal seçim ilkelerine dayanan ve temelleri John Holland tarafından atılan arama ve optimizasyon yöntemleridir [4, 5]. GA'ların çözüm için ihtiyaç duyduğu şey, problemin karar değişkenlerini uygun bir yöntemle kodlamak ve olası çözümlerin problem için kalitesini ölçen bir uygunluk fonksiyonu oluşturmaktır. Algoritma, birey ya da kromozom adı verilen aday çözümlerle (popülasyon) başlar ve çeşitli genetik ya da farklı operatörlerle uygunluk fonksiyon değerine bağlı olarak daha iyi kalitede yeni popülasyonun oluşturulması adımlarıyla ilerler. Bu adımlar belli bir durdurma kriterine kadar devam eder.

Evrimsel hesaplama yöntemleri arasındaki temel fark, bazen temsil ve ortaya çıkan çözüm olmakla birlikte çoğu zaman kullanılan operatörler ve bir sonraki popülasyonun seçim işleminden kaynaklanmaktadır.

Yakın zamanda yeni popülasyonun oluşturulmasında olasılık dağılımları kullanılmış ve daha öz bir kodlamayla GA performansı arttırılmıştır. Bu alanda yapılan çalışmalar genel olarak dağılım tahmin algoritmaları olarak bilinir [21] ve en çok kullanılanlar ise özlü GA [22], popülasyon temelli artımsal öğrenme [23], tek değişkenli marjinal dağılım algoritması [24] ve çok değişkenli normal dağılım algoritmasıdır [25].

Ayrıca, diferansiyel gelişim algoritması da özellikle sürekli değerli karar değişkenlerinin söz konusu olduğu problemlerde işleyiş olarak GA'ya benzer şekilde etkin olarak kullanılmaktadır [7]. Bazı yerel arama yöntemleriyle birleştirilmiş GA, melez GA ya da paralel GA olarak ta bilinen memetik algoritma [26] ve arama ya da optimizasyon esnasında problem bilgisinin birleştirilmesine izin veren kültürel algoritma [27] de evrimsel hesaplama alanındaki diğer algoritmalara örnek olarak verilebilir.

1.1.2. Karınca Koloni Algoritmaları

Sürü zekâsı alanının bir alt dalı olarak bilinen karınca koloni algoritması karınca kolonilerinin yiyecek toplamasını esas alan ve Dorigo tarafından önerilen bir algoritmadır [9]. Çeşitli versiyonları olan bu algoritmayı evrimsel hesaplama alanının bir alt dalı olarak görenler de vardır. Karıncalar koloniler halinde yiyecek toplarken en kısa yolu belirlerler ve bunu da geçmiş oldukları yollar üzerinde uçucu bir koku izi bırakarak hallederler. Karıncalar gidebileceği birden fazla yol olduğunda, iz miktarı fazla olan yolu daha çok tercih ederler.

Karınca koloni algoritmaları da gözlemlenen bu doğal süreçten esinlenerek geliştirilmiş popülasyon tabanlı algoritmalardır.

1.1.3. Arı Koloni Algoritmaları

Arı koloni algoritmaları arıların yiyecek bulma vb. davranışlarından esinlenerek geliştirilmiş sürü zekâsı alanının en yeni alt dallarındandır. Ancak bu alanda tam olarak çatı oturtulmuş değildir ve birçok farklı araştırmacı modelledikleri sisteme arı koloni algoritması ya da benzeri isimler vermiştir [28–42]. Yiyecek bulmada arıların yapmış olduğu dans etkileşimli bir davranıştır ve bu dansla yiyecek arayan başarılı arılar (öncü arılar) çiçek beneklerine olan yön ve uzaklık bilgilerini ve çiçek üzerindeki nektar miktarını kovandaki eşiyle paylaşır. Bu öncü arıların kolonideki diğer arıları farklı kaynak bulmak amacıyla verimli yerlere yerleştirmesi başarılı, etkileşimli bir mekanizmadır ve koloni hızlı bir şekilde tam olarak değişen nektar kaynaklarına göre kendini yerinde ve zamanında ayarlar. Bu algoritmalar da arıların koloni halindeki davranışlarından esinlenerek geliştirilmiştir.

1.1.4. Yapay Bağışıklık Sistemleri

Yapay bağışıklık sistemleri, teorik bağışıklık ve karmaşık problem alanlarına uygulanan gözlemlenmiş bağışık fonksiyonlar, ilkeler ve modellerden esinlenmiş hesapsal sistemlerdir [43]. İki temel bileşen (kemik iliği ve timus) ve iki ayrı teori (klonal seçim ve bağışık ağ) bağışıklık sistemini modellemek için kullanılır. Kemik iliği modeli, hücrelerin ve moleküllerin repertuarını üretmede kullanılır. Timus modeli öz / öz olmayan ayrımı yapmaya yetenekli hücre ve moleküllerin repertuarını üretmede kullanılır. Klonal seçim prensibi, bağışıklık sisteminin bir antijenik uyarıma karşı bağışıklık cevabının temel özelliklerini tanımlamak amacıyla kullanılır. Daha çok optimizasyon amaçlı kullanılan teori budur ve klonal seçim algoritması adıyla bilinir [44]. Diferansiyel denklemler temelli sürekli ağ modelleri başarılı bir şekilde optimizasyon problemlerine de uygulanmıştır. Bunlar aynı zamanda fark denklemleri temelli ayrık ağ modellerine de ilham olmuştur [45].

1.1.5. Parçacık Sürü Optimizasyonu

Parçacık Sürü Optimizasyonu (PSO) tekniği kuş ve balık sürülerinin hareketlerinden esinlenerek doğrusal olmayan nümerik problemlere optimal sonuçlar bulmak için ilk olarak 1995–1996 yıllarında sosyolog-psikolog James Kennedy ve elektrik mühendisi Russel Eberhart tarafından ortaya atılmış popülasyon tabanlı stokastik bir optimizasyon yöntemidir [46]. PSO'da her bir kuş parçacık olarak, kuş topluluğu da sürü olarak temsil edilir. 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 hatırlar. Çözüm uzayındaki her boyuttaki hızının ve yönünün her seferinde nasıl değişeceğini, komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının birleşiminden elde eder. Bu konu ile ilgili çalışmalar, tez kapsamında detaylı olarak incelenmiş ve yeni önerilerde bulunulmuştur.

1.1.6. Elektromanyetizma Algoritması

Elektromanyetizma algoritması Birbil ve Fang tarafından doğrusal olmayan fonksiyonların sınırlayıcısız global optimizasyonu için elektromanyetizma teorisindeki itme-çekme mekanizmasından esinlenerek geliştirilmiştir [16]. Her örneğin tüm şarjın amaç fonksiyon değerini kapsadığı şarjlı bir parçacık olarak uzaya serbest bırakıldığını varsayar. Şarj örnek popülasyon üzerinde noktanın itme ya da çekme büyüklüğünü belirler ve amaç fonksiyon değeri ne kadar büyükse çekim büyüklüğü de o ölçüde fazladır. Bu şarjlar hesaplandıktan sonra toplam kuvvetten bir yön belirlenir. Çekimler daha iyi alanlara doğru, itmeler ise ziyaret edilmeyen bölgelere doğru hareketi sağlar [47, 48]. Aslında bu algoritma PSO'ya benzemektedir. Temel fark arama uzayındaki hareketleri hesaplamadadır. PSO'daki gibi elektromanyetizma algoritmasında da her çözüm, her adımda yeni pozisyona doğru hareket eder. Fakat sadece komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının birleşiminden etkilenmez her çözüm popülasyondaki tüm diğer çözümlerden etkilenir [49].

1.2. Tezin Amaç ve Kapsamı

Bu tezde özellikle çok noktalı, sosyal ve biyolojik tabanlı stokastik bir yöntem olan ve kuş ve balık sürülerinin hareketlerinden esinlenerek doğrusal olmayan nümerik problemlere optimum sonuçlar bulmak için önerilmiş yumuşak hesaplama tekniklerinden olan PSO algoritmasına eklentiler yapılıp performansının arttırılması amaçlanmıştır. Bu amaçla yine yumuşak hesaplama tekniklerinden biri sayılan kaos, PSO ile birleştirilmiş ve on iki farklı PSO algoritması önerilmiştir. Önerilen bu algoritmaların performansları literatürde iyi sonuç verdiği belirlenmiş diğer PSO algoritmalarıyla karşılaştırılmış ve bu yöntemlerle eşdeğer düzeyde ya da daha iyi olduğu gösterilmiştir.

Ayrıca sürekli değerli problemlerde aralıkların temsil olarak kullanılması gereken durumlarda, PSO'ya yine yumuşak hesaplama tekniklerinden sayılan kaba kümelerin alt dalı olan aralık cebrinin kullanılabileceğinin gösterilmesi ve çok amaçlı optimizasyon problemleri için de çalışabilmesi için çeşitli düzenlemelerin yapılması amaçlanmıştır. Bir başka amaç da bu

yöntemleri ilk olarak veri madenciliği alanında kabul edilecek zaman dilimi içinde optimum çözümü bulunmayan problemlere uygulayarak etkili sonuçlar elde etmektir.

Bu tezin kapsamı;

- PSO'nun yerel optimum noktalara takılıp kalmasını engelleyerek global yakınsama hızını arttırmak için kaos tabanlı çeşitli yöntemler önermek,
- Kalite testi fonksiyonlarında, önerilen bu yöntemleri kullanan PSO ve literatürde iyi sonuçlar verdiği belirtilmiş diğer PSO algoritmalarıyla çeşitli karşılaştırmalar yapmak,
- Kaba kümelerin alt dalı olan aralık cebrinin PSO'da sürekli değerli problemlerde aralıkların temsil olarak kullanılması gereken durumlarda etkili olarak kullanılabileceğini göstermek ve bununla ilgili hesaplamaları göstermek (Kaba PSO),
- Veri madenciliğinin yöntemlerinden olan birliktelik kural madenciliği ve sınıflandırma kural madenciliğinin tanımını ve kullanılan performans ölçütlerini vermek,
- Önerilen kaba PSO algoritmasının gerçekten iyi sonuç verebileceği bir alana uygulamak ve özellikle sürekli değerli verilerde kural keşfi için etkili ve yeni bir yöntem sunmak,
- PSO, kaos ve kaba kümelerin üçünün birleşimiyle etkili PSO algoritmalar önermek ve bunları, etkin çözüm yöntemi bulunmayan sürekli değerli verilerde nicel birliktelik kurallarının otomatik keşfi alanında uygulayıp sonuçları karşılaştırmak,
- PSO'ya çok amaçlı optimizasyon problemleri için de çalışabilmesi için çeşitli düzenlemeler yapmak ve özellikle veri madenciliğinden sınıflandırma kural madenciliğini çok amaçlı bir optimizasyon problemi olarak karakterize edip önerilen bu PSO ile uygulamasını yapmak,
- Çok amaçlı PSO ile kaos ve kaba kümelerin birleşimiyle yeni PSO algoritmaları, çok amaçlı kaba kaotik PSO algoritmaları, önermek ve bunu da yine etkin çözümler bulmak amacıyla veri madenciliği alanında uygulamak

olacaktır.

1.3. Tezin Organizasyonu

Tez sekiz bölümden oluşmaktadır. İkinci bölümde, PSO algoritması tanıtılmış ve basit olarak nasıl çalıştığı küçük örneklerle açıklanmıştır. Ayrıca performansının arttırılabilmesi için literatürde yapılan çalışmalar ilgili başlıklar halinde özetlenmiştir.

Üçüncü bölümde, PSO'nun parametrelerinin belirlenmesi için rasgele tabanlı bir seçim söz konusu olduğunda farklı kaotik sistemler rasgele sayı dizilerinin yerine kullanılmış ve on iki farklı PSO önerilmiştir. Kullanılan kaotik haritalar tanıtılmış sonrasında da önerilen yöntemler

açıklanmıştır. Bu şekilde PSO'nun global yakınsama özelliğinin arttırılması ve lokal çözümde takılıp kalması önlenmeye çalışılmıştır. Daha sonra kalite testi fonksiyonları tanıtılarak önerilen yöntemlerin diğer PSO yöntemleriyle karşılaştırılması yapılmıştır. Son olarak da sonuçlara bağlı olarak analiz ve yorumlara yer verilmiştir. Bu bölümle ilgili aşağıdaki çalışmalar yayınlanmak üzere kabul edilmiştir [50, 51]:

- "Chaos Embedded Particle Swarm Optimization Algorithms", Chaos, Solitons & Fractals, Elsevier, http://dx.doi.org/10.1016/j.chaos.2007.09.063, (Kabul edildi).
- "Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları", 12. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi, Eskişehir, 2007.

Dördüncü bölümde, bazı sürekli karar değişkenleri içeren ve değişkenlerin aralıklarının kullanma zorunluluğu olduğu durum ve uygulamalarda alt ve üst sınırdan oluşan kaba değerlere bağlı olarak PSO'nun genişletilmesi sağlanmış ve eklentiler önerilmiştir. Aynı zamanda, bu şekilde kaba hesaplama alanına da ilave bir konu eklenmiştir. Ayrıca kabul edilecek zaman dilimi içinde optimum çözümü bulunmayan ve aslında bir arama ve optimizasyon problemi olan nicel nitelikler içeren veri tabanlarında veri madenciliği için yeni, etkili ve otomatik bir yöntem önerilmiştir ve bilgi kayıpları ve ön işlemelerden kaçınılarak bilgi keşfi yapılmıştır. Bu bölüm ile ilgili, özellikle diğer biyolojik tabanlı algoritmalar için kaba küme genişletilmesi ve nicel birliktelik kurallarının bir optimizasyon problemi olarak biyolojik tabanlı algoritmalarla modellenmesiyle ilgili aşağıdaki çalışmalar yayınlanmıştır [6, 8, 52, 53]:

- "MODENAR: Multi-Objective Differential Evolution Algorithm for Mining Numeric Association Rules", Applied Soft Computing, Elsevier.
- "An Efficient Genetic Algorithm for Automated Mining of Both Positive and Negative Quantitative Association Rules", Soft Computing, Springer Verlag, 10(3), 230-237, 2006.
- "Rough Immune Algorithm", International Symposium on Innovations in Intelligent Systems and Applications (INISTA-2005), 75-78, İstanbul, Turkey, Haziran, 2005.
- "Rough Differential Evolution Algorithm", Second International Conference on Electronics and Computer Engineering IKECCO 2005, Bishkek/KYRGYZSTAN, Nisan, 2005.

Beşinci bölümde, dördüncü bölümde önerilen kaba PSO algoritması ile üçüncü bölümde önerilen kaotik haritalı PSO algoritmalarının birleşimi ile Kaba Kaotik PSO (KKPSO)

algoritmaları önerilmiştir. Kaba değerlerle temsil ve hesaplamaların yapılması zorunluluğu olan yerlerde kullanılan kaba PSO algoritmasına kaotik haritalar eklenmiş ve algoritmanın performansının arttırılması amaçlanmıştır. Böylece; kaba kümeler, kaos ve optimizasyonun birlikte kullanılabileceği genel amaçlı melez arama ve optimizasyon algoritmaları önerilmiştir. Bu amaçla KKPSO teorileri açıklanmış ve performansının testi için nicel birliktelik kural madenciliği alanında uygulamalar yapılmıştır. Bu bölümle ilgili çalışmalar aşağıda belirtilen isimde kitap bölümü olarak yayınlanmak üzere kabul edilmiştir [54]:

- "Chaotic Rough Particle Swarm Optimization Algorithms", Swarm Intelligence: : Focus on Ant and Particle Swarm Optimization (Editörler: Professor Felix T. S. Chan ve Professor Manoj Kumar Tiwari), ARS Publishing, 2007.

Altıncı bölümde PSO'nun çok amaçlı optimizasyon problemlerinde kullanılabilmesi için farklı bir yöntem önerilmiş ve ilk olarak veri madenciliği uygulamaları yapılmıştır. Özellikle önerilen algoritmanın sınıflandırma kural madenciliği alanında etkili ve iyi performansa sahip olduğu görülmüştür. Bu bölümle ilgili, özellikle sınıflandırma kural madenciliğinin biyolojik tabanlı algoritmalarla modellenmesiyle ilgili ve PSO'nun tek amaçlı ve çok amaçlı sınıflandırma kural madenciliğinde kullanımı ile ilgili aşağıdaki çalışmalar yayınlanmıştır [10, 14, 55, 56]:

- "Mining Fuzzy Classification Rules Using an Artificial Immune System with Boosting", ADBIS 2005, Lecture Notes in Computer Science, Springer-Verlag, 3631, 283 – 293, Eylül, 2005.
- "FCACO: Fuzzy Classification Rules Mining Algorithm with Ant Colony Optimization", ICNC 2005, Lecture Notes in Computer Science, Springer-Verlag, 3612, 787 – 797, Ağustos, 2005.
- "Modified Particle Swarm Optimization Based Multi-Objective Rule Mining", INISTA 2007, 195-199, Istanbul, Turkey, 2007.
- "Sınıflandırma Kurallarının Parçacık Sürü Optimizasyon Algoritmasıyla Keşfi", 62-66, ASYU-INISTA 2006, Istanbul, Haziran, 2006.

Yedinci bölümde, üçüncü bölümde önerilen kaotik haritalı PSO algoritmaları, dördüncü bölümde önerilen kaba PSO algoritması ve altıncı bölümde önerilen çok amaçlı PSO birleştirilerek çok amaçlı kaba kaotik PSO algoritmaları önerilmiştir ve yine bir optimizasyon

problemi olarak algılanması gereken nicel değerler içeren veritabanlarında birliktelik kural keşfi alanında uygulamaları yapılıp performans karşılaştırmaları verilmiştir.

Sekizinci bölümde ise tezde yapılan çalışmalar değerlendirilmiş ve gelecek çalışmalara ışık tutması için yeni öneriler sunulmuştur.

2. PARÇACIK SÜRÜ OPTİMİZASYONU

2.1. Giriş

Gerçek hayatta sosyal varlıklar problemlerin çözümünü, ortak hareket ederek daha kısa zamanda bulabilmektedir. 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. Sosyal varlıkların bu tür davranışlarının sayısal ortamlardaki simülasyonlarıyla, bunların çeşitli problemlerin çözümünde kullanılabilir sezgiseller olduğu gösterilebilir.

Çoğu mühendislik problemlerinin çözülmesinde sezgisel yöntemlerin kullanılması son yıllarda önemli bir oranda artmaktadır. Bunun nedenleri, eğim bilgisine gerek duyulmaması ve bu yüzden matematiksel modelin çıkarılamadığı ve eğim bilgisinin türetilemediği ya da türetilmesinin çok maliyetli olduğu problemlerde iyi sonuçlar alınabilmesi; hesaplama gücünün iyi olması; uygulamasının basit olması ve dönüştürülebilir yani bir problem için yazılmış bir sezgisel programın kolaylıkla başka problemlere uygulanabilir olmasıdır.

PSO tekniği de popülasyon tabanlı sezgisel arama ve optimizasyon yöntemidir ve temelde işbirliğini esas alır. Bu yöntemde problem için aday çözümler parçacık olarak adlandırılır. Temel PSO'da bir parçacığın başlıca özellikleri şunlardır: *pozisyon*, *hız* (ya da daha açık olarak pozisyonu değiştirmek için ona uygulanan bir operatör), bilgiyi komşularla *değiştirme* kabiliyeti, bir önceki pozisyonu *hatırlama* kabiliyeti ve bir karar vermek için *bilgiyi kullanma* kabiliyeti [57].

Uyarlamalı Kültür Modeli parçacık sürülerinin temelini oluşturmaktadır. Bu teori kültürel uyarlamanın temelinde *değerlendir*, *karşılaştır* ve *taklit et* olarak üç ilkenin olduğunu belirtmektedir [57].

Yaşayan organizmalar için en hazır davranışsal karakteristik belki de uyarıyı *değerlendirme*, yani pozitif ya da negatif, çekici ya da itici olarak değerlendirme eğilimidir. Öğrenme, değerlendirme olamadan gerçekleşemez. Değerlendirmenin kendisi, tek başına *karşılaştırma* kabiliyeti olmadan kullanışsız ve imkânsızdır. Uyarlamalı Kültür Modelindeki ve parçacık sürülerindeki bireyler, kendilerini komşularıyla karşılaştırır ve sadece kendilerinden daha üstün olan komşularını taklit eder. *Taklit etme* alıcının açısından deneyim paylaşmanın en

basit formudur, sadece gözlemi değil, amaç gerçekleştirme ve zamanlama yeterliliğini de içerir [57].

PSO algoritmalarında bir parçacık, kendi deneyimini yani en iyi geçmiş pozisyonunu ve en başarılı komşusunun deneyimini hesaba katarak bir sonraki hareketinin yönüne karar verir.

2.2. Parçacık Sürü Optimizasyonu

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. PSO popülasyon tabanlı stokastik bir optimizasyon yöntemi olup çok parametreli ve çok değişkenli optimizasyon problemlerine çözümler üretmek için kullanılmaktadır [19, 58, 59].

Parçacık sürü kavramı basitleştirilmiş sosyal sistemin bir simülasyonu olarak ortaya çıkmıştır. Başlangıçtaki amaç, kuş ya da balık sürü koreografisinin grafiksel olarak simülasyonlarını yapmaktı. 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.

PSO'nun önemli özelliklerinden ikisi uygulama kolaylığı ve eğim bilgisine gerek duymamasıdır. GA kullanılarak çözülebilen çoğu problemi içeren farklı optimizasyon problemlerinin geniş bir kümesini çözmek için kullanılabilir. Buna, sinirsel ağ eğitimi [60] ve fonksiyon minimizasyonu gibi [61] problemler örnek olarak verilebilir. Ortalamada, GA ve PSO'nun aynı etkinliğe sahip olduğu (çözüm kalitesi) ancak PSO'nun hesapsal olarak daha etkili olduğu yani daha az fonksiyon değerlendirmesi kullandığı belirlenmiştir [62]. Ayrıca PSO'da belirlenmesi gereken parametre sayısı daha azdır ve uygulaması daha kolaydır.

Çoğu popüler optimizasyon algoritması, eğim-tabanlı algoritmalar gibi deterministiktir. Evrimsel algoritma ailesine bağlı algoritmalara benzer olarak PSO, hata fonksiyonundan türetilen eğim bilgisine ihtiyaç duymayan stokastik bir algoritmadır. Bu özellik, PSO' nun, eğim

bilgisinin elde edilemediği ya da elde edilmesinin hesapsal olarak çok maliyetli olduğu problemlerde kullanılmasını sağlar.

2.2.1. PSO Algoritması

PSO'nun temelinin sosyolojik esinlemeli olduğu söylenebilir. Çünkü algoritmanın orijinal fikri, kuşların sürü halinde toplanmasıyla ilişkilendirilmiş sosyolojik davranışlarına dayanır [58]. Kuş, balık ve hayvan sürülerinin bir "bilgi paylaşma" yaklaşımı uygulayarak çevrelerine adapte olabilme, zengin yiyecek kaynağı bulabilme ve avcılardan kaçabilme yeteneklerinden esinlenmiştir.

PSO, optimum ya da optimuma yakın çözüm bulmak için önce her biri aday çözümü sunan bireyler (parçacıklar) oluşturur. Bu bireylerin oluşturulması gelişigüzel, düzenli ya da her iki şekilde yapılabilir. Bireylerin bir araya gelmesinden çözüm için gerçekleştirilen popülasyonumuz (sürü) meydana gelir. Pratikte, 2 ve 100 arası boyuttaki çoğu gerçek problemler için 20 parçacıklı bir sürü oldukça iyi çalışmaktadır. Uyarlamalı sürü boyutu da kullanılabilir. PSO, bireyler arasındaki bilginin paylaşımını esas alır. Her bir parçacık kendi pozisyonunu sürüdeki en iyi pozisyona doğru ayarlarken, bir önceki tecrübesinden de yararlanır. s sürünün boyutu olsun. Her i parçacığı, birkaç karakteristiğe sahip bir nesne olarak temsil edilebilir. Bu karakteristikler aşağıdaki sembollere gösterilir:

x_i: Parçacığın mevcut pozisyonu;
v_i: Parçacığın mevcut hızı;
y_i: Parçacığın kişisel en iyi pozisyonu.

i parçacığıyla ilişkilendirilmiş kişisel en iyi pozisyon, parçacığın ziyaret ettiği (bir önceki x_i değeri) ve bu parçacık için en yüksek uygunluk değerini veren en iyi pozisyondur. Bir minimizasyon işi için daha düşük bir fonksiyon değeri sağlayan bir pozisyon daha yüksek uygunluğa sahip kabul edilir. f sembolü minimize edilen amaç fonksiyonunu göstermek üzere kişisel en iyi pozisyon için güncelleme denklemi t zaman aralığına bağlı olarak (2.1)'de gösterilmiştir.

$$y_i(t+1) = \begin{cases} y_i(t) & , f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1), & f(x_i(t+1)) < f(y_i(t)) \end{cases} \tag{2.1}$$

PSO'nun *global* ve *lokal* adlı iki versiyonu bulunmaktadır [63]. İki algoritma arasındaki fark verilen bir parçacığın direkt olarak etkileşim içinde olduğu parçacıkların kümesine bağlıdır

ve $\hat{y}$ sembolü, bu etkileşimi temsil için kullanılacaktır. İki modelin detayları aşağıda tam olarak açıklanacaktır. *Global* modelde kullanılan $\hat{y}$ 'nin tanımı Eşitlik (2.2)'de sunulmuştur.

$$\hat{y}(t) \in \{y_0(t), y_1(t), ..., y_s(t)\} \mid f(\hat{y}(t)) = \min\{f(y_0(t)), f(y_1(t)), ..., f(y_s(t))\} \quad (2.2)$$

Bu tanım, $\hat{y}$ 'nin herhangi bir parçacık tarafından şimdiye kadar keşfedilen en iyi pozisyon olduğunu belirtir. Algoritma iki bağımsız gelişigüzel diziyi kullanır, $r_1 \sim U(0, 1)$ ve $r_2 \sim U(0, 1)$. Bu diziler (2.3)'te gösterildiği gibi algoritmanın stokastik doğasını etkilemek için kullanılır. Burada c_1 ve c_2 katsayıları öğrenme faktörleridir ve *hızlanma katsayıları* olarak da adlandırılır. Bu katsayılar, bir iterasyonda bir parçacığın alabileceği adımın maksimum boyutunu etkiler ve her parçacığı kişisel en iyi ve global en iyi pozisyonlarına doğru çeken, stokastik hızlanmayı ifade eder. Düşük değerlerin seçilmesi parçacıkların hedef bölgeye doğru çekilmeden önce, bu bölgeden uzak yerlerde dolaşmalarına imkân verir. Ancak hedefe ulaşma süresi uzayabilir. Diğer yandan, yüksek değerlerin seçilmesi, hedefe ulaşmayı hızlandırırken, beklenmedik hareketlerin oluşmasına ve hedef bölgeye ulaşılmamasına sebep olabilir. c_1 ve c_2'nin değerleri $0 < c_1, c_2 \leq 2$ sabitleriyle sınırlanır. Ancak bilişsel katsayının olarak da isimlendirilen c_1'in biraz daha büyük seçilmesi ve $c_1 + c_2 = 4$ durumunun sağlanması halinde daha iyi sonuçların alınabileceği gösterilmiştir. Hız güncelleme adımı her boyut $j \in 1 \ldots n$ için ayrı olarak belirlenir. $v_{i,j}$, i. parçacıkla ilişkilendirilmiş hız vektörünün j. boyutunu gösterirse hız güncelleme denklemi (2.3)

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (2.3)$$

olur. Buradan c_1'in bu parçacığın kişisel en iyi pozisyonunun yönünde adım büyüklüğünü ayarladığı, c_2'nin de global en iyi parçacığın yönünde maksimum adım büyüklüğünü ayarladığı açıktır. $v_{i,j}$ değeri, parçacığın arama uzayından ayrılma olasılığını azaltmak için $[-v_{maks}, v_{maks}]$ değerine sıkıştırılmıştır. Eğer arama uzayı $[-x_{maks}, x_{maks}]$ sınırları ile tanımlanmışsa, v_{maks}'ın değeri tipik olarak $v_{maks} = k \times x_{maks}$ değerine ayarlanır ve $0.1 \leq k \leq 1.0$ olur [64]. Ayrıca v_{maks} arama uzayı büyüklüğünün yarısına eşitlenebilir.

Her parçacığın pozisyonu, bu parçacık için yeni hız vektörü kullanılarak

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2.4)$$

şeklinde güncellenir.

$x_{i,n}(t) \in [l_n, u_n]$ ve $1 \le n \le N$ olmak üzere l_n ve u_n n. boyut için alt ve üst sınırları göstermektedir. PSO algoritmasının çalışması esnasında, boyutlar için alt sınır ya da üst sınırın aşılması durumunda bir düzeltme işlemi uygulanması gerekmektedir. Bu işlem alt sınır ve üst sınır aşılması durumunda sırasıyla

$$x_i = x_i + \alpha \times \mathrm{r} \times (u_n(x_i) - l_n(x_i)) \tag{2.5}$$

$$x_i = x_i - \alpha \times \mathrm{r} \times (u_n(x_i) - l_n(x_i)) \tag{2.6}$$

olarak gerçekleştirilebilir. Burada $\alpha \in [0, 1]$ aralığında kullanıcı tanımlı bir değerdir ve $r \sim U(0, 1)$ olarak seçilir.

Global model daha hızlı bir yakınsama sunar, ancak sağlam olmayabilir. Bu model sürüdeki tüm parçacıklar karşısında *global en iyi parçacık* olarak adlandırılan sadece tek bir "en iyi çözüm"ü muhafaza eder. Bu parçacık, diğer tüm parçacıkları çeken bir çekici olarak hareket eder. Er geç tüm parçacıklar bu pozisyona doğru hareket edecektir ve eğer düzenli olarak güncellenmezse parçacık erken yakınsayabilir. $\hat{y}$ ve v_i için güncelleme denklemlerine bakıldığında $\hat{y}$ *global en iyi pozisyon*dur ve *global en iyi parçacık* olarak bilinen parçacığa aittir.

Lokal model çoklu çekicileri devam ettirerek erken yakınsamayı önlemeye çalışır. Her parçacık için daha sonra *lokal (kişisel) en iyi parçacık*, $\hat{y}_i$'nin seçildiği parçacıkların alt kümesi tanımlanır. $\hat{y}_i$ sembolü *lokal en iyi pozisyon* ya da *en iyi komşuluk* olarak adlandırılır. l büyüklüğünde bir komşuluk için, parçacık indisinin s'de katlandığı varsayılırsa, *lokal* güncelleme denklemleri aşağıdaki gibi olur:

$$N_i = \{y_{i-l}(t), y_{i-l+1}(t), ..., y_{i-1}(t), y_i(t), y_{i+1}(t), ..., y_{i-1}(t), y_{i+l}(t)\} \tag{2.7}$$

$$\hat{y}_i(t+1) \in N_i \mid f(\hat{y}_i(t+1)) = \min\{f(a)\}, \qquad \forall a \in N_i \tag{2.8}$$

Bütünlük için tekrar hız güncelleme denklemini yazılırsa:

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)] \tag{2.9}$$

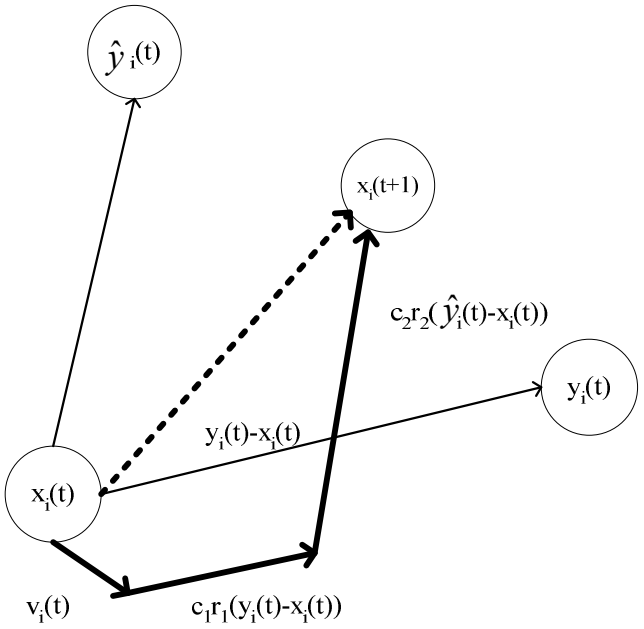
N_i'nin alt kümesi için seçilen parçacıkların arama uzayı domeninde birbirleriyle hiçbir ilişkisi yoktur, seçim sadece parçacığın indeks numarasına bağlıdır. Bu iki nedenden dolayı yapılmaktadır: herhangi bir kümeleme yapılmadığı için hesapsal olarak masraflı değildir ve

arama uzayında mevcut yerlerinden bağımsız olarak iyi çözümler hakkında bilginin tüm parçacıklara yayılmasını destekler.

$l = s$ durumunda *global* model *lokal* modelin özel bir durumudur. $l = 1$ ile yapılan deneyler *lokal* algoritmanın *global* versiyona göre daha yavaş yakınsadığını, fakat değersiz bir lokal minimuma takılma olasılığının daha az olduğunu göstermiştir.

Global model daha hızlıdır, fakat bazı problemler için lokal optimuma takılabilir. *Global* model hızlı bir sonuç almak için, *lokal* model de aramayı inceleştirmek için kullanılabilir.

Bir parçacığın (2.3) denklemindeki üç terime bağlı olarak ve (2.4) pozisyon güncelleme denklemine göre hareketi Şekil 2.1'de gösterilmiştir.



Şekil 2.1. Parçacığın hareketi

Algoritma, (2.3) ve (2.4)'te verilen güncelleme denklemlerinin tekrarlı tatbikinden oluşur. Şekil 2.2'de, PSO algoritmasının temel adımları görülmektedir. İki *if*'li ifade (2.1) ve (2.2) tanımlarının sırayla tatbikinin eşdeğeridir. Algoritmanın ilk adımında ifade edilen başlatma, aşağıda verilen adımlardan oluşur:

1. Her $x_{i,j}$ koordinatını tüm $i \in 1 \ldots s$ ve $j \in 1 \ldots n$ için $[-x_{maks}, x_{maks}]$ aralığında düzenli rassal dağılımdan türetilmiş bir değere ata (Parçacıkların başlangıç pozisyonları arama uzayı boyunca dağıtılır).

2. Her $v_{i,j}$'yi $i \in 1...s$ ve $j \in 1...n$ için [$-v_{maks}$, v_{maks}] aralığında düzenli rassal dağılımdan türetilmiş bir değere ata (Alternatif olarak parçacıkların hızları 0'a atanabilir çünkü başlangıç pozisyonları zaten gelişigüzel atanmıştır).
3. $y_i = x_i \ \forall i \in 1...s$ olarak ata (Alternatif olarak her parçacık için iki rassal vektör üretilebilir ve en uygun vektör y_i'ye daha az uygun olanı da x_i'ye atanabilir. Bu, ek fonksiyon değerlendirmesi gerektirir, bu yüzden genellikle ilk önce açıklanan daha basit yöntem kullanılır).

$x_{i,j}$ ve $v_{i,j}$'ye gelişigüzel başlangıç değerleri vererek sürüyü oluştur
Do
  For i = 1 **to** Parçacık sayısı
  if $f(x_i) < f(y_i)$ **then** $y_i = x_i$ // lokal (kişisel) en iyiyi güncelle
  $\hat{y}_i = \min(x_{komşular})$ // global en iyiyi güncelle
    For j = 1 **to** Optimize edilen boyut sayısı
      $v_{i,j} = v_{i,j} + c_1 r_{1,j}\left[y_{i,j} - x_{i,j}\right] + c_2 r_{2,j}\left[\hat{y}_j - x_{i,j}\right]$ // hız vektörünü güncelle
      $x_{i,j} = x_{i,j} + v_{i,j}$ // pozisyon vektörünü güncelle
    Next j
  Next i
Until Sonlandırma kriteri

Şekil 2.2. Orijinal PSO algoritmasının temel adımları [56]

Şekil 2.2'de ifade edilen sonlandırma kriteri çözülen problem tipine bağlıdır. Genellikle, algoritma sabit sayıda fonksiyon değerlendirmesine (sabit iterasyon sayısına) kadar ya da belirlenen bir hata sınırına ulaşıncaya kadar çalıştırılır. Literatürde on üç çeşit sonlandırma kriteri aşağıda isimleri verilen altı sınıfta toplanabilir [65].

1. **Referans kriteri**: Sürünün belirli bir yüzdesi bir optimuma yakınsamışsa sonlandırılır.
2. **Tükenme tabanlı kriter**: Belirli nesil sayısı, amaç fonksiyon değerlendirme sayısı ya da merkezi işlem birimi zamanı sonlandırma için kriter sayılır.
3. **İyileşme tabanlı kriter**: Eğer küçük ilerlemeler elde ediliyorsa algoritma sonlandırılabilir. Bunun da farklı versiyonları vardır [65, 66]:
    a. En iyi amaç fonksiyon değeri, belli nesil sayısı boyunca belirli bir eşiğin altındaysa sonlandırılır.

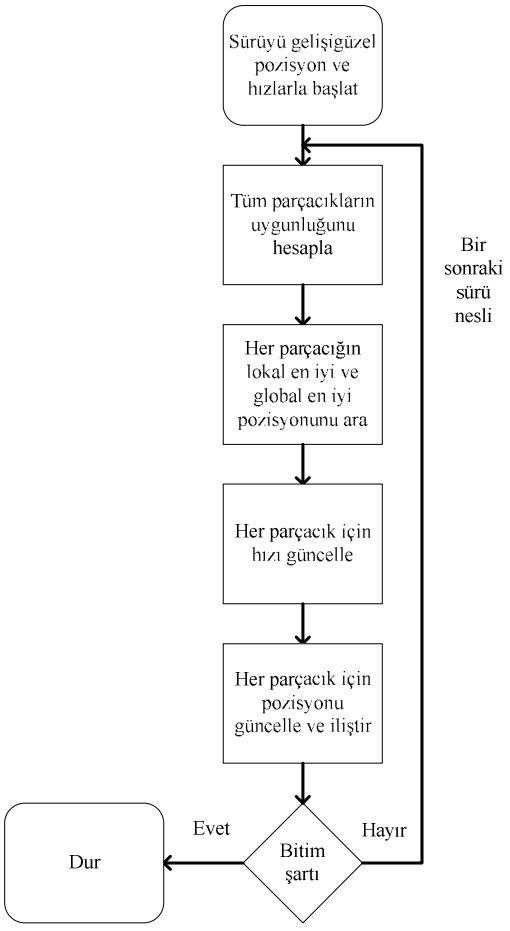
b. Ortalama amaç fonksiyon, değeri belli nesil sayısı boyunca belirli bir eşiğin altındaysa sonlandırılır.
c. Belirli bir nesil boyunca herhangi bir komşulukta bir ilerleme olmuyorsa sonlandırılır.

4. **Hareket tabanlı kriter**: Bireylerin hareketine bağlı sonlandırma kriteridir ve iki versiyonu bulunmaktadır [65, 66]:
    a. Ortalama amaç fonksiyon değerine bağlı olarak (amaç uzayı) popülasyondaki hareket, belirli nesil sayısı için belirli bir eşiğin altındaysa sonlandırılır.
    b. Pozisyonlara bağlı olarak (parametre uzayı) popülasyondaki hareket, belirli nesil sayısı için belirli bir eşiğin altındaysa sonlandırılır.
5. **Dağılım tabanlı kriter**: Bireyler birbirine çok yakınsa yakınsamaya ulaşılmıştır. Bu kriterin de, ilk üçü parametre uzayında sonuncusu da amaç uzayında uygulanan dört versiyonu bulunmaktadır [65, 66]:
    a. Her vektörden en iyi popülasyon vektörüne olan uzaklık, belirli bir eşiğin altındaysa sonlandırılır.
    b. Popülasyonun en iyilerinin belirli bir yüzdesinin en iyi popülasyon vektörüne olan uzaklığının, belirli bir eşiğin altında olup olmadığı test edilir. Bir önceki versiyon tüm popülasyon üyelerinin yakınsamasını bekler. Ancak optimum değer zaten bulunmuşsa fazladan hesapsal işlem yapılacaktır. Bu yöntem diğerine göre daha iyi olabilir [65, 66].
    c. Vektörlerin standart sapması belirli bir eşiğin altındaysa sonlandırılır.
    d. En iyi ve en kötü amaç fonksiyon değeri arasındaki fark belirli bir eşiğin altındaysa sonlandırılır.
6. **Birleşik kriter**: Bu kriter de iki şekilde uygulanabilir [65, 66]:
    a. Eğer belirli bir nesil boyunca ortalama iyileşme, belirli bir eşiğin altındaysa maksimum uzaklığın belirli eşiğin altında olup olmadığı test edilir.
    b. En iyi ve en kötü amaç fonksiyon değerleri arasındaki fark, belirli bir eşiğin altındaysa en iyi bireylerin belirli bir yüzdesinin en iyi çözüm olan maksimum uzaklığının belirli bir değerin altında olup olmadığı test edilir (en az belirli bir yüzdedeki bireyler uygun bölgede olmalıdır).

Algoritmanın çalışması özet olarak şöyledir: Başlangıçta, uygunluğa bağlı olarak bir parçacık, parçacıkların bir komşuluğunda en iyi parçacık olarak tanımlanır. Sonra tüm parçacıklar bu parçacığın yönünde ve daha önce keşfettikleri kendi en iyi çözümlerinin yönünde hızlanır. Bazen parçacıklar mevcut en iyi parçacığın ötesinde arama uzayını araştırarak

hedeflerinin dışına çıkacaklardır. Diğer parçacıkların yönleri değiştirdiği ve yeni 'en iyi' parçacığın yönüne gittiği durumda, tüm parçacıkların yoldayken daha iyi parçacıkları keşfetme olanakları da vardır. Arama uzayında farklı yönlerden mevcut en iyi çözüme yaklaşması sayesinde, bu komşu çözümlerin bazı parçacıklar tarafından keşfedilme şansı yüksektir.

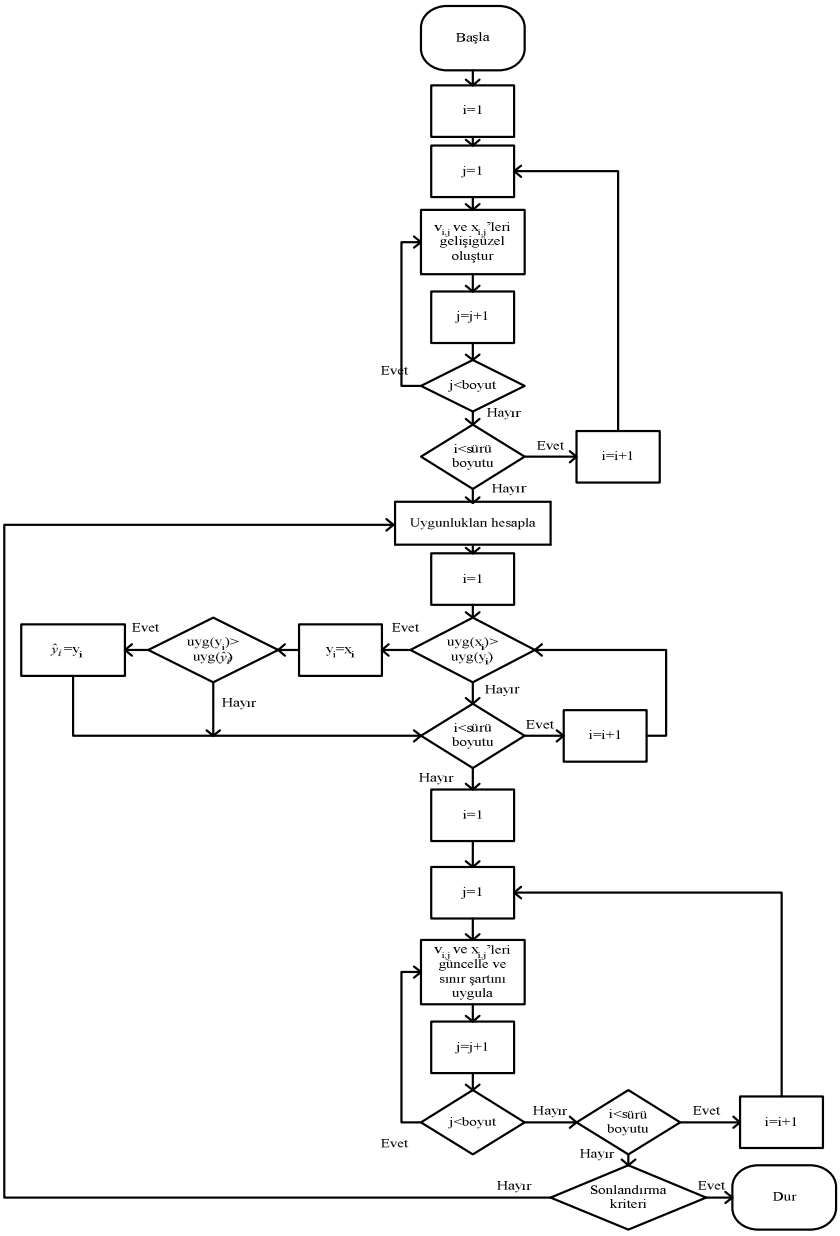
PSO'nun temel adımlarını gösteren akış diyagramı Şekil 2.3'te, bu akış diyagramının genişletilmiş versiyonu ise Şekil 2.4'te görülmektedir. Burada, *i* parçacık indisini; *j* boyut indisini; *boyut* problem boyutunu; *sürü boyutu* ise parçacık sayısını temsil etmektedir. *Sınır şartı uygula* kısmında boyutta sınırlama aşıldığında, yani minimum ve maksimum değerler dışına çıkıldığında sınırlandırılan aralığa dönmek için gereken işlemler yapılır.



Şekil 2.3. PSO'nun akış diyagramı

Hız güncelleme denklemi (2.3)'teki ikinci terim, *bilişle* ilişkilendirilmiştir. Yani sadece parçacığın kendi deneyimlerini hesaba katar. Üçüncü terim ise parçacıklar arasındaki *sosyal* etkileşimi temsil eder. Özet olarak, PSO hız güncelleme terimi bir bilişsel ve bir sosyal terimden oluşur. Bazı problemlerde ilk sonuçlar sosyal bileşenin daha önemli olabileceğini gösteriyor

olsa da, bu iki terimin bağıl önemi hakkında fazla bilgi yoktur. Parçacıklar arasındaki sosyal etkileşim işbirliğinin bir göstergesidir.

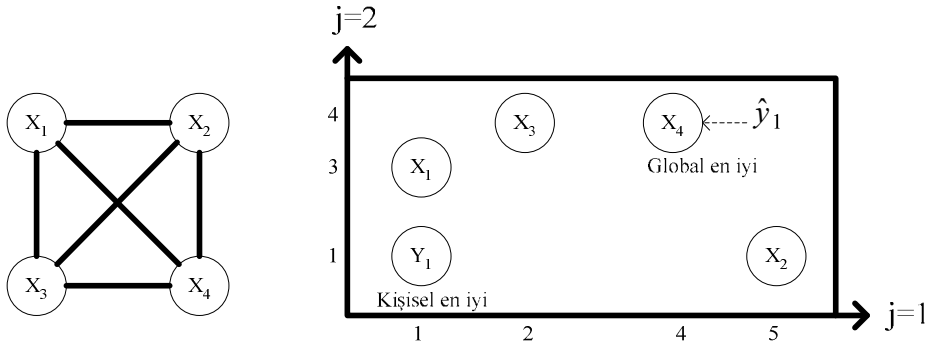


Şekil 2.4. PSO'nun genişletilmiş akış diyagramı

2.2.1.1. Orijinal PSO Algoritmasında Bir Parçacığın Hareketinin Sayısal Örneği

Sadece iki değişkenli basit bir problem ele alınırsa x_i parçacığı iki reel sayıdan oluşan bir vektör olarak temsil edilir. Yani $X_i = \langle x_{i1}, x_{i2} \rangle$ olur. Parçacık komşuluk yapısının *global*

model şeklinde olduğu ve bu komşuluk yapısının algoritma çalışması boyunca sabit olduğu varsayılacaktır. Şekil 2.5'te, dört parçacıktan oluşan sürünün *global* komşuluk yapısı ve arama uzayında parçacıkların pozisyonları görülmektedir. Pozisyonlar için *x* ekseni birinci boyut (*j*=1), *y* ekseni de ikinci boyut (*j*=2) olarak seçilmiştir.



Şekil 2.5. Komşuluk yapısı ve arama uzayında parçacıkların *t* anında pozisyonları

Sadece birinci parçacık (X_1) göz önüne alınıp bir adım sonra nasıl hareket edeceğine bakılırsa parçacığın hızı (2.3) denklemine göre güncellenecek ve sonra (2.4) ile bir sonraki pozisyona gelecektir. $c_1 = c_2 = 2$ seçildiği varsayılırsa;

$j = 1$ için $v_{1,1}(t) = 1$; $r_{1,1}(t) = 0.4$; $r_{2,1}(t) = 0.1$
$j = 2$ için $v_{1,2}(t) = 2$; $r_{1,2}(t) = 0.25$; $r_{2,2}(t) = 0.25$

j=1 boyutunda

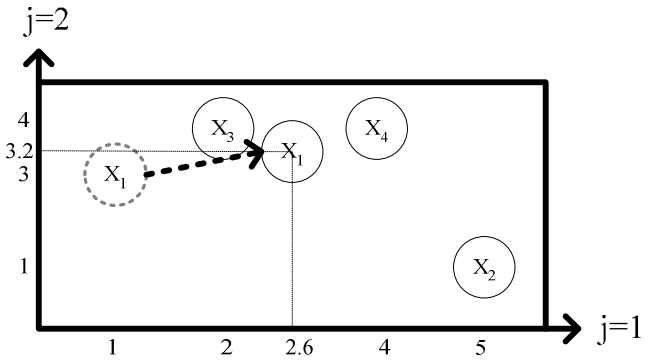
$$v_{1,1}(t+1) = 1 + 2 \times 0.4 \times [1-1] + 2 \times 0.1 \times [4-1] = 1.6$$
$$x_{1,1}(t+1) = 1 + 1.6 = 2.6$$

j=2 boyutunda

$$v_{1,2}(t+1) = 2 + 2 \times 0.25 \times [1-3] + 2 \times 0.1 \times [4-3] = 0.2$$
$$x_{1,2}(t+1) = 3 + 0.2 = 3.2$$

olur. Böylece X_1 parçacığı PSO güncelleme denklemlerinden sonra yeni (2.6, 3.2) pozisyonuna hareket edecektir. Bu durumda X_1'in yeni pozisyonu Şekil 2.6'da gösterildiği gibi olur. Bazı boyutlarda sınır aşma problemi ile karşılaşılınca X_1'in hızı için bazı sınırlamalar getirilebilir ve

boyut sınırlarında kalması sağlanabilir. Bu benzer güncelleme denklemleri, PSO'nun her iterasyonunda her parçacık için uygulanır.



Şekil 2.6. t+1 anında X_1 parçacığının pozisyonunun güncellenmesi

2.2.1.2. PSO ve Evrimsel Hesaplama

PSO'nun evrimsel hesaplama mantığına benzeyen bazı özellikleri vardır. Örneğin, PSO potansiyel çözümleri temsil eden bireylerin bir popülasyonunu devam ettirir ve bu da tüm evrimsel hesaplama mantığının ortak bir özelliğidir. Her iki yöntem de popülasyonu değerlendirmek için uygunluk değerlerine sahiptir. Her ikisi de popülasyonu güncellemeyi ve optimumu aramayı rastsal yöntemlerle yapar. Her iki sistem de başarıyı garanti etmez.

Eğer kişisel en iyi pozisyonlar (y_i) popülasyonun bir parçası olarak ele alınırsa, seçimin açık olarak zayıf bir formu oluşur. Bir ($\mu + \lambda$) evrim stratejileri algoritmasında çocuklar ataları ile yarışır ve eğer daha uygun iseler onların yerini alırlar. (2.1)'deki güncelleme denklemi bu mekanizmaya benzemektedir; buradaki fark her kişisel en iyi pozisyon (ata), mevcut pozisyon eski kişisel en iyi pozisyonundan daha uygun olursa, sadece kendi kişisel en iyi pozisyonu (çocuk)'nun yerini alabilir. Özetle, PSO'da seçimin zayıf bir şekli söz konusudur.

Hız güncelleme denklemi, gerçek-değerli GA'lardaki aritmetik çaprazlamaya benzemektedir. Normalde, aritmetik çaprazlama, içerilen ataların lineer karışımı olan iki çocuk üretir. $v_{i,j}(t)$ terimi olmayan (2.3)'teki PSO hız güncelleme denklemi iki ata içeren ve tek bir çocuk üreten aritmetik çaprazlamanın bir formu olarak düşünülebilir. Alternatif olarak, $v_{i,j}(t)$ terimi olmayan hız güncelleme denklemi bir mutasyon operatörü olarak da düşünülebilir.

$v_{i,j}(t)$ terimini modellemenin daha iyi bir yolu, her iterasyonu bir önceki popülasyonu yenisiyle değiştirme (ölüm ve doğum) olarak düşünmektense, bir adaptasyon süreci olarak düşünmektir. Bu şekilde x_i değerleri değiştirilmez hız vektörleri v_i kullanılarak adapte edilir. Bu, diğer evrimsel hesaplama algoritmaları ve PSO arasındaki farkı daha açık yapar. PSO pozisyon

ve hızla (pozisyondaki değişim) ilgili bilgiyi korur; tersine geleneksel evrimsel algoritmalar sadece pozisyonların izini saklar.

Böylece PSO ve çoğu evrimsel hesaplama algoritmaları arasında bir örtüşme olduğu görülmektedir, fakat PSO mevcut haliyle diğer evrimsel hesaplama algoritmalarda mevcut olmayan bazı karakteristiklere sahiptir. Özellikle, PSO parçacıkların pozisyonlarını modellediği gibi hızlarını da modeller. PSO'da parçacıklar kendilerini dâhili hızla güncellerler ve algoritma için önemli olan hafızaya sahiptirler. GA'larla karşılaştırıldığında, PSO'daki bilgi paylaşım mekanizması oldukça farklıdır. GA'larda kromozomlar bilgiyi birbirleriyle paylaşır. Böylece tüm popülasyon bir optimum bölgeye doğru tek bir grup olarak hareket eder. PSO'da, sadece *global en iyi* (ya da *kişisel en iyi*) bilgiyi diğerlerine dağıtır. Bu tek yönlü bir bilgi paylaşım mekanizmasıdır. GA'larla karşılaştırıldığında tüm parçacıklar, çoğu durumlarda lokal versiyonda bile hızlıca en iyi çözüme doğru yakınsama eğilimindedir.

GA'lara göre PSO'nun uygulaması daha basittir ve ayarlanması gereken daha az parametresi vardır. En iyi parametre kümesinin belirlenmesi hem zor hem de fazla zaman alabilmektedir. Ayrıca bir sonraki neslin belirlenmesi işleminde, GA'lar seçim stratejisinin belirlenmesi ve fazla hafıza gerektirirken, PSO pozisyon ve hız vektörlerini güncelleştirir ve strateji belirlemeye ihtiyaç duymaz.

2.2.1.3. PSO'nun Denklem Köklerinin Bulunmasında Kullanılması

PSO'nun adım adım çalışmasını görebilmek için basitçe bir denklemin köklerinin bulunması örnek olarak verilecektir. Bu tür problemler için PSO kullanılması çok zaman alır. Ancak PSO'nun adım adım çalışmasını görebilmek için bu basit örnek seçilmiştir. Denklemimiz f(x) = 3x-6 = 0, ve $x \in [0, 10]$ aralığında olsun. Amacımız, bu aralıkta denklemimizin kökünü bulmak olsun. Parçacık sayımızın dört olduğunu varsayalım ve $c_1 = c_2 = 1.3$ için bu tek boyutlu problemde PSO'nun iki iterasyon boyunca nasıl hareket edeceğini görelim. Denklemdeki x değerleri doğrudan parçacıkların pozisyon değerleridir. Uygunluk değeri olarak Uygunluk(x) = $1/|3x-6|$ seçilmiştir.

Öncelikle hız ve pozisyon değerleri gelişigüzel üretilir. Pozisyon değeri için [0, 1] aralığında üretilen gelişigüzel sayı 10 ile çarpılsın ve hız için ise [0, 1] aralığında değerler üretilsin. Buna göre başlangıçtaki hız ve pozisyon değerleri, ilk kişisel en iyi değerler, mevcut uygunluk, kişisel en iyi uygunluk değerleri Tablo 2.1'de verilmiştir.

Tablo 2.1. Sürüdeki ilk parametrik değerler

Parçacık hızı	Parçacık pozisyonu	Kişisel en iyi	Mevcut uygunluk	Kişisel en iyi uygunluk
0.4692	2.8441	2.8441	0.3949	0.3949
0.9883	0.6478	0.6478	0.2465	0.2465
0.4235	5.8279	5.8279	0.0871	0.0871
0.3340	5.1551	5.1551	0.1056	0.1056

Burada dikkat edilirse global en iyi değer 0.3949 ve bunun da indeksi 1 olarak görülmektedir. Yani 2.8441 değerine sahip parçacık global en iyidir. Birinci iterasyon sonrası parçacıkların hız ve pozisyon güncellemeleri Tablo 2.2'de görülmektedir.

Tablo 2.2. Birinci iterasyon sonrasında değerler

Parçacık hızı	Parçacık pozisyonu	Kişisel en iyi	Mevcut uygunluk	Kişisel en iyi uygunluk
0.4692	3.3133	2.8441	0.2538	0.3949
4.2956	1.1170	1.1170	0.3775	0.3775
-4.0705	6.2971	5.8279	0.0776	0.0871
-1.3000	5.6243	5.1551	0.0920	0.1056

Birinci iterasyon sonucu global en iyi değerin 0.3775 olduğu ve bunun indeksinin 2 olduğu yani 1.1170 değerine sahip parçacığın global en iyi olduğu görülmektedir. İkinci iterasyon sonucu güncellemelerle elde edilen yeni hız ve pozisyon değerleri ise Tablo 2.3'te verilmiştir.

Tablo 2.3. İkinci iterasyon sonrasında değerler

Parçacık hızı	Parçacık pozisyonu	Kişisel en iyi	Mevcut uygunluk	Kişisel en iyi uygunluk
-1.9525	1.3608	1.3608	0.5215	0.5215
-0.9173	-0.8355	1.1170	0.1176	0.3775
-5.9163	4.3446	4.3446	0.1422	0.1422
-4.4260	3.6718	3.6718	0.1994	0.1994

Burada global en iyinin 0.5215 olduğu görülmektedir. Bin iterasyon sonucu elde edilen sonuçlar ise Tablo 2.4'te verilmiştir.

Tablo 2.4. Bin iterasyon sonrasında değerler

Parçacık hızı	Parçacık pozisyonu	Mevcut uygunluk
1.0e+003 * -4.6795	2.0080	41.6667
1.0e+003 * -4.6748	2.0020	166.6667
1.0e+003 * -4.6812	2.0176	18.9394
1.0e+003 * -4.6798	2.0077	43.2900

Burada global en iyi 2.0020 değeridir. Denklemin gerçek kökü de 2.0'dır.

2.2.2. PSO'ya Modifikasyonlar

PSO'ya literatürde birçok ekleme, değişiklik ve geliştirmeler önerilmiştir. Önce PSO'nun ikili versiyonu anlatılacak ve daha sonra geliştirmeler verilecektir.

2.2.2.1. İkili PSO

PSO'nun ikili versiyonu Kennedy ve Eberhart tarafından geliştirilmiştir [67]. İkili kodlu GA ve PSO ile ilgili karşılaştırmalar yapabilmek için kullanışlı olduğu gibi, doğasında ikili olan problemleri temsil için de kullanışlıdır. Tipik bir uygulama, ağda iki düğüm arasında 1'in bağlantı durumunu, 0'ın ise bağlantının olmadığı durumu temsil ettiği sinirsel ağın bağlantılarını temsil etmek olabilir. Daha sonra ikili PSO ağ mimarisini geliştirmek için kullanılabilir.

İkili versiyon x_i ve y_i bileşen değerlerini {0, 1} kümesindeki elemanlardan alınacak şekilde kısıtlar. Ancak hızın, v_i, değeri üzerinde herhangi bir kısıtlama yoktur. Hızı, pozisyonları güncellemek için kullanırken, hız değeri de [0, 1] aralığına sınırlandırılır ve olasılık olarak ele alınır. Bu, sigmoid fonksiyonu kullanılarak

$$\mathrm{sig}(x) = \frac{1}{1+\exp(-x)} \tag{2.10}$$

şeklinde yapılır. İkili sürüde kullanılan hız terimi için güncelleme

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j} - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j - x_{i,j}(t)\right] \tag{2.11}$$

olur. Bu hız güncelleme denkleminin orijinal PSO'da kullanılandan farklı olmadığı görülmektedir. Genel pozisyon güncelleme denklemi yerine yeni bir olasılıksal güncelleme denklemi kullanılır, yani:

$$x_{i,j}(t+1) = \begin{cases} 0 & if \quad r_{3,j}(t) \geq \text{sig}\left(v_{i,j}(t+1)\right) \\ 1 & if \quad r_{3,j}(t) < \text{sig}\left(v_{i,j}(t+1)\right) \end{cases} \tag{2.12}$$

olur. Burada $r_{3,j}(t) \sim U(0, 1)$ düzenli olasılıksal bir değişkendir. Bu denkleme bakılırsa, $x_{i,j}$'nin değerinin $\text{sig}(v_{i,j}) = 0$ olduğunda 0 olarak kalacağı açıktır. Bu durum $v_{i,j}$ yaklaşık olarak -10'dan küçük olduğu zaman ortaya çıkacaktır. Aynı şekilde, sigmoid fonksiyonu $v_{i,j}>10$ olduğunda doyacaktır. Bunu önlemek için, $v_{i,j}$'nin değerinin ± 4 sınırında tutulması önerilmiştir [68]. Bu, $\text{sig}(4) \approx 0.018$ durum değiştirme olasılığı ile sonuçlanır. İkili PSO'yu açıklayan orijinal makale v_{maks} için daha büyük eşik, ± 6, önermiştir ve bu da yaklaşık olarak 0.0025 olasılığı ile sonuçlanır [67].

Daha sonraki bir çalışma, ikili PSO'yu çok modlu test fonksiyonunda GA ile karşılaştırmak için kullanmıştır [69]. Özelikle problem boyutu arttığında çoğu test fonksiyonunda, ikili PSO'nun GA'lara göre daha hızlı çözüme ulaştığı [69]'da gösterilmiştir.

Aynı vektörde hem ikili hem de sürekli değerleri kullanmak mümkündür. Bu versiyon *melez sürü* [68] olarak adlandırılmıştır. Aynı zamanda *ikili-sürekli sürü* olarak da kullanılmaktadır. Son zamanlardaki çalışmalar, PSO'nun basitçe ilgili nicelikleri gerekli olduğunda ayrıklaştırarak gelişigüzel ayrık temsilleri de içermesi için kabiliyetlerini genişletmiştir.

2.2.2.2. Yakınsama Oranı Geliştirmeleri

PSO'nun yakınsama oranını geliştirmek için literatürde birkaç yöntem önerilmiştir. Bu öneriler genellikle, algoritmanın yapısını değiştirmeksizin PSO güncelleme denklemlerinin değiştirilmesini içerirler. Bu, genellikle daha iyi bir lokal optimizasyon performansıyla bazen de çoklu lokal minimum içeren fonksiyonlarda performans düşüklüğüyle sonuçlanır.

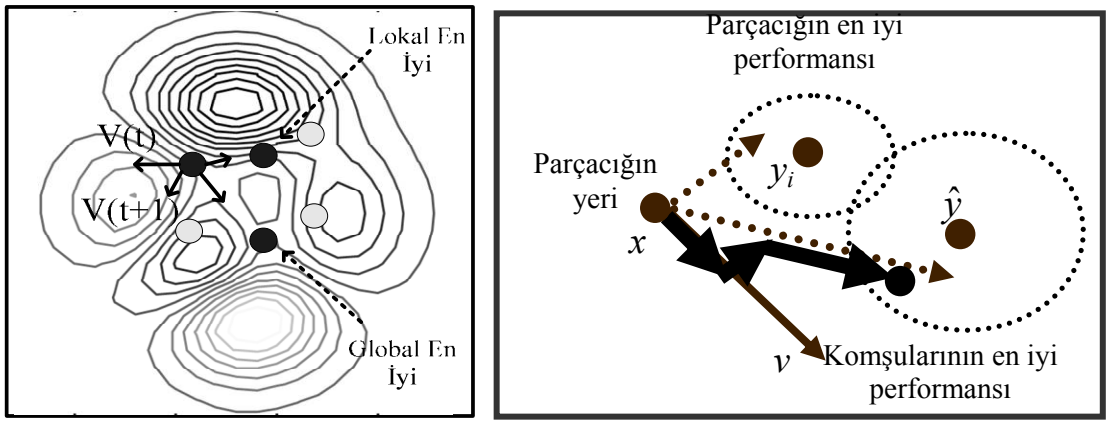
Atalet Ağırlığı

Günümüzde en yaygın kullanılan geliştirme Shi ve Eberhart [70] tarafından önerilen *atalet ağırlığı*dır. Atalet ağırlığı, bir önceki zaman adımı boyunca hızla ilişkilendirilmiş bir ölçekleme faktörüdür. Yeni hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (2.13)$$

olur. Orijinal PSO hız güncelleme denklemi $w = 1$ seçilerek elde edilebilir. Shi ve Eberhart w değerinin [0, 1.4] aralığındaki etkilerini ve zamanla değişen değerlerini incelemiştir. Bunların sonuçları, $w \in [0.8, 1.2]$ olarak seçmenin daha hızlı yakınsamayla sonuçlandığını ama daha büyük w değerlerinin (>1.2) daha fazla hata ile yakınsadığını göstermektedir.

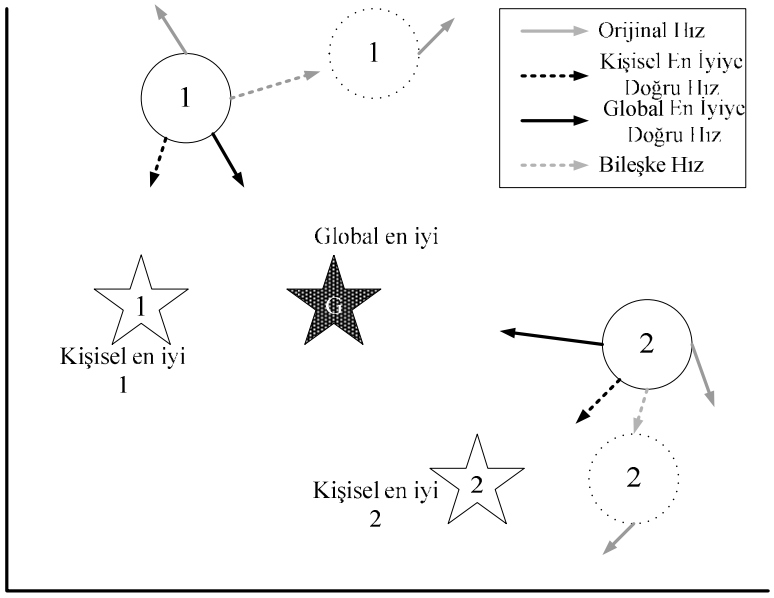
Şekil 2.7'de, bir parçacık için üç olası hareketin ağırlıklı kombinasyonu görülmektedir. Şekil 2.8 ise, sürüdeki parçacıkların hızlanmalarının iki boyutlu bir örneğidir.



Şekil 2.7. Üç olası hareketin ağırlıklı kombinasyonu

Üç sosyal / bilişsel katsayı sırayla parçacığın şu an kendisine, deneyimlerine ve komşularına ne kadar güvendiğini ölçer. Sosyal / bilişsel katsayılar her zaman adımında, verilen aralık içinde genellikle gelişigüzel seçilir.

Atalet ağırlığı, bir önceki hızın ne kadarının bir sonraki zaman adımında tutulacağını yönetir. w'nin etkisi kısaca şu şekilde özetlenebilir. $c_1 = c_2 = 0$ olarak seçildiğinde ve başlangıç hız değerinin sıfır olmadığı varsayıldığında 1.0'dan büyük bir w değeri, parçacığın değişmeyip olduğu gibi kalacağı maksimum hıza, v_{maks} (ya da $-v_{maks}$), ivmelenmesine neden olacaktır. 1.0'dan daha küçük w değeri parçacığın hızı sıfıra ulaşıncaya kadar yavaşlamasına neden olacaktır. $c_1, c_2 \neq 0$ olduğunda algoritmanın davranışını tahmin etmek daha zor olur, ama Shi ve Eberhart'ın sonuçlarına bağlı olarak w değerlerinin 1.0'a yakın değerleri tercih edilir [70].



Şekil 2.8. Parçacıkların hızlanmalarının iki boyutlu bir örneği

v_{maks} ile atalet ağırlığı arasındaki etkileşimi incelemek için de deneyler yapılmıştır [71]. Bu deneyde çalışılan tek fonksiyon için, 0.8 değerindeki bir atalet ağırlığının $v_{maks} = x_{maks}$ olduğu durumlarda bile iyi sonuçlar verdiği görülmüştür.

Bundan başka, dört farklı amaç fonksiyonu kullanarak atalet ağırlığının 0.9'dan 0.4'e lineer olarak azaltan deneyler yapılmıştır ve değişim

$$w = (w_1 - w_2) \times \frac{(maksiter - iter)}{maksiter} + w_2 \tag{2.14}$$

ile gösterilmiştir. Burada w_1 ve w_2 sırasıyla atalet ağırlığının başlangıç ve son değerleri; *iter* şu anki iterasyon sayısı ve *maksiter* de izin verilen maksimum iterasyon sayısıdır.

Bu ayarlar, PSO'nun simülasyonun başlangıcında (atalet ağırlığı büyük olduğunda) geniş bir alanı araştırmasına ve sonra daha küçük bir atalet ağırlığıyla aramayı inceltmesine izin verir. Atalet ağırlığı ısıl işlemde [72] karşılaşılan sıcaklık parametresine benzetilebilir. Isıl işlem algoritması sistemin sıcaklığını kademeli olarak azaltmak için kullanılan sıcaklık programı adı verilen bir işleme sahiptir. Sıcaklık ne kadar fazla olursa, algoritmanın da mevcut lokal minimumun çekim alanının o kadar dışındaki bir bölgeyi tarama olasılığı yüksektir. Bu yüzden uyarlamalı bir atalet ağırlığı, ısıl işlem algoritmasındaki sıcaklık programına eşdeğer olarak görülebilir.

Kısaca, PSO'da atalet ağırlığı global ve lokal arama yeteneğini dengelemek için kullanılır. Büyük atalet ağırlığı global arama, küçük ağırlık ise lokal arama yapılmasını kolaylaştırır. Atalet ağırlığı, lokal ve global arama arasındaki dengeyi sağlar ve bunun sonucunda yeterli optimal sonuca daha az iterasyonla ulaşılır. Buradaki her parçacık, sürüdeki sadece en iyi parçacığın değil sürüdeki diğer tüm parçacıkların tecrübelerinden de yararlanmış olur.

Zheng ve arkadaşları ise zamanla artan atalet ağırlığı kullanmış ve bazı durumlarda daha iyi sonuçlar alındığını belirtmişlerdir [73, 74]. Bunlar da Denklem (2.14)'ü kullanmışlardır sadece w_1 ve w_2'yi yer değiştirmişlerdir.

Zangh ve arkadaşları ise atalet ağırlığını [0, 1] aralığında gelişigüzel üretilen sayılarla çarparak kullanmışlardır [75]. Lineer azaltmalı atalet ağırlığının ilk değerinin maksimum iterasyon sayısına bağlı olması ve bunun da deney yapılmadan kestirilememesi, ayrıca algoritmanın ilk safhalarında lokal arama kabiliyeti eksikliği ve son safhalarında da global arama kabiliyeti eksikliğinden dolayı böyle bir yöntem önermişlerdir. Bu durumda güncelleme denklemi

$$v_{i,j}(t+1) = r_{0ij} v_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \tag{2.15}$$

olur; burada $r_{0,j} \sim U(0,1)$'dir. Yapılan deneylerde, lineer azaltmalı atalet ağırlığıyla performans karşılaştırması için sadece üç adet kalite test fonksiyonu kullanılmış ve ikisinde daha etkili sonuç alınırken diğerinde yakın sonuçlar alınmıştır.

Bulanık Atalet Ağırlığı

Shi ve Eberhart, atalet ağırlığını dinamik olarak adapte etmek için bir bulanık denetleyici önermiştir [76, 77]. Önerilen denetleyici giriş olarak mevcut atalet ağırlığını ve o ana kadar bulunan en iyi çözüme, f($\hat{y}$), tekabül eden fonksiyon değerini kullanır. Çoğu problem farklı ölçeklerde fonksiyon değerlerine sahip olduğundan f($\hat{y}$) değeri normalize edilmelidir. Denklem (2.16) fonksiyon değerlerini ölçeklemek için olası bir yöntemdir:

$$f_{norm}(\hat{y}) = \frac{f(\hat{y}) - f_{\min}}{f_{\max} - f_{\min}} \tag{2.16}$$

f_{maks} ve f_{min} değerleri problem bağımlıdır ve önceden bilinmelidir, ya da bazı tahminlerin mümkün olması gerekir.

Shi ve Eberhart, giriş değerlerinin ait olabileceği üç bulanık kümeye (*düşük*, *orta*, *yüksek*) karşılık üç bulanık üyelik fonksiyonu kullanmayı seçmişlerdir. Bulanık denetleyicini çıkışı ise atalet ağırlığının değerinde önerilen değişimdir [77].

Bulanık uyarlamalı atalet ağırlıklı PSO lineer azalan atalet ağırlığı kullanan PSO ile karşılaştırılmıştır. Sonuçlardan bulanık atalet ağırlıklı PSO'nun belli parametre ayarları için, test edilen bazı fonksiyonlarda daha iyi performans verdiği görülmüştür [76]. Aynı zamanda tek modlu fonksiyonlarda daha büyük performans elde edilmiştir. Bu durum şu şekilde açıklanmıştır: Tek modlu fonksiyonların lokal minimumu yoktur, bu yüzden atalet ağırlığı her iterasyonda belirlenebilir. Mantıksal olarak, algoritmanın çalışmasının başlangıcında büyük atalet ağırlığı PSO'nun minimizasyonun daha hızlı bulunduğu yaklaşık alana yerleşmesine izin verir. Bu alana ulaştıktan sonra atalet ağırlığı parçacığın hızını azaltmak için kademeli olarak azaltılmalıdır. Bu parçacıkların fonksiyonun yüzeyinde daha küçük özelliklere yerleşmesine izin verir. Bu işlem zamanla atalet ağırlığı azaltılarak tahmin edilebilir, ancak bu mekanizma PSO'nun daha küçük atalet ağırlığın daha da kullanılması gereken alana yerleşip yerleşmediğini bilmez. Bazen PSO bu alana ulaşmak için çok zaman alır, bazen de onu çok hızlıca bulur. Uyarlamalı bulanık denetleyici hangi tip davranışın daha uygun olduğunu yaklaşık olarak tahmin edebilir. Bulanık denetleyicideki kurallar etkili olarak atalet ağırlığını, PSO minimuma ne kadar yakınsa yani $f_{norm}(\hat{y})$ sıfıra ne kadar yakınsa, o oranda azaltır. Ancak çoklu lokal minimuma sahip olan bir fonksiyonla ilgilenirken optimum atalet ağırlığını bulmak daha zordur.

Uyarlamalı bulanık atalet ağırlık denetleyicisi atalet ağırlığını optimize etmek için umut verici bir yöntemdir ancak f_{maks} ve f_{min}'in bilinmesini gerektiren uygulama zorlukları genel biçimde kullanılmasını zorlaştırır.

Sınırlama Faktörü

Clerc tarafından yapılan bir çalışma *sınırlama faktörü*nün yakınsamayı sağlamaya yardım edebileceğini göstermiştir [78]. Sınırlama faktörü modeli w, c_1 ve c_2 değerlerinin yakınsamayı sağlayacak şekilde seçilmesini açıklar. Bu değerleri doğru seçerek $v_{i,j}$ değerlerinin $[-v_{maks}, v_{maks}]$ aralığında tutulması sağlanır. Sınırlandırma modelinin bir tanesine bağlı olarak değiştirilmiş hız güncelleme denklemi

$$v_{i,j}(t+1) = \chi\left(v_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right]\right) \tag{2.17}$$

olur. Burada

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{2.18}$$

ve $\varphi = c_1 + c_2$, $\varphi > 4$'tür. $c_1 = c_2 = 2.05$ olup toplanırsa $\varphi = c_1 + c_2 = 4.1$ ve $\chi = 0.7298$ olur. (2.17) denklemi t kaldırılırsa

$$v_{i,j}(t+1) = 0.7298\left(v_{i,j} + 2.05 r_{1,j}\left\lfloor y_{i,j} - x_{i,j}\right\rfloor + 2.05 r_{2,j}\left\lfloor \hat{y}_j - x_{i,j}\right\rfloor\right)$$

olur. $2.05 \times 0.7298 = 1.4962$ olduğundan bu (2.15)'te değiştirilmiş PSO hız güncelleme denkleminde $c_1 = c_2 = 1.4962$ ve $w = 0.7298$ kullanmayla eşdeğerdir.

Eberhart ve Shi yaptıkları çalışmada sınırlama faktörü kullanarak (hızı belli değerlerde tutmadan) daha hızlı yakınsama oranı elde etmişlerdir. Ancak bazı test fonksiyonlarında, sınırlama faktörlü PSO izin verilen iterasyon sayısında belirlenen hata değerine ulaşmada başarısız kalmıştır. Eberhart ve Shi'ye göre problem parçacıkların arama uzayının istenen bölgesinden çok uzaklara ayrılmasıdır [79]. Bu etkiyi hafifletmek için v_{maks} parametresini x_{maks} değerine ayarlayarak sınırlama faktörünü sınırda tutmayı uygulamışlardır. Bu test için kullanılan hemen hemen tüm fonksiyonlarda algoritma, yakınsama oranı ve hata eşiğine ulaşma açısından artan bir performans göstermiştir.

Seçim

Angeline evrimsel hesaplama alanından seçim fikrini alan yeni bir PSO önermiştir [80]. Angeline mevcut PSO'nun kişisel en iyi pozisyon ilave popülasyon üyeleri olarak düşünüldüğünde seçimin zayıf, örtük şekline sahip olduğunu söylemiştir. Karşılaştırma için Angeline tarafından kullanılan *global* modelde bir parçacığın sadece kişisel en iyi ve global en iyi parçacığa ulaşımı vardır. Bu, popülasyonun yarısındaki üyelerle (mevcut pozisyonlar) diğer yarısı (kişisel en iyi pozisyonlar) arasında muhtemel etkileşimlerin büyük oranda sınırlandığı anlamına gelir.

Evrimsel hesaplamadaki seçimin amacı genellikle arama uzayının yakın geçmişte umut verici sonuçlar veren özel bir bölgesine yönelmektir. Daha sonra bu bölge daha esaslı aranır. Angeline PSO'ya seçim eklemek için aşağıdaki yöntemi önermiştir:

1. Sürüden bir parçacık al. Bu parçacığın uygunluğunu sürüdeki diğer k parçacıkla karşılaştır ve bu bireyi her seferinde karşılaştırıldığı parçacıktan daha iyi uygunluğa sahipse bir işaret vererek ödüllendir. Bu işlemi her parçacık için tekrar et.

2. Parçacıkları bir önceki adımda toplanan işaretlere göre sırala.
3. Sürünün en üstteki yarısını seç ve bunların mevcut pozisyonlarını sürünün alt yarısının mevcut pozisyonlarına kopyala, kişisel en iyi değerlere bir işlem yapma.

Bu işlem PSO hız güncelleme denklemi yürütülmeden önce yapılır.

Angeline orijinal PSO ile (sınırlama faktörü ve atalet ağırlığı olmayan) seçimli PSO'yu karşılaştırmış ve değiştirilen PSO'nun tek modlu fonksiyonlarda ve Rastrigin fonksiyonunda orijinalinden önemli oranda daha iyi sonuçlar verdiğini görmüştür. Ancak Griewank fonksiyonunda performansı kötü çıkmıştır. Griewank fonksiyonunun birçok lokal minimumu vardır ve seçim mekanizması aslında lokal bir minimuma yakınsamayı desteklemiştir. Eğer bazı parçacıklar makul bir minimum keşfederse, sürünün diğer yarısı da aynı lokal minimumum alanına doğru hareket edebilir. Bu, algoritmanın arama uzayının büyük bölgelerini araştırma kabiliyetini etkiler ve böylece global minimumu bulmasını engeller.

Seçim böylece PSO'nun lokal arama kabiliyetini arttırır, ancak eşzamanlı olarak global arama kabiliyetini engeller.

Zamanla Değişen Hızlanma Katsayıları (ZDHK)

Ratnaweera ve Halgamuge [81] bilişsel bileşen c_1'i zamanla azaltan ve sosyal bileşen c_2'yi zamanla arttıran bir zamanla değişen hızlanma katsayısılı PSO önermiştir. Başlangıçta büyük c_1 değerleri ve küçük c_2 değerleri ile parçacıkların kişisel en iyiye doğru hareketlerinden ziyade arama uzayı boyunca hareket etmeleri istenmiştir. Optimizasyonun ileriki kısımlarında küçük c_1 değerleri ve büyük c_2 değerleri ise parçacıkların global optimuma doğru yakınsamalarını sağlamaktadır. ZDHK denklemleri

$$c_1 = \left(c_{1i} - c_{1f}\right) \times \left(\frac{maksiter - iter}{maksiter}\right) + c_{1f} \tag{2.19}$$

$$c_2 = \left(c_{2i} - c_{2f}\right) \times \left(\frac{maksiter - iter}{maksiter}\right) + c_{2f} \tag{2.20}$$

şeklindedir. Burada c_{1i} ve c_{2i}, c_1 ve c_2 katsayılarının başlangıç değerleri; c_{1f} ve c_{2f} de son değerleridir. c_1 ve c_2'nin değerlerinin en iyi aralığını bulmak için çeşitli kontrol problemleri ile simülasyonlar yapılmıştır ve c_1'in 2.5'ten 0.5'e ve c_2'nin 0.5'ten 2.5'e değiştiği aralıkta en iyi sonucun alındığı belirlenmiştir.

Başka bir çalışma da atalet ağırlığı ve hızlanma katsayılarını eş zamanlı uyarlayan ve [82]'de önerilen yöntemdir. Bu çalışma GLEnIyiPSO olarak adlandırılmıştır ve Denklem (2.21)

ve (2.22)'de gösterildiği gibi bu parametreleri parçacıkların global ve lokal en iyi pozisyonlarındaki terimlerini de içerecek şekilde günceller. Düzenlenen hız güncellemesi Denklem (2.23)'te gösterilmiştir.

$$w(t) = \left(1.1 - \frac{geniyi_t}{(peniyi_t)_{ortalama}} \right) \tag{2.21}$$

$$c(t) = \left(1 + \frac{geniyi_t}{peniyi_t} \right) \tag{2.22}$$

$$v_{i,j}(t+1) = w(t)v_{i,j}(t) + c_1 r_{1,j}(t)\left(peniyi_{i,j}(t) + geniyi_i(t) - 2x_{i,j}(t)\right) \tag{2.23}$$

Burada $peniyi_t$ parçacığın lokal en iyi değeri; $(peniyi_t)_{ortalama}$ parçacıkların lokal en iyi değerlerinin ortalaması; ve $geniyi_t$ de sürüdeki tüm lokal en iyi değer arasındaki en iyidir.

Yetiştirme

Angeline'nin çalışmasından sonra [80], LØvbjerg ve arkadaşları başka evrimsel hesaplama mekanizmalarını PSO algoritmasına uygulamışlardır [83]. GA tabiri olan *üreme* ve *yeniden birleşme*nin etkisini incelemeyi seçmişler ve *yetiştirme*den bahsetmişlerdir.

PSO'ya önerilen değişiklikler aşağıdaki gibi işler:

1. Yeni parçacığın hız ve pozisyonunu (2.3) ve (2.4)'teki gibi hesapla.
2. Her parçacığı P_b olasılığıyla (yetiştirme olasılığı) potansiyel ata olarak işaretle.
3. İşaretlenen parçacık havuzundan, iki aday seç ve denklem (2.24–2.27)'de detaylandırılan aritmetik çaprazlama operasyonu uygula ve orijinal atayla yer değişecek iki yeni çocuk oluştur.
4. Her parçacığın kişisel en iyi pozisyonlarını mevcut pozisyonlarına ayarla, örneğin $y_i=x_i$.

Ataların seçimi zayıf stokastik bir biçimde gerçekleşir, herhangi bir uygunluk-tabanlı seçim yapılmaz. Bu da çok sayıda lokal minimum içeren fonksiyonlarda uygunluk-tabanlı seçimle ilişkili potansiyel problemleri önler [83].

a ve *b* ata olarak seçilen iki parçacığın indisleri olsun. O zaman aritmetik çaprazlama şu şekilde işler:

$$x_a(t+1) = r_1 x_a(t) + (1.0 - r_1)x_b(t) \tag{2.24}$$

$$x_b(t+1) = r_1 x_b(t) + (1.0 - r_1) x_a(t) \tag{2.25}$$

$$v_a(t+1) = \frac{v_a(t) + v_b(t)}{\|v_a(t) + v_b(t)\|} \|v_a(t)\| \tag{2.26}$$

$$v_b(t+1) = \frac{v_a(t) + v_b(t)}{\|v_a(t) + v_b(t)\|} \|v_b(t)\| \tag{2.27}$$

Burada $r_1 \sim U(0, 1)$. İki pozisyonun aritmetik çaprazlanması ataların köşeleri oluşturduğu hiperküpte gelişigüzel iki yeni pozisyon üretir. Hız çaprazlaması sadece yönün etkilenmesini ve büyüklüğün etkilenmemesini sağlayacak şekilde iki atanın hızlarının toplam uzunluğunu normalize eder.

LØvbjerg tarafından sunulan sonuçlar, yetiştirmenin tek modlu fonksiyonlarda yakınsama hızını azalttığını ve böylece orijinal PSO'ya göre daha az etkili bir lokal optimizasyon yaptığını göstermiştir. Çoklu lokal minimuma sahip fonksiyonlarda ise etkili olduğu görülmüştür. *lokal* modelle ilgili bir karşılaştırma sunulmamıştır.

Yakınsama Garantili PSO (YGPSO)

Eğer bir parçacığın yörüngesi yakınsarsa, parçacık kişisel en iyi pozisyonu ve global en iyi parçacığın pozisyonu arasındaki yoldan türetilen bir değere doğru hareket edecektir [84]. Güncelleme denklemi (2.1)'den dolayı, parçacığın kişisel en iyi pozisyonu kademeli olarak global en iyi pozisyona doğru hareket eder ve en sonunda global en iyi parçacığın pozisyonuna yakınsar. Bu noktada, algoritma çözümü iyileştiremez çünkü parçacık hareket etmeyi durduracaktır. Algoritmanın gerçekten *f* fonksiyonunun minimumunu keşfedip keşfetmediği belli değildir, hatta parçacığın yakınsadığı pozisyonun lokal bir minimum olduğunun bile garantisi yoktur [84].

Atalet ağırlığı ve sınırlandırma faktörü içeren değişik PSO versiyonların hepsinin potansiyel bir tehlikesi vardır: eğer $x_i = y_i = \hat{y}$ ise hız güncelleme denklemi sadece $wv_{i,j}(t)$ değerine bağlı olacaktır. Diğer bir deyişle, eğer bir parçacığın mevcut pozisyonu global en iyi pozisyon / parçacık ile çakışırsa parçacık, eğer önceki hızı ve *w* sıfır değilse, sadece bu noktadan uzaklaşacaktır. Eğer önceki hızları sıfıra çok yakınsa, o zaman da tüm parçacıklar bir kez global en iyi parçacığı yakaladıklarında hareket etmeyi durduracaklardır ve bu da algoritmayı erken yakınsamaya götürebilecektir. Aslında bu, algoritmanın lokal bir minimuma

dahi yakınsadığını garanti etmez, sadece tüm parçacıların sürü tarafından şimdiye kadar keşfedilen en iyi pozisyona yakınsadığını gösterir [84].

Bunu önlemek için PSO'ya yeni bir terim eklenmiştir. τ global en iyi parçacığın indeksi olursa $y_\tau = \hat{y}$ 'dır. Global en iyi parçacık için yeni bir hız güncelleme denklemi önerilmiştir ve bu denklemi kullanan PSO, Yakınsama Garantili PSO (YGPSO) olarak adlandırılmıştır [84].

$$v_{\tau,j}(t+1) = -x_{\tau,j}(t) + \hat{y}_j(t) + wv_{\tau,j}(t) + \rho(t)(1 - 2r_{2,j}(t)) \tag{2.28}$$

ρ bir ölçekleme faktörüdür. Sürüdeki diğer parçacıklar her zamanki hız güncelleme denklemini kullanmaya devam eder. Kısaca, $-x_{\tau,j}(t)$ terimi parçacığın pozisyonunu $\hat{y}_j$ pozisyonuna "yeniden yerleştirir". Bu pozisyona, $wv_{\tau,j}(t)$ terimi ile temsil edilen mevcut arama yönünü temsil eden bir vektör eklenir. $\rho(t)(1 - 2r_{2,j}(t))$ terimi kenar uzunlukları $2\rho(t)$ olan bir örnek uzaydan gelişigüzel bir örnek üretir [84].

Global en iyi parçacık τ için hız güncelleme denklemi ve yeni hız güncelleme denklemi birleştirilirse yeni pozisyon güncelleme denklemi

$$x_{\tau,j}(t+1) = \hat{y}_j(t) + wv_{\tau,j}(t) + \rho(t)(1 - 2r_{2,j}(t)) \tag{2.29}$$

olur. ρ teriminin eklenmesi PSO'nun global en iyi pozisyon $\hat{y}$ civarındaki bir alanda gelişigüzel bir arama yapmasına neden olur. Bu arama alanının çapı ρ parametresiyle kontrol edilir. $\rho(y)$ her zaman adımından sonra

$$\rho(t+1) = \begin{cases} 2\rho(t) & \text{eger} \quad N_{basarilar} > s_c \\ 0.5\rho(t) & \text{eger} \quad N_{hatalar} > f_c \\ \rho(t) & \text{degilse} \end{cases} \tag{2.30}$$

denklemi ile uyarlanır. Burada $N_{hatalar}$ ve $N_{basarilar}$ terimleri sırasıyla ardışık hata ve başarıların sayısını göstermektedir ve bir hata da $f(\hat{y}(t)) = f(\hat{y}(t-1))$ olarak tanımlanır. Kabul edilebilir sonuçlar üretmek için $\rho(0) = 1.0$ varsayılan başlangıç değeri deneysel olarak bulunmuştur. s_c =

15 ve f_c = 5 seçilmiştir. Ayrıca dinamik olarak seçilebileceği belirtilmiştir. (2.30) denkleminin iyi-tanımlı olmasını garantilemek için

$$N_{basarilar}(t+1) > N_{basarilar}(t) \Rightarrow N_{hatalar}(t+1) = 0$$

ve

$$N_{hatalar}(t+1) > N_{hatalar}(t) \Rightarrow N_{basarilar}(t+1) = 0$$

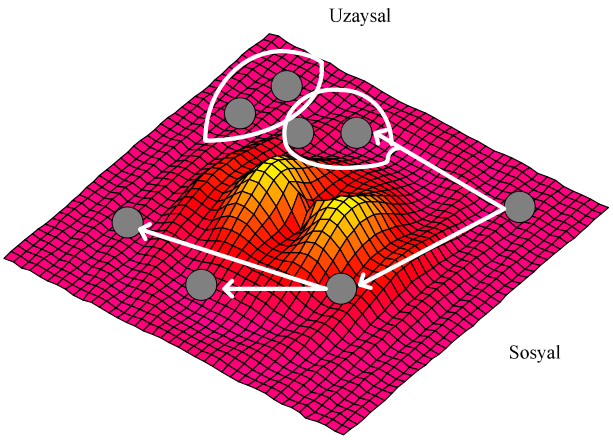
şeklinde ilave kurallar da uygulanmalıdır. Böylece, bir başarı durumunda hata sayacı sıfırlanır ve benzer şekilde başarı sayacı da bir hata oluştuğunda sıfırlanır [84].

2.2.2.3. Çeşitlilik Arttırma Geliştirmeleri

Bu bölümde sunulan modifikasyonlar daha çok GA'larda uygulanan yöntem olan popülasyonda çözüm çeşitliliğini arttırma amaçlıdır. Bu geliştirmeler genellikle yakınsama hızını azaltır, ancak çoklu lokal minimumla karşılaşıldığında daha iyi sonuçlar üretir.

Uzaysal Komşuluklar

Orijinal *lokal* PSO sürüyü indeks numaralarına bağlı olarak komşuluklara böler, yani x_1 ve x_2 uzaysal konumlarına bakılmayarak 1 çapında bir komşulukta komşu olarak göz önüne alınır. Parçacıkların uzaysal konumlarına bağlı farklı bir bölmeleme planı, Suganthan tarafından önerilmiştir [85].



Şekil 2.9. Komşuluklar

Algoritmanın her iterasyonu boyunca, her parçacıktan sürüdeki diğer tüm parçacıklara olan uzaklıklar, iki parçacık arasındaki en büyük uzaklığı d_{maks} adlı bir değişken ile takip edilerek hesaplanır. Her parçacık için $\|x_a - x_b\| / d_{maks}$ hesaplanır. Burada $\|x_a - x_b\|$ mevcut *a* parçacığı ile diğer *b* parçacığı arasındaki uzaklıktır. Bu oran, küçük orana tekabül eden komşu olan parçacıkları ya da büyük orana tekabül eden uzaktaki parçacıkları seçmede kullanılabilir. Suganthan seçim eşiğinin, küçük bir oranla başlayıp (örneğin bir *lokal* model) kademeli olarak artan şekilde iterasyon boyunca artması gerektiğini belirtmiştir. Oran 1'e yaklaştığında algoritma etkili olarak *global* modeli kullanacaktır.

Suganthan *frac* adlı eşiği

$$frac = \frac{3 \times k + 0.6 \times maksiter}{maksiter} \tag{2.31}$$

şeklinde hesaplamayı önermiştir [85]. Burada *k* mevcut iterasyon sayısı ve *maksiter* de izin verilen maksimum iterasyon sayısıdır. Diğer parçacık *b* eğer $\|x_a - x_b\| / d_{maks} > frac$ ise mevcut parçacığın komşuluğunda sayılır. *i* parçacığın komşuluğu

$$N_i = \{y_l\} \mid \frac{\|x_i - x_l\|}{d_{maks}} < frac, \quad l \in 1...s \tag{2.32}$$

şeklinde tanımlanır [84, 85]. Lokal en iyi parçacık daha sonra (2.7) denklemi kullanılarak seçilebilir. Bu denklemden sonra (2.9) denklemi parçacığın hızını güncellemek için kullanılabilir.

Suganthan aynı zamanda c_1, c_2 ve *w*'yi zamanla lineer olarak azaltmıştır, ancak sabit değerli c_1 ve c_2'nin daha iyi sonuçlar ürettiğini belirtmiştir. Aynı zamanda $c_1 = 2.5$ ve $c_2 = 1.5$ durumlarının çoğu test fonksiyonları için daha iyi sonuçlar ürettiği görülmüştür.

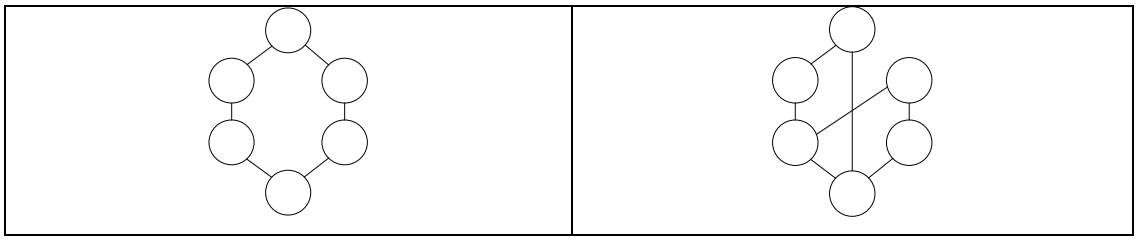
Değiştirilen komşuluk kuralı zamanla değişen *w* değeri ile birlikte *global* model ile karşılaştırıldığında tek modlu test fonksiyonları da dahil hemen hemen tüm test konfigürasyonlarında daha iyi sonuçlar vermiştir. Bu son özellik ilginçtir, çünkü *global* modelin tek modlu fonksiyonlarda *lokal* modelden daha iyi sonuçlar vermesi beklenir. Ancak *global* algoritma, sonuçları kesinlikle büyük miktarda etkileyecek zamanla değişen *w*'nin faydalarından yararlanamamaktadır [85].

Yakınsama durumunda, herhangi bir sosyal komşuluğun aynı zamanda bir uzaysal komşuluk olacağı söylenebilir. Komşuluğun da sayısının belirlenmesi için sürü boyutundakine benzer olarak uyarlamalı modeller bulunmaktadır [84].

Komşuluk Topolojileri

Denklem (2.4) ve (2.7) boyunca elde edilen 1'e eşit bir *l* değerli *lokal* model bir halka topolojisini tanımlar. Her parçacık, tüm komşuluğunu indeks uzayında kendine en yakın iki komşu olarak kabul eder. Tüm parçacıklar bilgiyi dolaylı olarak paylaşır çünkü $i + 1$ parçacığı hem i' nin hem $i + 2$'nin komşusudur, bunlar da $i - 1$ ve $i + 3$ komşularına sahiptir. i ve $2i$ parçacıkları arasında diğerine göre daha uzun yol aralarındaki bilgi değişimini yavaşlatır. Bu durum bunların arama uzayının farklı bölgelerini araştırmalarına ve hâlâ bilgi paylaşabilmelerine izin verir.

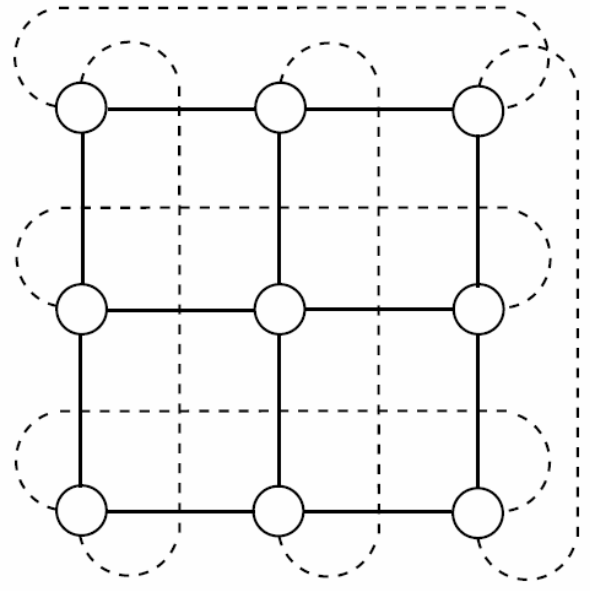
Kennedy *lokal* PSO için bilgi akışının değişebileceği alternatif komşuluk topolojileri oluşturmuştur [86]. Kennedy tarafından test edilen ilk topoloji değişken sayıda gelişigüzel yer değiştirilmiş bağlantılı orijinal halka topolojisidir. Tüm parçacıkların tek bir merkeze bağlandığı ancak direkt olarak birbirlerine direkt olarak bağlanmadığı bir "tekerlek" topolojisi de düşünülmüştür. Şekil 2.10, bazı linkler gelişigüzel değiştirilmeden önce ve sonra bu iki topolojiyi göstermektedir. Düşünülen son iki topoloji de gelişigüzel bağlantılı topoloji ve yıldız topolojisidir. Yıldız topoloji tamamen bağlantılı bir sürüyü temsil ettiğinde aslında bir *global* modeldir. Kennedy fonksiyon çok sayıda lokal minimuma sahip olduğunda, yıldız topoloji gibi çok bağlantılı topolojilerin iyi optimum bulmada zorluk yaşayacağını sanmıştır [57]. Ayrıca Kennedy ve Medes tarafından parçacıkların dört komşu parçacıkla bir ızgara ağı (iki boyutlu kafes) ile bağlandığı ve Şekil 2.11'de gösterilen bir Von Neuman topoloji de önermiştir [86]. Ancak bu topoloji yakınsama hızını düşürmektedir çünkü bulunan en iyi parçacık sürüdeki tüm parçacıkları etkilemeden önce kendi pozisyon bilgisini birkaç komşuluk boyunca yaymalıdır.



(a) Halka topolojisi (b) Gelişigüzel halka topolojisi

(c) Tekerlek topolojisi (d) Gelişigüzel tekerlek topolojisi

Şekil 2.10. Linkler gelişigüzel değiştirilmeden önce ve sonra iki olası komşuluk topolojisinin şekli



Şekil 2.11. Von Neuman topolojisi

Kennedy tarafından sunulan deneysel sonuçlar, topolojinin algoritmanın performansını önemli derecede etkilediğini fakat optimum topolojinin özel probleme bağlı olduğunu göstermektedir [57, 86]. Örneğin, tekerlek topolojisi çok lokal optimumlu bir fonksiyona uygulandığında en iyi sonucu vermiştir. Kennedy bunun topoloji boyunca bilginin daha yavaş yayılmasından kaynaklanabileceğini ve böylece algoritmanın çoklu lokal minimum karşısında daha sağlam olacağını ispatsız olarak ifade etmiştir. Ancak tek modlu fonksiyonlarda, yıldız topoloji (*global* model), birbirine daha az bağlı topolojilere göre bilginin daha hızlı yayılmasından dolayı daha iyi sonuçlar vermiştir.

Burada göz önünde bulundurulan tüm topolojiler parçacık indeksinde yapılır, arama uzayında yapılmaz.

Sosyal Tespit

Kennedy sosyal tespit olarak adlandırılan uzaysal komşuluk ve halka-topoloji yaklaşımlarının bir karışımı olan *lokal* PSO versiyonunu önermiştir [87]. Orijinal PSO'daki parçacıklar, kendileri tarafından keşfedilen önceki en iyi pozisyonlara (y_i'ler) ya da sürüdeki diğer parçacıklara ($\hat{y}_i$'ler) çekilir. İnsan sosyal etkileşim çalışmaları insanların gruptaki özel bir bireyin inançlarından çok bir grubun ortak inançlarını takip etmeye çalıştığını göstermektedir. Bu düşünce PSO'ya şu şekilde uygulanabilir: her parçacık arama uzayındaki bir kümenin bir üyesidir. Bu kümenin merkezi kümenin ortak inançlarına benzerdir. Parçacıklar da bireysel en iyiden ziyade bu kümelerin merkezine çekilebilir [84].

PSO'yu bilişsel (parçacığın önceki kişisel en iyi pozisyonuyla ilgili olarak) ya da sosyal (komşulukta önceki en iyi pozisyona bağlı olarak) terimin ya da her ikisinin uygun küme merkeziyle yer değişeceği şekilde uyarlamak mümkündür.

Bir *k*-means kümeleme algoritması kullanılarak, parçacıların kişisel en iyi değerlerinden birkaç küme $\boldsymbol{C}_l$, $l \in 1..\#küme$ oluşturulur. Küme sayısı, *#küme*, önceden seçilir. $\boldsymbol{C}^*$, tüm kümelerin setini göstersin. O zaman map($\boldsymbol{C}^*$, *i*), l = map($\boldsymbol{C}^*$, *i*) $\Rightarrow$ $y_l \in \boldsymbol{C}_l$ olan l değerini döndüren bir fonksiyondur. Bu fonksiyonla, *i* parçacığının ait olduğu kümeyi bulmak mümkündür.

i parçacığını içeren küme için merkez

$$\overline{C}(i) = \frac{1}{|C_l|}\sum_{a \in C_l} a, \qquad l = \text{map}(C^*, i) \tag{2.33}$$

şeklinde tanımlanır. $|C_l|$, l kümesindeki elemanların sayısını göstermektedir. (2.7) denkleminde tanımlandığı gibi 1 çapındaki komşuluk kullanılarak, komşuluktan en iyi bir *g* parçacığı

$$y_g \in N_i \mid f(y_g) \le f(y_B) \qquad \forall\, y_B \in N_i$$

şeklinde seçilir. *g*'yi içeren kümenin merkezi, denklem (2.33)'deki merkezin tanımı kullanılarak $\overline{C}(g)$ olur. Bu tanımlarla, *lokal* güncelleme denkleminin üç yeni çeşidi

$$v_{i,j}(t+1) = wv_{i,j} + c_1 r_{1,j}\left[\overline{C}(i)_j - x_{i,j}\right] + c_2 r_{2,j}\left[\hat{y}_{i,j} - x_{i,j}\right] \tag{2.34}$$

$$v_{i,j}(t+1) = wv_{i,j} + c_1 r_{1,j}\left[y_{i,j} - x_{i,j}\right] + c_2 r_{2,j}\left[\overline{C}(g)_j - x_{i,j}\right] \tag{2.35}$$

$$v_{i,j}(t+1) = wv_{i,j} + c_1 r_{1,j}\left[\overline{C}(i)_j - x_{i,j}\right] + c_2 r_{2,j}\left[\overline{C}(g)_j - x_{i,j}\right] \tag{2.36}$$

şeklinde tanımlanmıştır.

Denklem (2.34)'te sunulan ilk versiyon; parçacık, kendi kişisel deneyimini kullanmak yerine, ait olduğu uzaysal kümenin ortak deneyimini kullanır şeklinde yorumlanabilir. Komşuluk etkisi orijinal *lokal* modeldeki gibi değişmeden kalır.

İkinci versiyon tersi bir yaklaşım kullanır: Denklem (2.35), komşuluk etkisinin şimdi komşulukta en başarılı parçacıkların ait olduğu kümenin merkezine dayandığını göstermektedir.

Komşuluk, hâlâ parçacığın indeksine göre tanımlanır, uzaysal ilişkiler içermez. Güncelleme denkleminin bilişsel bileşeni etkilenmemektedir.

Son olarak, (2.36) biliş ve sosyal bileşenlerin her ikisini kişisel merkezlerle yer değiştirir. Parçacık o anki komşuluğundaki en başarılı parçacığın ait olduğu kümenin ve kendi kümesinin ortasına çekilir.

Küme merkezlerinin bulunmasıyla ilgili hesaplamalar ihmal edilemeyecek derecede bir zaman almaktadır, bu yüzden sosyal tespit yaklaşımı orijinal *lokal* PSO'dan biraz daha yavaştır [84].

Kennedy, Denklem (2.34)'e bağlı olarak ilk versiyonun bazı problemlerde orijinal *lokal* modelden daha iyi sonuçlar üretebildiğini bildirmiştir. Algoritmaların belirli bir hata sınırına ulaşmalarının ne kadar sürdüğünü görmek için zamanlarının ölçüldüğü ikinci deneyler, tespit algoritmalarının genel olarak orijinal *lokal* modelden daha yavaş olduğunu göstermiştir. Her iterasyon boyunca kümelerin oluşturulmasıyla ilgili genel maliyet yeni algoritmaları oldukça yavaşlatmaktadır [84].

Denklem (2.35) ve (2.36)'ya bağlı diğer iki algoritma parçacığın uzak bir kümenin merkezine geçmeye çalışmaması gerektiğini göstererek, genel olarak daha kötü sonuçlar vermiştir.

Alt Popülasyonlar

Bir algoritma tarafından korunan çözüm çeşitliliğini arttırmak için alt popülasyonlar kullanma fikri daha önce GA'larda kullanılmıştır. Alt popülasyon oluşturmak için orijinal popülasyon daha küçük popülasyonlar ayrılır. Algoritma (örneğin GA) alt popülasyondaki elemanlara olağan biçimde uygulanır. Zaman zaman üyeler alt popülasyonlar arasında yer değiştirir, ya da alt popülasyonlar arasında bilgi paylaşımını kolaylaştırmak için bazı etkileşim planları kullanılır. Fikir, her popülasyonun arama uzayının çok uzak bölgelerindeki çözümlerin olası zararlı etkilerinden uzak olarak arama uzayının daha küçük bir bölgesini tamamen araştırmasına izin vermektir [84].

LØvbjerg ve arkadaşları alt popülasyon fikrini PSO'ya uygulamıştır [83]. PSO'ya uyguladıkları daha önce açıklanan yetiştirme operatörü (bir aritmetik çaprazlama operatörü) alt popülasyonlar arası iletişimi etkilemek için de kullanılmıştır. Bunlar orijinal sürüyü her bloğun kendi global en iyi parçacığı koruduğu bloklara ayırırlar. Çaprazlama operatörü için ataları seçerken, atalardan birinin farklı bir alt popülasyondan seçilme olasılığı da vardır. Eğer bu olasılık oldukça küçük olursa, alt popülasyonlar kendi çözümlerini keşfetmek için zamana sahip

olacaktır ve sonra alt popülasyonlar arası yetiştirme yoluyla bu çözümü diğer alt popülasyonlarla paylaşabileceklerdir [84].

Asıl uygulamalarında yazarlar, alt popülasyonlar oluştururken parçacık sayısını 20'de sabit tutmayı seçmişlerdir ve böylece bir ikili alt popülasyon konfigürasyonu, her birinde 10 parçacık olan iki sürüden oluşacaktır. [83]'de sunulan sonuçlar, alt popülasyon oluşturan bu yöntemin daha iyi bir performans sağlamadığını göstermiştir. Alt popülasyon sayısı arttıkça yakınsama hızı da yavaşlamıştır.

Alt popülasyon tekniği PSO algoritmasına az fayda sağlıyormuş gibi görülmektedir. Alt popülasyonların oluşturulma şekli belki başarısızlığın sorumlusu olarak görülebilir, ancak fikir sağlamdır ve diğer evrimsel algoritmalara uygulandığında daha iyi çalışmaktadır [84].

Çoklu Başlatmalı PSO (ÇBPSO)

Van den Bergh YGPSO'nun bir uzantısı olarak çoklu başlatmalı PSO (ÇBPSO) önermiştir [84]. ÇBPSO şu şekilde çalışmaktadır:

1. Sürüdeki tüm parçacıkları gelişigüzel başlat.
2. Lokal bir minimuma yakınsayıncaya kadar YGPSO'yu uygula. Bu lokal optimumun pozisyonunu kaydet.
3. Birinci ve ikinci adımı sonlandırma kriteri sağlanıncaya kadar tekrarla.

Van den Bergh YGPSO'nun yakınsayıp yakınsamadığını belirlemeye bağlı olarak farklı ÇBPSO'lar önermiştir. Güzel bir yaklaşım

$$f_{oran} = \frac{f\left(\hat{y}(t)\right) - f\left(\hat{y}(t-1)\right)}{f\left(\hat{y}(t)\right)} \tag{2.37}$$

şeklinde amaç fonksiyonunda değişim oranını ölçmektir. f_{oran} kullanıcı tanımlı bir eşikten küçükse bir sayaç arttırılır. Sayaç belirli bir değere yaklaştığında sürünün yakınsadığı varsayılır. Van den Bergh'e göre, ÇBPSO çoğu test durumlarında YGPSO'dan daha iyi bir performans vermiştir. Ancak amaç fonksiyonunda boyut sayısı arttıkça, ÇBPSO'nun performansı önemli derecede azalmaktadır [84].

Mutasyon

Higashi ve Iba PSO'yu Gauss mutasyonuyla kullanarak melez bir yöntem sunmuştur [88]. Benzer olarak Esquivel ve arkadaşları PSO'nun erken yakınsama problemini gidermek için düzenli olmayan mutasyon operatörü isimli güçlü çeşitlilik bakım mekanizmasıyla *lokal* ve *global* modelin melez halinden oluşan bir yöntem önermiştir [89]. Esquivel ve arkadaşlarına göre *lokal* PSO'nun melez yaklaşımı ve düzenli olmayan mutasyon operatörü yapılan tüm deneylerde PSO ve YGPSO'dan daha iyi performans sergilemiştir [84, 89].

Çekici ve İtici PSO (ÇİPSO)

Çekici ve itici PSO (ÇİPSO) değişimli olarak bir çeşitlilik ölçüsüne bağlı çekim ve itim şeklinde iki fazı takip eder. Çekim fazında ÇİPSO PSO'yu hızlı bilgi akışına izin vermek için kullanır. Parçacıklar birbirini çeker ve böylece çeşitlilik azalır. Uygunluk ilerlemesinin %95'inin bu fazda olduğu bulunmuştur. Bu, çözümde ince ayarlamalar yapmada düşük çeşitliliğin önemini göstermektedir. İtme fazında parçacıklar o ana kadar bulunan en iyi çözümden itilirler ve böylece çeşitlilik artar. Riget ve Vesterstrøm'ın yaptığı deneylere göre ÇİPSO test edilen durumların çoğunda PSO ve GA'dan daha iyi sonuçlar vermiştir [90].

Dağıtan PSO (DPSO)

Dağıtan PSO (DPSO) erken yakınsamayı önlemek amacıyla Xie ve arkadaşları tarafından PSO'ya gelişigüzel mutasyonlar eklemek için önerilmiştir [91]. DPSO hız ve pozisyon güncellemelerinden sonra parçacıklara gelişigüzellik eklenmesi yoluyla şu şekilde negatif entropi (termodinamik, dağınım) getirir:

Eğer ($r_1(t)<c_v$) o zaman $v_{i,j}(t+1)=r_2(t)V_{maks}$
Eğer ($r_3(t)<c_l$) o zaman $x_{i,j}(t+1)=r_4(t)$

$r_1(t) \sim U(0,1)$, $r_2(t) \sim U(0,1)$ ve $r_3(t) \sim U(0,1)$; c_v ve c_l (0,1) aralığından kaotik faktörler ve $r_4(t) \sim U(S_{min}, S_{maks})$ ve S_{min}, S_{maks} arama uzayının alt ve üst sınırlarıdır. Kalite test problemlerine uygulandığında DPSO'nun PSO'dan daha performanslı olduğu gösterilmiştir [91].

Diferansiyel Gelişimli PSO (DGPSO)

Diferansiyel Gelişimli PSO (DGPSO) [92] mutasyon uygulamak için bir diferansiyel gelişim operatörü kullanır [93]. Bir $\ddot{y}(t)$ noktası şu şekilde hesaplanır:

Eğer ($r_1(t)<p_c$ YA DA $j=k_d$) O zaman

$$\ddot{y}_{i,j}(t)=\hat{y}_j(t)+\frac{\left(y_{1,j}(t)-y_{2,j}(t)+y_{3,j}(t)-y_{4,j}(t)\right)}{2} \tag{2.38}$$

olur. $r_1(t) \sim U(0, 1)$ ve $k_d \sim U(1, N_d)$'dir. $y_1(t)$, $y_2(t)$, $y_3(t)$ ve $y_4(t)$ kişisel en iyi pozisyonlar kümesinden gelişigüzel seçilirler. Sonra $f(\ddot{y}_i(t))< f(y_i(t))$ ise $y_i(t)=\ddot{y}_i(t)$ olarak seçilir. $x_i(t)$ yerine $y_i(t)$'yi mutasyona uğratmanın nedeni sürünün düzensizliğini önlemektir.

DGPSO tek iterasyonlarda hız ve pozisyon için (2.13) ve (2.4) denklemini, çift iterasyonlarda da (2.38) denklemini kullanır. Zhang ve Xie'ye göre DGPSO kalite testi fonksiyonlarına uygulandığında genel olarak PSO, diferansiyel gelişim, GA, evrim stratejileri ve bulanık uyarlamalı PSO'dan daha iyi sonuçlar vermiştir [92, 94].

Yaşam Çevrimi Modeli

Kendini uyarlayan PSO, GA ve tepe tırmanma algoritmalarının avantajlarını tek bir algoritmada birleştiren Yaşam Modeli isimli sezgisel bir arama yöntemi Krink ve Løvbjerg tarafından önerilmiştir [95]. Bu modelde potansiyel çözümleri temsil eden bireyler PSO parçacıkları olarak başlatılır, sonra çözüm için arama sırasında performanslarına göre GA bireylerine ya da tepe tırmanıcılara dönüşebilir. Sonra tekrar parçacığa dönüşürler. Bu işlem yakınsama sağlanıncaya kadar devam eder. Bu model PSO, GA ve tepe tırmanma yöntemleriyle karşılaştırılmış ve kalite testi fonksiyonlarında genel olarak iyi sonuçlar elde edilmiştir [95]. Ancak PSO beş test fonksiyonundan üçünde daha iyi ya da benzer sonuçlar üretmiştir.

Diğer melez bir yöntem ise Veeramachaneni ve arkadaşları tarafından önerilmiştir ve bu yöntemde PSO ve GA birleştirilmiştir [96]. Önerilen bu iki melez yöntemler için yapılan deney sonuçlarından, orijinal PSO'nun bu karmaşık melez yöntemlerden daha iyi sonuç verdiği gözlenmiştir.

Öz-örgütlenmiş Tehlikelilikli PSO (ÖÖTPSO)

Erken yakınsama probleminin üstesinden gelmek amacıyla sürü çeşitliliğini arttırmak için Løvbjerg ve Krink, PSO'yu öz-örgütlenmiş tehlikelilikle genişletmiştir [97]. Parçacıkların

birbirine ne kadar yakın olduklarını gösteren *tehlikelilik* adlı bir ölçü parçacıkların yerini değiştirerek sürü çeşitliğini arttırmak amacıyla kullanılmıştır. Yüksek tehlikelilik ölçüsüne sahip parçacık tehlikeliliğini, kendisinin bir komşuluğunda, kullanıcı tarafından belirlenen CL sayısındaki parçacığın tehlikelilik ölçüsünü arttırarak yayar. Daha sonra CL değeriyle kendi tehlikeliliğini azaltır ve kendi yerini değiştirir. İki tip yer değiştirme incelenmiştir. Birincisi parçacığı tekrar başlatırken, ikincisi parçacığı arama uzayında yüksek tehlikelilikle biraz daha öteye iter.

Løvbjerg ve Krink'e göre uygulama fonksiyonlarında ilk yerleştirme planı iyi sonuçlar vermiştir. Öz-örgütlenmiş tehlikelilikli PSO, deneydeki dört durumdan sadece birinde PSO'dan daha iyi sonuçlar vermiştir. Ancak, bir parçacığın tehlikelilik değerinin onda birinin kendi atalet ağırlığına (w=2) eklenmesiyle bu yöntemin PSO'ya karşı önemli oranda başarı sağladığı belirtilmiştir [97].

Uygunluk-Uzaklık Oran Tabanlı PSO (U-UOTPSO)

Perm ve arkadaşları hız güncelleme terimine, her parçacığın komşuluğunda daha iyi bir kişisel en iyi pozisyona sahip bir parçacığa doğru hareket etmesine izin verecek şekilde, yeni bir terim eklemiştir [98]. Bu şekilde algoritma Uygunluk-uzaklık oran tabanlı PSO (U-UOTPSO) olarak isimlendirilmiştir. Değiştirilen hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] + c_3 r_{3,j}(t)\left[y_{\eta,j}(t) - x_{i,j}(t)\right] \quad (2.39)$$

şeklindedir. Burada her $y_{\eta,j}(t)$

$$\frac{f(x_i(t)) - f(y_\eta(t))}{\left|f(y_{\eta,j}(t)) - f(x_{i,j}(t))\right|} \quad (2.40)$$

denklemi maksimize edilmek suretiyle seçilir.

Veeramachaneni ve arkadaşlarına göre U-UOTPSO, erken yakınsama olasılığını düşürmekte ve böylece lokal bir çözüme takılıp kalmayı önlemektedir [96]. Ayrıca $c_1r_1 = c_2r_2$=1 ve c_3r_3 = 2 seçildiğinde ve test edilen kalite testi fonksiyonlarında U-UOTPSO, birçok PSO versiyonundan; çekici ve itici PSO, dağıtan PSO, öz-örgütlenmişi tehlikelilikli PSO ve çoklu başlatmalı PSO; daha iyi sonuçlar vermiştir.

2.2.2.4. Paralel PSO

PSO algoritmasının paralel işlemlere uygun olduğu düşünülmüş ve [99]'da PSO'nun paralel versiyonları önerilmiştir. Burada önerilen yöntemler GA'nın paralelleştirilmiş modellerini örnek almıştır ve üç kategoriye ayrılmıştır. Global PSO, adalı PSO ve yayılmalı PSO.

Global PSO'da, ana işlemci global en iyi değeri bulur, diğer işlemciler ise parçacıkların amaç fonksiyonlarını hesaplar ve hız vektörünü günceller. Bu modelde haberleşme ek yükü sürü büyüklüğü ile doğru orantılıdır. Bununla beraber en iyi değer tüm sürü üzerinden hesaplanır, böylece her bir işlemcide global bilgi mevcuttur [99].

Adalı PSO yöntemi sürüyü alt sürülere böler. Her bir alt sürü kendi PSO algoritmasını çalıştırır. En iyi değeri bulma ve hız vektörünü güncelleme işlemi lokalde yapılır. Sürüyü belli bir iterasyon ilerlettikten sonra her sürünün en iyi sonucu komşu işlemcilere göç eder. Adalı PSO modelinde iki modül vardır. Birincisi komşu işlemcilerin birbirlerinden en iyi değerleri almaları ve göndermelerini sağlar, ikincisi ise her düğümde bulunan lokal optimuma göre hız vektörünü günceller. İlk modül belli bir zaman aralığında ya da kesmelerle çalışacaktır [99].

Yayılmalı PSO'da her bir parçacık ayrı bir yaşam alanı gibi düşünülüp tüm amaç fonksiyon bulma işlemleri lokalde yapılır. Yayılmalı PSO'da farklı olarak her iterasyonda işlemciler buldukları en iyi değerleri komşu işlemcilere gönderirler. Her işlemci komşudan gelen değerleri analiz eder ve bulunan en iyi değere göre hız vektörünü günceller. Bu model dağıtık öğrenmeyi kullanır çünkü her işlemci çözümü inceler ve bilgi değişim için komşularıyla işbirliği yapar. Bu modelin verimliliği halka topolojisinden tümü birbirine bağlı işlemci topolojisine kadar modelin bağlanırlığına bağlı olarak değişir [99].

Önerilen algoritmaların karmaşıklığı, ölçeklenebilirliği ve yayılmalı PSO'nun yakınsama özelliği incelenmiştir. Karmaşıklık analizi sonucu yayılmalı modelin her bir parçacığın farklı işlemciye yerleştirildiğinde ve her iterasyonda birbirleri ile haberleştirildiklerinde adalı modelin bir özel durumu olduğu görülmüştür. Ayrıca yayılmalı modelin global modele göre daha ölçeklenebilir olduğu, yani sürüdeki parçacık sayısı işlemci sayısından çok fazla olduğunda yayılmalı modelin hızlanmasının daha fazla olduğu görülmüştür. Yayılmalı modelin farklı topolojiler kullanılarak uygulaması gerçekleştirildiğinde ızgara, 2D-tor ve Hiperküp'te seri algoritmadan hızlı çalıştığı ancak halka topolojisine uygulandığında çok yavaş kaldığı belirlenmiştir. Sonuçta yayılmalı modelin hem ölçeklenebilir hem de sağlam bir model olduğu gösterilmiştir.

2.3. PSO Parametre Kontrolü

PSO'nun problemler için uygulanmasında iki ana adım bulunmaktadır: Çözümün temsili ve uygunluk fonksiyonu. PSO'nun avantajlarından birisi PSO'nun parçacık olarak gerçel sayıları almasıdır. GA'lardaki gibi ikili kodlamaya dönüştürme ihtiyacı ya da özel genetik operatörler kullanma ihtiyacı yoktur.

PSO'da ayarlanması gereken fazla parametre yoktur. Aşağıda parametre listesi ve tipik değerleri yer almaktadır.

Parçacık sayısı: Tipik sınır 20-40'tır. Gerçekte çoğu problem için 10 parçacık iyi sonuçlar almaya oldukça yeterlidir. Bazı özel ve zor problemler için 100 ya da 200 parçacık da kullanılabilir. Alternatif olarak uyarlamalı parçacık sayısı da kullanılabilir. Çünkü verilen bir sürü boyutunun gerçekten diğerinden iyi olduğu kimse tarafından ispatlanmamıştır. Bu yüzden algoritmanın zaman zaman parçacık sayısını değiştirmesine izin verilmiştir.

Mantık şudur: "En iyi parçacığın komşuluğunda yeterli gelişme olmuyorsa yeni bir parçacık üretilebilir" ya da "Yeterli gelişme oluyorsa en kötü parçacık ölebilir". Yeterli gelişme de "komşulukta parçacıkların en az yarısı için gelişme" olarak tanımlanmıştır.

Parçacıkların boyutu: Bu optimize edilecek probleme bağlıdır.

Parçacıkların aralığı: Bu da optimize edilecek probleme bağlıdır. Parçacıkların farklı boyutu için farklı aralıklar belirlenebilir.

***V_{maks}*:** Bu, bir parçacığın bir iterasyon boyunca alabileceği maksimum değişikliği yani hızı belirler. Genellikle parçacık aralığına göre belirlenir. Örneğin parçacık [-10, 10] aralığında ise parçacığın $V_{maks} = u_n(x_i)\text{-}l_n(x_i) = 20$ olarak belirlenebilir [100]. Ayrıca V_{maks} için arama uzayı boyutunun yarısı da kullanılabilir [101].

Öğrenme faktörleri: c_1 ve c_2 genellikle 2 olarak alınır. Fakat genellikle [0, 4] aralığında değerler alır. Ayrıca bu katsayılar da uyarlamalı olarak her adımda düzenlenebilir. Mantık şudur.

"Bir parçacık tüm komşularından ne kadar daha iyi ise o oranda kendi yoluna devam eder, ya da tam tersi." ya da "Bir parçacığın en iyi komşusu kendisinden ne kadar iyiyse bu parçacık o oranda onun doğrultusunda gider, ya da tam tersi".

Sonlandırma kriteri: Genelde PSO'nun çalıştıracağı maksimum iterasyon sayısı veya gerek duyulacak minimum hata sonlandırma kriteri olarak kullanılır. Bitim şartı optimize edilen probleme de bağlıdır. Literatürde kullanılabilecek tüm bitim şartları için [65] referansına başvurulabilir.

Global versiyon - Lokal versiyon: Global versiyon daha hızlıdır fakat bazı problemler için lokal optimuma takılabilir. Lokal versiyon biraz daha yavaştır ancak lokal bir optimuma takılma olasılığı daha azdır. Global versiyon hızlı bir sonuç almak için ve lokal versiyon aramayı inceleştirmek için kullanılabilir.

Atalet ağırlığı: Shi ve Eberhart tarafından önerilen bu terim için [70] referansına başvurulabilir.

2.4. Sonuçlar

Parçacık Sürü Optimizasyonu (PSO), biyolojik popülasyonlarda işbirlikçi davranış ve sürü halinde ilerleme fikrine bağlı olan ve diğerlerine oranla daha yeni bir sezgisel arama ve optimizasyon yöntemidir. Basit ve uygulaması çok kolay bir algoritmadır ve çok geniş bir uygulama alanı bulunmaktadır. Zor matematiksel problemlerini çok basit olarak çözebilen kısa cebirselsel adımlardan oluşur. Çalışma biçimi bakımından evrimsel hesaplama yöntemlerinden farklıdır. Zaman karşısında devam eden bireyler bir diğerinin problem uzayı aramasını etkiler. Bireysel deneyim ve sosyal öğrenmenin entegresi söz konusudur.

GA'lar kullanılarak çözülebilen çoğu problemi içeren farklı optimizasyon problemlerinin geniş bir kümesini çözmek için kullanılabilir ve etkili sonuçlar alınabilir.

3. KAOTİK HARİTALI PARÇACIK SÜRÜ OPTİMİZASYON YÖNTEMLERİ

3.1. Giriş

PSO'nun diğer evrimsel hesaplama temelli algoritma ve matematiksel temelli algoritmalara göre fazla hafıza gerektirmeyen, hesapsal olarak etkili ve uygulaması kolay optimizasyon yöntemi olduğu ve hızlı yakınsama özelliğinin olduğu ikinci bölümde belirtilmişti. PSO'nun arama başlangıcında global arama kabiliyetine sahip olması ve arama sonlarına doğru lokal arama kabiliyetine sahip olması istenmektedir. Bununla birlikte çok fazla lokal optimum noktası bulunan problemlerle çalışırken çalışma sonunda lokal optimum noktaları keşfetme olasılığı fazladır. Birçok araştırmacı ikinci bölümde de bahsedildiği gibi farklı ve değişik ayarlamalarla PSO'nun performansını arttırma yoluna gitmiştir.

PSO'nun yakınsama özelliği, çalıştırma sırasında parametreleri için rasgele sayı dizisi kullanan stokastik doğasına oldukça bağlıdır. PSO algoritmasında özellikle, farklı rasgele sayı dizileri kullanıldığında elde edilen son sonuçlar birbirine çok yakın olabilir ancak eşit olmayabilir. Aynı optimum değerlere ulaşabilmek için farklı iterasyon sayılarına ihtiyaç duyulabilir. Fakat diğer evrimsel hesaplama temelli algoritmalarda da olduğu gibi, PSO algoritmalarının performansını arttırmayı garanti eden özel bir sayı üretecine bağlı analitik sonuçlar yoktur [102].

Son zamanlarda kaotik sayı dizileri rasgele sayı dizilerinin yerini almış ve bazen güzel sonuçlar vermiştir. Bunlara örnek olarak güvenli iletişim [103], doğal fenomen modelleme [104], doğrusal olmayan devreler [105], DNA hesaplama [106], imge işleme [107] verilebilir. Ayrıca evrimsel algoritmalarının performanslarını arttırmak için de kullanılmıştır [102]. Kaotik sayı dizilerinin kullanılması teorik olarak bunların tahmin edilemezliği, yayılmış spektrumlu karakteristiği ve ergodik özelliklerinden dolayı artmıştır.

Tezin bu bölümünde PSO'nun parametrelerinin belirlenmesinde rasgele tabanlı bir seçim söz konusu olduğunda farklı kaotik sistemler rasgele sayı dizilerinin yerine kullanılmış ve on iki farklı PSO önerilmiştir. Bu şekilde PSO'nun global yakınsama özelliğinin arttırılması ve lokal çözümde takılıp kalması önlenmeye çalışılmıştır. Örneğin Trelea, çalışmasında atalet ağırlığı değerinin PSO'nun yakınsamasını etkileyen temel faktörlerin biri olduğunu belirtmiştir [108]. Ayrıca r_1 and r_2 değerleri de yakınsamayı etkileyen faktörlerdir. Ancak bu parametreler faz uzayında algoritmanın ergodik özelliğini garanti edemezler çünkü bunlar PSO'da rasgeledir.

Aşağıdaki alt bölümlerde sırayla kaotik haritalı PSO yöntemleri için kullanılan kaotik haritalar tanıtılmış sonrasında da önerilen yöntemler açıklanmıştır. Daha sonra kalite testi

fonksiyonları tanıtılarak önerilen yöntemlerin diğer PSO yöntemleriyle karşılaştırılması yapılmıştır. Son olarak da sonuçlara göre analiz ve yorumlarla bölüm sonuçlandırılmıştır.

3.2. Kaotik Haritalar

Kompleks fenomenleri modellemede, örneklemede, sayısal analizde, karar vermede ve özellikle bu tez konusunu da içeren sezgisel optimizasyon yöntemlerinde uzun periyotlu rasgele sayı dizileri çok önemli bir yer tutmaktadır. Üretilen sayılar için fazla depolama alanı kullanılmamalı ve istenen bir doğruluğa ulaşmak için fazla zamana gereksinim duyulmamalıdır. Bu şekilde üretilen sayılar bir uygulama için yeterince "rasgele" olurken başka bir uygulama için yeterince rasgele olmayabilir.

Kaos periyodik olmayan, yakınsamayan ve sınırlı olan, doğrusal olmayan dinamik sistemlerde bulunan deterministik, rasgele benzeri bir süreçtir. Ayrıca başlangıç şartları ve parametrelerine oldukça bağlıdır [109]. Kaosun doğası görünürde rasgele ve tahmin edilemezdir. Ayrıca kendi içerisinde bir düzene sahiptir. Hatta çoğu kez düzen içinde düzensizlik ya da düzensizlik içinde düzen olarak da tanımlanmaktadır. Matematiksel olarak, basit deterministik dinamik bir sistemin rasgeleliğidir ve kaotik sistem rasgelelik kaynağı olarak düşünülebilir.

Kaotik bir harita ayrık zamanlı dinamik bir sistemdir ve kaotik durumda ilerleyen

$$x_{k+1} = f(x_k), \quad 0 < x_k < 1, \; k = 0, \; 1, \; 2, \cdots \tag{3.1}$$

genel denklemiyle temsil edilebilir. Kaotik sayı dizisi

$$\{x_k : \; k = 0, \; 1, \; 2, \; \cdots \}$$

yayılmış spektrumlu rasgele sayı dizisi olarak kullanılabilir [102].

Kaotik sayı dizilerinin üretilmelerinin ve depolanmalarının kolay ve hızlı olduğu ispatlanmıştır. Sadece birkaç fonksiyon (kaotik harita) ve birkaç parametre (başlangıç şartı) çok uzun diziler için bile yeterlidir. Ayrıca, çok fazla sayıda farklı sayı dizisi basitçe başlangıç şartı değiştirilerek çok kolay bir şekilde üretilebilir. Bu sayı dizilerinin bir özelliği de deterministik olmaları ve tekrar üretilebilmeleridir.

Kaotik haritalı PSO'da ergodik, düzensiz ve stokastik özelliklerine sahip kaotik haritalar kullanılarak diğer PSO yöntemlerine oranla daha kolayca lokal çözümden kaçabilmeyi sağlamak amaçlanmıştır. Bu şekilde global yakınsamanın arttırılması hedeflenmiştir. Rasgele sayılar özel bir kaotik harita bir adım ilerletilerek üretilmektedir. Yani, PSO'da ilk iterasyondan itibaren rasgele sayı üretimine ihtiyaç duyulduğunda seçilen kaotik harita seçilen bir başlangıç

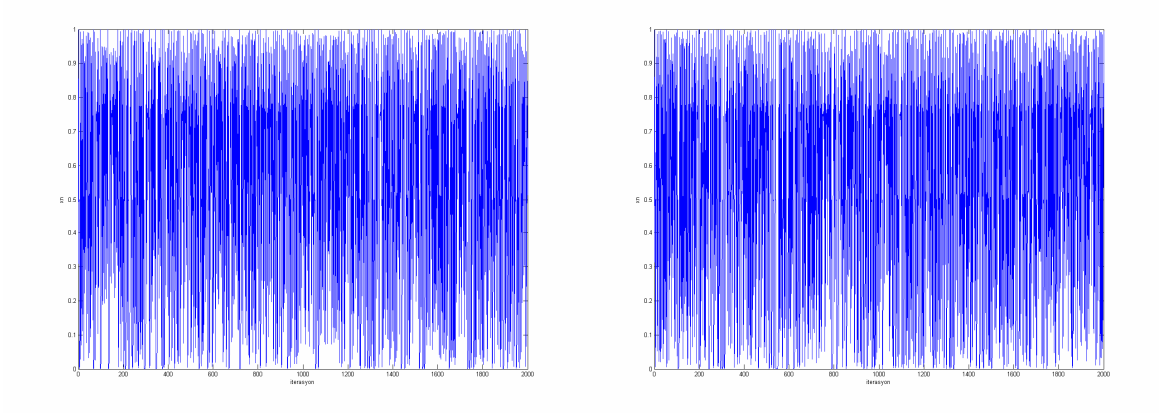
noktasından başlanarak birer adım ilerletilir. PSO parametreleri için kullanılacak kaotik sayı üreten haritalar aşağıda listelenmiştir.

3.2.1. Lojistik Harita

En basit ve en çok kullanılan haritalardan birisidir [110]. Bu kaotik davranış gösteren biyolojik popülasyonların doğrusal olmayan dinamiklerinde ortaya çıkmıştır. Lojistik harita Denklem (3.2)'de verilmiştir.

$$X_{n+1} = aX_n(1 - X_n) \tag{3.2}$$

Bu denklemde, n iterasyon sayısını göstermekte, X_n de n. kaotik sayıyı temsil etmektedir. Başlangıç $X_0 \in (0, 1)$ olduğunda $X_n \in (0, 1)$ olduğu görülmektedir. Ayrıca $X_0 \notin \{0.25, 0.5, 0.75\}$. Deneylerde a=4 seçilmiştir.



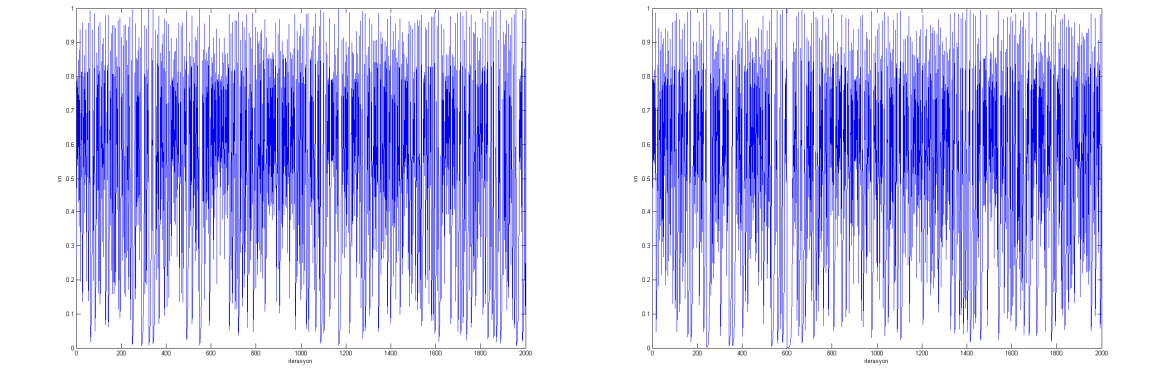
(a) (b)

Şekil 3.1. a) X_0=0.31 başlangıç şartlı lojistik harita b) X_0=0.3133 başlangıç şartlı lojistik harita

3.2.2. Çadır Harita

Çadır harita [111] lojistik haritaya benzemektedir. Bu da (0, 1) aralığında sayılar üretmektedir ve formülü Denklem (3.3)'te gösterilmiştir.

$$X_{n+1} = \begin{cases} X_n / 0.7 & , X_n < 0.7 \\ 10/3 X_n(1 - X_n) & , \text{degilse} \end{cases} \tag{3.3}$$



(a) (b)

Şekil 3.2. a) X_0=0.27 başlangıç şartlı çadır harita b) X_0=0.27114 başlangıç şartlı çadır harita

3.2.3. Sinüzoidal Yineleyici

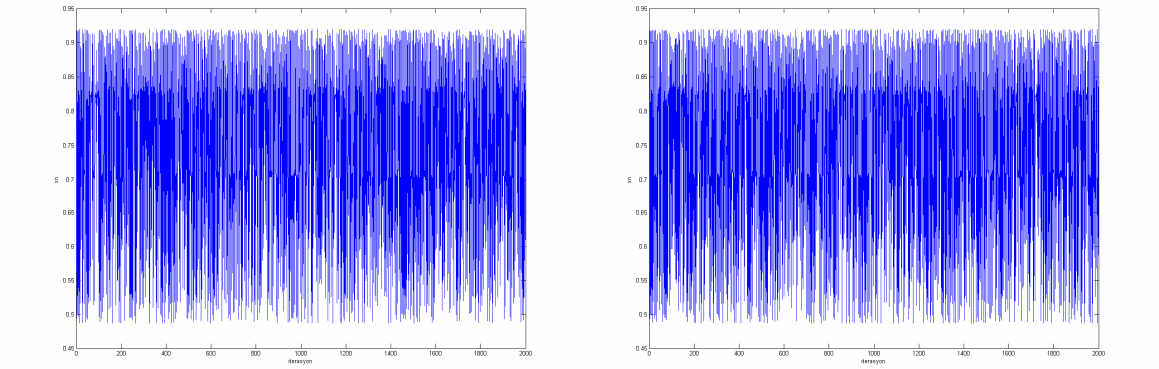
Kullanılan üçüncü üreteç sinüzoidal yineleyici [111] olarak adlandırılmaktadır ve Denklem (3.4)'teki gibi temsil edilmektedir.

$$X_{n+1} = ax_n^2 \sin(\pi x_n) \tag{3.4}$$

a=2.3 ve X_0=0.7 seçildiğinde denklem (3.5)'teki gibi basitleştirilebilir.

$$X_{n+1} = \sin(\pi x_n) \tag{3.5}$$

Bu harita da (0, 1) aralığında sayılar üretmektedir.



(a) (b)

Şekil 3.3. a) X_0=0.5637 başlangıç şartlı sinüzoidal yineleyici b) X_0=0.5637112 başlangıç şartlı sinüzoidal yineleyici

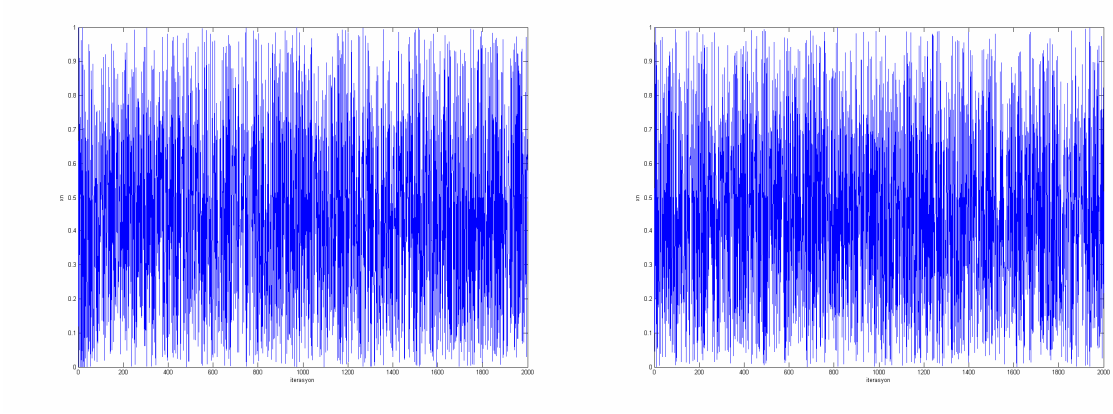
3.2.4. Gauss Haritası

Literatürde test amaçlı kullanılan Gauss haritası [111] Denklem (3.6)'daki gibi temsil edilmektedir.

$$X_{n+1} = \begin{cases} 0 & , X_n = 0 \\ 1/X_n \bmod(1) & , X_n \in (0,1) \end{cases} \tag{3.6}$$

$$1/X_n \bmod(1) = \frac{1}{X_n} - \left\lfloor \frac{1}{X_n} \right\rfloor \tag{3.7}$$

$\lfloor z \rfloor$, z'den küçük en büyük tamsayıyı temsil etmektedir. Bu harita da (0, 1) aralığında sayılar üretmektedir.



(a) (b)

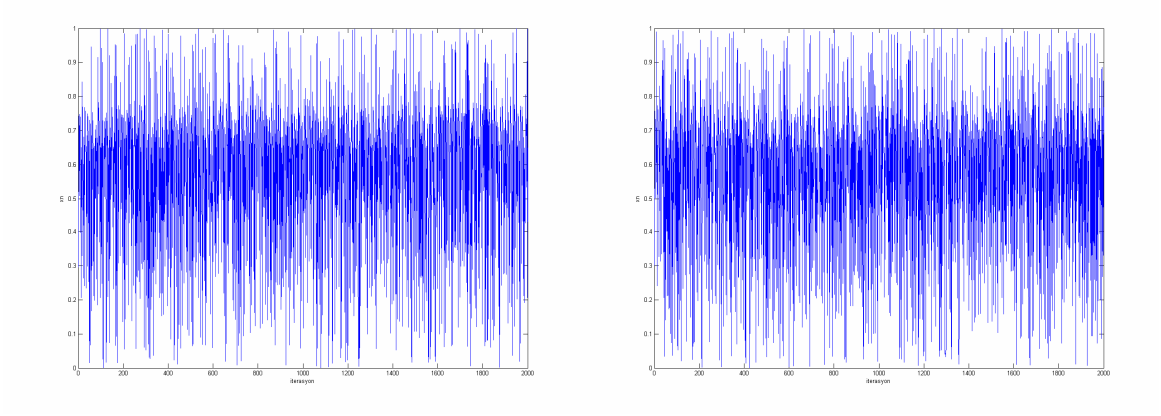
Şekil 3.4. a) X_0=0.12 başlangıç şartlı Gauss harita b) X_0=0.12345 başlangıç şartlı Gauss harita

3.2.5. Çember Harita

Çember harita [112] Denklem (3.8)'de gösterildiği şekilde temsil edilmektedir.

$$X_{n+1} = X_n + b - (a/2\pi)\sin(2\pi X_n) \bmod(1) \tag{3.8}$$

a=0.5 ve b=0.2 seçildiğinde bu harita da (0, 1) aralığında sayılar üretmektedir.



(a) (b)

Şekil 3.5. a) X_0=0.43 başlangıç şartlı çember harita b) X_0=0.43111 başlangıç şartlı çember harita

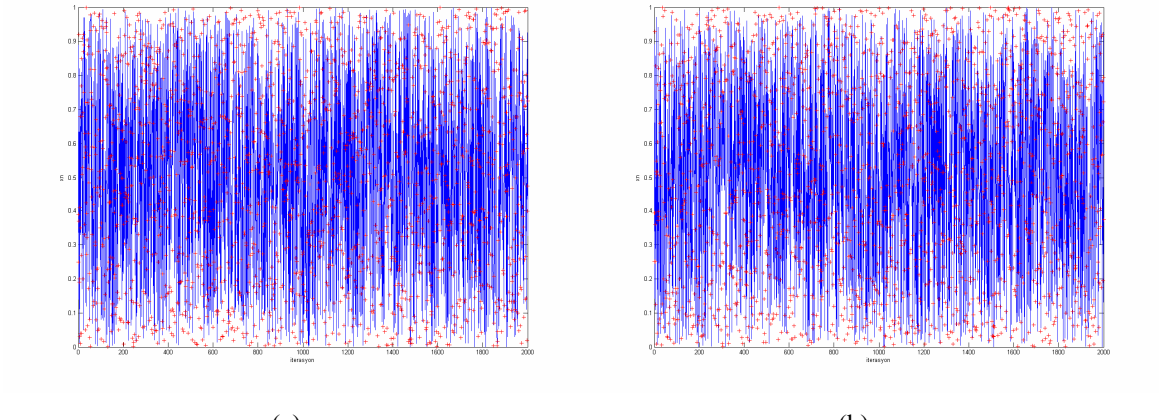
3.2.6. Arnold'ın Kedi Haritası

Arnold'ın kedi haritası 1960'larda bir kedinin resmini kullanarak etkilerini inceleyen Vladimir Arnold'dan sonra ismini almıştır. Denklem (3.9, 3.10) haritanın formülünü göstermektedir:

$$X_{n+1} = X_n + Y_n \bmod(1) \tag{3.9}$$

$$Y_{n+1} = X_n + 2Y_n \bmod(1) \tag{3.10}$$

$X_n \in (0, 1)$ ve $Y_n \in (0, 1)$ olduğu açıktır [113].



(a) (b)

Şekil 3.6. a) X_0=0.89 ve Y_0=0.25 başlangıç şartlı Arnold'ın kedi haritası b) X_0=0.89111 ve Y_0=0.25111 başlangıç şartlı Arnold'ın kedi haritası

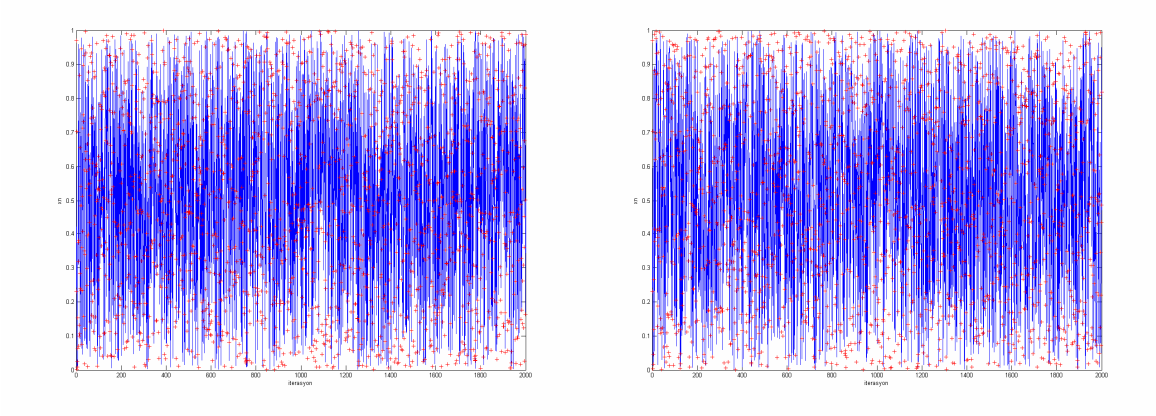
3.2.7. Sina Haritası

Sina haritası [114]

$$X_{n+1} = X_n + Y_n + a\cos\left(2\pi Y_n\right)\bmod(1) \tag{3.11}$$

$$Y_{n+1} = X_n + 2Y_n \bmod(1) \tag{3.12}$$

denklemleriyle tanımlanır. a=1 seçildiğinde bu harita (0, 1) aralığında sayılar üretir.



(a) (b)

Şekil 3.7. a) X_0=0.29 ve Y_0=0.97 başlangıç şartlı Sina haritası b) X_0=0.29012 ve Y_0=0.97012 başlangıç şartlı Sina haritası

3.2.8. Zaslavskii Haritası

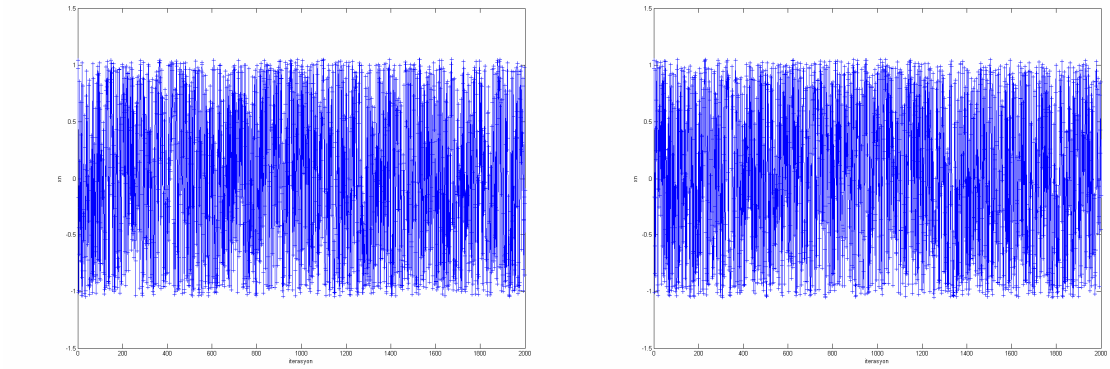
Zaslavskii haritası [115] da ilginç bir dinamik sistemdir ve

$$X_{n+1} = \left(X_n + v + aY_{n+1}\right)\bmod(1) \tag{3.13}$$

$$Y_{n+1} = \cos\left(2\pi X_n\right) + e^{-r}Y_n \tag{3.14}$$

şeklinde temsil edilmektedir.

Yayılmış karakteristikli ve büyük Lyapunov üstelleri ile tahmin edilemezliği ispatlanmıştır. Bu harita v=400, r=3, a=12 için en büyük Lyapunov üstelli garip çekici özelliği gösterir. Bu durumda $Y_{n+1} \in$ [-1.0512, 1.0512]'dir.



(a) (b)

Şekil 3.8. a) X_0=0.984, Y_0=0.971 başlangıç şartlı Zaslavskii Haritası **b)** X_0=0.985, Y_0=0.9709 başlangıç şartlı Zaslavskii Haritası

3.3. Kaotik Haritalı Parçacık Sürü Optimizasyon (KHPSO) Yöntemleri

Yeni kaotik haritalı PSO yöntemleri basitçe aşağıda sınıflandırılmış ve açıklanmıştır.

- **KHPSO1:** Başlangıç hız ve pozisyonların değerleri, sürü boyutuna ulaşılıncaya kadar seçilen kaotik haritanın yinelenmesiyle üretilir**.**

- **KHPSO2:** Denklem (2.13)'ün c_1 parametresi seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + KH_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \tag{3.15}$$

şeklinde ifade edilir. Burada KH_1 seçilen kaotik harita tabanlı bir fonksiyondur ve (0.5, 2.5) arasında değer alacak şekilde ölçeklenmiştir.

- **KHPSO3:** Denklem (2.13)'ün c_2 parametresi seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + KH_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \tag{3.16}$$

şeklinde ifade edilir. Burada KH_2 seçilen kaotik harita tabanlı bir fonksiyondur ve (0.5, 2.5) arasında değer alacak şekilde ölçeklenmiştir.

- **KHPSO4:** Denklem (2.13)'ün c_1 ve c_2 parametreleri seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + KH_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + KH_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.17)$$

şeklinde ifade edilir. Burada KH_1 ve KH_2 seçilen kaotik harita tabanlı fonksiyonlardır ve (0.5, 2.5) arasında değer alacak şekilde ölçeklenmiştir.

- **KHPSO5**: Denklem (2.13)'ün $r_{1,j}$ parametresi seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 KH_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.18)$$

şeklinde ifade edilir. Burada $KH_{1,j}$ seçilen kaotik harita tabanlı bir fonksiyondur ve (0.0, 1.0) arasında değerler alır.

- **KHPSO6:** Denklem (2.13)'ün $r_{2,j}$ parametresi seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 KH_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.19)$$

şeklinde ifade edilir. Burada $KH_{2,j}$ seçilen kaotik harita tabanlı bir fonksiyondur ve (0.0, 1.0) arasında değerler alır.

- **KHPSO7**: Denklem (2.13)'ün $r_{1,j}$ ve $r_{2,j}$ parametreleri seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 KH_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 KH_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.20)$$

şeklinde ifade edilir. Burada $KH_{1,j}$ ve $KH_{2,j}$ seçilen kaotik harita tabanlı fonksiyonlardır ve (0.0, 1.0) arasında değerler alır.

- **KHPSO8**: Denklem (2.13)'ün w, $r_{1,j}$ ve $r_{2,j}$ parametreleri seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = KH_1 v_{i,j}(t) + c_1 KH_{2,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 KH_{3,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.21)$$

şeklinde ifade edilir. Burada KH_1, KH_2, ve KH_3 seçilen kaotik harita tabanlı fonksiyonlardır ve (0.0, 1.0) arasında değerler alır.

- **KHPSO9:** Denklem (2.13)'ün w parametresi seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = KHv_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.22)$$

şeklinde ifade edilir. Burada KH seçilen kaotik harita tabanlı bir fonksiyondur ve (0.0, 1.0) arasında değer alır.

- **KHPSO10:** Denklem (2.13)'ün w ve c_1 parametreleri seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = KH_1 v_{i,j}(t) + KH_2 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.23)$$

şeklinde ifade edilir. Burada KH_1 ve KH_2 seçilen kaotik harita tabanlı fonksiyonlardır. KH_1 (0.0, 1.0) arasında değer alır ve KH_2 (0.5, 2.5) arasında değer alacak şekilde ölçeklenmiştir.

- **KHPSO11:** Denklem (2.13)'ün w ve c_2 parametreleri seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = KH_1 v_{i,j}(t) + r_2 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + KH_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.24)$$

şeklinde ifade edilir. Burada KH_1 ve KH_2 seçilen kaotik harita tabanlı fonksiyonlardır. KH_1 (0.0, 1.0) arasında değer alır ve KH_2 (0.5, 2.5) arasında değer alacak şekilde ölçeklenmiştir.

- **KHPSO12:** Denklem (2.13)'ün w, c_1 ve c_2 parametreleri seçilen kaotik haritayla değiştirilir ve hız güncelleme denklemi

$$v_{i,j}(t+1) = KH_1 v_{i,j}(t) + KH_2 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + KH_3 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (3.25)$$

şeklinde ifade edilir. Burada KH_1, KH_2 ve KH_3 seçilen kaotik harita tabanlı fonksiyonlardır. KH_1 (0.0, 1.0) arasında değer alırken KH_2 ve KH_3 (0.5, 2.5) arasında değer alacak şekilde ölçeklenmiştir.

Burada KHPSO1' in diğer KHPSO yöntemleriyle birlikte kullanılabileceği açıkça görülebilmektedir.

3.4. Kalite Testi Fonksiyonları

Matematiksel fonksiyonlara bağlı iyi tanımlanmış kalite testi fonksiyonları, optimizasyon yöntemlerinin performanslarını ölçmek ve test etmek için kullanılabilir. Bu kalite testi fonksiyonlarının doğası, karmaşıklığı ve diğer özellikleri tanımlarından kolaylıkla elde edilebilmektedir. Çoğu kalite testi fonksiyonların zorluk dereceleri parametrelerinin değiştirilmesiyle ayarlanabilir. Literatürde bulunan kalite testi fonksiyonlarından biri tek- modlu (sadece tek bir optimuma sahip) ikisi de çok-modlu (birçok lokal optimuma ancak tek bir global optimuma sahip) olmak üzere üç tane önemli fonksiyon, önerilen yöntemlerin istenen sonucu verebilme yeteneklerini test etmek için seçilmiştir. Bu fonksiyonlar evrimsel hesaplama ve diğer algoritma araştırıcıları tarafından yaygın olarak kullanılmıştır. Çünkü bu fonksiyonların karmaşıklıklıları çoğu mühendislik problemiyle eşdeğerdir. Tablo 3.1'de deneylerde kullanılan kalite testi fonksiyonlarının temel özellikleri görülmektedir. Aşağıdaki alt bölümlerde, bu fonksiyonların karakteristikleri açıklanmıştır.

Tablo 3.1. Kalite testi fonksiyonları özellikleri

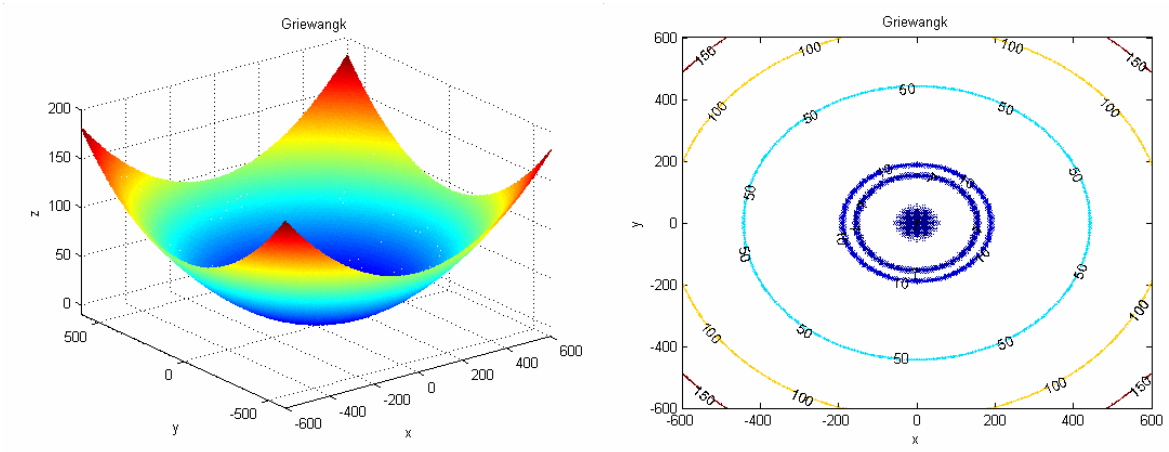
Fonksiyon numarası	***Fonksiyon adı***	***Alt sınır***	***Üst sınır***	***Optimum***	***Şekil***
1	Griewangk	-600	600	0	Çok-modlu
2	Rastrigin	-5.12	5.12	0	Çok-modlu
3	Rosenbrock	-2.048	2.048	0	Tek-modlu

3.4.1. Griewangk Fonksiyonu

Birinci test fonksiyonu düzenli dağıtılmış birçok yaygın lokal minimuma sahiptir [116]. Bu fonksiyon sürekli, çok-modlu, ölçeklenebilir, konveks ve ikinci dereceden bir fonksiyondur ve

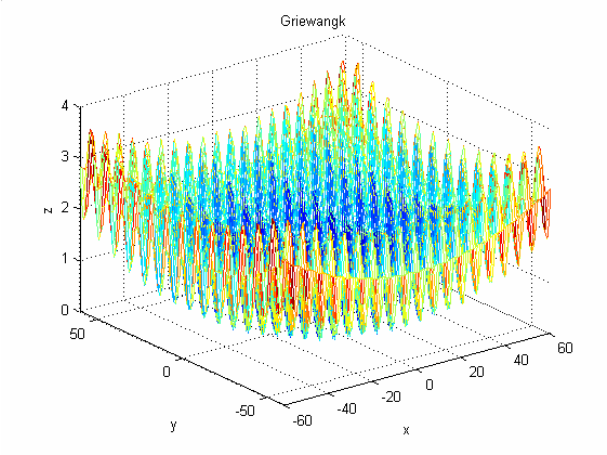
$$f_1(x) = \sum_{i=1}^{30}\left(\frac{x_i^2}{4000}\right) - \prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (3.26)$$

şeklinde ifade edilir. Sınırlar $-600 \le x \le 600$ olacak şekilde seçilmiştir. Global minimum noktası $x^* = (0, 0, \ldots, 10)$ ve $f_1(x^*) = 0$'dır. Şekil 3.9'da verilen sınırlarda fonksiyonun grafiği ve kontur eğrileri gösterilmiştir. Şekil 3.10'da ise fonksiyon, grafiği azaltılmış sınırlarla görülmektedir. Tanımındaki toplama terimi fonksiyona parabollük özelliği kazandırmaktadır. Bu fonksiyonda yerel minimum derecesi parabollük derecesinden daha üst seviyededir. Çarpım terimi baz alınarak arama uzayının boyutları artırılmakta ve yerel minimumların sayısı azaltılmaktadır. Arama aralığı ne kadar fazlaysa fonksiyon daha yatık (yassı) halde görülmektedir.



(a) (b)

Şekil 3.9. a) Griewangk fonksiyonu **b)** Kontur eğrileri



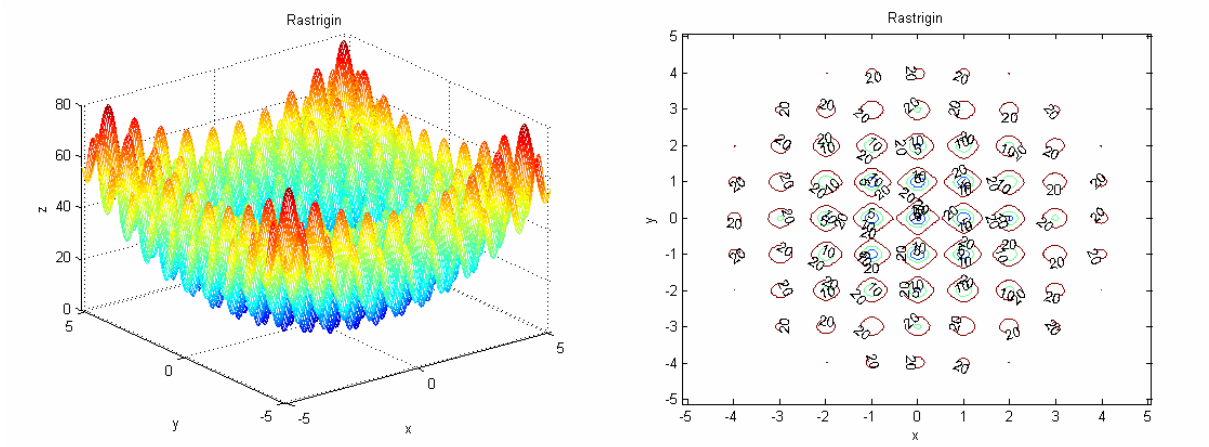
Şekil 3.10. Azaltılmış arama aralıklı Griewangk fonksiyonu

3.4.2. Rastrigin Fonksiyonu

İkinci kalite testi fonksiyonu da geniş arama uzayı ve çok sayıda lokal optimum noktalardan dolayı oldukça zor bir problemdir. Fonksiyon oldukça çok-modludur ve doğrusal değildir [117]. Lokal minimumların yeri düzgün dağılmıştır. Fonksiyon

$$f_2(x) = 10 \times 30 + \sum_{i=1}^{30} (x_i^2 - 10 \cdot \cos(2\pi x_i)) \tag{3.27}$$

şeklinde tanımlanır. Sınırlar $-5.12 \le x \le 5.12$ olacak şekilde seçilmiştir. Global minimum nokta $x^* = (0, 0, \ldots, 0)$ ve $f_2(x^*) = 0$. Fonksiyonun grafiği ve kontur eğrileri Şekil 3.11'de görülmektedir.



(a) (b)

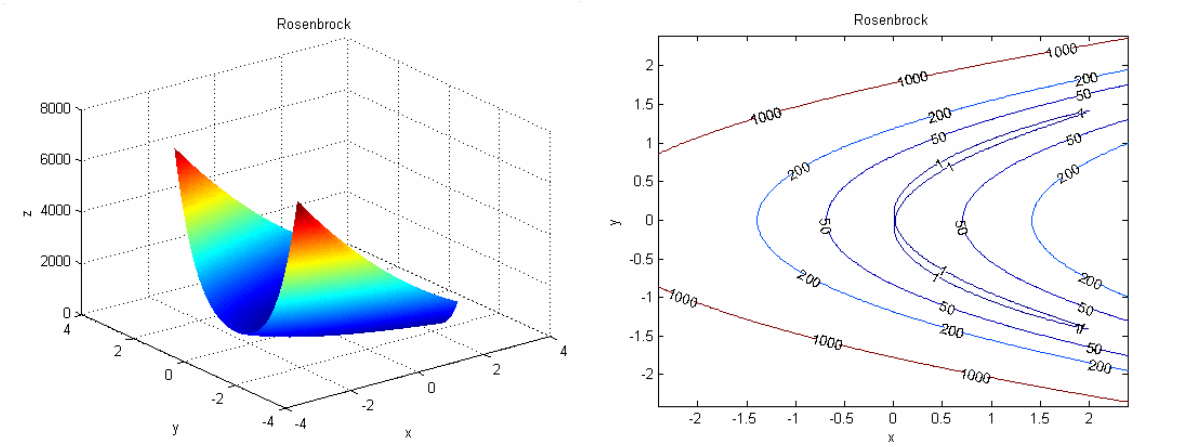
Şekil 3.11. a) Rastrigin fonksiyonu **b)** Kontur eğrileri

3.4.3. Rosenbrock Fonksiyonu

Üçüncü kalite testi fonksiyonu bazı değerleri arasında önemli etkileşimleri olan tek-modlu bir fonksiyondur. Birçok dar tepecik içerdiğinden dolayı zor bir fonksiyon olarak düşünülmektedir. Tepe noktaları çok keskindir. Global minimum nokta uzun, dar, parabolik şekilli düz bir vadide yer almaktadır. Vadiyi bulmak kolay olabilir ancak global optimuma yakınsamak zordur. Algoritmalar bu problemde ilerlenebilecek iyi noktaları tayin edememektedir. Bu yüzden bu fonksiyon optimizasyon algoritmalarının performanslarını ölçmek için sürekli kullanılmıştır [118, 119]. Fonksiyon

$$f_3(x) = \sum_{i=1}^{30} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{3.28}$$

şeklinde tanımlanır. Sınırlar $-2.048 \le x \le 2.048$ olacak şekilde seçilmiştir. Global minimumu $f_3(x^*) = 0$'dır ve $x^* = (1, 1, \ldots, 1)$'de yerleşmiştir. İki boyut için grafiği ve kontur eğrileri Şekil 12'de gösterilmiştir.



(a) (b)

Şekil 3.12. a) İki değişkenli Rosenbrock fonksiyonu **b)** Kontur eğrileri

3.5. Deneysel Sonuçlar

Tüm kalite testi fonksiyonları KHPSO yöntemleri ve literatürde iyi performans gösterdiği belirtilen diğer PSO yöntemleriyle çözülmüştür. Seçilen topoloji tüm parçacıkların diğerleriyle komşu olarak kabul edildiği *global* modeldir. Sürü boyutu tüm deneyler için 25 olarak belirlenmiştir. Algoritmaların sonlandırılması için, maksimum iterasyon sayısına ulaşma

ve minimum hatayı yakalama şeklinde iki kriter seçilmiştir. Aşağıdaki alt bölümlerde sırasıyla test fonksiyonları için elde dilen sonuçlar ve yorumlar sunulmuştur.

3.5.1. Griewangk Fonksiyonu için Sonuçlar

Griewangk fonksiyonu için farklı kaotik haritalar kullanan KHPSO yöntemleri ve literatürde iyi sonuçlar verdiği belirtilmiş üç PSO yönteminin 50 simülasyonu boyunca elde edilen uygunluk değerinin beş istatistiksel analizi yapılmıştır ve aşağıdaki tablolarda sırayla verilmiştir. İterasyon sayısı 2000 ve minimum hata 0.0001 seçilmiştir. Burada *bPSO* zamanla değişen atalet ağırlığı (*ZDAA*) [61] ve zamanla değişen hızlanma katsayılarını [81] içeren birleşimli PSO'yu temsil etmektedir. Zamanla değişen atalet ağırlığı için hızlandırma katsayıları literatürde de kullanılan $c_1 = c_2 = 2.00$ seçilmiştir. GLEnIyiPSO yöntemi de ikinci bölümde anlatıldığı gibi atalet ağırlığı ve hızlandırma katsayılarını, parçacıkların global ve lokal en iyi pozisyonlarındaki terimleri cinsinden içerir [82]. $V_{\max}$ tüm PSO yöntemleri için 600 olarak belirlenmiştir.

Tüm algoritmalar 50 kez çalıştırıp sonuçlar kaydedilmiştir. Adil bir değerlendirme için sürü tüm PSO'lar için global optimumu içermeyen sınırlar içerisinde başlatılmıştır. Kaydedilen sonuçlardan istatistiksel analizler gerçekleştirilmiş ve sırayla Tablo 3.2, Tablo 3.3 ve Tablo 3.4'te sunulmuştur. Her yöntem için Ortalama (Ort), En İyi (Min), En Kötü (Maks), Medyan (Med), Standart Sapma (SS) ve optimum değere yakınsamak için gereken ortalama iterasyon sayısı (Ort İter) hesaplanmış ve sonuçlar karşılaştırılmıştır.

Tablo 3.2. Griewangk fonksiyonu için PSO yöntemleri ile elde edilen sonuçlar

F1	bPSO	GLEnIyiPSO	ZDAA-PSO
Ort	0.034957333	0.014298924	0.012536245
Maks	0.181849495	0.083306690	0.051600123
Min	0.000100516	0.000074666	0.00008.8762
Med	0.017935979	0.009857288	0.007412101
SS	0.048103056	0.018505321	0.014532523
Ort İter	1395.89	934.82	1398.55

Sonuçlardan w, $r_{1,j}$ ve $r_{2,j}$'yi ayarlayan KHPSO7 ve KHPSO8'in diğer PSO yöntemlerine karşı üstünlüğü görülmektedir. Genel olarak bakıldığında da, yeni önerilen yöntemlerin sadece global optimum değeri üretmekle kalmayıp, daha hızlı bir yakınsama oranı verdiği görülmektedir. Önerilen yöntemlerdeki düşük standart sapma değeri de global optimum değeri elde etmede tutarlılık derecesini garantilemektedir. Diğer ilginç sonuç ise, iyi sonuçların kaotik sayı dizileri için Zaslavskii, Lojistik ve Çadır haritalar kullanıldığında elde edilmesidir.

Tablo 3.3. Griewangk fonksiyonu için ilk dört haritayı kullanan KHPSO yöntemleri ile elde edilen sonuçlar

Lojistik Harita												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0000961	0.0000792	0.0000862	0.0000972	0.0249573	0.0152989	0.0000981	0.0000831	0.0142989	0.0012639	0.0192392	0.0102889
Maks	0.0001129	0.0001466	0.0001166	0.0001313	0.0818494	0.0933066	0.0001319	0.0000995	0.0833066	0.0893656	0.0863066	0.0793556
Min	0.0000638	0.0000595	0.0000583	0.0000635	0.0001012	0.0000986	0.0000638	0.0000538	0.0000746	0.0001002	0.0000846	0.0000899
Med	0.0000990	0.0000991	0.0000792	0.0000871	0.0195359	0.0118683	0.0000991	0.0000870	0.0098572	0.0158425	0.0011572	0.0158426
SS	0.0000203	0.0000265	0.0000195	0.0000205	0.0471330	0.0215053	0.0000234	0.0000143	0.0185053	0.0159952	0.0195564	0.0157952
Ort İter	781.12	986.18	799.68	902.47	1309.89	1103.85	693.54	521.16	921.0	1399.44	934.01	1329.87
Çadır Harita												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0001366	0.0000942	0.0001112	0.0000873	0.0145673	0.0096298	0.0002696	0.0000963	0.0042989	0.0012359	0.0142392	0.0142989
Maks	0.0001829	0.0001653	0.0001964	0.0001293	0.0791844	0.0993064	0.0005319	0.0001587	0.0803560	0.0905969	0.0903066	0.0893453
Min	0.0000332	0.0000335	0.0000693	0.0000666	0.0001000	0.0001008	0.0000293	0.0000538	0.0000114	0.0001002	0.0001006	0.0001009
Med	0.0000860	0.0000881	0.0000771	0.0000921	0.0189535	0.0118685	0.0001391	0.0000893	0.0098575	0.0098245	0.0061572	0.0585846
SS	0.0000253	0.0000366	0.0001095	0.0000355	0.0447133	0.0483205	0.0000294	0.0000263	0.0395055	0.0999785	0.0265564	0.0357652
Ort İter	993.36	1050.44	1421.11	1125.12	1305.44	1336.11	798.06	598.46	1002.28	1209.98	1119.66	1032.22
Sinüzoidal Yineleyici												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0001168	0.0001112	0.0002162	0.0001035	0.0188673	0.0085298	0.0001696	0.0000923	0.0058989	0.0011235	0.0012391	0.0001213
Maks	0.0002229	0.0001853	0.0004964	0.0003293	0.0791844	0.0883064	0.0004029	0.0001477	0.0806513	0.0863596	0.0063066	0.0001753
Min	0.0000732	0.0000413	0.0000993	0.0000866	0.0001005	0.0001006	0.0000238	0.0000498	0.0000174	0.0001003	0.0000904	0.0000313
Med	0.0000980	0.0000981	0.0002771	0.0000991	0.0183635	0.0118235	0.0001281	0.0000749	0.0072573	0.0058245	0.0091572	0.0000878
SS	0.0000363	0.0000426	0.0002695	0.0000355	0.0314713	0.0283205	0.0000282	0.0000226	0.0199505	0.0339785	0.0765564	0.0000419
Ort İter	1009.58	1101.41	1002.44	1101.12	1015.44	1205.11	802.36	591.33	1362.38	1010.98	993.13	1021.36
Gauss Harita												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0009235	0.0001012	0.0001962	0.0001115	0.0001112	0.0079826	0.0001301	0.0001123	0.0000942	0.0000814	0.0000961	0.0192495
Maks	0.0005293	0.0002058	0.0004964	0.0004013	0.0001859	0.0990648	0.0004139	0.0001977	0.0001654	0.0001663	0.0001929	0.0791849
Min	0.0000769	0.0000313	0.0000999	0.0000796	0.0000413	0.0001002	0.0000168	0.0000498	0.0000339	0.0000492	0.0000342	0.0001002
Med	0.0009923	0.0000999	0.0002771	0.0000991	0.0000982	0.0082359	0.0001281	0.0000949	0.0000888	0.0000863	0.0000921	0.0193635
SS	0.0001355	0.0000726	0.0002695	0.0000385	0.0000429	0.0296050	0.0000232	0.0000255	0.0000366	0.0000266	0.0000296	0.0481030
Ort İter	1102.12	1001.47	1052.44	1221.21	1258.88	1205.06	962.36	639.92	1204.55	891.96	892.28	1209.91

Tablo 3.4. Griewangk fonksiyonu için 5., 6., 7. ve 8. haritayı kullanan KHPSO yöntemleri ile elde edilen sonuçlar

Çember Harita												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0001166	0.0001942	0.0019112	0.0000873	0.0149573	0.0142495	0.0002936	0.0001896	0.0010312	0.0012359	0.0215673	0.0406298
Maks	0.0001925	0.0002653	0.0041969	0.0001293	0.0956849	0.0791849	0.0005319	0.0004373	0.0022158	0.0893596	0.0801844	0.0803064
Min	0.0000838	0.0000435	0.0001003	0.0000666	0.0001005	0.0001012	0.0000293	0.0000338	0.0001003	0.0001002	0.0001000	0.0001008
Med	0.0000990	0.0000981	0.0018773	0.0000921	0.0199359	0.0196835	0.0001391	0.0001281	0.0009999	0.0098245	0.0190035	0.0418685
SS	0.0000153	0.0000466	0.0009095	0.0000355	0.0491630	0.0491030	0.0000294	0.0000312	0.0001026	0.0999785	0.0460633	0.0383205
Ort İter	952.36	1034.49	1121.11	1125.12	1465.89	1129.91	1001.06	932.39	1201.47	1209.98	1112.44	1206.15
Arnold'ın Kedi Haritası												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0005613	0.0001124	0.0000832	0.0000972	0.0001122	0.0208825	0.0005696	0.0002962	0.0055989	0.0042358	0.0132395	0.0142984
Maks	0.0008129	0.0001466	0.0001166	0.0001313	0.0001999	0.0800648	0.0009983	0.0004965	0.0913560	0.0888992	0.0803066	0.0803459
Min	0.0000738	0.0000995	0.0000983	0.0000635	0.0000517	0.0001007	0.0000985	0.0000999	0.0000214	0.0001003	0.0001003	0.0001009
Med	0.0006990	0.0000109	0.0000892	0.0000871	0.0000982	0.0381235	0.0005281	0.0002778	0.0063573	0.0063943	0.0591577	0.0585843
SS	0.0000513	0.0000195	0.0000225	0.0000209	0.0000429	0.0296050	0.0001082	0.0006695	0.0239505	0.0899786	0.0865564	0.0627652
Ort İter	962.90	986.18	1036.08	1483.18	1018.88	1409.63	869.66	1006.46	1008.88	1201.36	1087.66	1306.02
Sina Haritası												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0001696	0.0001543	0.0022112	0.0000873	0.0135957	0.0122988	0.0000892	0.0000826	0.0102989	0.0011139	0.0102395	0.0092889
Maks	0.0002012	0.0002513	0.0055962	0.0001293	0.1956849	0.0803063	0.0001319	0.0000989	0.0803066	0.0808855	0.0625065	0.0793556
Min	0.0001004	0.0000835	0.0001003	0.0000666	0.0000802	0.0000986	0.0000238	0.0000538	0.0000590	0.0001000	0.0000700	0.0000899
Med	0.0000155	0.0001611	0.0019770	0.0000921	0.0999359	0.0116683	0.0000991	0.0000853	0.0108578	0.0103052	0.0095728	0.0158426
SS	0.0000159	0.0001066	0.0009990	0.0000255	0.0493130	0.0205053	0.0000234	0.0000183	0.0185158	0.0150452	0.0102556	0.0157952
Ort İter	1003.37	1021.49	12631.11	1105.12	1109.09	1013.68	689.29	7091.62	932.05	1120.39	941.01	1329.87
Zaslavskii Haritası												
F1	KHPSO1	KHPSO2	KHPSO3	KHPSO4	KHPSO5	KHPSO6	KHPSO7	KHPSO8	KHPSO9	KHPSO 10	KHPSO 11	KHPSO 12
Ort	0.0000961	0.0000884	0.0000826	0.0000972	0.0098673	0.0100983	0.0000892	0.0000881	0.0112985	0.0000817	0.0000991	0.0008982
Maks	0.0001219	0.0001366	0.0001066	0.0001210	0.0291844	0.0803063	0.0001319	0.0000995	0.0833060	0.0001663	0.0001929	0.0893453
Min	0.0000609	0.0000420	0.0000500	0.0000635	0.0001000	0.0000882	0.0000198	0.0000410	0.0000724	0.0000493	0.0000342	0.0000900
Med	0.0000991	0.0000981	0.0000788	0.0000887	0.0103035	0.0106683	0.0000991	0.0000870	0.0098342	0.0000863	0.0000101	0.0585846
SS	0.0000203	0.0000265	0.0000195	0.0000210	0.0314713	0.0019505	0.0000224	0.0000130	0.0174053	0.0000276	0.0000276	0.0356652
Ort İter	789.33	979.10	729.61	913.44	1002.02	1117.06	687.08	589.59	906.09	897.37	899.27	1009.35

3.5.2. Rastrigin Fonksiyonu için Sonuçlar

Algoritmaların stokastik özelliklerini görebilmek için her biri yüz kez çalıştırılmıştır. Bu deneyde maksimum iterasyon 5000 olarak belirlenmiş ve algoritmaların potansiyellerinin bulunması amaçlanmıştır. Denklem (3.29)'da tanımlanmış algoritma başarı oranı, farklı PSO yöntemlerinden elde edilen sonuçların karşılaştırılması amacıyla kullanılmıştır.

$$S = 100 \frac{NT_{basarili}}{NT_{tum}} \Big|_{Q_{seviye}} \tag{3.29}$$

$N_{basarili}$ izin verilen maksimum iterasyon sayısında Q_{seviye} üzerinde sonuç bulan deneme sayısıdır. $N_{tüm}$ tüm deneme sayısıdır. Q_{seviye}, algoritma Q_{seviye} toleransına yakınsadığında algoritmayı sonlandırma şartıdır ve

$$\left| f\left(x_i(t)\right) - f\left(x^*\right) \right| < Q_{seviye} \tag{3.30}$$

şeklinde tanımlanır.

Tablo 3.5'te sınırlayıcısız temel algoritma (Temel-PSO), ZDAA-PSO [74] ve stokastik atalet ağırlıklı PSO (StZDAA-PSO) [75] algoritmalarından elde edilen sonuçlar görülmektedir. StZDAA-PSO için atalet ağırlığı değerler (0.5, 1) aralığındadır. Önerilen KHPSO yöntemlerinden elde edilen sonuçlar ise Tablo 3.6 ve Tablo 3.7'de gösterilmiştir. Bu zor problem için, KHPSO yöntemlerinin diğer PSO yöntemlerine göre biraz daha iyi sonuç verdiği görülmektedir. Kısmi kalitelerin (Q_{seviye}) toplanmasıyla hesaplanmış farklı kalite seviyelerinde KHPSO yöntemlerinin toplam başarı oranı Tablo 3.8'de gösterilmiştir. KHPSO7, KHPSO8 ve KHPSO12 yöntemlerinin en iyi performansa sahip yöntemler olduğu görülmektedir. KHPSO7 Zaslavskii haritasıyla kullanıldığında en iyi sonuç alınmıştır.

Tablo 3.5. Rastrigin fonksiyonu için PSO algoritmalarının başarı oranları

Q_{seviye}	Temel-PSO	ZDAA-PSO	StZDAA-PSO
1	0	0	0
10	0	1	2

Tablo 3.6. Rastrigin fonksiyonu için ilk dört haritayı kullanan KHPSO algoritmalarının başarı oranları

Lojistik Harita												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	1	1	0	1	1	1	1	2	1	1	2
10	2	2	3	2	3	3	3	5	3	2	2	4
Çadır Harita												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	1	0	0	0	1	1	2	1	0	1	1
10	1	2	2	2	2	2	3	3	1	2	2	2
Sinüzoidal Yineleyici												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	0	1	1	1	1	1	0	0	0	1	1
10	1	2	2	2	2	2	4	3	2	1	1	2
Gauss Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	0	0	0	1	0	1	0	0	1	1	1
10	1	1	1	1	1	1	2	2	1	1	1	2

Tablo 3.7. Rastrigin fonksiyonu için 5., 6., 7. ve 8. haritayı kullanan KHPSO algoritmalarının başarı oranları

Çember Harita												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	1	0	0	0	1	1	1	1	0	0	1
10	2	2	1	1	1	2	3	3	1	2	2	2
Arnold'ın Kedi Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	1	1	0	0	1	1	2	1	1	0	2
10	2	2	2	2	1	2	3	3	2	3	2	3
Sina Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	1	1	0	1	1	0	1	0	0	0	2
10	2	2	2	3	2	2	4	3	1	2	2	3
Zaslavskii Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	1	0	1	1	1	1	2	0	0	0	0
10	2	3	2	3	2	4	5	4	2	2	2	3

Tablo 3.8. Rastrigin fonksiyonu için farklı kalite seviyelerinde KHPSO yöntemlerinin toplam başarı oranları

Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	5	6	4	2	5	7	7	9	5	3	4	**10**
10	13	16	15	16	14	19	**27**	26	13	15	14	21

3.5.3. Rosenbrock Fonksiyonu için Sonuçlar

Tablo 3.9'da Rosenbrock fonksiyonu için sınırlayıcısız temel algoritma (Temel-PSO), ZDAA-PSO ve stokastik atalet ağırlıklı PSO (StZDAA-PSO) algoritmalarından elde edilen sonuçlar görülmektedir. KHPSO yöntemlerinden elde edilen sonuçlar ise Tablo 3.10, Tablo 3.11 ve Tablo 3.12'de gösterilmiştir.

Tablo 3.9. Rosenbrock fonksiyonu için PSO algoritmalarının başarı oranları

Q_{seviye}	Temel-PSO	ZDAA-PSO	ZDAAStochastic-PSO
1	0	0	0
10	0	1	2

Tablo 3.10. Rosenbrock fonksiyonu için ilk dört haritayı kullanan KHPSO algoritmalarının başarı oranları

Lojistik Harita												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	1	0	0	0	1	1	0	0	0	0	1
10	1	1	2	2	2	3	4	3	1	2	2	2
Çadır Harita												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	1	1	0	0	1	1	1	0	0	1	1
10	1	1	1	2	1	3	3	3	2	2	1	2
Sinüzoidal Yineleyici												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	0	1	0	0	1	1	1	0	0	0	1
10	1	1	2	1	1	2	3	3	2	1	2	2
Gauss Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	0	1	0	1	1	1	0	0	0	1	1
10	1	1	1	2	1	1	3	2	0	1	1	2

Bu fonksiyon sonuçlarından da KHPSO yöntemlerinin diğerlerine göre daha iyi sonuç verdiği görülmektedir. Zaslavskii haritası kullanan KHPSO7 yönteminin diğerleri arasında en iyi yöntem olduğu da sonuçlardan kolayca görülebilmektedir. Kaosun optimizasyon problemleri için bazı optimum noktaları arayan bazı yöntemler için aranan bir süreç olduğu anlaşılmaktadır.

Tablo 3.11. Rosenbrock fonksiyonu için 5., 6., 7. ve 8. haritayı kullanan KHPSO algoritmalarının başarı oranları

Çember Harita												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	1	0	0	0	0	1	1	1	0	0	1
10	0	1	1	1	2	1	3	4	1	1	0	2
Arnold'ın Kedi Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	0	0	0	0	1	2	1	0	0	0	1
10	1	2	1	1	1	2	4	3	2	1	1	2
Sina Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	0	0	0	1	1	1	1	2	1	1	0	0
10	1	1	2	2	1	1	3	2	2	2	1	2
Zaslavskii Haritası												
Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	0	0	0	1	1	2	3	1	1	1	2
10	1	1	2	1	2	2	4	4	1	1	2	2

Tablo 3.12. Rosenbrock fonksiyonu için farklı kalite seviyelerinde KHPSO yöntemlerinin toplam başarı oranları

Q_{seviye}	KHPSO 1	KHPSO 2	KHPSO 3	KHPSO 4	KHPSO 5	KHPSO 6	KHPSO 7	KHPSO 8	KHPSO 9	KHPSO 10	KHPSO 11	KHPSO 12
1	1	3	3	1	3	7	**10**	9	3	2	3	8
10	7	9	12	12	11	15	**27**	24	11	11	10	16

3.6. Sonuçlar

Tezin bu bölümünde PSO'nun parametreleri ayarlamak için farklı kaotik haritalar kullanılmıştır. Bu işlem, klasik PSO'da rasgele sayıya her ihtiyaç duyulduğunda kaotik sayı üreteci kullanılarak yapılmıştır. On iki kaotik haritalı PSO yöntemi önerilmiştir ve kalite testi fonksiyonlarında sekiz kaotik harita analiz edilmiştir. PSO ve kompleks dinamik gibi farklı alanlarda gelişen sonuçların birleştirilmesi bazı optimizasyon problemlerinde kaliteyi

arttırabilmektedir ve kaos istenen bir süreç olabilmektedir. Ayrıca özellikle KHPSO7 ve KHPSO8 yöntemlerinin çözüm kalitesini arttırdığı, yani bazı durumlarda lokal çözümlerden kaçarak global arama kabiliyetini arttırdığı görülmüştür. Önerilen bu yöntemler henüz yenidirler. Bunların dağıtık ve paralel versiyonlarıyla optimize edilmiş parametreler kullanılarak ayrıntılı deneyler yapılabilir.

4. KABA PSO

4.1. Giriş

Farklı bilgi tipleri için uygun temsil bulma problemi yapay zekâ ve ilgili çalışma alanlarının süregelen bir konusudur. Bazı pratik durumlarda hesaplamalarda belirsizlik, hata ve değişkenlik içeren sistemleri tam olarak temsil ve modelleme için matematiksel ve hesapsal yöntemler ortaya konulmalıdır ve aralıkları değişken olarak kullanmak tercih edilebilir. Tolerans aralığı sağlamak ve değişkenlerin doğru değerlerini kaydedeme gibi durumlar değişkenlerin aralıklarının kullanma zorunluluğunu ortaya çıkarır [120]. Aralıklarla temsil parçalardan oluşmayan sayısal nitelikler için veri bütünlüğünü sağlar ve bilgi kaybı oluşmasını engeller.

Tezin bu bölümünde de alt ve üst sınırdan oluşan kaba değerlere bağlı olarak PSO'nun genişletilmesi sağlanmış ve aralıkların birer karar değişkeni olarak kullanılması durumunda PSO'ya yapılacak eklentiler gösterilmiştir. Aynı zamanda, bu şekilde kaba hesaplama alanına da ilave bir konu eklenmiştir. Ayrıca seçilen uygulama alanı, nicel nitelikler içeren veri tabanlarında veri madenciliği, için de yeni, etkili ve otomatik bir yöntem önerilmiştir ve bilgi kayıpları ve ön işlemlerden kaçınılarak kural keşfi yapılmıştır.

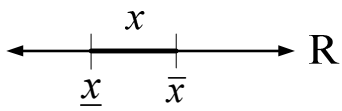
4.2. Kaba Parçacık Sürü Optimizasyon Algoritması (KPSOA)

Lingras tarafından önerilen kaba örüntüler; günlük hava sıcaklığı, hisse bedeli aralığı, arıza sinyali, saatlik trafik hacmi ve günlük finansal göstergeler gibi değişkenler için değişkenler kümesi ya da aralığını etkili şekilde temsil edebilen bir kaba değer oluşturan bir alt ve bir üst sınıra dayanır [121]. Giriş-çıkışta ya da ara safhaların herhangi bir yerinde, aralıkta ya da daha genel olarak sınırlı ve küme üyelikli kesinsizlik ve belirsizliği kapsayan problemler, kaba örüntülerin kullanılmasıyla halledilebilir. Kaba küme teorisindeki gelişmeler, alt ve üst sınır genel fikrinin çok farklı tip uygulamalar için kullanışlı olabilecek geniş bir çatı sağlayabileceğini göstermiştir [120].

Nesneler, örnekler ya da kayıtlar sınırlı nitelikler kümesiyle tanımlanabilir. Bir nesnenin tanımı, n bir nesneyi tanımlayan nitelik sayısı olursa, n-boyutlu bir vektördür. Bir örüntü, sınıfa ait nesnelerin bazı niteliklerinin değerlerine bağlı nesnelerin bir sınıfıdır.

x bir nesnenin tanımındaki bir nitelik, $\underline{x}$ ve $\overline{x}$ ise $\underline{x} \leq \overline{x}$ olmak şartıyla x'in alt ve üst sınırları (son noktalar) olsun. Her nitelik değişkeninin kaba örüntü değeri Denklem (4.1)'de gösterildiği gibi alt ve üst sınırlardan oluşur. Bu, şematik olarak Şekil 4.1'de gösterilmiştir. Bu, reel sayı R kümesinin kapalı, öz ve sınırlı alt kümesidir.

$$x = \left(\underline{x}, \overline{x}\right) \tag{4.1}$$



Şekil 4.1. Bir kaba değer

$0 \leq \underline{x}$ ise kaba değer *pozitif kaba değer* olarak adlandırılır ve $x > 0$ yazılır. Tersine, eğer $\overline{x} \leq 0$ ise kaba değer *negatif kaba değer* olarak adlandırılır ve $x < 0$ yazılır. Pozitif ya da negatif kaba değerler *işaretli kaba değerler*dir. Eğer $\underline{x} = 0$ ya da $\overline{x} = 0$ ise kaba değer *sıfır sınırlı kaba değer* olarak adlandırılır. Sıfır sınırlı pozitif kaba değer *sıfır-pozitif kaba değer* ve benzer olarak sıfır sınırlı negatif kaba değer *sıfır-negatif kaba değer* olarak adlandırılır. Hem pozitif hem negatif değeri olan kaba değer *sıfırı araya alan kaba değer* olarak adlandırılır. Bu tanımlamalar Tablo 4.1'de özetlenmiştir.

Tablo 4.1. Kaba değerler ile ilgili tanımlar

Tanım	**Şart**
pozitif kaba değer ($x>0$)	Ancak ve ancak $\underline{x}>0$
negatif kaba değer ($x<0$)	Ancak ve ancak $\overline{x}<0$
sıfır-pozitif kaba değer ($x \geq 0$)	Ancak ve ancak $\underline{x}=0$
sıfır-negatif kaba değer ($x \leq 0$)	Ancak ve ancak $\overline{x}=0$
sıfırı araya alan kaba değer ($x<>0$)	Ancak ve ancak $\overline{x}>0$ ve $\underline{x}<0$

Bir kaba değer x'in *orta noktası*, *yarıçapı* ve *genişliği* aşağıdaki gibi tanımlanır.

$$orta(x) = \left(\overline{x} + \underline{x}\right)/2 \tag{4.2}$$

$$yarıçap(x) = \left(\overline{x} - \underline{x}\right)/2 \tag{4.3}$$

$$genişlik(x) = \left(\overline{x} - \underline{x}\right) = 2yarıçap(x) \tag{4.4}$$

x = (*orta*(x)-*yarıçap*(x), *orta*(x)+*yarıçap*(x)) olduğundan, kaba değerler son noktalar yerine orta noktalar ve çaplar cinsinden de temsil edilebilir.

Kaba değerler bir hesaplamada sadece alt ve üst sınırların anlamlı olarak düşünüldüğü bir nitelik için değerler kümesini ya da bir aralığı temsil etmede kullanışlıdır. Hesapsal matematiğin çoğu alanında çok popüler olabilir. Örneğin kaba değerlerle hesaplama yaparak, bazı hatalarla, bir fonksiyonu tek bir değerden ziyade bütün bir aralık üzerinde değerlendirmek mümkündür. Kaba değerlerle çalışma daima tam ya da fazla tahmin edilen sınırlar üreteceğinden bir fonksiyonda kritik bir değeri kaçıramaz. Bu yüzden, global maksimum ya da minimum bulmada ya da diğer optimizasyon problemlerinde kullanışlı olabilir.

Aslında geleneksel bir örüntü alt ve üst sınırları değişkenin değerine eşit olan bir kaba değer olarak görülebilir [121, 122]. Kaba değerler üzerindeki bazı işlemler aşağıda gösterilmiştir:

$$x + y = \left(\underline{x}, \overline{x}\right) + \left(\underline{y}, \overline{y}\right) = \left(\underline{x} + \underline{y}, \overline{x} + \overline{y}\right) \tag{4.5}$$

$$x - y = \left(\underline{x}, \overline{x}\right) + \left(-\left(\underline{y}, \overline{y}\right)\right) = \left(\underline{x} - \overline{y}, \overline{x} - \underline{y}\right) \tag{4.6}$$

$$x \times y = \left(\min\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right), \max\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right)\right) \tag{4.7}$$

$$\frac{1}{x} = \left(\frac{1}{\overline{x}}, \frac{1}{\underline{x}}\right), \qquad 0 \notin \left(\underline{x}, \overline{x}\right) \tag{4.8}$$

$$\frac{x}{y} = \frac{\left(\underline{x}, \overline{x}\right)}{\left(\underline{y}, \overline{y}\right)} = \frac{\left(\underline{x}, \overline{x}\right)}{\left(\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right)}, \qquad 0 \notin \left(\underline{y}, \overline{y}\right) \tag{4.9}$$

$$c \times x = c \times \left(\underline{x}, \overline{x}\right) = \left(\underline{x}, \overline{x}\right) \times c = \begin{cases} \left(c \times \underline{x}, c \times \overline{x}\right) & if \quad c \geq 0 \\ \left(c \times \overline{x}, c \times \underline{x}\right) & if \quad c < 0 \end{cases} \tag{4.10}$$

Aslında bu işlemler geleneksel aralık cebrinden alınmıştır [122].

Toplama ve çarpma işlemlerinin cebirsel özellikleri Tablo 4.2'de açıklanmıştır.

Tablo 4.2. Cebirsel özellikler

Cebirsel özellikler	**Tanım**	**Şart**
Yer değiştirme	$x+y=y+x$	-
	$xy=yx$	
Birleşme	$(x+y)+z=x+(y+z)$	-
	$(xy)z=x(yz)$	
Etkisiz eleman	$0+x=x$	-
	$1.x=x$	
Dağılma	$x(y+z)=xy+xz$	$\underline{x}=\overline{x}$
	$x(y+z)=xy+xz$	$y\geq 0$ ve $z\geq 0$
	$x(y+z)=xy+xz$	$y\leq 0$ ve $z\leq 0$
	$x(y+z)=xy+xz$	$x\geq 0,\ \underline{y}=0$ ve $\overline{z}=0$
	$x(y+z)=xy+xz$	$x\leq 0,\ \underline{y}=0$ ve $\overline{z}=0$
	$x(y-z)=xy-xz$	$y\geq 0$ ve $z\leq 0$
	$x(y-z)=xy-xz$	$y\leq 0$ ve $z\geq 0$

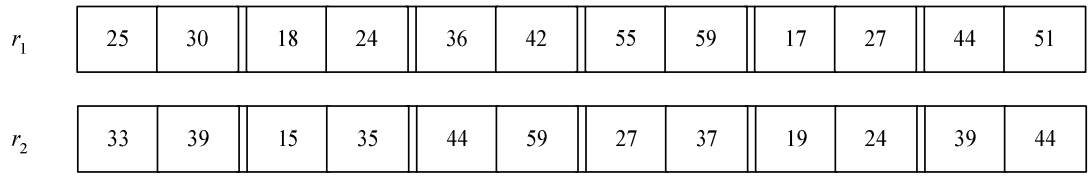
Bir r kaba parçacığı kaba parametre r_i'nin bir katarıdır:

$$r=\left(r_i \mid 1\leq i\leq n\right) \tag{4.11}$$

Bir kaba parametre r_i, biri alt parametre ($\underline{r_i}$) olarak adlandırılan alt sınır için biri de üst parametre ($\overline{r_i}$) olarak adlandırılan üst sınır için kullanılan geleneksel parametre çiftidir.

$$r_i=\left(\underline{r_i},\overline{r_i}\right) \tag{4.12}$$

Şekil 4.2'de kaba parçacık örnekleri gösterilmiştir.



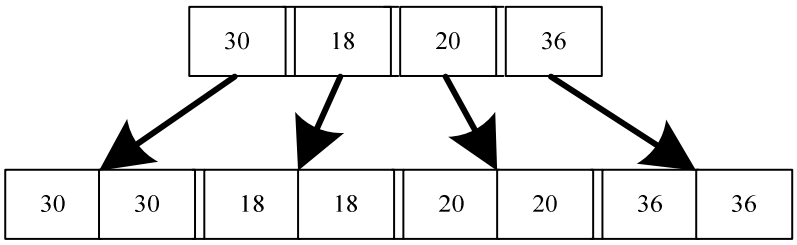
Şekil 4.2. Kaba parçacıklar

Her kaba parametrenin değeri, değişken için aralıktır. Aralığın kullanılması, kaba parçacık tarafından temsil edilen bilginin kesin olmadığını gösterir. Ancak bazı uygulamalarda da, kesinsizlik söz konusu olmaz; direkt olarak bu tür temsilin kullanılması ihtiyacı doğabilir. Bazen *kesinlik* olarak adlandırılan bilgi ölçüsü uygunluk seviyesini değerlendirmede kullanışlı olabilir [121].

$$kesinlik\,(r) = -\sum_{1\le i\le n}\left(\frac{\overline{r_i} - \underline{r_i}}{Sinir_{maks}(r_i)}\right) \tag{4.13}$$

$Sinir_{maks}(r_i)$ kaba parametre r_i'nin değeri için izin verilen maksimum sınırı göstermektedir.

PSO algoritmalarında kullanılan geleneksel parametre ve parçacıklar, Şekil 4.3'te gösterildiği gibi bunların kaba eşdeğerlerinin özel durumlarıdır. Geleneksel bir *p* parçacığı için *kesinlik*(*p*) maksimum değer olan sıfıra eşittir.



Şekil 4.3. Geleneksel parçacıklar ve onların kaba eşdeğerleri

Sınır kısıtlı problemlerde, hız ve pozisyon güncelleme denklemlerinden sonra karar değişkenlerinin değerlerinin izin verilen aralıkta kalması gerekmektedir. Bu teknik KPSOA için de genelleştirilebilir. Alt sınırın üst sınırdan küçük olması şartı zaten KPSOA'da sağlanmaktadır.

4.3. Nicel Birliktelik Kural Madenciliği ve İlgili Çalışmalar

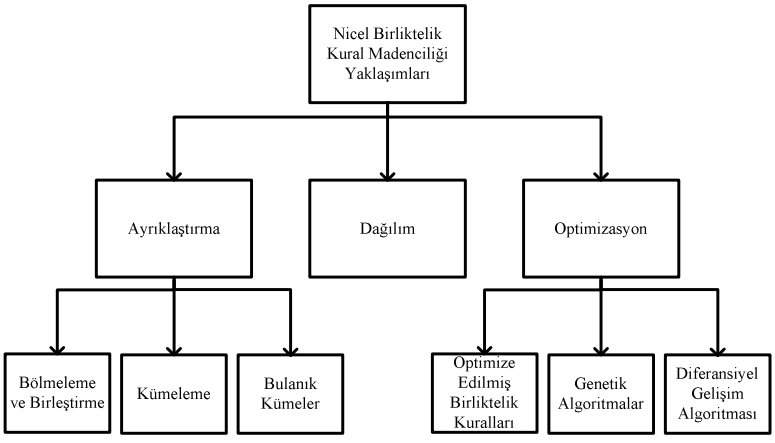
Market verisi üzerinden çalışan birliktelik kural madenciliği algoritması ilk olarak [123]'te tanıtılmıştır. Bu algoritma ve bundan sonra önerilen çoğu algoritma birliktelik kurallarının keşfi için iki aşama takip etmişlerdir: birincisi yoğun nesne kümelerini bulmak, ikincisi de elde edilen yoğun nesne kümelerinden kuralları keşfetmek. Keşfedilen kurallar belirli destek ve güven değerlerine sahiptir. Bu şekilde ikili birliktelik kuralları anlamlı olmasına rağmen ilgilenilen veri nesneleri çoğu durumda kategorik ya da niceldir. Bu yüzden nicel birliktelik kural madenciliği algoritmaları önerilmiştir. Bir nicel birliktelik kuralında nitelikler ikili değerlerle sınırlandırılmamış, nicel (yaş, maaş, sıcaklık gibi) ve kategorik (cinsiyet, marka) değerleri de almıştır. Böylece nicel birliktelik kuralları ikili birliktelik kurallarından daha anlamlıdır [124].

Bir personel veritabanında nicel bir birliktelik kuralı şu şekildedir:

"Yaş $\in$ [25, 36] $\wedge$ Cinsiyet=Erkek $\Rightarrow$ Maaş $\in$ [2000-2400] $\wedge$ Arabası_Var=Evet"
(Güven = 4%, Destek = 80%).

Bu nicel birliktelik kuralında "Yaş $\in$ [25, 36] $\wedge$ Cinsiyet=Erkek" kuralın ata kısmı "Maaş $\in$ [2000-2400] $\wedge$ Arabası_Var=Evet" ise sonuç kısmıdır. Bu kural "işçilerin %4'ü (destek) erkektir ve 25 ve 36 yaşları arasındadır ve 2000 ile 2400 TL arası maaş almaktadır ve arabaları vardır" ve "25 ve 36 yaşlarındaki erkeklerin %80'i (güven) 2000 ile 2400 TL arası maaş almaktadır ve arabaları vardır" demektedir.

Nicel birliktelik kurallarının keşfi için üç temel yaklaşım bulunmaktadır ve bunlar Şekil 4.4'te gösterilmiştir.



Şekil 4.4. Nicel birliktelik kural madenciliği yaklaşımları

[125]'te nicel birliktelik kural madenciliği nitelik alanlarının küçük aralıklara bölünmesi ve birleştirilen aralıklar yeterli desteğe sahip oluncaya kadar bitişik aralıkların daha büyük aralıklar şeklinde birleştirilmesiyle yapılmıştır. Aslında, nicel problem ikili problem haline dönüştürülmüştür. Fakat bu teknikte, bir nitelik diğer nitelikler hesaba katılmadan ayrıklaştırılmış ve nitelik etkileşimleri göz ardı edilmiştir.

Daha sonra farklı araştırmacılar kümeleme tekniklerini kullanmışlardır. Miller ve Yang [126] aralıkların anlamını arttıran uzaklık-tabanlı birliktelik kural madenciliği süreci önermişlerdir ve aralıkları belirlemek için de Birch kümelemeyi uygulamışlardır. Lent ve arkadaşları [127] nicel nitelikler için kümeleme amacıyla BitOp olarak adlandırılan geometrik tabanlı bir algoritma önermişlerdir. Bunlar anlamlı bölgeleri hesaplamak ve birliktelik kurallarının keşfini desteklemek için kümelemenin olası bir çözüm olduğunu göstermişlerdir. Vannucci ve Colla orijinal örnek dağılımını korumaya çalışan denetimsiz ayrıklaştırma için önerilen tekniklerin sınırlamalarını kaldırmak amacıyla bir sinirsel ağ, kendini organize eden harita, önermiştir [128]. Bu çalışmaların çoğu aykırı verilere hassas kalmakta ve verinin dağılımını yansıtmamaktadır.

[129]'de nicel birliktelik kurallarının madenciliği için yine kümeleme kullanan bilgi-teorili bir yaklaşım önerilmiştir. Nitelikler arasındaki bilgi verici ilişkileri gösteren bir çizge inşa edilmiştir. Sonra çizgede işe yaramayan nitelik kümelerini ve böylece bu nitelikler arasında birleştirilen aralıkları budamak için gruplar kullanılmıştır.

Bazı araştırmacılar nicel veriyi bulanık kümelerle bölmüşler ve keşfedilen kuralları *bulanık birliktelik kuralları* olarak adlandırmışlardır [130]. Bu kurallar şu şekildedir:

$$A=X \Rightarrow B=Y$$

Burada A ve B niteliklerin alt kümesi olan nesne kümelerini içermektedir. X ve Y ise A ve B'de ilgili nitelik kümesiyle ilişkili bulanık kümeleri içermektedir.

Ancak, tüm bu teknikler kullanıcıdan ön bilgi isterler. Nicel nitelikler için aralıkların seçilmesi güven ve destek değerlerine oldukça duyarlıdır [123]. Sınırlar ve bulanık üyelik kümelerinin şekilleri uzman kişiler tarafından belirlenmelidir ve böylece otomatik ayrıklaştırmanın gerekli olduğu durumlarda bu teknikler kullanılamaz. Nicel nitelikler için, hepsi bireysel olarak düşünüldüğünde, anlamlı bir ayrıklaştırma bulmak zordur. Böylece klasik iki adımlı yaklaşımlar nicel birliktelik kural madenciliği için artık uygun olmamaktadır.

Aumann ve Lindell nicel bir değerin dağılımını birliktelik kurallarına dahil edilip edilmeme kriteri olarak kullanmıştır [131]. Bunlar birliktelik kurallarının ilginç davranışlar

sergileyen (kural atası) bir popülasyon alt kümesi (kural sonucu) olarak düşünüleceğini iddia etmişlerdir. İki tip nicel kural araştırmışlardır:

"kategorik $\Rightarrow$ nicel" kurallar

ve

"nicel $\Rightarrow$ nicel" kurallar

Bu nicel birliktelik kural algoritmalarındaki genel kısıtlama kuralın atasında ya da sonucunda izin verilen değer sayısıdır. Ayrıca, hem ikili hem de nicel değerlerin kuralın ata ya da sonuç kısmında yer almasına izin verilmemektedir. Nicel niteliklerin ayrıklaştırılması kaçınılmaz biçimde bilgi kaybına neden olmaktadır. Ayrıklaştırma niteliğin orijinal dağılımını yansıtmamakta ve ayrıklaştırılmış aralıklar kuralları gizleyebilmektedir (aralık çok büyükse düşük kararlılıktaki kurallar kaçırılabilir; çok küçük olduğunda da kural keşfetmek için yeterli veri bulunmayabilir). Aralıklar semantik olarak anlamsız olabilir ve uzmanlara mantıklı gelmeyebilir. Ayrıca birkaç nicel değerin kümülatif etkisi kolaylıkla temsil edilemeyebilir.

Bazı araştırmacılar da nicel değerler için nicel aralıklar bulmak amacıyla geometrik ortalamayı kullanmışlardır [132]. Ancak kuralın atası sadece tek bir kategorik değerle sınırlandırılmıştır. Keşfettikleri kurallar şu formdadır:

"$A \in [v_1, v_2] \Rightarrow C$"

ya da bunun genişletilmiş formu

"$A \in [v_1, v_2] \wedge C_1 \Rightarrow C_2$"

Burada A nicel nitelikler C, C_1 ve C_2 ikili ifadelerdir.

Fukuda ve arkadaşları [133] ve Yoda ve arkadaşları [134] ata kısımda iki nicel değer ve sonuçta bir ikili nesne olacak şekilde farklı bir biçimi önermişlerdir.

Tüm bu yaklaşımların temel problemi madencilik algoritmasından önce verinin hazırlanmasıdır. Kullanıcı tarafından ya da otomatik süreçle oluşturulan bu hazırlık birçok bilgi kaybını beraberinde getirir çünkü kurallar daha önce oluşturulan aralıklardan ayrılarak üretilecektir. Nicel veri için oluşturulan aralıklar keşfedilen kurallardan değerli bilgiyi kolaylıkla elde edebilmek amacıyla uzman kişiler için yeterince özlü ve anlamlı olmayabilir.

Ayrıca, bulanık küme yaklaşımı haricinde bu yaklaşımların bazı sakıncaları bulunmaktadır. İlk problem insan algısına göre sezgisel olmayan aralıklar arasındaki keskin sınır tarafından ortaya çıkar. Algoritmalar aralıkların sınırlarına yakın olan elemanları ya ihmal eder ya da çok önemser. Ayrıca, ön bilgi olmadan aralık tekniği için üyelik derecesinin ayırt edilmesi kolay değildir. Benzeri şekilde, bulanık kümelerle bölmeleme de kolay değildir çünkü nicel nitelik değerleri için en uygun bulanık kümeye karar vermek zordur [135, 136]. Nicel niteliklerin karakteristikleri genel olarak bilinmez ve alan uzmanları tarafından en uygun bulanık kümelerin her zaman temin edilmesi gerçekçi değildir. Bu yüzden bazı araştırmacılar ayrı bir ön işlem olarak nicel nitelikler için bulanık kümeleri bulmada evrimsel algoritma kullanmışlardır [137].

Aslında nicel birliktelik kurallarının keşfi basit bir ayrıklaştırma işleminden daha ziyade zor bir optimizasyon problemidir. Bu yüzden bazı araştırmacılar bunu bir optimizasyon problemi olarak karakterize etmiş ve birliktelik kurallarını global optimizasyon algoritmalarıyla bulma yoluna gitmişlerdir. Sadece yoğun nesne kümelerini bulmak için evrimsel algoritma kullanma fikri [138]'de kullanılmıştır. Ancak kodlama değişken boyuttan dolayı genetik operatörler için fazla etkili değildir. Ayrıca, sadece destek optimize edilmiş ve yoğun nesne kümeleri üretilmiştir. [6]'da etkili ve düzenlenmiş bir evrimsel algoritma ile tüm kurallar tek çalıştırmada bulunmuştur. [8]'de ise çok amaçlı diferansiyel gelişim algoritması önerilmiştir. En son yeni çalışma ise genetik algoritma kullanan QuantMiner adlı algoritmadır [139]. Ancak bu çalışmada kullanıcı tarafından kuralın atasında ve sonucunda yer alacak niteliklerin belirlenmesi yani bir şablonun oluşturulması gerekmektedir ve yapılan iş bu şablon için en uygun aralıkların bulunmasıdır. Fakat hangi niteliğin kuralda yer alacağının ve yer alacaksa da nerede yer alacağının (ata ya da sonuç kısmı) belirlenmesi işi uzman ya da kullanıcıya bırakılamaz ve bu işleri veri madenciliği algoritmasının kendisinin yapması gerekir. Aksi takdirde ilgilenilen gerçek bir veri madenciliği problemi ve süreci olmaz. Bu çalışma sadece oluşturulan şablon için aralık belirleme işini yapmaktadır. Önceden belirlenen kural şablonları ile elde edilen kurallar ilgili veritabanında en iyi kurallar olmayabilir ve bu da en iyi kuralların keşfini engeller. QuantMiner sadece uygun aralık bulmaktadır. Yani, kullanışlı bile olamamasına rağmen kullanıcı ya da uzman bilgisi gerektirir.

Birliktelik kurallarının keşfi için PSO kullanan herhangi bir çalışma yoktur. Tezin bu bölümünde, PSO nicel niteliklerin aralıkların optimizasyonunu ve kuralların keşfini eş zamanlı olarak ve herhangi bir ön işlem ve uzman bilgisi gerektirmeden otomatik yapacak şekilde tasarlanmıştır. Tasarlanan PSO, ayrıca her veritabanı için belirlenmesi güç olan minimum destek ve minimum güven değerlerine ihtiyaç duymadan veritabanından bağımsız bir yaklaşım sunar. Genelde kullanılanın aksine yüksek kaliteli birliktelik kurallarını yoğun nesne kümeleri üretmeden direkt olarak keşfeder.

4.4. Nicel Birliktelik Kural Keşfinde KPSOA

4.4.1. Parçacık Temsili

Arama boyunca üretilen ve düzenlenen parçacıklar kuralları temsil etmektedir. Her parçacık nesneleri ve aralıkları temsil eden karar değişkenlerinden oluşur. *i*. nesnenin *i*. karar değişkeninde kodlandığı pozisyonel bir kodlama kullanılmıştır. Her karar değişkeni de üç kısımdan oluşur. Karar değişkeninin birinci kısmı kuralın ata ya da sonuç kısmını (A-S) temsil eder ve sıfır ile bir arasında değerler alır. Eğer bu değer $0.00 \leq A\text{-}S_i \leq 0.33$ ise bu nesne kuralın ata kısmında yer alacaktır. $0.33 < A\text{-}S_i \leq 0.66$ ise nesne kuralın sonuç kısmında yer alacaktır ve eğer $0.66 < A\text{-}S_i \leq 1.00$ ilgili nesne kuralda yer almayacaktır. 0.00 ve 0.33 arasında değer alan tüm karar değişkenleri kuralın ata kısmını, 0.33 ve 0.66 arasında değer alanlar ise kuralın sonuç kısmını oluşturacaktır. İkinci kısım alt sınırı (AS) temsil ederken üçüncü kısım da üst sınırı (ÜS) temsil etmektedir. Bir parçacığın yapısı Şekil 4.5'te görülmektedir. Burada *m*, keşif yapılan verinin nitelik sayısıdır.

Değişken$_1$			***Değişken$_2$***			...			***Değişken$_m$***		
$A\text{-}S_1$	AS_1	$ÜS_1$	$A\text{-}S_2$	AS_2	$ÜS_2$				$A\text{-}S_m$	AS_m	$ÜS_m$

Şekil 4.5. Parçacık temsili

Bu parçacık temsilinin uyarlamasında karar değişkeninin ikinci ve üçüncü kısmı tek bir değer olarak, yani kaba değer olarak, işlem görecektir. İlk bakışta bu temsilin sadece nicel nitelikler içeren veri tabanları için kullanılabileceği görülmektedir. Ancak, bunu ayrık ya da nominal değerler için genişletmek çok kolay ve düz bir adımdır. Nicel nitelikler temsilin başında ayrık olanlar ise sonunda yer alırlar. Ayrık nitelikler için sadece $A\text{-}S_i$ ve D_i kullanılır ve D_i niteliğin değeridir. Yani, ayrık ya da nominal niteliklerin değerleri için AS_i ve $ÜS_i$ yerine sadece D_i kullanılır.

4.4.2. Uygunluk Fonksiyonu

Keşfedilen kurallar yüksek destek ve güven değerlerine sahip olmalıdır. Kurallar keşfedilirken eş zamanlı olarak ilgili niteliklerin aralıkları da keşfedilecek kuralı optimize edecek şekilde ayarlanmaktadır. Yani bu aralıkların belirlenmesi uygunluk fonksiyonunun bir parçası olarak ele alınmıştır. Seçilen uygunluk fonksiyonu (4.14)'te gösterilmiştir:

$$Uygunluk = \alpha_1 \times kapsama(Ata+Sonuç) + \alpha_2 \times \frac{kapsama(Ata + Sonuç)}{kapsama(Ata)} - \alpha_3 \times (NA-N(*)) - \alpha_4 \times Aralik - \alpha_5 \times İşaret \quad (4.14)$$

Seçilen bu uygunluk fonksiyonu beş bölümden oluşmaktadır. Burada *Ata* ve *Sonuç* sırasıyla kuralın ata ve sonuç kısmını içeren ayrık nesne kümeleridir. *kapsama(Ata+Sonuç)* Ata ve Sonucu içeren kayıtların veritabanındaki toplam kayıt sayısına oranıdır. Bu birinci kısım bir birliktelik kuralının istatistiksel anlamı olarak görülebilen destek değeri olarak düşünülebilir. İkinci kısım da kuralın güven değerini temsil etmektedir. Üçüncü kısım parçacıktaki nitelik sayısını göstermektedir. Bu az sayıda nitelik barındıran kısa kuralların oluşmasına destek vermektedir. *N*A veritabanındaki nitelik sayısı *N*(*) da parçacığın ilk parçasında 0.66 ile 1.00 arasında değer alan nitelik sayısıdır. Bu terimle veri madenciliğinde önemli olan okunabilirlik ve anlaşılabilirlik arttırılmaya çalışılmıştır. Uzun kurallar gereksiz ve önemsiz bilgi içerebilmekte ve bu da kuralın başarısını ve etkili şekilde işlenebilmesini engelleyebilmektedir. Ancak, son kullanıcıya daha kaliteli bilgi verebilmek için bazı veritabanlarında ve kullanıcı isteklerine ve gereksinimlerine göre bu terim uygunluk fonksiyonunun değerini arttırıcı şekilde de (işareti "+" yapılarak) kullanılabilir. Uygunluk fonksiyonundaki dördüncü kısım nesne kümesi ve kurala uyan aralıkların genliğini azaltmak için kullanılmaktadır. Bu şekilde aynı sayıda kaydı kapsayan ve aynı sayıda nitelik içeren bir iki kuraldan aralıkları küçük olan en iyi bilgiyi vermektedir ve bunun uygunluğu daha iyidir. *Aralik* (4.15)'te gösterildiği şekilde hesaplanır ve gen_m, *Aralik*'ın etkisini uygunluk fonksiyonunda dengelemek için her nitelik için kararlaştırılan genlik faktörüdür.

$$Aralik = \frac{\ddot{U}S_m - AS_m}{gen_m} \quad (4.15)$$

Uygunluk fonksiyonundaki son bölüm KPSOA'nın sonraki aramalarda farklı kuralları keşfetmesi için düzenlenmiştir. KPSOA her çalıştırmada tek bir kural keşfetmektedir. İşaret terimi de bir kaydın niteliğinin daha önce bir kural atası ya da sonucu tarafından kapsanıp kapsanmadığını göstermek için kullanılmaktadır.

α_1 , α_2 , α_3 , α_4 ve α_5 kullanıcı tanımlı parametrelerdir ve bu parametreler değiştirilerek uygunluk fonksiyonundaki kısımların etkileri değiştirilebilir. Uygunluk hesaplamasında *Aralik* kısmı nicel nitelikleri temsil eden parçacık kısımlarını ilgilendirmektedir.

4.4.3. Mutasyon

KPSOA'da mutasyon da kullanılmıştır. Mutasyon bir parçacığın tek bir kaba karar değişkeninin p_{mut} olasılığıyla değişime uğraması şeklinde gerçekleşir. Genetik algoritmada kullanılan dört çeşit mutasyon KPSOA'ya da uyarlanmıştır [138].

- **Tüm aralığı sağa doğru kaydırma**: Alt ve üst sınırlardaki değerler arttırılır.
- **Tüm aralığı sola doğru kaydırma**: Alt ve üst sınırlardaki değerler azaltılır.
- **Aralık boyutunu arttırma**: Alt sınır değeri azaltılırken üst sınır değeri arttırılır.
- **Aralık boyutunu azaltma**: Alt sınır değeri arttırılırken üst sınır değeri azaltılır.

Parçacık pozisyonları eğer mutasyona uğramış parçacıklar daha iyi uygunluğa sahip olurlarsa bu değerle güncellenir.

4.4.4. Sınır Aralıklarının Arıtılması

KPSOA çalışması sonunda keşfedilen kuralın nitelik sınırlarına bir iyileştirme işlemi uygulanır. Bu ilgili parçacıkta aralık boyutunun destek değeri orijinal destek değerinden küçük olana kadar aralığın azaltılması işlemini içerir. Bu şekilde daha kaliteli kuralların oluşması sağlanır.

4.4.5. Parametre Kontrolü

Deneylerde kullanılan parametre değerleri Tablo 4.3'te gösterilmiştir. Hız ve pozisyon için maksimum ve minimum değerler karar değişkenlerin değerlerine bağlıdır. α_1, α_2, α_3, α_4, ve α_5 sırasıyla 0.8, 0.8, 0.05, 0.1, ve 0.2 olarak seçilmiştir.

Tablo 4.3. Kullanılan parametre değerleri

Parametreler	Sürü boyutu	İterasyon sayısı	Mutasyon olasılığı	Hızlanma katsayıları	Atalet ağırlığı
Değerler	25	1000	0.5	2	0.9'dan 0.4'e

4.5. Deneysel Sonuçlar

KPSOA ilk olarak dört nicel nitelik içeren 1000 kayıtlı sentetik bir veritabanında değerlendirilmiştir. Tüm alan değerleri [0, 100] aralığına ayarlanmıştır. Değerler Tablo 4.4'te gösterilen şekilde, önceden belirlenen kümelerde gruplanmış biçimde, düzenli olarak dağıtılmıştır. Bu dağılım tamamen gelişigüzeldir. Bazı aralıklar küçük bazıları ise büyük boyuta

sahiptir. Bu kümeler için destek ve güven değerleri sırasıyla %25 ve %100 olarak seçilmiştir. Bu kümeler dışındaki değerler bu kurallardan daha iyi kural oluşturmayacak şekilde dağıtılmıştır. Birliktelik kural madenciliğinde uygunluk fonksiyonu için uygun ağırlıklar belirlenerek bu kuralların keşfedilmesi istenmiştir. Amaç her ayarlanan bölgenin aralığını doğru olarak bulmaktır. KPSOA'nın her niteliğin nicel aralığı için en uygun değerlerle birliktelik kurallarını keşif yeteneği test edilmiştir.

Tablo 4.4. Sentetik olarak oluşturulan kümeler

Kümeler
$A_1 \in [1–10] \wedge A_2 \in [15–30]$
$A_1 \in [15–45] \wedge A_3 \in [60–75]$
$A_2 \in [65–90] \wedge A_4 \in [15–45]$
$A_3 \in [80–100] \wedge A_4 \in [80–100]$

Tablo 4.5. KPSOA tarafından bulunan kurallar

Kural	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [1\text{-}10] \Rightarrow A_2 \in [15\text{-}30]$	25	100	100
$A_1 \in [15\text{-}45] \Rightarrow A_3 \in [60\text{-}75]$	25	100	
$A_3 \in [80\text{-}100] \Rightarrow A_4 \in [80\text{-}98]$	25	100	
$A_2 \in [65\text{-}89] \Rightarrow A_4 \in [15\text{-}43]$	25	100	
$A_2 \in [15\text{-}30] \Rightarrow A_1 \in [1\text{-}10]$	25	100	
$A_3 \in [60\text{-}75] \Rightarrow A_1 \in [15\text{-}45]$	25	100	
$A_4 \in [80\text{-}98] \Rightarrow A_3 \in [80\text{-}100]$	25	100	
$A_4 \in [15\text{-}45] \Rightarrow A_2 \in [65\text{-}89]$	25	100	

Tablo 4.5'te KPSOA tarafından bulunan kurallar gösterilmektedir. Tablodan görüldüğü gibi sentetik olarak oluşturulan kümelere göre keşfedilen kurallar, yüksek destek ve güven değerine sahiptir; ayrıca anlaşılabilirdir. KPSOA veritabanından bağımsızdır çünkü her veritabanı için belirlenmesi güç olan minimum destek ve güven değerlerine bağlı değildir. Eğer destek ve güven değerleri kullanılsaydı ve destek değeri %25'ten büyük seçilseydi bu veritabanındaki niteliklerin değerlerine göre hiçbir kural keşfedilemeyecekti. Ancak bu veritabanının bazı doğru ve anlaşılabilir kurallar içerdiği bilinmektedir. KPSOA minimum destek ve güven eşik değerlerini kullanmadan tüm bu kuralları otomatik olarak keşfetme yeteneğine sahiptir. Ayrıca kuralın ata ve sonuç kısmının önceden ayarlanmasıyla oluşturulmuş kural şablonunun belirlenmesine ihtiyaç duymaz.

Tablo 4.6. Farklı seviyelerdeki gürültü sonrası keşfedilen kurallar

***gürültü_seviyesi* = 4%**			
Keşfedilen kurallar	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [1\text{-}10] \Rightarrow A_2 \in [15\text{-}29]$	24.2	100	95.3
$A_1 \in [15\text{-}45] \Rightarrow A_3 \in [60\text{-}73]$	24.0	100	
$A_3 \in [80\text{-}100] \Rightarrow A_4 \in [80\text{-}96]$	23.6	96.7	
$A_2 \in [65\text{-}90] \Rightarrow A_4 \in [15\text{-}46]$	24.2	98.3	
$A_2 \in [15\text{-}29] \Rightarrow A_1 \in [1\text{-}10]$	24.1	100	
$A_3 \in [60\text{-}73] \Rightarrow A_1 \in [15\text{-}45]$	24.0	100	
$A_4 \in [80\text{-}96] \Rightarrow A_3 \in [80\text{-}100]$	23.7	96.6	
$A_4 \in [15\text{-}46] \Rightarrow A_2 \in [65\text{-}89]$	24.2	98.3	
***gürültü_seviyesi* = 6%**			
Keşfedilen kurallar	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [1\text{-}11] \Rightarrow A_2 \in [14\text{-}31]$	23.3	98.9	93.1
$A_1 \in [15\text{-}45] \Rightarrow A_3 \in [56\text{-}73]$	23.6	99.0	
$A_3 \in [80\text{-}100] \Rightarrow A_4 \in [84\text{-}95]$	23.3	94.5	
$A_2 \in [65\text{-}89] \Rightarrow A_4 \in [14\text{-}49]$	23.8	97.8	
$A_2 \in [14\text{-}31] \Rightarrow A_1 \in [1\text{-}11]$	23.3	98.9	
$A_3 \in [56\text{-}73] \Rightarrow A_1 \in [15\text{-}45]$	23.6	99.0	
$A_4 \in [84\text{-}95] \Rightarrow A_3 \in [80\text{-}100]$	23.3	94.5	
$A_4 \in [14\text{-}49] \Rightarrow A_2 \in [65\text{-}89]$	23.8	97.8	
***gürültü_seviyesi* = 8%**			
Keşfedilen kurallar	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [1\text{-}11] \Rightarrow A_2 \in [14\text{-}29]$	22.4	97.6	91.9
$A_1 \in [15\text{-}45] \Rightarrow A_3 \in [62\text{-}76]$	22.9	98.0	
$A_3 \in [79\text{-}100] \Rightarrow A_4 \in [82\text{-}98]$	22.6	93.3	
$A_2 \in [65\text{-}90] \Rightarrow A_4 \in [15\text{-}48]$	23.7	95.8	
$A_2 \in [14\text{-}29] \Rightarrow A_1 \in [1\text{-}11]$	22.4	97.4	
$A_3 \in [62\text{-}76] \Rightarrow A_1 \in [15\text{-}45]$	22.9	98.0	
$A_4 \in [82\text{-}98] \Rightarrow A_3 \in [79\text{-}100]$	22.8	93.4	
$A_4 \in [15\text{-}48] \Rightarrow A_2 \in [65\text{-}90]$	23.7	95.8	

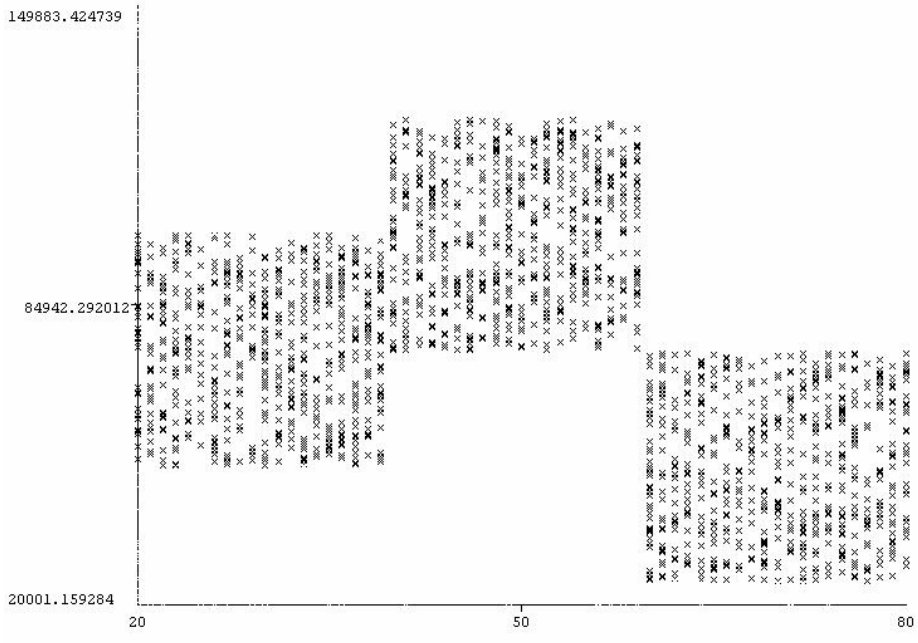
Önerilen algoritmanın etkinliliği test etmek için KPSOA, gürültülü sentetik veritabanında da çalıştırılmıştır. Gürültü, kümenin ikinci nesnesinin aralığına ait olmayan değerler yerleştirmek suretiyle oluşturulmuştur. Bu yüzden, kayıtların belli bir yüzdesi ikinci nesnenin önceden belirlenen aralığında yer almaz. Örneğin, ilk küme için kayıtların belli bir yüzdesi, ikinci nesne $A_2 \in$ [15-30]'da bulunmaz, ancak [0-14] ya da [31-100] aralığında dağıtılmıştır.

Algoritmanın, kuralların ata ya da sonuç kısmı için en uygun aralıkları bulup bulamayacağı test edilmiştir. Bu test üç seviyeli gürültü ile yerine getirilmiştir (*gürültü_seviyesi*'nin 4%, 6% ve 8% değerleri için). Deneysel sonuçlar Tablo 4.6'da gösterilmiştir. Bu tabloda, keşfedilen kurallar, destek ve güven değerleri ve toplam kayıtta keşfedilen kural tarafından kapsanan kayıt yüzdeleri verilmiştir. Aralıkların sınırlarının sentetik olarak üretilenlere hemen hemen uyduğu görülebilmektedir. Bu KPSOA'nın test edilen veri içinde belirli yüzdelerdeki gürültünün üstesinden geldiğini göstermektedir.

Başka bir deneyde KPSOA'nın etkinliliğini diğer algoritmalarla karşılaştırmak amacıyla yapılmıştır. [140]'taki ikinci fonksiyon kullanılarak sentetik bir veritabanı oluşturulmuştur. Fonksiyon Şekil 4.6'da gösterilmiştir. Amaç her oluşturulan bölgenin aralığını en uygun şekilde bulmaktır. Yaş ve maaş niteliklerinin aldığı değerlere bağlı olarak her kayda bir grup atanmıştır. Şekil 4.7 sadece Grup A'ya ait 5000 kaydın yer aldığı fonksiyonun dağılımının grafiksel gösterimidir.

$$\text{Eğer } ((yaş < 40) \wedge (50\text{K} \leq maaş \leq 100\text{K})) \vee$$
$$((40 \leq yaş < 60) \wedge (75\text{K} \leq maaş \leq 125\text{K})) \vee$$
$$((yaş \geq 60) \wedge (25\text{K} \leq maaş \leq 75\text{K})) \Rightarrow \text{Grup } A$$
$$\text{Değilse} \Rightarrow \text{Grup } B$$

Şekil 4.6. Deney için kullanılan fonksiyon



Şekil 4.7. Deney için kullanılan fonksiyonun grafiksel gösterimi

Veritabanı, kayıtların alt ve üst sınırlar arasında düzgün dağıtılmasıyla oluşturulmuştur. *maaş* niteliği için sınır noktalar 20000'den 150000'e ve *yaş* niteliği için sınır noktalar 20'den 80'e kadardır. Grup için üçüncü nitelik de eklenmiştir. Fonksiyona göre kayıtların %37.9'u Grup A'ya aittir. Birliktelik kurallarında *yaş* ve *maaş* nitelikleri ata kısımda ve Grup niteliği de sonuç kısmında olacaktır. Bu yüzden parçacık temsili de bu duruma riayet edecektir.

GA ve KPSOA haricindeki algoritmalar nicel niteliklerin aralıklarını oluşturmak için otomatik bir ayrıklaştırma süreci içermektedirler ve bu algoritmalar Weka programı ile çalıştırılmıştır. OneR [141] algoritmasında nicel nitelikleri ayrıklaştırmak için kullanılan sepet sayısı 6 olarak belirlenmiştir. PART [142] algoritması için budama amacıyla kullanılan güven faktörü 0.25 seçilmiş ve azaltılmış hata budaması için C4.5 budaması kullanılmıştır. Ridor [143] algoritmasında bir kuralda örneklerin toplam minimum ağırlığı 1 olarak belirlenmiş ve standart değerleriyle çalıştırılmıştır. KPSOA'nın yeni bir teknik olduğu ve GA'lar ile diğer yöntemlerin eski teknikler olduğu ve uzun süreden beri kullanıldığını unutmamak gerekmektedir. *Kapsama* ve *NegatifNoktalar* olarak adlandırılan iki kalite ölçüsü keşfedilen kurallar için karşılaştırma yapmak amacıyla kullanılmıştır. Bunlar, (4.16) ve (4.17) denklemleriyle tanımlanmıştır.

$$Kapsama = \frac{N_{Pozitif}}{N_{TumPozitif}} \tag{4.16}$$

$$NegatifNoktalar = \frac{N_{Negatif}}{N_{TumNegatif}} \tag{4.17}$$

$N_{TumPozitif}$ tüm pozitif kayıtların (Grup A) sayısıdır. $N_{Pozitif}$ algoritmalar tarafından keşfedilen kurallar tarafından kapsanan kayıtların sayısıdır. $N_{TumNegatif}$ negatif kayıtların (Grup B) sayısıdır ve $N_{Negatif}$ kural tarafından yanlışlıkla kapsanan negatif kayıt sayısıdır. Aslında *Kapsama* doğru kapsanan kayıtların oranıdır ve *NegatifNoktalar* da kuralların güvenilirliğini göstermektedir.

Sonuçların adil karşılaştırılabilmesi için başlangıç sürü ve GA'lar için başlangıç popülasyonu farklı bir şekilde başlatılmıştır. Veritabanındaki aynı bir kayıt seçilmiş ve kural bu kayda bağlı olarak üretilmeye çalışılmıştır. Seçilen a_i niteliğinin her v_i değeri için alt limit v_i-θ ve üst limit $v_i + \theta$, v_i'nin bir yüzdesi olarak belirlenmiş ve böylece başlangıç sürü ve popülasyonun en az bir kayıt içermesi sağlanmıştır.

Algoritmalardan elde edilen sonuçlar Tablo 4.7'de görülmektedir. KPSOA Grup A için sadece üç kural keşfetmiştir. Ancak GA dışında diğer algoritmalar çok fazla kurala ihtiyaç duymuştur. Az sayıdaki kural son kullanıcıya anlaşılabilirlik ve okunabilirlik sunmaktadır ve çoğu durumda istenen bir özelliktir. Ayrıca KPSOA tarafından elde edilen *Kapsama* değerleri daha iyidir. Başka önemli bir nokta da diğer algoritmalar tarafından elde edilen büyük *NegatifNoktalar* değerleri daha az güveni gösterir ve bu da ayrıklaştırma işlemi için bu algoritmalar tarafından seçilen aralıklardandır. Bu algoritmalar uygun kesme noktaları belirleyememiştir ve bu da nicel birliktelik kuralların keşfindeki en önemli problemdir. Bu algoritmalar tarafından keşfedilen kurallar, anlaşılabilir değildir ve artık nitelik ve değerlere sahiptir. Örnek olarak Ridor tarafından keşfedilen bir kural aşağıdadır:

"(*yaş* > 39.5) $\wedge$ (*maaş* $\leq$ 122707.776803) $\wedge$ (*maaş* > 28267.898833) $\wedge$ (*yaş* $\leq$ 61.5) $\wedge$ (*maaş* > 74840.212238) $\wedge$ (*yaş* $\leq$ 59.5) $\Rightarrow$ *Grup* = *A*" (kapsanan kayıt=408)

Burada, kuralın ata kısmında sadece iki nitelik bulunmasına yani çok az sayıda nitelikle uğraşılmasına rağmen, dikkat edilirse "(*yaş* <= 61.5)" ve "(*yaş* <= 59.5)" kuralın ata kısmındadır ve "(*yaş* <= 59.5)" atık niteliktir. Benzer şekilde "(*maaş* > 28267.898833)" ve "(*maaş* > 74840.212238)" kuralın ata kısmındadır ve burada da "(*maaş* > 28267.898833)" atık niteliktir. Bu şekilde kuralın anlaşılması ve işletilebilmesi zorlaşmaktadır.

Yine PART tarafından keşfedilen bir kural aşağıdadır:

"($maaş > 24927.14358$) $\wedge$ ($maaş > 49974.276261$) $\wedge$ ($yaş \leq 60$) $\wedge$ ($maaş \leq 100532.277915$) $\wedge$ ($yaş \leq 39$) $\wedge$ ($maaş \leq 98002.227109$) $\Rightarrow$ $Grup = A$" (kapsanan kayıt=609)

Bu kuralda da dikkat edilirse "($maaş > 24927.14358$)" ve "($maaş > 49974.276261$)"; benzer şekilde "($maaş \leq 100532.277915$)" ve "($maaş \leq 98002.227109$)" aynı kuralın ata kısmında yer almakta ve bu basit ve kısa kural bile atık nitelik ve değerler içermektedir.

OneR ise sadece maaş niteliğine göre kuralları üretebilmiştir.

KPSOA ise, çok az ve artık nitelik ve değer içermeyecek şekilde veritabanında doğru ve anlaşılabilir kuralları bulabilmiştir.

Tablo 4.7. Fonksiyon 2 için farklı algoritmalar tarafından elde edilen sonuçlar

	OneR	**PART**	**Ridor**	**GA**	**KPSOA**
Kural sayısı	59	3	5	3	3
Kapsama	70.44	66.80	99.89	99.83	100
NegatifNoktalar	18.03	20.20	0.003	0.001	0.00

Algoritmaların etkinliliğini doğrulayan sezgisel bir ölçü kuralların aralıklarının sentetik olarak oluşturulanlarla karşılaştırılması ve böylece kuralların kalitelerinin belirlenmesidir. KPSOA tarafından keşfedilen kuralların ortalama destek ve güven değerleri 12.21 ve 100.00'dır Ortalama destek sayısı üç ile çarpılırsa (3 kural keşfedilmiştir) %37.9'a yakın değer elde edilmektedir ve bu da pratik olarak tüm kayıtların kapsandığı anlamına gelmektedir. Güven değeri de %100'dür çünkü bölgelerde diğer gruptan kayıt bulunmamaktadır.

KPSOA, aynı zamanda 6 tane halka açık gerçek veritabanında da değerlendirilmiştir: Basketball, Bodyfat, Bolts, Pollution, Quake, ve Sleep. Bu veritabanları Bilkent Üniversitesi Fonksiyon Yaklaştırma Deposunda hazır bulunmaktadır [144]. KPSOA'nın bir karakteristiği de stokastikliğidir. Böylece algoritma farklı çalıştırılmalarda dalgalanmalara sahip olabilir. En iyi sonucun alınabilmesi için algoritma birkaç kez çalıştırılabilir. Algoritma on kez çalıştırılmış ve bu çalıştırmaların ortaklama değerleri sunulmuştur.

Bir önceki deneyde de görüldüğü gibi diğer birliktelik kural keşfi yapan algoritmalarla KPSOA'yı karşılaştırmak zor ve adaletsiz olabilir. Çoğu madencilik algoritması niteliklerin aralıklarının belirlenmesini istemekte ve önişlem gerektirmektedir. Bu yüzden anlamlı olarak karşılaştırma, literatürde birliktelik kurallarının arama sırasında nicel nitelikleri ayrıklaştıran ve evrimsel hesaplama tabanlı iki algoritmayla yapılabilir.

Tablo 4.8'de, KPSOA ve [6]'da sunulan algoritma tarafından bulunan yüksek kaliteli kural sayısı ve bu kuralların güven değerleri (standart sapmalarla birlikte) ile birlikte

veritabanındaki kayıt sayısı ve nicel nitelik sayısı görülmektedir. Deneysel sonuçlarda kural ve güven değerleri kullanılmıştır çünkü [6]'daki algoritma da yoğun nesne kümelerini bulmadan direkt olarak kuralları keşfetmektedir. Ayrıca o algoritma da evrimsel hesaplama tabanlı (bir GA) bir algoritmadır ve aralıkları ve kuralları eş zamanlı olarak bulur. [6]'daki algoritma için popülasyon sayısı 100 seçilmiş ve sadece pozitif birliktelik kurallarını keşfedecek şekilde uyarlanmıştır. Altı veritabanından dördünde, KPSOA fazla sayıda daha kaliteli kural bulmuştur ancak fark çok fazla değildir. Güven değerleri bakımından, bu algoritmayla rekabet edebilir görülmektedir.

Tablo 4.8. [6]'da önerilen algoritmayla karşılaştırmalar

Veritabanı	**Kayıt sayısı**	**Nitelik sayısı**	**Kural sayısı**		**Güven(%)**	
			[6]	**KPSOA**	**[6]**	**KPSOA**
Basketball	96	5	33.8	34.2	60 ∓ 1.2	60 ∓ 2.8
Bodyfat	252	18	44.2	46.4	59 ∓ 3.8	61 ∓ 1.8
Bolts	40	8	39.0	36.4	65 ∓ 1.9	64 ∓ 2.0
Pollution	60	16	41.2	44.6	68 ∓ 4.8	66 ∓ 4.7
Quake	2178	4	43.8	46.4	62 ∓ 5.1	63 ∓ 2.8
Sleep	62	8	32.8	37.6	64 ∓ 2.3	64 ∓ 2.8

Tablo 4.9'da ise KPSOA, GAR [138] ve [6]'da önerilen algoritmalardan elde edilen sonuçlar karşılaştırılmaktadır. GAR algoritması sadece yoğun nesne kümelerini bulmak için bir evrimsel algoritma kullanmaktadır. Bu yüzden, kurallara bağlı değerler ile ilgili karşılaştırmalar yapılmamıştır. "Destek(%)" sütun değerleri ortalama desteği, "Boyut" sütun değerleri kuralda bulunan ortalama nitelik sayısını göstermektedir. "Genlik(%)" sütun değerleri ise kümeye bağlı aralıkların ortalama boyutunu göstermektedir.

KPSOA altı veritabanından üçünde yüksek destek değerli kurallar (nesne kümelerine bağlı) bulmuştur ve fark yine fazla değildir. Tüm veritabanları için KPSOA tarafından elde edilen boyut değerleri GAR algoritmasının bulduklarından daha küçüktür ve altı veritabanından ikisinde [6]'da önerilen algoritmadan elde edilenden daha küçüktür. KPSOA tarafından elde edilen genlik değerleri GAR tarafından elde edilen değerlerden daha küçüktür ya da o değerlere eşittir. Altı veritabanından dördünde de [6]'da önerilen algoritmadan elde edilen genlik değerlerinden daha küçük değerlere sahiptir

Tablo 4.9. Destek, boyut ve genlik sonuçlarının karşılaştırılması

Veritabanı	Destek(%)			Boyut			Genlik(%)		
	KPSOA	GAR	[6]	KPSOA	GAR	[6]	KPSOA	GAR	[6]
Basketball	36.44	36.69	32.21	3.21	3.38	3.21	19	25	20
Bodyfat	65.22	65.26	63.29	6.94	7.45	7.06	25	29	27
Bolts	28.48	25.97	27.04	5.14	5.29	5.14	19	34	27
Pollution	43.85	46.55	38.95	6.46	7.32	6.21	15	15	14
Quake	38.74	38.65	36.96	2.22	2.33	2.10	17	25	19
Sleep	36.52	35.91	37.25	4.19	4.21	4.19	5	5	4

Gerçek verilerdeki son deneysel sonuçlar, Tablo 4.10'dadır ve burada da keşfedilen kurallar tarafından kapsanan kuralların yüzdesi verilmiştir. Bu sonuçlardan da KPSOA algoritmasının diğer evrimsel hesaplama tabanlı algoritmalarla rekabet edebileceği görülebilmektedir.

Tablo 4.10. Keşfedilen kurallar tarafından kapsanan kayıtların yüzdesi

Veritabanı	Kayıtlar(%)		
	KPSOA	GAR	[6]
Basketball	100.00	100.00	100.00
Bodyfat	86.11	86.00	84.12
Bolts	79.80	77.50	77.5
Pollution	95.02	95.00	95.0
Quake	87.92	87.50	87.6
Sleep	81.02	79.03	79.81

Sonuç olarak KPSOA kuralları, aralıkları çok fazla seçmeden ve yüksek güven ve destek değerlerine sahip olacak şekilde keşfetmiştir. Ayrıca kurallardaki nitelik sayısı da azdır. Böylece veritabanlarında keşfedilen kurallar, doğru, okunabilir ve anlaşılabilirdir.

4.6. Sonuçlar

Tezin bu bölümünde kaba örüntü fikrine bağlı olarak kaba parçacık ve kaba karar değişkenleri kullanan KPSOA önerilmiştir. Yani aralıkların birer karar değişkeni olarak kullanılması durumunda PSO'ya yapılacak eklentiler açıklanmıştır. PSO algoritmalarında kullanılan geleneksel parçacık ve değerlerin bunların kaba eşdeğerlerinin özel bir durumu olduğu gösterilmiştir.

Aynı zamanda, bu şekilde kaba hesaplama alanına da ilave bir konu eklenmiştir. Ayrıca seçilen ilk uygulama alanı, nicel nitelikler içeren veri tabanlarında veri madenciliği, için de yeni, etkili ve otomatik bir yöntem önerilmiştir ve bilgi kayıpları ve ön işlemelerden kaçınılarak bilgi keşfi yapılmıştır.

KPSOA pratik uygulamalar için kullanışlı genişletmeler sunmaktadır. Kaba küme teorili PSO algoritmaları ve bunun farklı arama ve optimizasyon problemlerinde daha ayrıntılı deneyleri ileriki çalışmalar olarak düşünülebilir.

5. KABA KAOTİK PSO

5.1. Giriş

Tezin bu bölümünde ise, üçüncü bölümde önerilen kaotik haritalı PSO algoritmaları ile dördüncü bölümde önerilen kaba PSO algoritmasının birleşimi ile kaba kaotik PSO (KKPSO) algoritmaları önerilmiştir. Kaba değerlerle temsil ve hesaplamaların yapılması zorunluluğu olan yerlerde kullanılan kaba PSO algoritmasına, kaotik haritalar eklenmiş ve algoritmanın performansının arttırılması amaçlanmıştır. Böylece; kaba kümeler, kaos ve optimizasyonun birlikte kullanılabileceği genel amaçlı, melez arama ve optimizasyon algoritmaları önerilmiştir. Aynı zamanda kaba hesaplama alanına yeni bir çalışma konusu eklenmiştir. Bu amaçla, KKPSO teorileri açıklanmış ve performansının testi için nicel birliktelik kural madenciliği alanında uygulamalar yapılmıştır. Seçilen bu uygulama alanı için önerilen yöntemin çok uygun ve etkili olduğu gösterilmiştir.

5.2. Kaba Kaotik PSO (KKPSO)

Üçüncü bölümde, PSO'nun parametrelerinin belirlenmesinde rasgele tabanlı bir seçim söz konusu olduğunda farklı kaotik sistemler rasgele sayı dizilerinin yerine kullanılmış ve farklı kaotik haritalı PSO algoritmaları önerilmiştir. Aynı algoritmalar, kaba PSO için de kullanılmış ve kaba kümelerin aralık cebri konusu ile kaosun kaotik haritalar konusunun PSO'ya eklenmesiyle oluşturulan bu yeni algoritmalara kaba kaotik PSO (KKPSO) adı verilmiştir.

Tablo 5.1. Kaotik haritalı PSO algoritmalarının kısa özeti

Ad	**Etki**	**Kaotik Haritanın Ölçekli Değerleri**		
KPSO1	Başlangıç hız ve pozisyon	Her karar değişkeninin alt ve üst sınırları		
KPSO2	c_1	0.5 – 2.5	-	-
KPSO3	c_2	0.5 – 2.5	-	-
KPSO4	c_1 ve c_2	0.5 – 2.5	0.5 – 2.5	-
KPSO5	r_1	0.0 – 1.0	-	-
KPSO6	r_2	0.0 – 1.0	-	-
KPSO7	r_1 ve r_2	0.0 – 1.0	0.0 – 1.0	-
KPSO8	w, r_1 ve r_2	0.0 – 1.0	0.0 – 1.0	0.0 – 1.0
KPSO9	w	0.0 – 1.0	-	-
KPSO10	w ve c_1	0.0 – 1.0	0.5 – 2.5	-
KPSO11	w ve c_2	0.0 – 1.0	0.5 – 2.5	-
KPSO12	w, c_1 ve c_2	0.0 – 1.0	0.5 – 2.5	0.5 – 2.5

KHPSO algoritmalarının kısa bir özeti Tablo 5.1'de verilmiştir. Burada ilk sütun PSO'nun adını, ikinci sütun kaotik haritaların PSO'daki etkilediği parametreleri ve üç alt kısımdan oluşan üçüncü sütun ise bu parametrelerin seçilen kaotik haritanın ürettiği değerlerin ölçeklenmesiyle alacağı değerleri göstermektedir. Tablo, örnek olarak şu şekilde yorumlanabilir. KPSO3 sadece ikinci hızlanma katsayısını etkiler ve ilgili kaotik haritadan alınan değerler 0.5 ile 2.5 arasında ölçeklenmiştir. Eğer kaba temsiller kullanılırsa bu isimlere, "Kaotik" kelimesini temsil eden birinci "K"'dan önce "Kaba"'yı temsil eden ikinci bir "K" eklenir. Yani KPSO1 için kaba temsil kullanılırsa bu KKPSO1 olarak adlandırılır.

5.3. KKPSO Algoritmalarının Veri Madenciliğinde Uygulamaları

KKPSO algoritmaları, kategorik ya da nicel değerler içeren veritabanlarında birliktelik kurallarının keşfi için kullanılmıştır. Bunun için de dördüncü bölümde de uygulaması yapılan ve Agrawal ve arkadaşları tarafından kullanılan ikinci fonksiyon [135], test amaçlı kullanılmıştır. Bu veritabanındaki ilk uygulamada, 5000 adet veri oluşturulmuş ve bu verinin %37.9'unun *A* grubuna ait olması sağlanmıştır. Bu amaçla dördüncü bölümde açıklanan parçacık temsili, uygunluk fonksiyonu ve mutasyon çeşitleri kullanılmıştır. Algoritma parametre değerleri Tablo 5.2'de verilmiştir.

Tablo 5.2. Kullanılan parametre değerleri

Parametreler	**Sürü boyutu**	**İterasyon sayısı**	**Mutasyon olasılığı**
Değerler	20	1000	50%

Yapılan deneyde, önerilen on iki KKPSO algoritmalarının nicel birliktelik kurallarının keşfinde performansları karşılaştırılmıştır. Algoritmaların etkinliliğini doğrulayan sezgisel bir ölçü, kuralların aralıklarının sentetik olarak oluşturulanlarla karşılaştırılması ve böylece kuralların kalitelerinin belirlenmesidir. KPSOA tarafından keşfedilen kuralların ortalama destek ve güven değerleri 12.21 ve 93.33'tür. Burada hızlandırma katsayıları için 2 değeri, atalet ağırlığı için ise 0.9–0.4 arası azalan değerler seçilmiştir. KKPSO1 algoritması için de aynı değerler kullanılmıştır.

KKPSO algoritmaları tarafından elde edilen sonuçlar Tablo 5.3'te verilmiştir. Tüm algoritmalar üç kural keşfetmiştir ve bu kurallara ait ortalama destek ve güven değerleri sunulmuştur. Ortalama destek sayısı üç ile çarpıldığında Grup *A*'ya ait kayıtların oranı olan %37.9'a yakın değerler elde edilebilmektedir. Güven değerleri de %100'e yakındır. Bu da oluşturulan ilgili bölgelerde, diğer gruba ait kaydın olmadığını göstermektedir. KKPSO7,

KKPSO8 ve KKPSO12 algoritmaları Zaslavskii haritasıyla kullanıldığında iyi performans göstermiştir. KKPSO7 algoritmasının Zaslavskii haritasıyla kullanıldığında keşfettiği kurallar Tablo 5.4'tedir.

Tablo 5.3. KKPSO algoritmaları tarafından elde edilen kuralların ortalama destek ve güven değerleri (Grup A=Toplam veri sayısının %37.9'u)

Lojistik Harita												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	11.24	10.64	10.96	11.98	10.78	10.80	12.48	12.28	10.82	10.66	11.02	11.54
Güven (%)	98.14	98.12	97.01	97.64	97.42	97.96	99.01	98.98	98.42	98.24	98.56	99.08
Sinüzoidal Yineleyici												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	11.20	11.24	10.86	11.62	10.86	10.96	12.38	12.28	10.58	10.90	11.06	11.24
Güven (%)	98.48	98.46	98.08	98.06	97.56	97.02	99.08	98.96	98.54	97.98	99.02	98.96
Gauss Haritası												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	11.91	11.05	10.86	10.24	10.84	10.97	12.28	12.23	10.81	10.23	9.99	11.98
Güven (%)	97.22	97.62	98.68	98.16	98.26	97.64	99.04	99.06	98.48	97.86	97.56	98.62
Zaslavskii Haritası												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	11.05	11.26	10.24	10.22	10.97	10.96	12.62	12.41	10.96	10.82	10.88	11.94
Güven (%)	97.56	97.55	97.22	98.08	98.48	98.69	99.65	99.12	98.62	98.86	97.16	98.84

Tablo 5.4. Zaslavskii haritası kullanan KKPSO7 algoritması tarafından bulunan kurallar

Kural	Destek(%)	Güven(%)
Eğer $yaş \in [20, 40] \wedge maaş \in [50136, 99869] \Rightarrow$ Grup A	12.63	98.94
Eğer $yaş \in [41, 59] \wedge maaş \in [76779, 12469] \Rightarrow$ Grup A	12.62	100
Eğer $yaş \in [61, 80] \wedge maaş \in [25440, 73998] \Rightarrow$ Grup A	12.61	100

İkinci uygulamada ise gruplar dengeli dağıtılmıştır. Yani A grubuna ait kayıt sayısı, toplam kayıt sayısının yarısı (2500) olarak belirlenmiştir. Böylece, keşfedilecek kuralların toplam destek değerinin %50'ye yakın olması beklenmektedir. KKPSO algoritmaları tarafından elde edilen sonuçlar Tablo 5.5'te verilmiştir. Tüm algoritmalar, her biri ata kısımda iki nitelik

içeren üç kural keşfetmiştir ve bu kurallara ait ortalama destek ve güven değerleri sunulmuştur. Ortalama destek sayısı üç ile çarpıldığında, Grup *A*'ya ait kayıtların oranı olan %50'ye yakın değerler elde edilebilmektedir. Güven değerleri de %100'e yakındır ve bu da oluşturulan ilgili bölgelerde diğer gruba ait kaydın olmadığını göstermektedir. Keşfedilen kuralın gücü sayılan güven değerlerinin yüksek olması, önerilen algoritmaların doğru ve tutarlı kural keşfettiklerini göstermektedir. Yine KKPSO7, KKPSO8 ve KKPSO12 algoritmalarının Zaslavskii haritasıyla kullanıldığında iyi performans gösterdiği tespit edilmiştir. KKPSO8 algoritmasının Zaslavskii haritasıyla kullanıldığında keşfettiği kurallar Tablo 5.6'dadır. Burada özellikle destek değerleri arasındaki küçük fark, aslında veritabanındaki kayıtların özelliklerine göre büyük önem arz edebilmektedir.

Nicel niteliklere ait bölme noktalarının önerilen yöntemlerle doğru olarak bölündüğü, ayrıca fazla nitelik ya da fazla kuralın elde edilmediği, böylece güven ve destek değerleri yüksek, anlaşılabilir kuralların keşfedildiği görülmektedir.

Tablo 5.5. KKPSO algoritmaları tarafından elde edilen kuralların ortalama destek ve güven değerleri (Grup A=Toplam veri sayısının %50'si)

Lojistik Harita												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	13.86	13.94	13.96	13.98	13.92	13.98	13.90	14.96	14.88	14.86	14.24	14.62
Güven (%)	100	100	100	100	99.88	100	100	100	100	100	100	100
Sinüzoidal Yineleyici												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	14.20	14.26	13.96	14.52	14.44	14.96	15.24	14.28	14.32	14.92	14.62	14.20
Güven (%)	100	100	100	98.98	100	100	100	100	100	100	100	100
Gauss Haritası												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	14.68	15.58	14.80	14.64	14.44	14.91	14.62	15.28	14.54	14.54	13.84	15.68
Güven (%)	100	100	100	100	100	100	100	100	100	100	100	100
Zaslavskii Haritası												
	KKPSO 1	KKPSO 2	KKPSO 3	KKPSO 4	KKPSO 5	KKPSO 6	KKPSO 7	KKPSO 8	KKPSO 9	KKPSO 10	KKPSO 11	KKPSO 12
Destek (%)	15.26	15.36	15.62	15.48	15.88	15.28	15.94	16.06	15.66	15.80	15.36	15.88
Güven (%)	100	100	100	100	100	100	100	100	100	100	100	100

Tablo 5.6. Zaslavskii haritası kullanan KKPSO8 algoritması tarafından bulunan kurallar

Kural	Destek(%)	Güven(%)
Eğer $yaş \in [20, 39] \wedge maaş \in [50120, 99902] \Rightarrow$ Grup A	16.12	100
Eğer $yaş \in [40, 59] \wedge maaş \in [75972, 12488] \Rightarrow$ Grup A	15.98	100
Eğer $yaş \in [61, 80] \wedge maaş \in [25094, 74908] \Rightarrow$ Grup A	16.08	100

5.4. Sonuçlar

Tezin bu bölümünde, kaba değerlerle temsil ve hesaplamaların yapılması zorunluluğu olan yerlerde kullanılan kaba PSO algoritmasına kaotik haritalar eklenmiş ve algoritmanın performansının arttırılması amaçlanmıştır. Bu amaçla uygulama alanı veri madenciliği seçilmiş ve nicel birliktelik kurallarının keşfinde bu önerilen algoritmaların performans karşılaştırılması yapılmıştır. Nicel birliktelik kurallarının keşfi ve nicel niteliklerin aralıklarının otomatik bölünmesi işini, eş zamanlı ve doğru olarak yapan algoritmalar literatürde pek fazla yoktur ve bu gibi durumlarda bu problemi bir optimizasyon problemi olarak ele alıp çözebilen yumuşak hesaplama tekniklerine ihtiyaç duyulabilmektedir. Tezin bu bölümünde de bu zor problem için farklı yumuşak hesaplama tekniklerinin birleşimiyle etkili sonuçlar verebilecek KKPSO algoritmaları tanıtılmış ve veri madenciliği uygulamaları yapılmıştır.

Bu bölüm, üçüncü bölümde önerilen on iki farklı kaotik haritalı PSO'nun, aralığın karar değişkeni olarak kullanıldığı uygulama alanları için kaba değişkenin parçacık temsilinde ve hesaplamalarında kullanıldığı kaba PSO algoritmasıyla birleştirilmesinin bir sonucudur. Önerilen yöntemler çok farklı alanlarda uygulama alanı bulabilir ve etkili sonuçlar verebilir.

6. ÇOK AMAÇLI PARÇACIK SÜRÜ OPTİMİZASYONU

6.1. Giriş

Çoğu mühendislik problemleri ile ilgili gerçek uygulamada genellikle birden fazla amacın optimizasyonu ile ilgilenilir. Yani bir amaç fonksiyonu yerine k tane amaç fonksiyonunun maksimize ya da minimize edilmesi söz konusudur. Problemin optimum çözümü ya da çözümleri, tüm amaç fonksiyonlarını birlikte (simültane olarak) optimize eden çözümdür. Böyle bir çözüme ulaşmak çok zordur [145]. Çünkü tek bir amaca ilişkin olarak en iyi değeri veren bir çözüm, birden fazla amaç söz konusu olduğunda aynı sonucu vermeyebilir. Yani, genellikle göz önüne alınan amaçlar, diğer bir deyimle değerlendirme ölçütleri birbiri ile çelişkili ve negatif yönde etkileşimli olabilmektedir. Örneğin bir veri madenciliği süreci sonunda elde edilen kuralların yüksek doğruluğa ve güce sahip olması, ilginç, kolay okunabilir ve anlaşılabilir olması istenmektedir. Başka bir örnek ise alınmak istenen herhangi bir cihazın çok fonksiyonlu, kaliteli ve kolay kullanılabilir olması ve fiyatının düşük olması istenir. Kısaca, birden fazla amaç dikkate alındığında amaçlar arasında ödünleşimler söz konusu olabilmektedir.

Çok amaçlı optimizasyon son birkaç yıldır önemli bir araştırma alanı haline gelmiştir [145]. Aslında sınıflandırma kural madenciliği de aynı birimle ölçülebilen ve çoğu zaman çelişen değişik amaçlarla çok amaçlı bir optimizasyon problemi olarak tasarlanabilir. Sınıflandırma kural madenciliği için çok amaçlı yaklaşım kullanan çok kısıtlı çalışma vardır [146–150].

Farklı veri madenciliği algoritmaları için çok amaçlı yaklaşım olarak en çok kullanılan üç yöntem; verilen bir formüle göre (ağırlıklı formül) tüm amaçları ağırlıklandırma, amaçlara bazı öncelik şemaları atama (sözlüksel) ve tek bir çözümden ziyade olası en iyi çözümler kümesini bulan Pareto baskınlık yaklaşımıdır. Literatürde PSO'yu çok amaçlı problemler için düzenlemeyi içeren yayınlar bulunmaktadır [151–153]. Aynı zamanda tek amaçlı PSO kullanarak sınıflandırma kural madenciliği yapan birkaç çalışma da vardır [154–156]. Fakat bunlar çok amaçlı özelliğe sahip değildir. Tezin bu bölümünde de PSO'nun çok amaçlı optimizasyon problemlerinde kullanılabilmesi için farklı bir yöntem önerilmiş ve ilk olarak veri madenciliği tekniklerinden sınıflandırma kural madenciliği alanında uygulamaları yapılmıştır. Doğru ve anlaşılabilir sınıflandırma kuralların, düzenlenen PSO ile keşfi amaçlanmıştır.

6.2. Çok Amaçlı Optimizasyon

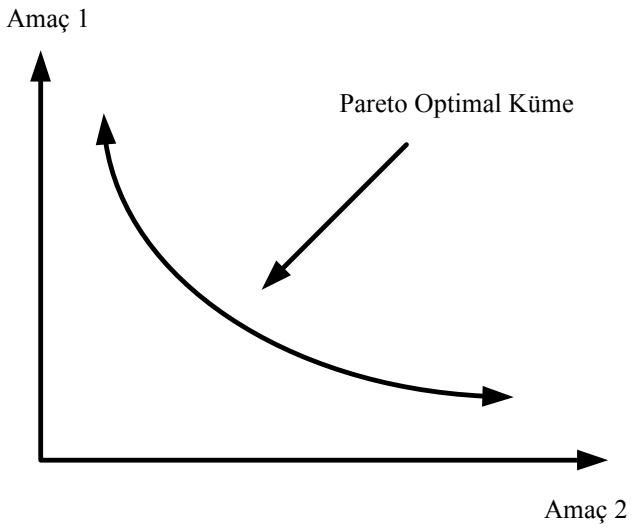
Çok amaçlı optimizasyon iki ya da daha fazla amaç fonksiyonunu eş zamanlı optimize etme işidir. Genellikle amaç fonksiyonları istenen bir sonucun farklı özelliklerini dikkate alır. Çoğu zaman bu amaçlar, tüm fonksiyonları eş zamanlı olarak optimize eden tek bir sonucun bulunmadığı bir çelişki içindedirler. Bunun yerine bir optimal çözümler kümesi söz konusu olur. Bu küme de Pareto optimallık kavramı kullanılarak tanımlanır ve çoğunlukla Pareto optimal küme olarak adlandırılır [157].

Bir çözüm vektörü $x \in X$, X arama kümesinde x'e baskın olan herhangi bir çözüm olmadığı zaman Pareto optimal olur. Baskın olma ise şu şekilde tanımlanır:

Bir çözüm vektörü x aşağıdaki şartlar sağlandığında y'ye baskındır denir:

- $\forall i \in \{1, 2, ..., I\}\ f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{y})$ ve
- $\exists i \in \{1, 2, ..., I\}\ f_i(\boldsymbol{x}) > f_i(\boldsymbol{y})$.

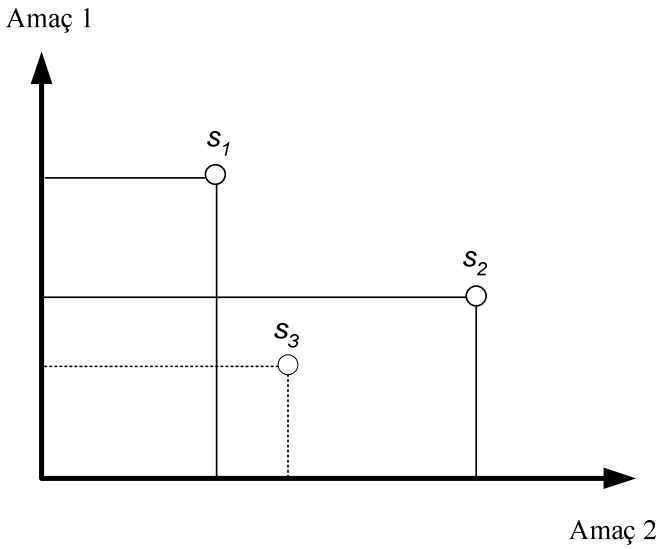
I amaç fonksiyon sayısıdır. Bu tanım şu anlama gelmektedir: x, her amaca dair y kadar iyiyse ve x'in y'den daha iyi olduğu konusunda en az bir amaç varsa y'ye baskındır denir. Bu şekilde Pareto optimal çözümler Pareto optimal kümeyi oluşturur [157]. Minimizasyonu istenen iki amaç fonksiyonlu bir problemin örnek bir Pareto optimal kümesi Şekil 6.1'de gösterilmiştir.



Şekil 6.1. Pareto optimal küme

Baskınlık ve Pareto optimallık, çok basit olarak Şekil 6.2'de tasvir edilmiştir. s_1, s_2 ve s_3 gibi üç tane çözümün ve maksimize edilecek iki amacın olduğunu varsayalım. s_1 amaç 1 için en

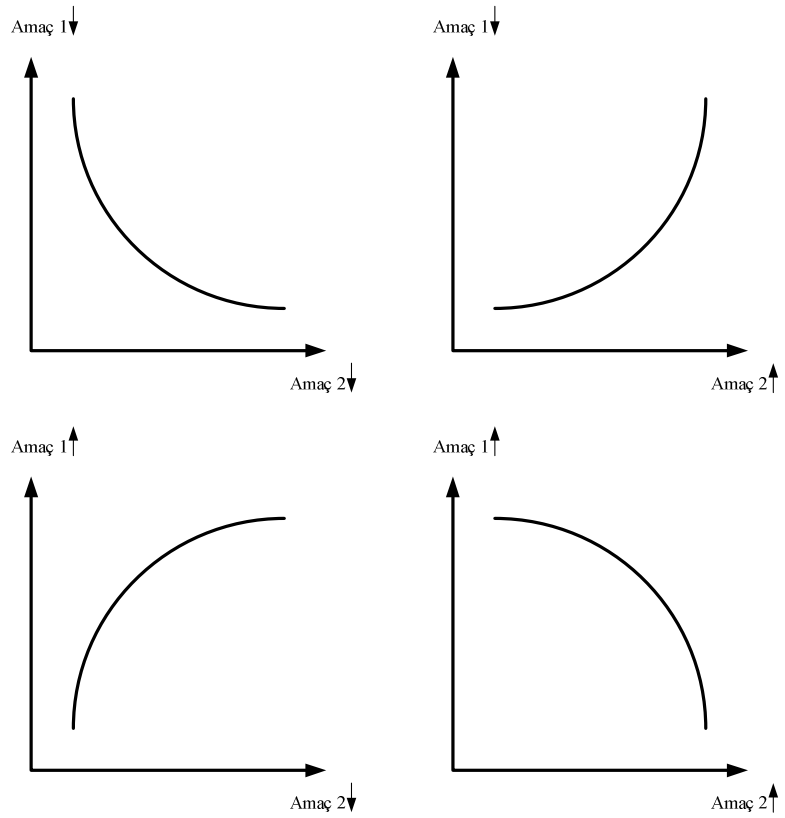
yüksek değere sahip olduğundan herhangi bir çözüm tarafından baskın değildir. Benzer olarak s_2 de amaç 2 için en yüksek değere sahip olduğundan hiçbir çözüm s_2'ye baskın değildir. s_3 amaç 2 için yüksek değere sahip olduğundan, s_1 tarafından baskın değildir, ancak iki amaç için de s_2'den daha düşük değere sahip olduğundan s_2 tarafından baskındır. Böylece s_1 ve s_2'ye baskın herhangi bir çözüm bulunmamakta, s_3'e ise bir çözüm, s_2, baskın olmaktadır. Bu yüzden baskın olmayan ya da Pareto optimal çözüm *Pareto_Küme*=$\{s_1, s_2\}$ şeklinde gösterilir.



Şekil 6.2. Baskınlık ve Pareto optimallik kavramı

Şekil 6.3 ise minimizasyon ya da maksimizasyona ilişkin optimizasyon konfigürasyonlarına bağlı iki amaçlı problemler için olası Pareto optimal kümeleri göstermektedir. Burada ↑ maksimizasyonu, ↓ ise minimizasyonu temsil etmektedir.

Çok amaçlı optimizasyonda amaç, farklı Pareto optimal çözüm kümelerini bulmaktır. Evrimsel çok amaçlı optimizasyonda, tek bir evrimsel algoritma çalışmasıyla çözümler kümesi bulunur. Uygulaması yapılan kural madenciliği alanında da, tüm amaçlar eş zamanlı düşünüldüğünde geniş anlamda herhangi bir kuralın diğerinden üstün olmadığı optimal, yüksek kaliteli sınıflandırma kurallarının PSO'nun tek çalıştırılmasında bulunması hedeflenmiştir.



Şekil 6.3. İki amaçlı bir problem için örnek Pareto optimal kümeler

6.3. Sınıflandırma Kural Madenciliği ve İlgili Çalışmalar

Sınıflandırma kurallarının madenciliği, en çok kullanılan ve insan düşünce yapısına en yatkın veri madenciliği tekniklerinden biridir. Bu teknik ile bir veri kümesinden kullanıcıların çok kolay anlayacağı kurallar çıkarılır. Bu teknikte veri için, nitelikler kullanılarak dağılıma göre bir model bulunur. Bulunan bu model, başarısı belirlenerek niteliğin gelecekteki ya da bilinmeyen değerini tahmin etmek için yani en doğru sınıfa atamak için kullanılır. Kısaca sınıflandırmada, yeni gelen her bir örnek, önceden sınıflandırılmış bir takım sınıflar üzerinde yapılan bir eğitim neticesinde ortaya çıkan bir modele göre daha önce belirlenmiş olan bir sınıfa atanmaktadır. Bu bağlamda kullanılan belki de en önemli değerlendirme kriterleri, tahmini doğruluk ve anlaşılabilirliktir. Tahmini doğruluk, genelleme olarak ta bilinir ve oluşturulan modelin daha önce görülmemiş örnekleri sınıflandırmada ne kadar performanslı olduğunun bir ölçüsüdür. Anlaşılabilirlik ise, oluşturulan modelin kullanıcılar tarafından anlaşılabilirliğini ölçer [56].

Sınıflandırma için çeşitli yöntemler ve algoritmalar bulunmaktadır. Karar ağaçları, sınıflandırma için güçlü bir modeldir. Bunlar, C4.5 [158] ve CART [159] gibi tekniklerle oluşturulur ve 'böl-ve-yönet' stratejisini uygular. Veri, ayrı alt kümelere ayrılır ve algoritma her kümeye tekrarlı olarak uygulanır. Karar ağaçlarının en önemli avantajı, türetilen bir model olarak karar verme işlemine açık bir şekilde hakim olmasıdır. Çok sayıda işlem yapmaya gerek duymadan sınıflandırma işlemini gerçekleştirebilir. Ancak bunlar, oluşturulmaları sırasında eğitim verisinde örneklerin saf alt kümelerini belirleme eğilimindedir. Bu da yanlış ya da tutarsız olan örneklere aşırı uymaya neden olabilir ve böylece son modelin genelleme gücünü azaltır. Bu problemin üstesinden gelmek için kural budama ve benzeri yardımcı yordamlar kullanılmaktadır. Ayrıca, karar ağaçları, tahmin için kullanıldığı durumlarda tahmin edilecek değişkenin sürekli değerler alması durumunda uygun sonuçlar üretememektedir.

Karar listeleri de eğitim verisinden çıkarılan bilginin açık bir temsilini belirli şekilde göstermesiyle karar ağaçlarına benzer. Ancak bunlar 'ayır-ve-yönet' yaklaşımını kullanır ve bir kural, eğitim verisinin bir alt kümesini kapsamak için oluşturulur ve sonra daha fazla kural, kalan örnekleri tekrarlı olarak kapsamak için üretilir. Bu strateji, ilk olarak AQ ailesinde [160] uygulanmıştır ve daha sonra CN2 [161] gibi algoritmalara temel teşkil etmiştir. Algoritmanın sonunda sıralı EĞER-O ZAMAN kurallarının listesi elde edilir ve yeni bir örneğin sınıflandırılmasında sırayla uygulanır. Eğer listedeki ilk kural örneği kapsamıyorsa, o zaman bir sonraki denenir. İkincisi de çalışmazsa, listedeki üçüncü kural denenir ve böylece devam eder. Bir örnek bir kural tarafından sınıflandırılırsa, daha fazla kural denenmez. Eğer kuralların hiçbiri örneği kapsamıyorsa, o zaman karar listesinin en altındaki varsayılan bir kural işletilir. Yani varsayılan kurala ulaşan tüm sınıflandırılmamış örnekler bu kuralın sınıf etiketiyle işaretlenir.

Sıralı listelerin bir dezavantajı, bireysel kuralların kendilerinin anlaşılma bakımından zor olabilmesidir. Bir listedeki bir kural, önceki tüm kuralların bağlamında ele alınmalıdır. Karar ağaçları gibi, karar listeleri de gürültülü eğitim verisine aşırı uyma problemiyle karşı karşıyadır ve bu yüzden genellikle kural budama işlemi uygulanır.

Evrimsel hesaplama, özellikle GA ve genetik programlama da etkili şekilde sınıflandırma kural madenciliğinde kullanılmıştır [162, 163]. Bu yaklaşımla arama uzayı üzerinde global bir arama yapılır ve kaba seçim algoritmalarına göre nitelik etkileşimiyle daha iyi baş edilebilir. Ayrıca açıklanabilir sonuçlar üretirler ve çok değişik tiplerdeki verileri işleme özelliğine sahiptir. Ancak, optimal sonucun üretildiğine dair bir garanti bulunmamaktadır ve bazen ağır işlem yükü gerektirebilir. Ayrıca, sürü zekâsı tekniklerinden karınca koloni optimizasyon algoritması temelli algoritmalar [164] ve yapay bağışıklık sistemlerinden klonal seçim algoritması da sınıflandırma kurallarının keşfi için kullanılmıştır [14]. Sürü zekâsı

konusunun yeni ve aktif araştırma konusu PSO algoritmasının sınıflandırma kural madenciliğinde kullanımı çok yenidir ve şu ana kadar sadece iki çalışma yapılmıştır. Yakın zamanda Sousa ve arkadaşları tarafından kullanılmıştır [165]. Aynı zamanlarda Liu ve arkadaşları tarafından da kullanılmıştır [166].

Sınıflandırmaya ayrıca örnek-tabanlı öğrenme, yapay sinir ağları, lojistik gerileme ve Bayesian ağları yaklaşımları da vardır. Bu metotların çoğunun temel dezavantajı, tahmini doğrulukları bazı durumlarda iyi olmasına rağmen, açıklayıcı güçlerinin eksikliğidir. Literatürdeki bu yöntemlere bazen bulanık mantık ta eklenerek bulanık kurallar üretilmiştir [10].

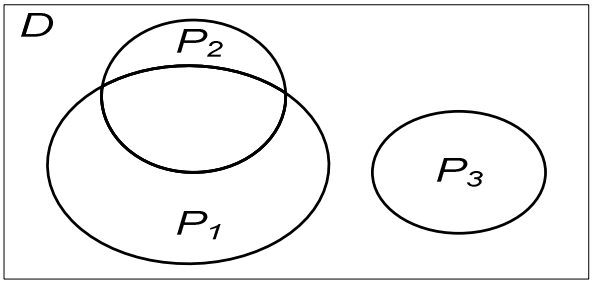
Kural budama gürültülü eğitim verisine aşırı uymadan kaçınmak için gerekli bir işlemdir. Karar listelerinde kural budama için iki temel strateji vardır. Birincisi komple bir kural kümesi oluşturulur ve sonra nitelikleri kurallardan elimine edilerek ya da bireysel kurallar silinerek kural kümesi basitleştirilir. Bu global olarak kural kümesinin önceden tanımlı bazı budama kriterlerine bağlı olarak optimize edilmesiyle yapılır. İkinci strateji ise artımsal budama olarak adlandırılır. Çünkü her kural, algoritmayla oluşturulduktan hemen sonra basitleştirilir [56].

Tezin bu bölümünde ise sınıflandırma kural madenciliği çok amaçlı bir optimizasyon problemi olarak ele alınmış ve doğru, anlaşılabilir kural listesi düzenlenen PSO algoritmasıyla elde edilmeye çalışılmıştır. Bu yöntemde, yoğun işlem gerektiren budama işlemine gerek duyulmamakta ve bu iş kural keşif aşamasında direkt halledilmektedir. Ayrıca bu yöntem 'ayır-ve-yönet' stratejini kullanmaz. Onun yerine, veritabanını azaltmadan her seferinde her sınıf için Pareto tabanlı çok amaçlı optimizasyon fikrini uygular. Bu şekilde [165] ve [166]'da önerilen yöntemlerde ortaya çıkabilecek kurallar arasında beklenmedik etkileşimler ortadan kalkacaktır. Bu etkileşimler, bir örnek farklı sınıfların birkaç kuralı tarafından kapsandığı zaman ortaya çıkabilir.

[165] ve [166]'da önerilen algoritmalarda parçalanma problemi ortaya çıkabilir [167]. Kapsama algoritmaları bir kural üretildiğinde tüm eğitim verisindeki kapsanan örnekleri çıkarır ve iterasyonlardan sonra eğitim örneklerinin sayısını azaltır ve lokal olarak önemli ancak global olarak önemsiz kuralların üretilmesine yol açar. Bu çalışmada önerilen yöntemle, bu global önemli kuralların aranması sağlanır.

Şekil 6.4, bu tür kapsama algoritmalarında ortaya çıkabilecek olası bir parçalanma problemini göstermektedir. *D* eğitim verisinde önem sırasına göre listelenmiş P_1, P_2 ve P_3 gibi üç kuralın olduğunu varsayalım. Ardışık kapsama algoritmaları önce P_1'i keşfeder ve P_1 tarafından kapsanan pozitif örnekleri çıkarır. Bu taşınmadan dolayı kalan veride P_2, P_3'ten daha az önemli hale gelir ve arama P_2 yerine P_3'ü bulur. İterasyonlardan sonra keşfedilen kurallar daha çok lokal olarak önemli olacaktır ve global olarak önemli olan kuralları kaybetme şansı

sürekli artacaktır. Bu çalışmada önerilen yöntemle parçalanma probleminin önüne geçilmiştir. Çünkü hiçbir eğitim verisi çıkarılmaz ve algoritma tüm eğitim verisine bağlı olarak çalışır.



Şekil 6.4. Parçalanma problemi

6.4. PSO Tabanlı Çok Amaçlı Kural Madenciliği için bir Model

6.4.1. Parçacık Temsili

Her parçacık bir kuralı temsil eder ve algoritma bitim şartı sağlandığında her sınıf için doğru ve anlaşılabilir kuralların kümesini oluşturur. Her parçacık, [0, 1] aralığında iki kısımdan oluşan reel değerli elemanların birleşimidir ve Şekil 6.5'te gösterilmiştir. *m* tane karar niteliği varsa her parça *m* elemandan oluşur. Böylece parçacık boyutu 2*m* olur. Bir parçacığa bağlı olarak bir kural oluşturmak için, parçacıkta kodlanmış veri iki kısımda orijinal bilgiye dönüştürülür. Eğer birinci (Nitelik-varlık kısmı-*NV*) kısımda *i*. elemanın değeri 0.5'ten büyükse o zaman *i*. nitelik kuralın ata kısmında yer alacak aksi durumda yer almayacaktır. Parçacıkta ikinci kısım Nitelik-değer (*ND*) kısmı olarak adlandırılır. *ND* kısmı nispeten komplekstir, çünkü farklı tipteki niteliklerin dikkate alınması gerekmektedir.

Tamsayı tipi için dönüşüm Denklem (6.1)' de verilmiştir [166].

$$V_{\text{orj}}[i]=tavan(v_i \times (V_{imax}-V_{imin})+V_{imin}) \quad (6.1)$$

Nominal tipteki nitelikler için ise dönüşüm Denklem (6.2)'deki gibidir [166].

$$V_{\text{orj}}[i]= DiziDeger_i(tavan(v_i) \times DegerSayisi_i) \quad (6.2)$$

Burada $V_{\text{orj}}[i]$ *i*. nitelik için parçacıktan dönüştürülen değer, v_i parçacıktaki *i*. değer anlamına gelmektedir. Denklem (6.1)'de *i*. niteliğin tipi tamsayı ya da reeldir ve V_{imax} *i*. niteliğin maksimum değerini V_{imin} de minimum değerini göstermektedir. Denklem (6.2)'de niteliğin tipi

nominaldir ve $DiziDeger_i$ i. niteliğin her farklı nominal değerini tutan dizidir ve $DegerSayisi_i$ de i. niteliğin toplam değişik değer sayısıdır. *tavan()* bilinen tavan fonksiyonudur [166].

$Eleman_1$		$Eleman_2$		...	...	$Eleman_m$	
NV_1	ND_1	NV_2	ND_2	...	...	NV_m	ND_m

Şekil 6.5. Parçacıktaki kural temsili

6.4.2. Amaç Tasarımı

Keşfedilen kuralların yüksek tahmini doğruluk ve anlaşılabilirliğe sahip olması gerekmektedir. Bu yüzden tahmini doğruluk ve anlaşılabilirlik ölçüleri tanımlanmalıdır.

$$Tahmini\ Doğruluk = \frac{|A \& S| - 1/2}{|A|} \tag{6.3}$$

$|A \& S|$ kuralın hem ata (A) hem de sonuç (S) kısmını karşılayan kayıt sayısıdır. $|A|$ ise sadece kuralın ata kısmındaki şartları sağlayan kayıtların sayısıdır.

$$Anlaşılabilirlik = 1 - \frac{|\text{NV kismi} > 0.5|}{|\text{NV kismi}|} \tag{6.4}$$

Anlaşılabilirlik ölçüsünde $|\text{NV kismi}|$ parçacıkta kodlanan şartların (karar niteliği) sayısıdır ve benzer olarak $|\text{NVkismi} > 0.5|$ da parçacığın NV kısmında 0.5'ten daha büyük değere sahip olan şartlarının sayısıdır.

6.4.3. Çok Amaçlı Yaklaşım

Kullanılan teknik [153]'te önerilen tekniğe dayanmaktadır. O ana kadar bulunan global en iyi parçacıkları depolamak için bir harici depo (HD) muhafaza edilir ve en başta sürüde tüm baskın olmayan parçacıklarla doldurulur. Daha iyi bir çözüm bulunur bulunmaz, HD bu çözümlerle güncellenir [153].

Bu yaklaşımda her parçacıktan iki parçacık türetilir. i. parçacık için, x_i pozisyon ve v_i de hız olmak üzere, $çocuk_1$ ve $çocuk_2$ şeklinde iki parçacık türetilir ve bunların pozisyon ve hızları şu şekilde gösterilir:

$x_l=x_r=x_i$

$v_l=v_r=v_i$

Elde edilen Pareto optimal kümenin aralarını maksimize etmek için komşuluk ötesi arama gerçekleştirilir ve burada ilk çocuk parçacık, *çocuk*$_1$, için bir global optimum seçilir.

Komşuluk ötesi arama için HD'de p tane aday çözümün olduğunu varsayalım. Her çözüm için benzerlik ölçüsü (*Ben*) hesaplanır:

$$Ben(i,j)=\sum_{i=1}^{n}\left[\left|f_t(x_i)-f_t(x_j)\right|\le\varepsilon\right] \tag{6.5}$$

Bir çözümün benzerlik değeri ne kadar fazlaysa bu çözümün komşuluğunda daha fazla aday çözüm yerleşmiş demektir. Bu yüzden, minimum benzerliğe sahip çözüm, *çocuk*$_1$'in global optimumu olarak seçilecektir.

Sonra *çocuk*$_1$'in hız ve pozisyonu, PSO güncelleme denklemleriyle güncellenir. Eğer yeni çözüm x_l' uygunsa ve HD'deki herhangi bir çözüm tarafından baskın değilse, HD x_l' eklenmesiyle ve x_l' tarafından baskın bulunan çözümlerin çıkarılmasıyla güncellenir.

İkinci çocuk parçacık, *çocuk*$_2$, için global optimum komşuluk arama stratejisi kullanılarak HD'den seçilir. Komşuluk arama stratejisi, HD'de *çocuk*$_2$'ye en yakın çözüm *çocuk*$_2$'nin global optimumu olarak seçilecektir fikrine dayanmaktadır. Bu strateji parçacıkların kendi komşuluklarında arama yapmalarını sağlar ve daha yüksek yakınsama hızı sunar [153].

çocuk$_2$ için hız ve pozisyon güncellemeleri PSO güncelleme denklemleriyle yapıldıktan sonra yeni bir çözüm x_r' elde edilir. Sonra HD x_r' ile güncellenir. Çocuk parçacıklar uçmayı tamamladıklarında, i. parçacık ta güncellenmelidir. Eğer x_l' uygunsa, i. parçacık *çocuk*$_1$ ile güncellenir. Aksi durumda *çocuk*$_2$ ile güncellenir. Bu kısımdaki uygunluk ise, amaçların belirlenen ağırlıklarla çarpılıp toplanmasıyla elde edilen değerin büyüklüğüyle ölçülür.

Son olarak i. parçacığın kişisel en iyi pozisyonu (p_i) x_i ve p_i arasındaki Pareto baskınlık ilişkisi karşılaştırılarak güncellenir. p_i x_i'ye baskınsa, p_i hala i. parçacığın kişisel en iyi pozisyonu olacaktır. Aksi durumda x_i p_i'nin yeni değeri olarak tanımlanacaktır. Biri birilerine baskın değillerse, amaçların belirlenen ağırlıklarla çarpılıp toplanmasıyla elde edilen değeri büyük olan p_i'nin yeni değeri olacaktır.

6.4.4. Parametre Kontrolü

Veritabanlarında sınıflandırma kuralları keşfetmek için tasarlanan çok amaçlı PSO'da parçacık sayısı 30 olarak belirlenmiştir. Sonlandırma kriteri ise 500 iterasyon olarak sabitlenmiştir. Atalet ağırlığı komşuluk ötesi arama stratejisi kullanıldığında 1.2'den 0.8'e azalan şekilde seçilmiştir. Komşuluk arama stratejisi kullanıldığında ise 0.9'dan 0.4'e doğru azalır. $\lambda = 1$; $V_{min} = X_{min} = 0$; $V_{maks} = X_{maks} = 1$. Hızlanma katsayıları da $c_1 = c_2 = 2$ olarak belirlenmiştir. *Tahmini doğruluk* için ağırlık 0.7, *anlaşılabilirlik* için ise 0.3 seçilmiştir.

6.5. Deneysel Sonuçlar

UCI makine öğrenmesi veritabanından "Monk1" ve "Mushroom" veritabanı ilk deneyler için kullanılmıştır [168]. Monk1 veritabanında 124 kayıt ve 7 nitelik bulunmaktadır. Sınıf niteliği "class"'tır ve "0" ya da "1" değerini alır. Mushroom veritabanında ise 8124 kayıt ve 23 nitelik bulunmaktadır. Bu veritabanında da sınıf niteliği "class"'tır ve "e" ya da "p" değerini alır [168].

Elde edilen kural listesinin diğer algoritmalarla karşılaştırılması için her iki veritabanı iki kısma ayrılmıştır: Eğitim kümesi (kayıtların 2/3'si) ve test kümesi (kayıtların 1/3'i). Yani eğitim verileri üzerinden çalıştırılan algoritma test verileri üzerinden değerlendirilmiştir. Monk1 için test verisi sayısı 18 tanesi birinci sınıfa (0), 25 tanesi de ikinci sınıfa (1) ait olmak üzere toplam 43'tür. Mushroom için ise test verisi sayısı 1410 tanesi birinci sınıfa (e), 1353 tanesi de ikinci sınıfa (p) ait olmak üzere toplam 2763'tür. Tüm veri üzerinden, önerilen algoritma ile elde edilen kural listesi Tablo 6.1'de verilmiştir. Bu tabloda tüm veri üzerinden; sınıflar (S), önerilen algoritma ile elde edilen kural listesi, kuralların tahmini doğrulukları (TD) ve anlaşılabilirlik değerleri (A) gösterilmiştir. Bu veritabanında karar listesi şeklinde kuralları üreten diğer algoritmaların elde ettiği sonuçlar, karşılaştırmalı olarak Tablo 6.2'de gösterilmiştir. Ridor [143] algoritmasında bir kuralda örneklerin toplam minimum ağırlığı 1 olarak belirlenmiş ve standart değerleriyle çalıştırılmıştır. PART [142] algoritması için budama amacıyla kullanılan güven faktörü 0.25 seçilmiş ve azaltılmış hata budaması için C4.5 budaması kullanılmıştır. OneR [141] algoritmasında nicel nitelikleri ayrıklaştırmak için kullanılan sepet sayısı 6 olarak belirlenmiştir. Prism algoritması da sadece nominal niteliklerle çalışır, budama yapmaz ve kayıp niteliklerle çalışamaz [169]. NNge algoritması ise iç içe geçmeyen genelleştirilmiş örnekleri kullanan en yakın komşu benzeri bir algoritmadır [170]. Bu algoritma için genelleştirme için deneme sayısı ve ortak bilgi için dizin sayısı 5 olarak belirlenmiştir.

Tablo 6.1. Monk1 veritabanından keşfedilen kurallar

S	Keşfedilen kural atası	TD	A
0	Eğer (attribute#2 = 2)	0.98	0.83
0	Eğer (attribute#5 = 2)	0.98	0.83
0	Eğer (attribute#5 = 3)	0.98	0.83
1	Eğer (attribute#5 = 1)	0.98	0.83

Tablo 6.2. Monk1 veritabanından elde edilen sonuçların karşılaştırılması

	Çok Amaçlı PSO	Ridor	PART	OneR	Prism	NNge
Kural Sayısı	4	6	7	4	25	10
Ortalama Anlaşılabilirlik	0.830	0.583	0.738	0.830	0.500	0.00

Bu tablodan, önerilen algoritmanın az sayıda ve anlaşılabilir kuralları keşfettiği görülmektedir. Fazla sayıda, anlaşılması zor ve uzun kuralları keşfetmemiştir. Sadece OneR algoritması hariç diğerlerinden daha iyi performans göstermiştir. NNge algoritması tüm nitelikleri içeren kurallar keşfetmiştir. Yani kuralların hepsinde altı nitelik yer almaktadır. Ayrıca bu algoritma nitelik değer çifti kısmında değer olarak tek değil birden fazla değer içeren bir küme bulundurduğundan anlaşılması ve işletilebilmesi güç kurallar keşfetmiştir. Keşfettiği bir kural şudur:

"Eğer (attribute#1 $\in$ {1,2,3}) $\wedge$ (attribute#2 $\in$ {2,3}) $\wedge$ (attribute#3 $\in$ {1,2}) $\wedge$ (attribute#4 $\in$ {1,2,3}) $\wedge$ (attribute#5 $\in$ {1}) $\wedge$ (attribute#6 $\in$ {1,2}) $\Rightarrow$ Sınıf = 1 (Kapsanan kayıt = 22)"

Monk1 test verisi üzerinde kullanılan bu algoritmanın ve diğer algoritmaların elde ettiği sonuçlar, karşılaştırmalı olarak Tablo 6.3'te verilmiştir. Bu tabloda, algoritma isimleri ve hemen altında ilgili sınıf için doğru ya da yanlış sınıflandırılan kayıt sayıları verilmiştir. Bunlara göre toplam sayı ve yüzde oranları da bu değerlerin hemen altında verilmiştir. Örneğin, Ridor algoritması, 13'ü birinci sınıfa (0), 18'i de ikinci sınıfa (1) ait 31 tane doğru (toplam kayıtların %72.093'ü); 5'i birinci sınıfa, 7'si de ikinci sınıfa ait 12 tane yanlış (toplam kayıtların %27.907'si) sınıflandırma yapmıştır. Bu tablodan, kullanılan bu yeni algoritmanın, doğruluk açısından Ridor, OneR, Prism ve NNge algoritmalarına göre daha iyi PART

algoritması ile aynı derecede performans gösterdiği görülmektedir. Prism algoritması 4 kaydı sınıflandıramamıştır.

Tablo 6.3. Monk1 test veritabanından elde edilen sonuçların karşılaştırılması

	Çok Amaçlı PSO		**Ridor**		**PART**		**OneR**		**Prism**		**NNge**	
	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış
Sınıf = 0	16	2	13	5	16	2	18	0	14	0	13	5
Sınıf = 1	23	2	18	7	23	2	15	10	14	11	23	2
Toplam	39	4	31	12	39	4	33	10	28	11	36	7
Yüzde (%)	90.6977	9.3023	72.093	27.907	90.6977	9.3023	76.7442	23.2558	65.1163	25.5814	83.7209	16.2791

Daha büyük veritabanı üzerinde, Mushroom, yapılan uygulamalardan elde edilen sonuçlar Tablo 6.4'tedir. Bu tabloda, tüm veri üzerinde, önerilen algoritma ile elde edilen kural listesi, kuralların tahmini doğrulukları ve anlaşılabilirlik değerleri görülmektedir. Bu veritabanında diğer algoritmaların elde ettiği sonuçlar karşılaştırmalı olarak Tablo 6.5'te gösterilmiştir. Bu tablodan, önerilen algoritmanın az sayıda ve anlaşılabilir kuralları keşfettiği görülmektedir. Fazla sayıda, anlaşılması zor ve uzun kural keşfetmemiştir. Virgülden üç rakam sonrasına yuvarlama yapılmıştır. Bu tablodan da, yine NNge algoritmasının anlaşılması ve işletilebilmesi güç olan kuralları keşfettiği görülmektedir. Keşfettiği bir kural aşağıdadır:

"Eğer cap-shape $\in$ {f,k,x} $\wedge$ cap-surface $\in$ {f,s,y} $\wedge$ cap-color $\in$ {b,e,g,n,p,w,y} $\wedge$ bruises? $\in$ {f,t} $\wedge$ odor $\in$ {c,f,p,s,y} $\wedge$ gill-attachment $\in$ {f} $\wedge$ gill-spacing $\in$ {c,w} $\wedge$ gill-size $\in$ {b,n} $\wedge$ gill-color $\in$ {b,g,h,k,n,p,u,w} $\wedge$ stalk-shape $\in$ {e,t} $\wedge$ stalk-root $\in$ {b,e,?} $\wedge$ stalk-surface-above-ring $\in$ {f,k,s} $\wedge$ stalk-surface-below-ring $\in$ {f,k,s} $\wedge$ stalk-color-above-ring $\in$ {b,n,p,w} $\wedge$ stalk-color-below-ring $\in$ {b,n,p,w} $\wedge$ veil-type $\in$ {p} $\wedge$ veil-color $\in$ {w} $\wedge$ ring-number $\in$ {o} $\wedge$ ring-type $\in$ {e,l,p} $\wedge$ spore-print-color $\in$ {h,k,n,w} $\wedge$ population $\in$ {s,v,y} $\wedge$ habitat $\in$ {d,g,l,p,u} $\Rightarrow$ Sınıf = p (Kapsanan kayıt = 3760)"

Mushroom test verisi üzerinde kullanılan bu algoritmanın ve diğer algoritmaların elde ettiği sonuçlar, karşılaştırmalı olarak Tablo 6.6'da verilmiştir. Bu tablodan kullanılan bu yeni algoritmanın doğruluk ve anlaşılabilirlik açısından OneR algoritmasına göre daha iyi, diğerleri ile aynı derecede performans gösterdiği görülmektedir. Bu karşılaştırmada orijinal Prism algoritmasının sonuçları verilememiştir, çünkü bu algoritma kayıp veriler olduğu durumda

çalışamamaktadır (stalk-root niteliğinde 2480 değer kayıptır). Diğer algoritmalar kayıp değerler için analitik olmayan ortalama olarak tanımlanabilen mod değerini, yani en fazla tekrar eden değeri kullanır.

Tablo 6.4. Mushroom veritabanından keşfedilen kurallar

S	Keşfedilen kural atası	TD	A
p	Eğer (odor = f)	100	0.95
p	Eğer (odor = c)	100	0.95
p	Eğer (odor = s)	100	0.95
p	Eğer (odor = y)	100	0.95
p	Eğer (gill-size = b)	100	0.95
e	Eğer (gill-size = b) $\wedge$ (ring-number = o)	100	0.91
e	Eğer (bruises? = f)	99	0.95

Tablo 6.5. Mushroom veritabanından elde edilen sonuçların karşılaştırılması

	Çok Amaçlı PSO	Ridor	PART	OneR	NNge
Kural Sayısı	7	11	12	9	7
Ortalama Anlaşılabilirlik	0.944	0.904	0.921	0.950	0.00

Tablo 6.6. Mushroom test veritabanından elde edilen sonuçların karşılaştırılması

	Çok Amaçlı PSO		RIDOR		PART		OneR		NNge	
	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış
Sınıf = e	1410	0	1410	0	1410	0	1410	0	1410	0
Sınıf = p	1353	0	1353	0	1353	0	1315	38	1353	0
Toplam	2763	0	2763	0	2763	0	2725	38	2763	0
Yüzde (%)	100	0	100	0	100	0	98.6247	1.3753	100	0

"Zoo" ve "Nursery" veritabanları ise önerilen çok amaçlı PSO algoritması, yakın zamanda sınıflandırma kurallarının madenciliği için çok amaçlı GA'yı kullanan çalışma ile [147] karşılaştırmak amacıyla seçilmiştir. Zoo veritabanı 101 kayıt ve 18 nitelik içermektedir. Bu veritabanında sınıf niteliği "type"'dır ve yedi farklı değer bulunmaktadır. Genelleme

yeteneği olmayan hayvan isimlerini içeren nitelik, önişlem olarak veritabanından çıkarılmıştır. Nursery veritabanı ise, 12960 kayıt ve 9 nitelikten oluşmaktadır. Bu veritabanında sınıf niteliği "recommendation"'dır ve beş farklı değeri vardır. Bunlar "Not_recom" (NC), "Recommend" (R), "Very_recom" (VR), "Priority" (P) ve "Spec_prior" (SP)'dir [168].

Önerilen bu yöntem ile elde edilen sonuçlar [147]'de GA tarafından elde edilen sonuçlarla karşılaştırılmıştır. [147]'de üç GA önerilmiştir ancak INPGA adlı en iyi sonucu veren algoritma, karşılaştırma için seçilmiştir. Her veritabanı eğitim kümesi (kayıtların 1/3'i) ve test kümesi (kayıtların 2/3'si) şeklinde iki kısma ayrılmıştır. Çok amaçlı PSO her sınıf için ayrı olarak çalıştırılmış ve bu değiştirilmemiş tahmin edilen sınıf için ilgili kural kümesi elde edilmiştir.

Tablo 6.7 ve Tablo 6.8'de sırasıyla Zoo ve Nursery veritabanlarından PSO ile elde edilen sonuçlar görülmektedir. S "sınıfı", TD "tahmini doğruluğu" ve A "anlaşılabilirliği" göstermektedir.

Tablo 6.7. Zoo veritabanından keşfedilen kurallar

S	Keşfedilen kural atası	TD	A
1	Eğer (eggs = 0) ∧ (venomous = 0) ∧ (domestic = 0)	0.91	0.81
1	Eğer (milk = 1)	0.90	0.94
1	Eğer (hair = 1) ∧ (eggs = 0) ∧ (venomous = 0) ∧ (domestic = 0)	0.95	0.75
2	Eğer (feathers = 1) ∧ (toothed = 0)	0.97	0.88
2	Eğer (hair = 0) ∧ (feathers = 1) ∧ (venomous = 0) ∧ (legs = 2) ∧ (domestic = 0)	1.00	0.69
3	Eğer (eggs = 1) ∧ (predator = 1) ∧ (toothed = 1) ∧ (catsize = 0)	0.99	0.75
3	Eğer (eggs = 1) ∧ (predator = 1) ∧ (catsize = 0)	0.98	0.81
4	Eğer (aquatic = 1) ∧ (breathes = 0) ∧ (tail = 1)	0.80	0.81
5	Eğer (airborne = 0) ∧ (aquatic = 1) ∧ (toothed = 1) ∧ (catsize = 0)	1.00	0.75
6	Eğer (airborne = 1) ∧ (fins = 0) ∧ (tail = 0)	0.83	0.81
7	Eğer (predator = 1) ∧ (breathes = 0) ∧ (domestic = 0)	0.87	0.81
7	Eğer (predator = 1) ∧ (breathes = 0) ∧ (tail = 0) ∧ (domestic = 0)	0.88	0.75

Tablo 6.9'da yöntemlerin ortalama performansları görülmektedir. Bu tablodan, düzenlenen PSO'nun daha yeni bir teknik olmasına ve bu alanda ilk kez uygulanıyor olmasına rağmen kullanılan veritabanlarında veri madenciliğinin sınıflandırma kural madenciliği alanında iyi performansa sahip olduğu görülmektedir.

Tablo 6.8. Nursery veritabanından keşfedilen kurallar

S	Keşfedilen kural atası	TD	A
NR	Eğer (parents = pretentious) ∧ (children = 3) ∧ (housing =convenient) ∧ (health = not_recom)	0.76	0.50
NR	Eğer (parents = usual) ∧ (form = foster) ∧ (housing = less_conv) ∧ (finance = inconv) ∧ (health = not_recom)	0.95	0.38
R	Eğer (has_nurs = proper) ∧ (finance = convenient)	0.75	0.75
R	Eğer (has_nurs = proper) ∧ (finance = convenient) ∧ (health = recommended)	0.81	0.63
VR	Eğer (housing = less_conv) ∧ (finance = inconv) ∧ (social = slightly_prob)	0.88	0.63
VR	Eğer (housing = less_conv) ∧ (finance = inconv) ∧ (social = slightly_prob) ∧ (health = recommended)	0.90	0.50
P	Eğer (parents = great_pret) ∧ (social = slightly_prob) ∧ (health = recommended)	0.81	0.63
SP	Eğer (parents = usual) ∧ (has_nurs = very_crit) ∧ (form = more)	0.79	0.63

Tablo 6.9. Ortalama performanslar

Veritabanı / Yöntem	Zoo		Nursery	
	TD	A	TD	A
PSO	0.92	0.80	0.83	0.58
INPGA	0.91	0.84	0.77	0.63

Her iki veritabanında, iki yöntem tarafından keşfedilen kuralların kalitesini değerlendirmek için 10 katlı çapraz geçerlilik işlemi çalıştırılmıştır. Tablo 6.10 ve Tablo 6.11'de sırasıyla Zoo ve Nursery veritabanlarında elde edilen sonuçlar verilmiştir. Tahmini doğruluğa bağlı olarak tüm sonuçlar bütün olarak dikkate alındığında, PSO'nun bir dereceye kadar INPGA'dan daha iyi bir performansa sahip olduğu görülmektedir.

Tablo 6.10. Zoo veritabanında tahmini doğruluk (%)

Sınıf	PSO	INPGA
1	100 ± 0.0	100 ± 0.0
2	100 ± 0.0	100 ± 0.0
3	95 ± 13.8	0.0 ± 0.0
4	100 ± 0.0	100 ± 0.0
5	100 ± 0.0	100 ± 0.0
6	90 ± 10.0	90 ± 10
7	83.9 ± 10.2	85.5 ± 0.0

Tablo 6.11. Nursery veritabanında tahmini doğruluk (%)

Sınıf	PSO	INPGA
NR	41.8 ± 14.4	12.8 ± 9.8
R	0.0 ± 0.0	0.0 ± 0.0
VR	100 ± 0.0	100 ± 0.0
P	0.0 ± 0.0	0.0 ± 0.0
SP	100 ± 0.0	100 ± 0.0

6.6. Sonuçlar

Veri madenciliği süreci sonunda keşfedilen kuralların yüksek doğruluğa ve güce sahip olması, ilginç, kolay okunabilir ve anlaşılabilir olması istenmektedir. Yani, genellikle göz önüne alınan amaçlar, diğer bir deyimle değerlendirme ölçütleri birbiri ile çelişkili ve negatif yönde etkileşimli olabilmektedir. Literatürde de daha çok tek amaç, yani doğruluk göz önüne alınmakta ve bazen diğer amaçlar göz ardı edilebilmektedir. Farklı amaçların da ortaya çıkması durumunda, veri madenciliği aslında çok amaçlı bir optimizasyon problemi olarak karakterize edilebilir. Bu çalışmada da; basit ancak esnek, sağlam ve global aramada etkili olan, ayrıca kaba seçim algoritmalarına göre nitelik etkileşimleriyle iyi baş edebilen PSO algoritması çok amaçlı olarak tasarlanmış ve sınıflandırma kurallarının etkili keşfi yapılmıştır. Önerilen yöntem, üzerinde fazla bir optimizasyon yapılmadığı halde, tahmini doğrulukları yüksek, anlaşılabilir kurallar keşfetmiştir.

Farklı amaçlar da, örneğin ilginçlik, düşünülüp fonksiyon halinde ifade edilerek amaç sayısı arttırılmış çalışmalar yapılabilir. Ayrıca Pareto kümedeki baskınlık kavramına bulanık mantık eklenerek, amaçların direkt değerleri yerine bulanıklaştırılmış amaç değerleri

kullanılarak bulanık Pareto kümeler kullanılabilir ve özellikle belirsizlik durumunda etkili sonuçlar alınabilir. Algoritmanın paralel ve dağıtık versiyonları, farklı amaçlar ve optimize edilebilecek parametreler kullanılarak daha ayrıntılı deneyler yapılabilir ve veri madenciliğinin birliktelik kural madenciliği, ardışık örüntü keşfi ve kümeleme kural madenciliği gibi alanlarında etkili şekilde kullanılabilir.

7. ÇOK AMAÇLI KABA KAOTİK PARÇACIK SÜRÜ OPTİMİZASYONU

7.1. Giriş

Altıncı bölümde PSO'ya çok amaçlı problemleri çözebilmesi için çeşitli eklenti ve düzenlemeler yapılmıştı. İlk uygulama alanı olarak da karar değişkenleri sürekli değer içermeyen veritabanlarında sınıflandırma kurallarının çok amaçlı madenciliği olarak seçilmişti. Tezin bu bölümünde ise, karar değişkenleri sürekli değerli olan problemler için PSO çok amaçlı olarak tasarlanmış ve ilk etkili uygulaması nicel birliktelik kurallarının keşfinde yapılmıştır. Seçilen uygulama alanı için, aralıkları da aynı anda dikkate alıp direkt kuralları otomatik olarak üreten etkili algoritma sayısı sadece iki tanedir [6, 8]. Problemi çok amaçlı bir optimizasyon problemi olarak ele alan ve çözüm sunan çalışma ise bulunmamaktadır. Nicel birliktelik kurallarının keşfi aslında zor bir problemdir ve kendi içerisinde birçok amacın optimize edilmesini içerir. Bu amaçlardan bazıları şöyle sıralanabilir:

- Güven değerinin maksimizasyonu
- Destek değerinin maksimizasyonu
- Kuralın anlaşılabilir olmasının sağlanması
- Kuralın yeterince ilginç olmasının sağlanması
- Niteliklerin aralıklarının mümkün olacak şekilde minimizasyonu

Dikkat edilirse, genellikle göz önüne alınan amaçlar, diğer bir deyimle değerlendirme ölçütleri birbiri ile çelişkili ve negatif yönde etkileşimlidir. Sonuçta elde edilecek kural listesidir ve bu kuralların oluşturulmasında Pareto baskınlık yaklaşımı kullanılmıştır. Bu yaklaşım içinde bir formüle göre (ağırlıklı formül) tüm amaçları ağırlıklandırma yaklaşımı da kullanılmıştır. Tezin bu bölümünde ayrıca üçüncü bölümde önerilen kaotik haritalı PSO, dördüncü bölümde önerilen kaba PSO ve altıncı bölümde önerilen çok amaçlı PSO birleştirilmiş ve çok amaçlı genel algoritmalar önerilmiştir. Önerilen algoritmalar Çok Amaçlı Kaba Kaotik PSO (ÇAKKPSO) algoritmaları olarak adlandırılmıştır. Bu algoritmaların, karar değişkenleri içerisinde sürekli değerleri de içeren problemler için etkili, çok amaçlı bir yöntem olarak kullanılması hedeflenmiştir.

7.2. Çok Amaçlı Kaba Kaotik PSO Algoritmaları

Bu bölümde önerilen yöntemler tezin üçüncü bölümünde önerilen kaotik haritalı PSO, dördüncü bölümünde önerilen kaba PSO ve altıncı bölümünde önerilen çok amaçlı PSO'nun birleştirilmiş halidir ve sürekli karar değişkeni de içeren problemler için genel çok amaçlı optimizasyon problemi olarak etkili olarak kullanılabilir. Kaotik haritalı PSO algoritmaları PSO'nun parametrelerinin belirlenmesinde rasgele tabanlı bir seçim söz konusu olduğunda farklı kaotik sistemlerin kullanılmasını içermektedir ve bu amaçla on iki farklı PSO kullanılabilir. Kaba PSO alt ve üst sınırdan oluşan kaba değerlere bağlı olarak PSO'nun genişletilmesidir ve aralıkların birer karar değişkeni olarak kullanılması durumunda PSO'ya eklentileri içermektedir. Çok amaçlı PSO'da tek algoritmanın tek çalışmasında çözümler kümesinin üretilmesi ile uğraşılır. Uygulaması yapılan kural madenciliği alanında da, tüm amaçlar eş zamanlı düşünüldüğünde, geniş anlamda herhangi bir kuralın diğerinden üstün olmadığı en uygun yüksek kaliteli kuralların PSO'nun tek çalıştırılmasında bulunması hedeflenir.

Uygulama olarak nicel birliktelik kurallarının çok amaçlı keşfi seçildiğinden parçacık temsili, dördüncü bölümde önerilen kaba PSO'nun uygulamasında kullanılan temsilin aynısıdır. Çok amaçlı yaklaşım da altıncı bölümde anlatıldığı şekildedir. Alt sınır ve üst sınır aşılması durumunda düzeltme işlemi uygulanır. Diğer eklenti ve farklılıklar aşağıdaki alt bölümlerde açıklanmıştır.

7.2.1. Amaçlar

Keşfedilen kurallar yüksek *destek* ve *güven* değerlerine sahip olmalıdır. Ancak bunlar aynı önem sahip değillerdir. Algoritmaların tüm nicel veritabanlarında çalışabilmesi için amaçlara farklı önem verilmesi için ağırlıklandırma yönteminin kullanılması gerekmektedir. Bu yüzden güven ağırlığının desteğin ağırlığından küçük seçilmesi mantıklıdır çünkü bazı gürültülü veritabanlarında kuralın ata ve sonuç kısmında desteği sadece 1 olan ve böylece güven değeri %100 olan kurallar keşfedilebilir. Bu durumda da bu kural diğerleri tarafından baskın olmayacaktır. Algoritmanın çalışacağı veritabanı ile ilgili önceden çok gürültülü olduğu bilgisi varsa, güven değeri çok küçük seçilebilir ya da amaçlardan çıkarılabilir.

Başka bir amaç ise *anlaşılabilirlik*tir. Bu amacın kullanılmasındaki gaye, veri madenciliğinde önemli olan okunabilirlik ve anlaşılabilirliğin arttırılmaya çalışılmasıdır. Uzun kurallar gereksiz ve önemsiz bilgi içerebilmekte ve bu da kuralın başarısını ve etkili şekilde işlenebilmesini engelleyebilmektedir. Anlaşılabilirlik kuralın hem atasında hem de sonuç

kısmındaki nitelik sayısına bağlı olarak ölçülmüştür. Bu amaçla, (7.1)'deki gibi formülize edilen Ghosh ve Nath'ın anlaşılabilirlik ifadesi kullanılmıştır [171].

$$Anlaşılabilirlik = \frac{\log\left(1+\left|Sonuç\right|\right)}{\log\left(1+\left|Ata \cup Sonuç\right|\right)} \tag{7.1}$$

Burada $\left|Sonuç\right|$ ve $\left|Ata \cup Sonuç\right|$ sırasıyla kuralın sonuç kısmında ve toplam kuralda bulunan nitelik sayısını göstermektedir.

Kuralların keşfi sırasında amaçlanan, nesne kümesi ve kurala uyan aralıkların genliğini azaltmaktır. Bu şekilde aynı sayıda kaydı kapsayan ve aynı sayıda nitelik içeren bir iki kuraldan aralıkları küçük olan, en iyi bilgiyi vermektedir. Aralıkların genliği, denklem (7.2)'deki gibi hesaplanır.

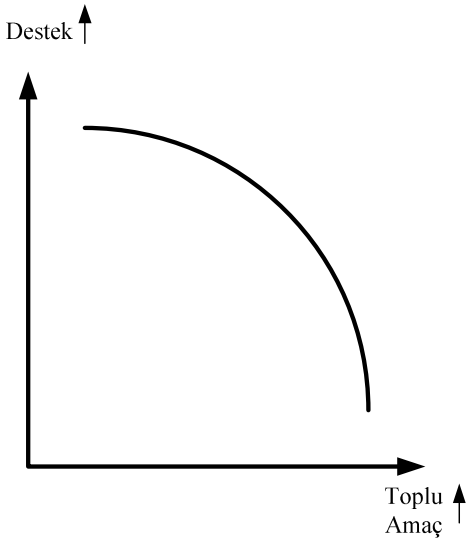
$$Aralıkların\ genliği = \frac{1}{m} \times \sum_{i=1}^{m} \frac{a_i - \ddot{u}_i}{\max(A_i) - \min(A_i)} \tag{7.2}$$

Burada, m nesne kümesindeki nesne sayısı, a_i ve $\ddot{u}_i$ nesne kümesinde kodlanmış i niteliğiyle ilgili olarak alt ve üst sınırı göstermektedir. $\max(A_i)$ ve $\min(A_i)$, i niteliğiyle ilgili olarak aralıkların izin verilen limitleridir. Bu amacın minimize edilmek istendiği unutulmamalıdır.

Bu durumda; destek, güven ve anlaşılabilirlik maksimizasyon amaçları iken aralıkların genliği minimizasyon amacıdır. Anlaşılması açısından tüm amaçların maksimize edilmesi düşünülebilir ve bu minimizasyon amaçları 1'den çıkarılarak maksimizasyon amacı haline getirilebilir. Tüm amaçlar [0, 1] aralığındadır. Ancak burada dikkat edilmesi gereken bazı hususlar vardır. Özellikle anlaşılabilirlik ve aralıkların genliği amaçları, Pareto baskınlık tanımında düşük destek ve düşük güven değerlerine sahip olan kuralların keşfedilmesini olanak verebilir. Yani aralık amacı çok düşük olup güven ve destek değeri de çok düşük olan kuralların keşfedilmesi ya da sadece iki nitelikten oluşan (bir kuralda en az iki nitelik bulunmaktadır) ancak yine aynı şekilde güven ve destek değeri de çok düşük olan kuralların keşfedilmesi olasıdır. Bu durumu ortadan kaldırmak için anlaşılabilirlik ve aralıkların genliği amaçları, güven amacı ile birleştirilmiş ve toplu amaç olarak düşünülmüştür. Yani bu durumda amaçlar, destek ve bu anlaşılabilirlik, aralıkların genliği ve güven amaçlarını içeren toplu amaçtan oluşur. Kısaca dört amaç iki amaç haline getirilmiştir. Toplu amaç denklem (7.3)'te gösterildiği gibi formülize edilmektedir.

$$Toplu\ amaç = 0.8 \times Güven + 0.1 \times Anlaşılabilirlik - 0.1 \times Aralıkların\ genliği \qquad (7.3)$$

Bu durumda istenen, Şekil 7.1'de gösterildiği gibi yüksek destek değerli ve yüksek toplu amaç değerli kuralların keşfidir. Amaçların değerlendirilmesinde virgülden iki rakam sonrası dikkate alınmamıştır, yuvarlama yapılmıştır.



Şekil 7.1. Keşfedilecek kuralların özellikleri

7.2.3. Filtreleme ve Sınır Aralıklarının Arıtılması İşlemleri

Başka çözümler tarafından baskın olmayan çözümler HD arttıkça, atık kural oluşması durumunu önlemek için, bu çözümlerin kapsadığı benzer kayıtların sayısı belli bir eşik değerden yüksek çıkması durumunda destek değeri küçük olan atılır. Bu eşik değeri, benzer kayıtların bir oranı olarak seçilebilir. Böylece amaç uzayında birbirine çok yakın noktaların elenerek birbirinden uzak olanların tutulması sağlanır.

Önerilen algoritmanın çalışması sonunda keşfedilen kuralların nitelik sınırlarına bir iyileştirme işlemi uygulanır. Bu iyileştirme, ilgili parçacıkta aralık boyutunun destek değeri orijinal destek değerinden küçük olana kadar aralığın azaltılması işlemini içerir. Bu şekilde daha kaliteli kuralların oluşması sağlanır.

7.2.4. Parametre Kontrolü

Destek için ağırlık 0.7, *toplu amaç* için ise 0.3 seçilmiştir. Diğer parametreler Tablo 7.1'de verilmiştir. Filtreleme için eşik değeri %80 seçilmiştir.

Tablo 7.1. Kullanılan parametre değerleri

Parametreler	Sürü boyutu	İterasyon sayısı	Mutasyon olasılığı	Hızlanma katsayıları	Atalet ağırlığı
Değerler	30	1000	0.5	2	Komşuluk ötesi arama stratejisi kullanıldığında 1.2'den 0.8'e; komşuluk arama stratejisi kullanıldığında ise 0.9'dan 0.4'e doğru azalır.

7.3. Deneysel Sonuçlar

ÇAKKPSO algoritmaları dört nicel nitelik içeren 1000 kayıtlı sentetik bir veritabanında değerlendirilmiştir. Tüm alan değerleri [0, 100] aralığına ayarlanmıştır. Değerler Tablo 7.2'de gösterilen şekilde önceden belirlenen kümelerde gruplanmış biçimde düzenli olarak dağıtılmıştır. Bu dağılım tamamen gelişigüzeldir. Bu kümeler için destek ve güven değerleri sırasıyla %25 ve %100'tür. Bu kümeler dışındaki değerler bu kurallardan daha iyi kural oluşturmayacak şekilde dağıtılmıştır. Amaç, Pareto optimal kümeleri bulmaktır. ÇAKKPSO algoritmasının her niteliğin nicel aralığı için en uygun değerlerle birliktelik kurallarını keşif yeteneği test edilmiştir. KHPSO7 yani hız güncelleme denkleminde rasgele sayı dizileri (r_1 ve r_2) için kaotik harita kullanan algoritma ve en iyi sonucun alındığı Zaslavskii kaotik harita kullanılmıştır.

Tablo 7.2. Sentetik olarak oluşturulan kümeler

Kümeler
$A_1 \in [1–10] \wedge A_2 \in [15–30]$
$A_1 \in [15–30] \wedge A_3 \in [60–70]$
$A_2 \in [65–80] \wedge A_4 \in [15–25]$
$A_3 \in [80–90] \wedge A_4 \in [80–95]$

Tablo 7.3'te Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması tarafından bulunan ve birbirine baskın olmayan kurallar görülmektedir. Tablodan görüldüğü gibi sentetik olarak oluşturulan kümelere göre keşfedilen kurallar, yüksek destek ve güven değerine sahiptir; ayrıca anlaşılabilirdir. ÇAKKPSO da veritabanından bağımsızdır, çünkü her veritabanı için belirlenmesi güç olan minimum destek ve güven değerlerine bağlı değildir. Eğer destek ve güven değerleri kullanılsaydı ve destek değeri %25'ten büyük seçilseydi, bu veritabanındaki niteliklerin değerlerine göre hiçbir kural keşfedilemeyecekti. Ancak bu veritabanının bazı doğru ve anlaşılabilir kurallar içerdiği bilinmektedir. ÇAKKPSO algoritması tek çalıştırmada, minimum destek ve güven eşik değerlerini kullanmadan, kuralları otomatik

olarak keşfetme yeteneğine sahiptir. Ayrıca kuralın ata ve sonuç kısmının önceden ayarlanmasıyla oluşturulmuş kural şablonunun belirlenmesine ihtiyaç duymaz.

Tablo 7.3. ÇAKPSOA tarafından bulunan kurallar

Kural	Destek(%)	Güven(%)	Kayıtlar(%)
$A_1 \in [1–10] \Rightarrow A_2 \in [15–30]$	25	100	100
$A_1 \in [15–30] \Rightarrow A_3 \in [60–70]$	25	100	
$A_3 \in [80–90] \Rightarrow A_4 \in [80–93]$	25	100	
$A_2 \in [65–79] \Rightarrow A_4 \in [15–25]$	25	100	
$A_2 \in [15–30] \Rightarrow A_1 \in [1–10]$	25	100	
$A_3 \in [60–70] \Rightarrow A_1 \in [15–30]$	25	100	
$A_4 \in [80–93] \Rightarrow A_3 \in [80–90]$	25	100	
$A_4 \in [15–25] \Rightarrow A_2 \in [65–79]$	25	100	

Etkinliliği test etmek için Zaslavskii kaotik haritalı ÇAKKPSO algoritması, gürültülü sentetik veritabanında da çalıştırılmıştır. Gürültü, kümenin ikinci nesnesinin aralığına ait olmayan değerler yerleştirmek suretiyle oluşturulmuştur. Bu yüzden, kayıtların belli bir yüzdesi ikinci nesnenin önceden belirlenen aralığında yer almaz. Örneğin, ilk küme için kayıtların belli bir yüzdesi, ikinci nesne $A_2 \in [15–30]$ aralığında değildir, [0–14] ya da [31–100] aralığında dağıtılmıştır.

Algoritmanın, kuralların ata ya da sonuç kısmı için en uygun aralıkları bulup bulamayacağı test edilmiştir. Bu test üç seviyeli gürültü ile yerine getirilmiştir (*gürültü_seviyesi*'nin 4%, 6% ve 8% değerleri için). Deneysel sonuçlar Tablo 7.4'te gösterilmiştir. Bu tabloda, keşfedilen kurallar, destek ve güven değerleri ve toplam kayıtta keşfedilen kural tarafından kapsanan kayıt yüzdeleri verilmiştir. Aralıkların sınırlarının sentetik olarak üretilenlere hemen hemen uyduğu görülebilmektedir. Bu, ÇAKKPSO algoritmasının test edilen veri içinde belirli yüzdelerdeki gürültünün üstesinden geldiğini göstermektedir. Ancak burada keşfedilen kuralların sadece destek ve güven değerlerine bakıp Pareto baskınlık ilişkisi düşünülmemelidir. Destek ve toplu amaç, bu ilişkide düşünülmelidir.

ÇAKKPSO algoritmaları aynı zamanda 6 halka açık gerçek veritabanında da değerlendirilmiştir: Basketball, Bodyfat, Bolts, Pollution, Quake, ve Sleep. Bu veritabanları Bilkent Üniversitesi Fonksiyon Yaklaştırma Deposunda bulunmaktadır [144]. KPSOA'nın bir karakteristiği de onun stokastik oluşudur. Böylece algoritma farklı çalıştırılmalarda dalgalanmalara sahip olabilir. En iyi sonucun alınabilmesi için algoritma birkaç kez

çalıştırılabilir. Algoritma on kez çalıştırılmış ve bu çalıştırmaların ortaklama değerleri sunulmuştur.

Tablo 7.4. Farklı seviyelerdeki gürültü sonrası keşfedilen kurallar

***gürültü_seviyesi* = 4%**			
Keşfedilen kurallar	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [1–11] \Rightarrow A_2 \in [15–29]$	23.9	98.8	95.4
$A_1 \in [15–30] \Rightarrow A_3 \in [60–69]$	23.7	98.9	
$A_2 \in [64–78] \Rightarrow A_4 \in [15–25]$	23.7	98.9	
$A_2 \in [15–29] \Rightarrow A_1 \in [1–11]$	23.9	98.8	
$A_4 \in [80–93] \Rightarrow A_3 \in [80–91]$	23.7	98.9	
$A_4 \in [14–23] \Rightarrow A_2 \in [64–79]$	23.7	98.9	
***gürültü_seviyesi* = 6%**			
Keşfedilen kurallar	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [1–10] \Rightarrow A_2 \in [15–32]$	23.0	97.1	93.1
$A_1 \in [15–30] \Rightarrow A_3 \in [58–69]$	23.0	97.1	
$A_2 \in [64–78] \Rightarrow A_4 \in [14–26]$	22.9	97.5	
$A_2 \in [14–29] \Rightarrow A_1 \in [1–12]$	23.0	97.1	
$A_4 \in [80–94] \Rightarrow A_3 \in [80–92]$	22.9	97.5	
$A_4 \in [14–23] \Rightarrow A_2 \in [62–79]$	22.9	97.5	
***gürültü_seviyesi* = 8%**			
Keşfedilen kurallar	**Destek(%)**	**Güven(%)**	**Kayıtlar(%)**
$A_1 \in [2–12] \Rightarrow A_2 \in [15–32]$	22.5	95.4	91.8
$A_1 \in [15–30] \Rightarrow A_3 \in [58–70]$	22.5	95.4	
$A_2 \in [64–78] \Rightarrow A_4 \in [14–27]$	22.4	95.9	
$A_2 \in [13–29] \Rightarrow A_1 \in [1–12]$	22.4	95.9	
$A_4 \in [79–94] \Rightarrow A_3 \in [80–92]$	22.5	95.4	
$A_4 \in [14–24] \Rightarrow A_2 \in [62–79]$	22.4	95.9	

Dördüncü bölümde de açıklanan diğer birliktelik kural keşfi yapan algoritmalarla bu bölümde önerilen ÇAKKPSO algoritmalarını karşılaştırmak zor ve adaletsiz olabilir. Çoğu madencilik algoritması niteliklerin aralıklarının belirlenmesini istemekte ve önişlem gerektirmektedir. Bu yüzden anlamlı olarak karşılaştırma, literatürde birliktelik kurallarının

arama sırasında nicel nitelikleri ayrıklaştıran ve evrimsel hesaplama tabanlı iki algoritmayla yapılabilir.

Tablo 7.5'te Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması ve [6]'da sunulan algoritma tarafından bulunan yüksek kaliteli kural sayısı ve bu kuralların güven değerleri (standart sapmalarla birlikte) ile birlikte veritabanındaki kayıt sayısı ve nicel nitelik sayısı görülmektedir. Deneysel sonuçlarda kural ve güven değerleri kullanılmıştır çünkü [6]'daki algoritma da yoğun nesne kümelerini bulmadan direkt olarak kuralları keşfetmektedir. Ayrıca o algoritma da evrimsel hesaplama tabanlı (bir GA) bir algoritmadır ve aralıkları ve kuralları eş zamanlı olarak bulur. [6]'daki algoritma için popülasyon sayısı 100 seçilmiş ve sadece pozitif birliktelik kurallarını keşfedecek şekilde uyarlanmıştır. Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO güven değerleri bakımından bazen daha kaliteli kural bulmakla birlikte genel olarak bu algoritmayla rekabet edebilir görülmektedir.

Tablo 7.5. [6]'da önerilen algoritmayla karşılaştırmalar

Veritabanı	**Kayıt sayısı**	**Nitelik sayısı**	**Kural sayısı**		**Güven(%)**	
			[6]	**ÇAKKPSO**	**[6]**	**ÇAKKPSO**
Basketball	96	5	33.8	15.0	60 ∓ 1.2	60 ∓ 2.4
Bodyfat	252	18	44.2	14.9	59 ∓ 3.8	61 ∓ 1.4
Bolts	40	8	39.0	14.9	65 ∓ 1.9	63 ∓ 2.2
Pollution	60	16	41.2	14.5	68 ∓ 4.8	67 ∓ 4.5
Quake	2178	4	43.8	15.0	62 ∓ 5.1	63 ∓ 2.8
Sleep	62	8	32.8	14.5	64 ∓ 2.3	64 ∓ 2.7

Tablo 7.6'da ise, Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması, GAR [138] ve [6]'da önerilen algoritmalardan elde edilen sonuçlar karşılaştırılmaktadır. GAR algoritması sadece yoğun nesne kümelerini bulmak için bir evrimsel algoritma kullanmaktadır. Bu yüzden, kurallara bağlı değerler ile ilgili karşılaştırmalar yapılmamıştır. "Destek(%)" sütun değerleri ortalama desteği, "Boyut" sütun değerleri kuralda bulunan ortalama nitelik sayısını göstermektedir. "Genlik(%)" sütun değerleri ise kümeye bağlı aralıkların ortalama boyutunu göstermektedir.

Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması, altı veritabanından üçünde yüksek destek değerli kurallar (nesne kümelerine bağlı) bulmuştur ve fark yine fazla değildir. Tüm veritabanları için Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması tarafından elde edilen boyut değerleri GAR algoritmasının bulduklarından daha küçüktür ve altı veritabanından ikisinde [6]'da önerilen algoritmadan elde edilenden daha

küçüktür. Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması tarafından elde edilen genlik değerleri GAR tarafından elde edilen değerlerden daha küçüktür ya da o değerlere eşittir. Altı veritabanından dördünde de, [6]'da önerilen algoritmadan elde edilen genlik değerlerinden daha küçük değerlere sahiptir.

Tablo 7.6. Destek, boyut ve genlik değerlerine göre sonuçların karşılaştırılması

Veritabanı	Destek(%)			Boyut			Genlik(%)		
	ÇAKKPSO	GAR	[6]	ÇAKKPSO	GAR	[6]	ÇAKKPSO	GAR	[6]
Basketball	36.40	36.69	32.21	3.21	3.38	3.21	19	25	20
Bodyfat	65.24	65.26	63.29	6.94	7.45	7.06	25	29	27
Bolts	28.39	25.97	27.04	5.14	5.29	5.14	19	34	27
Pollution	43.90	46.55	38.95	6.46	7.32	6.21	15	15	14
Quake	38.78	38.65	36.96	2.22	2.33	2.10	16	25	19
Sleep	36.68	35.91	37.25	4.19	4.21	4.19	5	5	4

Gerçek verilerdeki son deneysel sonuçlar, Tablo7.7'dedir ve burada da keşfedilen kurallar tarafından kapsanan kuralların yüzdesi verilmiştir. Bu sonuçlardan da Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritmasının diğer evrimsel hesaplama tabanlı algoritmalarla rekabet edebileceği görülebilmektedir.

Tablo 7.7. Keşfedilen kurallar tarafından kapsanan kayıtların yüzdesi

Veritabanı	Kayıtlar(%)		
	KPSOA	GAR	[6]
Basketball	100.00	100.00	100.00
Bodyfat	86.10	86.00	84.12
Bolts	79.78	77.50	77.5
Pollution	95.04	95.00	95.0
Quake	87.90	87.50	87.6
Sleep	80.82	79.03	79.81

Sonuç olarak Zaslavskii kaotik haritalı KHPSO7 kullanan ÇAKKPSO algoritması kuralları, aralıkları çok fazla seçmeden ve yüksek güven ve destek değerlerine sahip olacak şekilde keşfetmiştir. Ayrıca kurallardaki nitelik sayısı da azdır. Böylece veritabanlarında keşfedilen kurallar, doğru, okunabilir ve anlaşılabilirdir. İstenen amaçlar doğrultusunda kuralları keşfettiği görülmüştür. Amaçlara ilginçlik vb. kriterler de eklenebilir ancak bu çalışmada amaçların dışında tutulmuştur. Bazen nesnel bazen de öznel ölçüler kullanan ilginçlik kriteri çok yüksek oranda kullanıcıya bağımlıdır.

7.4. Sonuçlar

Tezin bu bölümünde kaos, kaba kümeler, çok amaçlı optimizasyon fikirleri PSO ile birleştirilmiş ve genel amaçlı "çok amaçlı kaba kaotik PSO" (ÇAKKPSO) algoritmaları önerilmiştir. Kaba örüntü fikrine bağlı olarak kaba parçacık ve kaba karar değişkenleri kullanan KPSOA, PSO'da parametrelerin belirlenmesinde rasgele sayıya her ihtiyaç duyulduğunda kaotik sayı üreteci kullanan KHPSO ve çok amaçlı PSO algoritmalarının birleştirilmesiyle etkili çözümler verebilecek bu algoritmanın ilk uygulaması da nicel birliktelik kurallarının keşfi alanında yapılmıştır.

Nicel birliktelik kurallarının keşfi, çok amaçlı bir optimizasyon problemi olarak karakterize edilmiştir. Pareto tabanlı PSO ile, tek çalıştırmada nicel niteliklerin aralıklarının optimizasyonu ve doğru, güçlü ve anlaşılabilir kuralların keşfi eş zamanlı ve herhangi bir ön işlem ve uzman bilgisi gerektirmeden, bilgi kayıpları olmadan otomatik olarak yapılmıştır. Tasarlanan PSO, ayrıca her veritabanı için belirlenmesi güç olan minimum destek ve minimum güven değerlerine ihtiyaç duymadan, veritabanından bağımsız bir yaklaşım sunar. Genelde kullanılanın aksine yüksek kaliteli birliktelik kurallarını yoğun nesne kümeleri üretmeden direkt olarak keşfeder.

Aynı zamanda, bu şekilde kaba hesaplama alanına da ilave bir konu eklenmiştir. Üçüncü bölümde ortaya çıkarılan kaosun optimizasyonda istenen bir süreç olabilmesi fikri bu bölümde kullanılmış ve en etkili kaotik haritalı PSO ve ilgili kaotik harita kullanılmış ve tatmin edici sonuçlar alınmıştır. ÇAKKPSO pratik uygulamalar için kullanışlı genişletmeler sunmaktadır. Farklı arama ve optimizasyon problemlerinde ve özellikle paralel ve dağıtık versiyonlarıyla daha ayrıntılı deneyler ileriki çalışmalar olarak düşünülebilir.

8. SONUÇLAR

Bu tezde çok noktalı, sosyal ve biyolojik tabanlı stokastik bir yöntem olan ve kuş ve balık sürülerinin hareketlerinden esinlenerek doğrusal olmayan nümerik problemlere optimum sonuçlar bulmak için önerilmiş yumuşak hesaplama tekniklerinden Parçacık Sürü Optimizasyonu (PSO) algoritmasının performansının arttırılması amacıyla çeşitli eklenti ve düzenlemeler yapılmıştır. Özellikle yerel optimum noktalara takılıp kalmayı engelleyerek global yakınsama hızını arttırmak amaçlanmıştır. Bu amaçla, yine yumuşak hesaplama tekniklerinden biri sayılan kaos PSO ile birleştirilmiştir. PSO'nun parametrelerinin belirlenmesinde rasgele tabanlı bir seçim söz konusu olduğunda, örneğin başlangıç sürüsü, atalet ağırlığı, hızlanma katsayıları ve hız ve pozisyon güncelleme denklemlerindeki rassal sayılar, farklı kaotik sistemler rasgele sayı dizilerinin yerine kullanılmış ve on iki farklı PSO algoritması önerilmiştir. Önerilen bu algoritmalara kaotik haritalı PSO algoritmaları adı verilmiş ve bu algoritmalarda sekiz farklı kaotik haritanın etkisi incelenmiştir. Önerilen bu algoritmaların literatürde iyi sonuç verdiği belirlenmiş diğer PSO algoritmalarıyla performanslarının karşılaştırılması sonucu önerilen yöntemlerin çoğunun bu yöntemlerle eşdeğer düzeyde ya da daha iyi performansa sahip olduğu görülmüştür. PSO ve kompleks dinamik gibi farklı alanlarda gelişen sonuçların birleştirilmesinin bazı optimizasyon problemlerinde kaliteyi arttırabileceği ve kaosun istenen bir süreç olabileceği belirtilmiştir.

Ayrıca aralıkların temsil olarak kullanılması gereken durumlarda PSO ile birlikte yine yumuşak hesaplama tekniklerinden sayılan kaba kümelerin alt dalı olan aralık cebrinin etkili olarak kullanılabileceği gösterilmiş, bununla ilgili PSO hesaplamaları açıklanmıştır. Bu amaçla alt ve üst sınırdan oluşan kaba değerlere bağlı olarak PSO'nun genişletilmesi sağlanmış ve bu temsil ve ilgili güncellemeleri içeren algoritma kaba PSO algoritması adıyla tanıtılmıştır. Kaba PSO algoritmasının özellikle veri madenciliğinde sürekli değerli verilerde kural keşfi için yeni, etkili ve otomatik yöntem olarak kullanılabileceği ilk uygulama olarak gösterilmiştir. Nicel birliktelik kurallarının keşfi ve nicel niteliklerin aralıklarının otomatik bölünmesi işini eş zamanlı ve doğru olarak yapan algoritmalar literatürde pek fazla yoktur ve bu amaçla bu problem bir kaba PSO tarafından çözülebilecek bir optimizasyon problemi olarak ele alınmış ve bilgi kayıpları ve ön işlemlerden kaçınılarak kural keşfi yapılmıştır. Veritabanlarında kaba PSO ile keşfedilen kuralların, doğru, okunabilir ve anlaşılabilir olduğu gösterilmiştir. Aynı zamanda kaba hesaplama alanına yeni bir çalışma konusu eklenmiştir.

Kaba değerlerle temsil ve hesaplamaların yapılması zorunluluğu olan yerlerde kullanılan kaba PSO algoritmasına kaotik haritalar eklenmiş ve böylece PSO, kaos ve kaba kümelerin üçünün birleşimiyle performansı arttırmak amacıyla genel amaçlı kaba kaotik PSO

algoritmaları önerilmiştir. Bu algoritmalar, etkin çözüm yöntemi bulunmayan sürekli değerli verilerde nicel birliktelik kurallarının otomatik keşfi alanında uygulanmış ve doğruluk, okunabilirlik ve anlaşılabilirlik açısından başarılı sonuçlar elde edilmiştir. Önerilen kaba kaotik PSO algoritması çok farklı alanlarda uygulama alanı bulabilir.

PSO algoritmasının birden fazla amacın eşzamanlı optimizasyonunu söz konusu olduğu çok amaçlı optimizasyon problemleri için de çalışabilmesi için algoritmaya çeşitli düzenlemeler ve eklentiler yapılmıştır. Sınıflandırma kurallarının madenciliği problemi aynı birimle ölçülebilen ve çoğu zaman çelişen değişik amaçlarla çok amaçlı bir optimizasyon problemi olarak tasarlanmış ve önerilen çok amaçlı PSO algoritması ile belirlenen amaçlar doğrultusunda kuralların keşfi yapılmıştır. Algoritma ilk kez uygulandığı halde, yani üzerinde fazla bir optimizasyon yapılmadığı halde, tahmini doğrulukları yüksek, anlaşılabilir kurallar keşfetmiştir.

Ayrıca çok amaçlı PSO ile kaos ve kaba kümelerin birleşimiyle yeni PSO algoritmaları, çok amaçlı kaba kaotik PSO algoritmaları, önerilmiş ve bunlar da yine etkin çözümler bulmak amacıyla veri madenciliği alanında uygulanmıştır. Literatürde nicel birliktelik kurallarının keşfi problemini çok amaçlı bir optimizasyon problemi olarak ele alan ve çözümler sunan herhangi bir çalışma bulunmamaktadır. Oysaki nicel birliktelik kurallarının keşfi zor bir problemdir ve kendi içerisinde birçok amacın eşzamanlı optimize edilmesini içerir. Bu amaçlardan tezde kullanılanlar şunlardır:

- Güven değerinin maksimizasyonu
- Destek değerinin maksimizasyonu
- Kuralın anlaşılabilir olmasının sağlanması
- Niteliklerin aralıklarının mümkün olacak şekilde minimizasyonu

Dikkat edilirse, genellikle göz önüne alınan amaçlar, diğer bir deyimle değerlendirme ölçütleri birbiri ile çelişkili ve negatif yönde etkileşimlidir. Sonuçta elde edilecek kural listesidir ve bu kuralların oluşturulmasında Pareto baskınlık yaklaşımı kullanılmıştır. Bu yaklaşım içinde ağırlıklı formüle göre tüm amaçları ağırlıklandırma yaklaşımı da kullanılmıştır. Önerilen algoritmalarla kurallar; veritabanından bağımsız, yoğun nesne kümeleri üretilmeden ve aranan amaçlar doğrultusunda, başarılı şekilde keşfedilmiştir.

8.1. Öneriler

Bu tezde önerilen algoritmalar genel amaçlıdır ve veri madenciliğinin diğer alt dallarında ve özellikle kümeleme kurallarının madenciliği ve ardışık örüntü keşfinde kullanılabilir. Ayrıca diğer optimizasyon ve arama tabanlı mühendislik problemlerinde de

kolayca kullanılabilir. Önerilen bu yöntemler henüz yenidirler. Bunların dağıtık ve paralel versiyonlarıyla optimize edilmiş parametreler kullanılarak ayrıntılı deney ve uygulamalar yapılabilir.

Ayrıca PSO algoritması belli iterasyon boyunca herhangi bir ilerleme ya da belli bir oranda iyileşme sağlayamadığı zaman, bu tıkanmanın olduğu noktada lokal arama olarak kaotik haritalar belli iterasyon boyunca devreye girebilir ve böylece algoritmanın lokal optimum noktalara takılıp kalması önlenebilir. Farklı yerel arama yöntemleriyle de birleştirilip performans karşılaştırmaları yapılabilir. Burada önerilen algoritma ve teknikler, farklı yapı ve türdeki PSO'larla birleştirilip performans testleri yapılabilir.

Seçilen uygulama alanları, nicel birliktelik kurallarının ve sınıflandırma kurallarının madenciliği, için farklı yumuşak hesaplama yöntemleri kullanılıp performans karşılaştırmaları yapılabilir. Ayrıca bu tezde veri madenciliği için önerilen yöntemler, farklı evrimsel hesaplama alanındaki yöntemlere de uyarlanabilir. Çok amaçlı PSO' da, farklı amaçlar, örneğin ilginçlik, düşünülüp fonksiyon halinde ifade edilerek amaç sayısı arttırılmış çalışmalar yapılabilir. Ayrıca Pareto kümedeki baskınlık kavramına bulanık mantık eklenerek, amaçların direkt değerleri yerine bulanıklaştırılmış amaç değerleri kullanılarak bulanık Pareto kümeler kullanılabilir ve özellikle belirsizlik durumunda iyi sonuçlar alınabilir.

KAYNAKLAR

1. Murty, K. G., 2003, Optimization Models For Decision Making: Volume 1, Internet Edition, Chapter 1: Models for Decision Making, 1-18.
2. 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, Cilt: 16 Sayı: 1, 2-15.
3. Simha, R., Carnahan, J., 2001, Nature's Algorithms: Natural and Social Metaphors in Algorithm Design, IEEE Potentials, Vol. 20, No. 2, 21-24.
4. Holland, J. H., 1975, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.
5. Goldberg, D. E, 1989, Genetic Algorithms in Search, Optimization and Machine Learning, Kluwer Academic Publishers, Boston, MA.
6. Alataş, B., Akın, E., 2006, An Efficient Genetic Algorithm for Automated Mining of Both Positive and Negative Quantitative Association Rules, Soft Computing, Springer Verlag, 10(3), 230-237.
7. Storn, R., Price, K., 1995, Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, ICSI.
8. Alataş, B., Akın, E., Karcı, A., 2007, MODENAR: Multi-Objective Differential Evolution Algorithm for Mining Numeric Association Rules, Applied Soft Computing, doi:10.1016/j.asoc.2007.05.003.
9. Dorigo, M., Maziezzo, V., Colorni, A., 1996, The Ant System: Optimization by a Colony of Cooperating Ants. IEEE Trans. on Systems, Man and Cybernetics B,26(1), 29-41.
10. Alataş, B., Akın E., 2005, FCACO: Fuzzy Classification Rules Mining Algorithm with Ant Colony Optimization, ICNC 2005, Lecture Notes in Computer Science, Springer-Verlag, 3612, 787 – 797.
11. Hu, S., Liao, X., Mao, X., 2003, Stochastic Hopfield Neural Networks**,** J. Phys. A: Math. Gen. 36 2235-2249, doi:10.1088/0305-4470/36/9/303.
12. Karaboga, D., Basturk, B., 2007, A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization.
13. de Castro, L.N., Von Zuben, F.J**.,** 2002, Learning and Optimization Using the Clonal Selection Principle, IEEE Transaction on Evolutionary Computation, 6:3, 239-251.

14. Alataş, B., Akın E., 2005, Mining Fuzzy Classification Rules Using an Artificial Immune System with Boosting, ADBIS 2005, Lecture Notes in Computer Science, Springer-Verlag, 3631, 283 – 293.
15. Cerny, V., 1985, Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, J. Opt. Theory Appl., 45, 1, 41-51.
16. Birbil, S. I., Fang, SC., 2003, An Electromagnetism-like Mechanism for Global Optimization, Journal of Global Optimization, Volume 25, Number 3, 263–282.
17. Glover, F., Laguna, M., 1997, Tabu Search. Kluwer, Norwell, MA
18. Reynolds, R. G., 1994, An Introduction to Cultural Algorithms. In A. V. Sebald and L. J. Fogel, editors, Proceedings of the Third Annual Conference on Evolutionary Programming, 131-139. World Scientific, River Edge, New Jersey.
19. Kennedy, J., Eberhart, R. C., 1995, Particle Swarm Optimization, Proc. of IEEE International Conference on Neural Networks, volume 4, 1942-1948, Australia.
20. Rechenberg, I., 1971, Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (Doktora Tezi). Fromman-Holzboog tarafından tekrar basıldı, 1973.
21. Lozano, J.A., Larrañga, P., Inza, I., Bengoetxea, E., (Eds.), 2006, Towards a New Evolutionary Computation, Advances in estimation of distribution algorithms. Springer.
22. Harik, G. R., Lobo, F. G., Goldberg, D. E., 1999, The compact genetic algorithm. IEEE Trans. on Evol. Comp., 3(4):287—297.
23. Baluja, S., 1994, Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report CMU-CS-94-163, Comp. Sci. Dep., Carnegie Mellon U.
24. Muehlenbein, H., Paass, G., 1996, From Recombination of Genes to Estimation of Distributions, Proc. PPSN-IV, Springer.
25. Larra˜naga, P., Lozano, JA, Bengoetxea, E., 2001, Estimation of Distribution Algorithms Based on Multivariate Normal and Gaussian Networks. Technical report, Dept. of Computer Science and Artificial Intelligence, University of the Basque Country, KZAA-IK- 1-0
26. Moscato, P., 1989, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Caltech Concurrent Computation Program, C3P Report 826.
27. Reynolds, R.G., 2004, Peng, B, Cultural Algorithms: Modeling of How Cultures Learn to Solve Problems, IEEE ICTAI 2004, 166-172.

28. Basturk, B., Karaboga, D., 2006, An Artificial Bee Colony (ABC) Algorithm for Numeric Function Optimization, IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA.
29. Pham, D.T., Koç, E., Ghanbarzadeh, A., 2006, Otri S., Rahim S., Zaidi M., The Bees Algorithm – A Novel Tool for Complex Optimisation Problems, IPROMS 2006 Proceeding 2nd International Virtual Conference on Intelligent Production Machines and Systems, Oxford, Elsevier.
30. Yang, X.S., 2005, Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms, IWINAC 2005, LNCS 3562, Yang, J. M. and J.R. Alvarez (Eds.), Springer-Verlag, Berlin Heidelberg, 317-323.
31. Lucic, P., Teodorovic, D., 2003, Computing with Bees: Attacking Complex Transportation Engineering Problems, International Journal on Artificial Intelligence Tools, 12(3), 375-394.
32. Teodorovic, D., Dell'Orco, M., 2005, Bee Colony Optimization - A Cooperative Learning Approach to Complex Transportation Problems, Advanced OR and AI Methods in Transportation, 51-60.
33. Nakrani, S., Tovey, C., 2003, On Honey Bees and Dynamic Allocation in an Internet Server Colony, Proceedings of 2nd International Workshop on the Mathematics and Algorithms of Social Insects, Atlanta, Georgia, USA.
34. Wedde, H.F., Farooq, M., Zhang, Y., 2004, BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior, Ant Colony, Optimization and Swarm Intelligence, Eds. M. Dorigo, Lecture Notes in Computer Science 3172, Springer Berlin, 83-94.
35. Bianco, G.M., 2004, Getting Inspired from Bees to Perform Large Scale Visual Precise Navigation, Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 619-624.
36. Chong, C.S., Low, M.Y.H., 2006, Sivakumar A.I., Gay K.L., A Bee Colony Optimization Algorithm to Job Shop Scheduling, Proceedings of the 37th Winter Simulation, Monterey, California, 1954-1961.
37. Drias, H., Sadeg, S., Yahi, S., 2005, Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem, IWAAN International Work Conference on Artificial and Natural Neural Networks, Barcelona, Spain, 318-325.
38. Quijano, N., Passino, K.M., 2007, Honey Bee Social Foraging Algorithms for Resource Allocation Theory and Application, American Control Conference, New York City, USA.

39. Abbass H.A., 2001, MBO: Marriage in Honey Bees Optimization A Haplometrosis Polygynous Swarming Approach, CEC2001 Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, 207-214.
40. Bozorg Haddad, O., Afshar A., Mariano M.A., 2006, Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization, Water Resources Management, 20, 661-680.
41. Koudil, M., Benatchba, K., Tarabet, A., Sahraoui, E.B., 2007, Using Artificial Bees to Solve Partitioning and Scheduling Problems in Codesign, Applied Mathematics and Computation, 186(2), 1710-1722
42. Benatchba, K., Admane, L., Koudil, M., 2005, Using Bees to Solve a Data Mining Problem Expressed as a Max-Sat One, In Proceedings of IWINAC'2005, International Work Conference on the Interplay between Natural and Artificial Computation, Canary Islands, Spain, 212- 220.
43. De Castro, L.N., Timmis, J., 2002, Artificial Immune Systems: A New Computational Intelligence Systems, Springer, 357 sayfa.
44. De Castro, L. N., Von Zuben, F. J., 2000, The Clonal Selection Algorithm with Engineering Applications, GECCO'2000, 36-37.
45. Alataş, B., Akın, E., 2004, Yapay Zekada Yeni Bir Alan: Yapay Bağışıklık Sistemleri, YA-EM'2004 Kongresi, 464-466, Adana.
46. Kennedy, J., Eberhart, R. C., 1995, Particle Swarm Optimization, Proc. of IEEE International Conference on Neural Networks, volume 4, 1942-1948, Australia.
47. Birbil, S. I., Fang, S., Sheu, R. 2004, On the Convergence of a Population-Based Global Optimization Algorithm. J. of Global Optimization 30, 2–3.
48. Wu, P., Yang, K. J., Fang, H. C., 2006, A Revised EM-Like Algorithm + K-OPT Method for Solving the Traveling Salesman Problem, ICICIC '06, 546- 549.
49. Tušar, T., Korošec, P., Papa, G., Filipič, B., Šilc, J., 2007, A Comparative Study of Stochastic Optimization Methods in Electric Motor Design, Applied Intelligence, 27: 101-111.
50. Alataş, B., Akın, E., Özer, A. B., (Kabul edildi), Chaos Embedded Particle Swarm Optimization Algorithms, Chaos, Solitons & Fractals, Elsevier, DOI: 10.1016/j.chaos.2007.09.063.
51. Alataş, B., Akın, E., Özer, A. B, 2007, Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları, 12. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi, Eskişehir.

52. Alataş, B., Akın, E., 2005, Rough Immune Algorithm, International Symposium on Innovations in Intelligent Systems and Applications (INISTA-2005), 75-78, İstanbul, Turkey.
53. Alataş, B., Akın, E., 2005 Rough Differential Evolution Algorithm, Second International Conference on Electronics and Computer Engineering IKECCO 2005, Bishkek/KYRGYZSTAN, 175-180.
54. Alataş, B., Akın, E., 2007, Chaotic Rough Particle Swarm Optimization Algorithms, Swarm Intelligence: : Focus on Ant and Particle Swarm Optimization (Editörler: Professor Felix T. S. Chan ve Professor Manoj Kumar Tiwari), ARS Publishing
55. Alataş, B., Akın, E., 2007, Modified Particle Swarm Optimization Based Multi-Objective Rule Mining, INISTA 2007, 195-199, Istanbul, Turkey.
56. Alataş, B., Akın, E., Karcı, A., 2006, Sınıflandırma Kurallarının Parçacık Sürü Optimizasyon Algoritmasıyla Keşfi, 62-66, ASYU-INISTA 2006, Istanbul, Haziran.
57. Kennedy, J., Eberhart, R. C., Shi, Y., 2001, Swarm Intelligence, Morgan Kaufmann Publishers, 512 sayfa
58. Sevkli M., 2005, Atölye Tipi Çizelgeleme Problemlerine Parçacık Sürü Optimizasyonu Yaklaşımı ve Genetik Algoritma Modeli ile Karşılaştırılması, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi.
59. Eberhart, R. C., Kennedy, J., 1995, A New Optimizer Using Particle Swarm Theory, Proc. of Sixth International Symposium on Micro Machine and Human Science, 39-43, Japan.
60. van den Berg, F., Engelbrecht, A. P., 2000, Cooperative Learning in Neural Networks using Particle Swarm Optimizers, South African Computer Journal, 26: 84-90.
61. Shi, Y., Eberhart, R. C., 1999, Empirical Study of Particle Swarm Optimization, Proc of IEEE Congress on Evolutionary Computation, 1945-1949, USA.
62. Hassan, R., B. de Weck, C. O., 2004, A Comparison of Particle Swarm Optimization and the Genetic Algorithm.
63. Eberhart, R. C., Simpson, P., Dobbins, R., 1996 Computational Intelligence PC Tools, Chapter 6, 212-226, Academic Pres Professional.
64. Corne, D., Dorigo, M., Glower, F. (editors), 1999 New Ideas in Optimization, Chapter 14, 217-279, McGraw Hill.
65. Zielinski, K., Peters, D., Laur, R., 2005, Stopping Criteria for Single-Objective Optimization Optimization, Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore.
66. Zielinski, K., Peters, D., Laur, R., 2007, Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimization Algorithm, Informatica, 31, 51–59.

67. Kennedy, J., Eberhart, R. C., 1997, A Discrete Binary Version of the Particle Swarm Algorithm, Proc of the Conference on Systems, Man, and Cybernetics, 4104-4109.
68. Eberhart, R. C., Kennedy, J., 2001, Swarm Intelligence, Morgan Kaufmann, 510 sayfa.
69. Kennedy, J., Spears, w. M., 1998, Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator, Proc of the International Conference on Evolutionary Computation, 78-83, Alaska.
70. Shi, Y., Eberhart, R. C., 1998, A Modified Particle Swarm Optimizer, IEEE International Conference on Evolutionary Computation, Alaska.
71. Shi, Y., Eberhart, R. C., 1998, Parameter Selection in Particle Swarm Optimization, Evolutionary Computation 7 Proc of EP 98, 591-600.
72. Otten, R. H. J. M., Ginneken, L. P. P. P., 1989, The Anneling Algorithm, Kluwer, Boston, USA.
73. Zheng, Y. L., Ma, L. H., Zhang, L. Y., Qian, J. X., 2003, On the Convergence Analysis and Parameter Selection in Particle Swarm Optimization, Proc. of the 2003 IEEE International Conference on Machine Learning and Cybernetics. Piscataway, NJ, USA: IEEE Press, pp. 1802-1807.
74. Zheng, Y.L., Ma, L.H., Zhang, L.Y., Qian, J.X., 2003, Empirical Study of Particle Swarm Optimizer with an Increasing Inertia Weight, Proc. of the 2003 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press, 221-226.
75. Zhang, L., Yu, H., Hu, S., 2003, A New Approach to Improve Particle Swarm Optimization, Lecture Notes in Computer Science (LNCS) No. 2723: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003), Chicago, IL, USA. 134-142.
76. Shi, Y., Eberhart, R. C., 2001, Fuzzy Adaptive Particle Swarm Optimization, Proc of the Congress on Evolutionary Computation, Korea.
77. Shi, Y., Eberhart, R. C., 2001, Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight, Proc of the Workshop on Particle Swarm Optimization, USA.
78. Clerc, M., 1999, The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization, Proc of the IEEE Congress on Evolutionary Computation, 1951-1957, USA.
79. Eberhart, R. C., Shi, Y., 2000, Comparing inertia weights and constriction factors in particle swarm optimization, Proc. Congress on Evolutionary Computation 2000, San Diego, CA, pp 84-88.

80. Angeline, P. J., 1999, Using Selection to Improve Particle Swarm Optimization, Proc. of IJCNN'99, USA.
81. Ratnaweera, A., Halgamuge, S. K., 2004, Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients, IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, 240-255
82. Arumugam M. S., Rao M. V. C., 2006, On the Performance of the Particle Swarm Optimization Algorithm with Various Inertia Weight Variants for Computing Optimal Control of a Class of Hybrid Systems, Discrete Dynamics in Nature and Society, 1-17.
83. LØvbjerg, M., Rausmussen, T. K., Krink, T., 2001, Hybrid Particle Swarm Optimizer with Breeding and Subpopulations, Proc of GECCO, USA.
84. van den Bergh, F., 2001, An analysis of Particle Swarm Optimizers, PhD Thesis, University of Pretoria.
85. Suganthan, P. N., 1999, Particle Swarm Optimizer with Neighborhood Operator, Proc of the IEEE Congress on Evolutionary Computation, 1958-1961, USA.
86. Kennedy, J., Medes, R., 2002, Population Structures and Particle Swarm Performance, IEEE Congress on Evolutionary Computation (CEC. 2002), Honolulu, USA.
87. Kennedy, J., 2000, Stereotyping: Improving Particle Swarm Performance with Cluster Analysis, Proc. of the IEEE Congress on Evolutionary Computing, 1507-1512, USA.
88. Higashi, N. Iba, H., 2003, Particle Swarm Optimization with Gaussian Mutation, Proc. of the IEEE Swarm Intelligence Symposium, pages 72—79.
89. Esquivel, S. C., Ando, C. C., Carlos A., 2003, On the Use of Particle Swarm Optimization with Multimodal Functions, Proceedings of the Congress on Evolutionary Computation (CEC'2003), Canberra, Australia, IEEE press.
90. Riget, J. Vesterstrøm, J.S., 2002, A Diversity-Guided Particle Swarm Optimizer the ARPSO, Technical Report 2002-02, Department of Computer Science, University of Aarhus.
91. Xie, X. F, Zhang. W. J., Yang, Z. L., 2002, A Dissipative Particle Swarm Optimization, Congress on Evolutionary Computation, 1456-1461
92. Zhang, W. J., Xie, X. F., 2003, DEPSO: Hybrid Particle Swarm with Differential Evolution Operator, IEEE Int. Conf. on Systems, Man & Cybernetics (SMCC), 3816-3821.
93. Storn, R., Price, K., 1997, Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, J. Global Optimization 11, 341-359.
94. Omran, M. G. H, 2005, Particle Swarm Optimization Methods for Pattern Recognition and Image Processing, University of Pretoria, PhD Thesis.

95. Krink, T., Løvbjerg, M., 2002, The LifeCycle model: Combining Particle Swarm Optimisation, Genetic Algorithms and HillClimbers, Proc. of Parallel Problem Solving from Nature, vol. VII, pp. 621-630.
96. Veeramachaneni, K., Peram, T., Mohan, C., Osadciw, L. A., 2003, Optimization Using Particle Swarm with Near Neighbor Interactions, Genetic and Evolutionary Computation Conference, Chicago, Illinois.
97. Lovbjerg, M., Krink, T., 2002, Extending Particle Swarm Optimizers with Self-Organized Criticality. The 4th IEEE Congress on Evolutionary Computation, Hawaii, 1588-1593.
98. Perm, T. Vesramaohaneni, K., Mohan, C.K., 2003 Fitness- Distance-Ratio Hased Particle Swam Optimization, Proc. IEEE Swarm Intelligence Symposium SlS2003, Indianapolis, Indiana, USA, 174 -181.
99. Belal, M., El-Ghazawi T., 2004, Parallel Models For Partıcle Swarm Optimizers, The International Journal of Intelligent Computing and Information Sciences, Vol. 4, No. 1.
100. Zielinski, K., Laur, R., 2006, Constrained Single-Objective Optimization Using Particle Swarm Optimization, Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, Canada, 443-450.
101. He, S., Wu, Q.H., Wen, J.Y., Saunders, J.R. and Paton, R.C., 2004, A Particle Swarm Optimizer with Passive Congregation, Biosystem, V. 78. 135-147.
102. Caponetto, R., Fortuna, L., Fazzino, S., Xibilia, M.G., 2003, Chaotic Sequences to Improve the Performance of Evolutionary Algorithms, IEEE Trans Evol. Comput., 7(3):289–304.
103. Wong, K., Man, K. P., Li, S., Liao, X., 2005, More Secure Chaotic Cryptographic Scheme based on Dynamic Look-up Table, Circuits, Systems & Signal Processing, vol. 24, no. 5, pp. 571-584.
104. Brännström, Å., 2004, Modelling Animal Populations, Umeå University, PhD Thesis.
105. Arena, P., Caponetto, R., Fortuna, L., Rizzo, A., La Rosa, M., 2000, Self Organization in Non-recurrent Complex System, Int. J. Bifurcation and Chaos, vol. 10, no. 5, 1115–1125.
106. Manganaro, G., de Gyvez, J. P., 1997, DNA Computing Based on Chaos, Proc. 1997 IEEE International Conference on Evolutionary Computation. Piscataway, NJ: IEEE Press, 255–260.
107. Belkhouche, F., Qidwai, U., Gokcen, I., Joachim, D., 2004, Binary Image Transformation Using Two-dimensional Chaotic Maps**,** Pattern Recognition, ICPR 2004, Proc. of the 17th International Conference on Volume 4, Issue , 23-26, 2004, 823 - 826 Vol. 4.
108. Trelea, I.C., 2003, The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection, Inf. Process. Lett. 85 (6), 317–325

109. Schuster, H. G., 1988, Deterministic Chaos: An Introduction. Second Revised Edition, Physick- Verlag GmnH, D-6940 Weinheim, Federal Republic of Germany.
110. May, R.M., 1976, Simple Mathematical Models with Very Complicated Dynamics, Nature 261: 459.
111. Peitgen, H., Jurgens, H., Saupe, D., 1992, Chaos and Fractals. Berlin, Germany: Springer-Verlag.
112. Zheng, W. M., 1994, Kneading Plane of the Circle Map, Chaos, Solitons and Fractals 4, 1221.
113. Arnold, V. I., Avez, A., 1967, Problèmes Ergodiques de la Mécanique Classique, Paris, Gauthier-Villars.
114. Sinai, Y. G., 1972, Russ. Math. Survey, 27, 21.
115. Zaslavskii, G. M., 1978, The Simplest Case of a Strange Attractor, Physic Letters A, vol. 69, 145-147.
116. Griewangk, A. O., 1981, Generalized Descent of Global Optimization. Journal of Optimization Theory and Applications, 34: 11.39.
117. Rastrigin, L. A., 1974, Extremal Control Systems. In Theoretical Foundations of Engineering Cybernetics Series, Moscow, Nauka, Russian.
118. De Jong, K., 1975, An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan.
119. Digalakis, J. G., Margaritis, K. G., 2002, An Experimental Study of Benchmarking Functions for Genetic Algorithms. International Journal Comput. Math. 79(4): 403-416.
120. Lingras, P., 1996, Rough Neural Networks, International Conference on Information Processing and Management of Uncertainty, Granada, Spain, pp. 1445-1450.
121. Lingras, P., Davies, C., 2001, Applications of Rough Genetic Algorithms, Computational Intelligence: An International Journal, 3(17), 435-445.
122. Lingras, P., Davies, C., 1999, Rough Genetic Algorithms, Proceedings of the 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing RSFDGrC 1999, Lecture Notes In Computer Science; Vol. 1711, 38-46.
123. Agrawal, R., Imielinski, T., Swami, A. N., 1993, Mining Association Rules Between Sets of Items in Large Databases, Proc. ACM SIGMOD, Washington, D.C., 207-216.
124. Tong, Q., Yan, B., Zhou, Y., 2005, Mining Quantitative Association Rules on Overlapped Intervals. ADMA 2005: 43-50
125. Srikant, R., Agrawal, R., 1996, Mining Quantitative Association Rules in Large Relational Tables, Proc. ACMSIGMOD, 1-12.

126. Miller, R.J., Yang, Y., 1997, Association Rules over Interval Data, Proc. ACM SIGMOD International Conf. Management of Data, 29, 452–461.
127. Lent, B., Swami, A., Widom J., 1997, Clustering Association Rules, Proc. IEEE International Conf. on Data Eng., 220–231.
128. Vannucci, M., Colla, V., 2004, Meaningful Disretization of Continuous Features for Association Rules Mining by means of a SOM, ESANN2004 European Symposium on Artificial Neural Networks, Belgium, 489-494.
129. Ke, K., Cheng, J. Ng, W., 2006, MIC Framework: An Information-Theoretic Approach to Quantitative Association Rule Mining, ICDE '06, 112-114.
130. David, L. O., Li, Y., 2007, Mining Fuzzy Weighted Association Rules, 40th Annual Hawaii International Conference on System Sciences HICSS-2007, 53-62.
131. Aumann, Y., Lindell, Y., 2003, A Statistical Theory for Quantitative Association Rules, Journal of Intelligent Information Systems, 20:3, 255–283
132. Fukuda, T., Yasuhiko, M., Sinichi, M., Tokuyama, T., 1996, Mining Optimized Association Rules for Numeric Attributes, Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM Pres, 182-191.
133. Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama T., 1996, Data Mining Using Two-dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization, Proc. ACM SIGMOD Int. Conf. Management of Data, ACM Pres, 13–23.
134. Yoda, K., Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama, T., 1997, Computing Optimized Rectilinear Regions for Association Rules. Proc. 3rd International Conference on Knowledge Discovery and Data Mining, AAAI Pres, 96–103.
135. Alatas, B., Arslan A., 2005, A Novel Approach Based on Genetic Algorithm and Fuzzy Logic for Mining of Association Rules (in Turkish), Firat University, Journal of Science and Engineering, 17 (1) , 42-51.
136. Alatas, B., Arslan A., 2004, Mining of Fuzzy Association Rules with Genetic Algorithms (in Turkish), Gazi University, Journal of Polytechnic, 7 (4), 269-276.
137. Kaya, M., Alhajj., R., 2005, Genetic Algorithm Based Framework for Mining Fuzzy Association Rules, Fuzzy Sets and Systems, Vol. 152, Issue 3, 587-601.
138. Mata, J., Alvarez, J. L., Riquelme, J. C., 2002, Discovering Numeric Association Rules via Evolutionary Algorithm, 6-th Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD-02 (LNAI) Taiwan 2336, 40-51.
139. Alatas, B., Akin, E., 2006, An Efficient Genetic Algorithm for Automated Mining of both Positive and Negative Quantitative Association Rules, Soft Computing, Springer-Verlag, 10(3), 230-237

140. Agrawal, R., Imielinski, T., Swami, A., 1993, Database Mining: A Performance Perspective, IEEE Transactions on Knowledge and Data Engineering, v.5 n.6, 914-925.
141. Holte, R.C., 1993, Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, Machine Learning. 11:63-91.
142. Frank, E., Witten, I. H., 1998, Generating Accurate Rule Sets Without Global Optimization, Fifteenth International Conference on Machine Learning, 144-151.
143. Gaines, B. R., Compton, P., 1995, Induction of Ripple-Down Rules Applied to Modeling Large Databases, J. Intell. Inf. Syst.. 5(3):211-228.
144. Guvenir, H. A., 2000, Uysal, I., Bilkent University Function Approximation Repository, http://funapp.cs.bilkent.edu.tr.
145. Coello, C. A. C., 2006, Evolutionary Multiobjective Optimization: A Historical View of the Field, IEEE Computational Intelligence Magazine, Vol. 1, No. 1, 28-36.
146. Kshetrapalapuram, K. K., Kirley, M., 2005, Mining Classification Rules Using Evolutionary Multi-objective Algorithms, Knowledge-Based Intelligent Information and Engineering Systems, Part 3, Proceedings,Springer, 959--965, LNCS Vol. 3683.
147. Dehuri, S., Mall, R., 2006, Predictive and Comprehensible Rule Discovery Using a Multi-objective Genetic Algorithm, Elsevier Knowledge-Based Systems 19, 413–421.
148. Setzkorn, C., 2006, On The Use Of Multi-Objective Evolutionary Algorithms For Classification Rule Induction, Phd Thesis, University of Liverpool, UK, 2006.
149. Pila, A.D., Giusti, R., Prati, R.C., Monard, M.C., 2006, A Multi-Objective Evolutionary Algorithm to Build Knowledge Classification Rules with Specific Properties**,** Sixth International Conference on Hybrid Intelligent Systems (HIS'06).
150. de la Iglesia, B., Philpott, M.S., Bagnall, A.J., 2003, V.J. Rayward-Smith, Data Mining Rules Using Multi-objective Evolutionary Algorithms, IEEE CEC'03, Vol:3, 1552-1559.
151. Meng, H. Y., Zhang, X. H., Liu, S. Y., 2005, Intelligent Multiobjective Particle Swarm Optimization Based on AER Model, EPIA 2005, 178-189.
152. Alvarez-Benitez, J. E., Everson, R. M., Fieldsend, J. E., 2005, A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts, In C. Coello-Coello et al., editors, Evolutionary Multi-Criterion Optimization, volume 3410, 459–73, Springer, 2005.
153. Zhang, Y., Huang, S., 2004, A Novel Multiobjective Particle Swarm Optimization for Buoys-arrangement Design, Intelligent Agent Technology, 24- 30.
154. Liu, Y., Shi, Q. Z. Z., Chen, J., 2004, Rule Discovery with Particle Swarm Optimization, AWCC 2004 LNCS 3309, 291-296.
155. Zhao, X., Zeng, J., Gao, Y., Yang, Y., 2006, Particle Swarm Algorithm for Classification Rules Generation, IEEE Intelligent Systems Design and Applications, 957-962.

156. Sousa, T., Silva, A., Neves, A., 2004, Particle Swarm based Data Mining Algorithms for Classification Tasks, Elsevier Parallel Computing, 30, 767–783.
157. Coello, C. A., 2000, An Updated Survey of GA-based Multi-objective Optimization Techniques, ACM Computing Surveys, 32(2), 109–143.
158. Quinlan, J.R., 1992, C4.5: Programs for Machine Learning, Morgan Kaufmann.
159. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984, Classification and Regression Trees, Wadsworth.
160. Michalski, R.S., 1969, On the Quasi-Minimal Solution of the Covering Problem, Fifth Int. Symp on Information Processing, 125-128, Bled, Yugoslavia, 1969.
161. Clark, P., Niblett, T., 1989, The CN2 Induction Algorithm, Machine Learning Journal, 261-283, The Netherlands, Kluwer, 1989.
162. Alataş, B., Arslan, A., 2004, Mining of Interesting Prediction Rules with Uniform Two-Level Genetic Algorithm, IJCI Vol 1, Iss 4, 276-281.
163. Gündoğan, K.K., Alataş, B., Karcı, A., 2004, Mining Classification Rules by Using Genetic Algorithms with Non-random Initial Population and Uniform Operator, Turkish Journal of Electrical Engineering & Computer Science, Vol 12, Iss 1, 43-52.
164. Alataş, A., Akın, E., 2004, Sınıflandırma Kurallarının Karınca Koloni Algoritmasıyla Keşfi, ELECO2004, 357-361, Bursa
165. Sousa, T.F., Silva, A.P., Silva, A.F., 2004, Particle Swarm Based Data Mining Algorithms for Classification Tasks, Parallel Comput., 30(5-6):767-783
166. Liu, Y., Qin, Z., Shi, Z., Chen, J., 2004, Rule Discovery with Particle Swarm Optimization, AWCC 2004, 291-296.
167. Dung, N.D., 2003, Using Prior Knowledge in Rule Induction, Master Thesis, Japan Advanced Institute of Science and Technology.
168. Newman, D. J., Hettich, S., Blake, C.L., Merz, C. J., 1998, UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
169. Cendrowska, J., 1987, PRISM: An Algorithm for Inducing Modular Rules, International Journal of Man-Machine Studies. 27(4):349-370.
170. Roy, S., 2002, Nearest Neighbor With Generalization. Christchurch, New Zealand.
171. Ghosh, A., Nath, B. T., 2004, Multi-Objective Rule Mining using Genetic Algorithms, Information Sciences, 163, 123-133.

ÖZGEÇMİŞ

05–11–1977 tarihinde Elazığ'da doğdu. 1996 yılında Elazığ Anadolu Lisesi'nden mezun oldu. 1997 yılında Fırat Üniversitesi Bilgisayar Mühendisliği bölümüne birinci olarak girdi ve bu bölümde lisans eğitimini 2001 yılında birincilikle bitirdi. Aynı yıl, aynı bölümün Kuramsal Temeller bilim dalında yüksek lisans eğitimine ve araştırma görevlisi olarak çalışmaya başladı. 2003 yılında bu bilim dalında yüksek lisans eğitimini tamamladı. Aynı yıl Elektrik-Elektronik Mühendisliği bölümünde doktora eğitimine başladı ve halen Fırat Üniversitesi Bilgisayar Mühendisliği bölümünde araştırma görevlisi olarak çalışmaktadır. İlgi alanları veri madenciliği, optimizasyon ve yumuşak hesaplamadır.