T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

DOĞADAN ESİNLENEN VE SES YANKISINA DAYALI ÇÖZÜM ARAMA ALGORİTMASI İLE BELGE KÜMELEME

Sinem BÜYÜKSAATÇI

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

Danışman

Prof. Dr. Alp BARAY

Temmuz,2015

İSTANBUL

Bu çalışma 13/07/2015 tarihinde aşağıdaki jüri tarafından Endüstri Mühendisliği Anabilim Dalı Endüstri Mühendisliği programında Doktora Tezi olarak kabul edilmiştir.

Tez Jürisi:

İmza
Prof. Dr. Alp BARAY (Danışman)
İstanbul Üniversitesi
Mühendislik Fakültesi

İmza
Prof. Dr. Şakir ESNAF
İstanbul Üniversitesi
Mühendislik Fakültesi

İmza
Prof. Dr. Selim ZAİM
İstanbul Teknik Üniversitesi
İşletme Fakültesi

İmza
Prof. Dr. Mehmet Mutlu YENİSEY
İstanbul Üniversitesi
Mühendislik Fakültesi

İmza
Doç. Dr. Hür Bersam BOLAT
İstanbul Teknik Üniversitesi
İşletme Fakültesi

ÖNSÖZ

Bu çalışmamı sunarken;

Doktora öğrenimim boyunca yardım ve desteğini esirgemeyen, paylaştığı bilgi ve tecrübelerle her konuda yol gösterici olan kıymetli tez danışmanım ve hocam Prof. Dr. Alp BARAY'a,

Tez çalışmalarım sırasında yapıcı eleştirileri ile fikir veren değerli hocalarım Prof. Dr. Şakir ESNAF ve Prof. Dr. Mehmet Mutlu YENİSEY'e,

Bitmek tükenmek bilmeyen sorularıma sabırla katlanan, tezimin uygulama aşamasında bana destek veren ve yol gösteren sevgili çalışma arkadaşım ve dostum Yrd. Doç. Dr. Tarık KÜÇÜKDENİZ'e,

Hocalarım Prof. Dr. Alp BARAY, Prof. Dr. Şakir ESNAF ve Prof. Dr. Mehmet Mutlu YENİSEY başta olmak üzere psikolojik olarak beni yükselten tüm diğer hocalarıma ve yardımlarını esirgemeyen tüm diğer çalışma arkadaşlarıma,

Yurtiçi doktora bursu ile bana maddi destek sağlayan TÜBİTAK-Bilim İnsanı Destekleme Daire Başkanlığı'na,

Ayrıca, bugüne kadar benden maddi ve manevi desteklerini hiçbir zaman esirgemeyen, her zaman arkamda olan, varlıklarından güç aldığım aileme çok teşekkür ederim.

Temmuz,2015 Sinem BÜYÜKSAATÇI

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Simgeler	Açıklama
T	: Sıcaklık
$f(x)$	: Amaç fonksiyonu
T_0	: Başlangıç sıcaklığı
T_f	: Son sıcaklık
N_{maks}	: Maximum iterasyon sayısı
x^0	: Başlangıç tahmini değeri
p_c	: Çaprazlama olasılığı
p_m	: Mutasyon olasılığı
γ	: Feromon buharlaşma oranı ya da ışık emme katsayısı
$v_i^{(t)}$	: *i.* parçacığın *t* anındaki hızı
$x_i^{(t)}$	: *i.* parçacığın *t* anındaki konumu
$v_i^{(t)}$	: *i.* parçacığın *t* anındaki hızı
w	: Atalet ağırlığı
c_1	: Bireysel öğrenme katsayısı
c_2	: Küresel öğrenme katsayısı
x_i^{best}	: *i.* parçacığın bilinen en iyi konumu
x_{gbest}	: Sürünün bilinen en iyi konumu
$rand$	: Rastgele sayı
g^*	: Küresel en iyi değer
r_{pa}	: Frekans ayarlama oranı
r_{kabul}	: Armoni hafıza kabul oranı
p	: Arama uzayı boyutu
S	: Bakteri sayısı
S_r	: Nesil başına üreme sırasında ortadan kaldırılan bakteri sayısı
N_c	: Kemotakstik adım sayısı
N_s	: Yüzme uzunluğu sınırı
N_{re}	: Üreme adım sayısı
N_{ed}	: Eliminasyon ve dağılma olayı sayısı
P_{ed}	: Her bir bakterinin elimine/dağılma olasılığı
$J(i,j,k,l)$	: *i.* bakterinin *j.* kemotaktik, *k.* üreme ve *l.* eliminasyon adımındaki uygunluk değeri
I_i	: *i.* ateşböceğinin parlama yoğunluğu
p_a	:Guguk kuşu tarafından yumurtlanan yumurtanın ev sahibi kuş tarafından bulunma olasılığı
q	: Alt-mempleks boyutu
P_x	: Küresel en iyi kurbağa
P_b	: Yerel en iyi kurbağa
P_w	: Küresel en kötü kurbağa
S_{maks}	: Maksimum adım büyüklüğü

$\boldsymbol{K}$: Küme sayısı
$\boldsymbol{N}$: Belge sayısı
$\boldsymbol{M}$: Terim sayısı
$\boldsymbol{C_i}$: *i.* küme
$\boldsymbol{n_i}$: *i* kümesinin boyutu
$\boldsymbol{d_j}$: *j.* belge vektörü
$\boldsymbol{n_{ij}}$: *j* belgesi içindeki *i* teriminin sayısı
$\boldsymbol{w_{ij}}$: j belgesi içindeki i teriminin terim ağırlığı
$\boldsymbol{dist}$: Uzaklık
$\mathbf{Sim}(\boldsymbol{C_j}, \boldsymbol{C_k})$: *j* kümesi ile *k* kümesi arasındaki benzerlik
$\boldsymbol{tf_{ij}}$: j belgesi içindeki i teriminin terim frekansı
$\boldsymbol{idf_i}$: i. terimin ters belge frekansı
$\boldsymbol{df_i}$: i teriminin belge kümesi içindeki sıklığı
$\mathbf{cos}(\boldsymbol{d_j}, \boldsymbol{d_k})$: *j* belgesi ile k belgesi arasındaki kosinüs değeri
$\|\boldsymbol{d_j}\|$: *j* belgesinin uzunluğu (normu)
$\boldsymbol{d_{t,j}}$: *j* belgesindeki t-inci eleman
$\boldsymbol{D_j}$: *j* belgesindeki terimlerin seti
$\mathbf{J}(\boldsymbol{d_j}, \boldsymbol{d_k})$: *j* belgesi ile k belgesi arasındaki Jaccard katsayısı
$\boldsymbol{c_i}$: *i.* kümenin ağırlık merkezi vektörü
$\boldsymbol{c}$: Tüm belge setinin ağırlık merkezi vektörü
$\mathbf{I}$: İçsel ölçüt fonksiyonu
$\boldsymbol{\mathcal{E}}$: Dışsal ölçüt fonksiyonu
$\boldsymbol{H}$: Hibrid ölçüt fonksiyonu
$\boldsymbol{d_j^T}$: *j.* belge vektörünün transpozesi
$\boldsymbol{d}(\boldsymbol{C_r})$: *r.* kümenin küme içi uzaklıkları toplamı
$\boldsymbol{\overline{S_i}}$: *i.* kümeye atanan belgelerin küme merkezine uzaklıklarının ortalaması
$\boldsymbol{SS_B}$: Kümeler arası varyansların toplamı
$\boldsymbol{SS_w}$: Küme içi varyansların toplamı
$\boldsymbol{E_j}$: *j.* belgenin entropisi
$\boldsymbol{p_{ij}}$: j. kümedeki bir belgenin i. kategoriye ait olma olasılığı
$\boldsymbol{F}$: F ölçüsü değeri
$\boldsymbol{v}$: Ses hızı
$\boldsymbol{f}$: Frekans
$\boldsymbol{\lambda}$: Dalga boyu
$\boldsymbol{A_0}$: Ses yüksekliği
$\boldsymbol{r}$: Sinyal emisyon oranı
$\boldsymbol{A^t}$: *t* zamanındaki ortalama ses yüksekliği

Kısaltmalar **Açıklama**

TB : Tavlama Benzetimi
GA : Genetik Algoritmalar
PSO : Parçacık Sürü Optimizasyonu
BBAA : Bakteriyel Besin Arama Algoritması
AHH : Arama Hafızası Havuzu

SBAA : Seçilen Boyutun Arama Aralığı
DBS : Değişen Boyutların Sayısı
KPD : Kendi Pozisyonunu Değerlendirme
UD : Uygunluk Değeri
KO : Karışım Oranı
KSA : Kurbağa Sıçrama Algoritması
TREC : Text Retrieval Conference
TD-IDF : Term Frequency-Inverse Document Frequency (Terim Frekansı-Ters Belge Frekansı)
ICS : Intra Cluster Similarity (Küme İçi Benzerlik)
DBI : Davies-Bouldin İndeksi
CHI : Calinski-Harabasz indeksi
SI : Silhouette indeksi

ÖZET

DOKTORA TEZİ

DOĞADAN ESİNLENEN VE SES YANKISINA DAYALI ÇÖZÜM ARAMA ALGORİTMASI İLE BELGE KÜMELEME

Sinem BÜYÜKSAATÇI

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Endüstri Mühendisliği Anabilim Dalı

Danışman : Prof. Dr. Alp BARAY

1980'li yıllardan günümüze kombinatoryal optimizasyon teorisi alanında büyük gelişmeler katedilmiş ve sezgisel algoritmalar olarak adlandırılan yaklaşık algoritmalar araştırma ve uygulamalarda önemli bir hale gelmiştir. Bununla birlikte, klasik sezgisel algoritmaların karmaşık optimizasyon problemleri üzerinde yeterince etkin olmadığı durumlar için metasezgisel algoritmalar olarak adlandırılan ve yine yaklaşık algoritmalar sınıfında yer alan algoritmalara yönelim başlamıştır.

Bir metasezgisel algoritma biçimsel olarak; arama uzayını keşfetmek ve kuvvetlendirmek için farklı akıllı kavramları birleştirerek ve optimuma yakın çözümler bulmak amacıyla bilgi yapılandırmasında öğrenme stratejileri kullanarak sezgisel algoritmalara rehberlik eden iteratif bir oluşum süreci olarak tanımlanabilmektedir. Metasezgisel algoritmalar "yeterince küçük" işlem zamanında "yeterince iyi" bir çözüm bulmak amacıyla özel olarak geliştirilmiştir.

Metasezgisel algoritmaların temelleri önemli ölçüde değişkenlik göstermektedir. Bazı algoritmalar optimizasyon sürecini, görünüşte optimizasyon ile alakasız olan hayvan sürülerinin davranışları, doğal evrim, fiziksel değişimler vb. gibi yaklaşımlar kullanarak açıklarken, bazıları biyolojik temellere dayanmaktadır. Ancak genel olarak tüm metasezgisel algoritmalar yapılarında rastgelelik bulundurmaktadır.

Bu doktora tezi hayvan davranışından esinlenerek geliştirilmiş ve literatürde yeni sayılabilecek metasezgisellerden biri olan yarasa algoritması ile belge kümeleme uygulaması incelenmiştir. Literatürde sıkça kullanılan belge setlerine farklı ölçüt

fonksiyonları ve farklı benzerlik ölçüleri ile yapılan analizler sonucunda yarasa algoritmasının performansı, kümeleme işlemlerinde çoğunlukla kullanılan k-ortalamalar ve parçacık sürü optimizasyonu algoritması ile karşılaştırılmıştır. Çalışma, literatürde sıkça kullanılan belge setlerini yarasa algoritması ile kümeleyen ilk çalışma olması yönünden önem taşımaktadır.

Temmuz 2015, 126 sayfa

Anahtar kelimeler: Metasezgiseller, Yarasa Algoritması, Belge Kümeleme

SUMMARY

Ph.D. THESIS

DOCUMENT CLUSTERING WITH A NATURE-INSPIRED AND ECHOLOCATION BASED SEARCH ALGORITHM

Sinem BÜYÜKSAATÇI

İstanbul University

Institute of Graduate Studies in Science and Engineering

Department of Industrial Engineering

Supervisor : Prof. Dr. Alp BARAY

From 1980s to present, a great deal of effort has been invested in the field of combinatorial optimization theory in which approximate algorithms, often called heuristic algorithms, have become an important area of research and applications. Besides metaheuristics algorithms that are a class of approximate methods have started to be trend where classical heuristics and optimization methods have failed to be effective and efficient on complex optimization problems.

A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions. Metaheuristics are developed specifically to find a solution that is "good enough" in a computing time that is "small enough".

The underlying foundations of different metaheuristics vary significantly. While some of the metaheuristics model the optimization process by using a metaphor seemingly unrelated to optimization, such as the behavior of animal swarms, natural evolution, physical changes etc., others based on the biological basis. But in general, metaheuristics frameworks rely heavily on the use of randomness.

In this thesis, document-clustering process will be conducted with bat algorithm, which is one of the animal inspired metaheuristic algorithms. As a result of analysis with different criterion functions and different similarity measures on document sets from literature, bat algorithm performance will be compared with k-means and particle swarm optimization algorithms used mostly in the clustering process.

July 2015, 126 pages

Keywords: Metaheuristics, Bat Algorithm, Document Clustering

1. GİRİŞ

Hem teorik hem de pratikte önemli birçok optimizasyon problemi, hedeflere ulaşmak için gerekli değişkenler dizisinin en iyi şekilde seçimi ile ilgilenmektedir. Çözüm değişkenleri dizisi gerçek değerli olabileceği gibi kesikli değerlerle de kodlanabilmektedir. Kesikli değişkenlerin söz konusu olduğu problemler literatürde kombinatoryel optimizasyon (KO) problemleri olarak adlandırılmaktadır.

Kombinatoryel optimizasyon problemlerinin önemi nedeniyle, bu problem tipini çözebilecek pek çok algoritma geliştirilmiştir. Bu algoritmaları tam ve yaklaşık algoritmalar olarak sınıflandırmak mümkündür. Tam algoritmalar, KO problemlerinin her sonlu boyutlu örneği için sınırlı zaman içinde optimum çözüm bulmayı garanti etmektedir. Ancak KO problemleri NP-hard yapıdadır ve bu durum uygulamalarda yüksek hesaplama sürelerine yol açmaktadır. Bu nedenle, KO problemlerini çözmek için yaklaşık algoritmaların kullanılması son 40 yılda daha fazla ilgi görmüştür. Bu algoritmalar, önemli ölçüde kısalmış sürede optimuma yakın sonuçlar bulmakta ve kendi içinde yapıcı algoritmalar ve yerel arama algoritmaları olarak ayrılmaktadırlar. Yapıcı algoritmalar başlangıçta boş kısmi çözüme bileşenler ekleyerek sıfırdan çözüm üretmektedirler. Yerel arama algoritmaları ise bir çözümden başlayıp iteratif olarak geçerli çözüm yerine daha iyi bir çözüm bulmaya çalışmaktadırlar.

1980'li yıllarda, verimli ve etkili bir arama alanı keşfetmek amacıyla temel sezgisel yöntemleri üst düzey yaklaşımlarla birleştirmeye çalışan yeni bir yaklaşık algoritma türü ortaya çıkmıştır. Metasezgisel algoritmalar olarak adlandırılan bu algoritmalar günümüze kadar, hem kesikli hem de sürekli değişkenlerin çözüm olduğu birçok problemin çözümünde kullanılmıştır.

Bir metasezgisel algoritma biçimsel olarak; arama uzayını keşfetmek ve kuvvetlendirmek için farklı akıllı kavramları birleştirerek ve optimuma yakın çözümler bulmak amacıyla bilgi yapılandırmasında öğrenme stratejileri kullanarak sezgisel algoritmalara rehberlik eden iteratif bir oluşum süreci olarak tanımlanabilmektedir.

Metasezgisel algoritmalar genel olarak “yeterince küçük” işlem zamanında “yeterince iyi” bir çözüm bulmak amacıyla özel olarak geliştirilmiştir.

Metasezgisel algoritmaların temelleri önemli ölçüde değişkenlik göstermektedir. Bazı algoritmalar optimizasyon sürecini, görünüşte optimizasyon ile alakasız olan hayvan sürülerinin davranışları, doğal evrim, fiziksel değişimler vb. gibi yaklaşımlar kullanarak açıklarken, bazıları biyolojik temellere dayanmaktadır. Ancak genel olarak tüm metasezgisel algoritmalar yapılarında rastgelelik bulundurmaktadır.

Bu doktora tezi kapsamında yapılacak olan çalışma, hayvan davranışından esinlenerek geliştirilmiş ve literatürde yeni sayılabilecek metasezgisellerden biri olan yarasa algoritması ile belge kümeleme uygulaması üzerinedir. Çalışma, literatürde sıkça kullanılan belge setlerini yarasa algoritması ile kümeleyen ilk çalışma olması yönünden önem taşımaktadır. Ayrıca algoritmanın performansı, belge kümeleme analizinde en çok kullanılan algoritmalar ile karşılaştırılarak test edilmiştir.

Bilgisayar ve iletişim teknolojilerindeki hızlı gelişmeler, birçok sektörde büyük bir teşvik uyandırmış; işlem yönetimi, bilgi alma ve veri analizi uygulamalarında kullanılmak üzere büyük miktarda veri tabanlarının ve bilgi depolarının oluşturulması sağlanmıştır. Böylece internet üzerindeki metin belgelerinin hacmi, dijital kütüphaneler ve depoları, haber kaynakları, şirket çapında intranet, blog makaleler ve e-postalar gibi dijital kişisel belgeler büyük bir büyüme göstermiştir.

Günümüzde hala devam eden elektronik belgelerin sayısındaki artış, bu belgelerin elle verimli bir şekilde organize ve analiz edilmesini zorlaştırmaktadır. Bu zorluklar, metin belgelerinin otomatik olarak etkin ve verimli bir şekilde organize edilmesini, verinin daha anlamlı hale getirilmesini beraberinde getirmiştir.

Geleneksel istatistiki tekniklerle büyük boyuttaki veriyi çözmek mümkün değildir. Bu sebeple, verileri işlemek ve çözümlemek için özel tekniklere ihtiyaç duyulmuş ve veri madenciliği çalışmaları hızla yayılmıştır.

Veri madenciliği, çok büyük veri hacimleri arasında tutulan, potansiyel olarak faydalı ve anlaşılır bilgilerin çıkarıldığı ve arka planda veritabanı yönetim sistemleri, istatistik, yapay zekâ, makine öğrenmesi gibi işlemlerin bulunduğu veri analizi teknikleri

bütünüdür. Verilerden yola çıkarak her alanda daha hızlıca karar alabilmeyi vaat eden veri madenciliği, son dönemin en cazip ve çekici meslek dallarından biri haline gelmiştir.

Veri madenciliği uygulamalarında kullanılan en önemli yöntemlerden biri kümelemedir. Kümeleme, verinin benzer özellikteki gruplara ayrılması işlemidir. "*Küme*" olarak adlandırılan her bir grup, kendi içinde birbirine benzeyen nesnelerden oluşurken diğer kümelerdeki nesnelerden farklılık göstermektedir. "*Belge kümeleme*" de, bilgi çekme dünyasının en önemli konu başlıklarından biridir. Bilimsel veri arama, bilgi alma ve metin madenciliği, mekansal veri tabanı uygulamaları, Web analizi, müşteri yönetimi, pazarlama, tıbbi teşhis, hesaplamalı biyoloji gibi birçok alanda belge kümeleme etkin rol oynamaktadır.

Literatürde belge kümeleme ile ilgili olarak yapılmış çok sayıda çalışma yer almaktadır. Ancak birçok algoritma, yüksek kaliteli kümeleme çözümü elde etmek için gerekli bazı özel özellikleri ve zorlukları tam olarak karşılayamadığından, hala farklı algoritmalarla hızlı ve verimli kümeleme çözümleri denenmektedir. Bu nedenle, bu tez çalışmasında metasezgisel algoritmalardan biri olan yarasa algoritması ile belge kümeleme çalışması ele alınacaktır.

Doktora tezinin ana hatlarına bakılacak olursa; ikinci bölümü genel kısımlar oluşturmaktadır. Bu bölümde, doğadan esinlenerek geliştirilen metasezgisel algoritmalar ve belge kümeleme başlıkları ayrıntılı olarak açıklanacak, yarasa algoritmasının temelini oluşturan yarasaların genel davranışları ve kullandıkları akustik özelliğine yer verilecektir. Bunların dışında, yarasa algoritmasının literatürdeki uygulama alanları yine bu bölümde ortaya konacaktır. Ayrıca, metasezgisel algoritmalar ile belge kümeleme konusunda literatürde yer alan çalışmaların ayrıntılı bir incelemesi de ikinci bölümün kapsamındadır.

Üçüncü bölümde, yöntem olarak seçilen yarasa algoritması detaylı bir şekilde ele alınacak; algoritmanın varsayımlarından, işleyişinden ve standart formülasyonundan bahsedilecektir.

Dördüncü bölüm, belge kümeleme analizlerinin yer aldığı bölümdür. Bu bölümde literatürde yer alan 3 farklı belge seti içinden, belge ve terim sayısı farklı rastgele 6 set

oluşturulacak ve bu 6 set üzerinde MATLAB programı kullanılarak hem K-ortalamalar algoritması, hem yarasa algoritması hem de parçacık sürü optimizasyonu algoritması ile kümeleme işlemi uygulanacaktır. Kümeleme sırasında farklı benzerlik ölçüleri ve ölçüt (amaç) fonksiyonlarından faydalanılıp sonuçlar tablolar halinde karşılaştırılacaktır. Ayrıca, yarasa algoritmasının belge kümeleme sonuçlarının başarısı çeşitli göstergelerle değerlendirilecektir.

Doktora tezinin beşinci bölümü sonuç bölümüdür. Bu bölümde, tez çalışması kapsamında oluşturulan katkılar özetlenerek gelecekteki çalışmalar için öneriler sunulacaktır.

2. GENEL KISIMLAR

Bu bölümde, öncelikle optimizasyon kavramı üzerinde durularak optimizasyon algoritmaları sınıflandırılacak ve doğadan esinlenerek geliştirilen metasezgisel algoritmaların bir kısmı detaylı anlatılacaktır. Tez çalışmasında kullanılacak yöntem olan ve doğadan esinlenerek geliştirilen yarasa algoritmasının genel yapısına ve yarasa algoritması ile yapılmış literatürde yer alan çalışmalara yine bu kısımda yer verilecektir. Daha sonra tezin uygulama konusu olan belge kümeleme ile ilgili olarak; bir belgenin gösterimi, belge kümeleme türleri, belge kümeleme algoritmaları, belge kümelemede benzerlik ölçüleri, kümeleme ölçüt fonksiyonları, kümeleme algoritmalarının değerlendirilmesi gibi çeşitli başlıklar ayrıntılı bir biçimde açıklanacaktır. Bunların dışında, doğadan esinlenerek geliştirilen metasezgisel algoritmalar kullanılarak yapılan belge kümeleme çalışmaları ile ilgili literatür yine bu bölümde sunulacaktır.

2.1. OPTİMİZASYON

Çağlar boyunca insanlar sürekli optimizasyon süreci ile içiçe olmuşlardır. Ancak optimizasyon süreci ile ilgili dev adım ellili yılların başında dijital bilgisayarların gelişiyle atılmıştır. Son yıllarda da, optimizasyon teknikleri hızla gelişmiş ve bu kapsamda önemli ilerlemeler sağlanmıştır. Aynı zamanda dijital bilgisayarın daha hızlı, daha çok yönlü ve daha verimli bir hale gelmesiyle de inatçı olarak düşünülen karmaşık optimizasyon problemlerinin çözümü mümkün bir hal almıştır (Antoniou ve Lu, 2007).

Optimizasyon, verilen koşullar altında belirli bir amacı gerçekleştiren en iyi sonucu bulma çabasıdır. Optimizasyon süreci; model, en iyileyici (optimizer) ve simülatör olmak üzere üç bileşenden oluşmaktadır. Matematiksel ya da sayısal model, fiziksel problemin matematiksel denklemler kullanılarak ifade edilmiş halidir. Bu, herhangi bir modelleme ve optimizasyon için ilk önemli adımdır. İkinci önemli adım, doğru algoritma veya en iyileyici kullanmaktır. Böylece tasarım değişkenlerinin optimal kombinasyon seti bulunabilmektedir. Optimizasyonun önemli bir özelliği, arama işleminin yakınsamasına yol açan bilinen bir sonuçtan yeni sonuçlar üretmek ya da yeni sonuçları aramaktır. Bu arama sürecinin nihai amacı, genellikle zor olsa da, küresel optimuma yakın çözümleri bulmaktır. Optimizasyona işlem zamanı ve maliyet

açısından bakıldığında da en önemli adım, verimli bir değerlendirici ya da simülatörün kullanılmasıdır. Birçok uygulamada, bir kez doğru model temsili yapılmış ve uygulanmışsa da, bir optimizasyon süreci genellikle binlerce ve milyonlarca amaç fonksiyonu yapılandırmalarını içermektedir. Dolayısıyla bu adım, genel işlem süresinin %50'si ile %90'ı içeren en zaman alıcı adımdır (Koziel ve Yang, 2011).

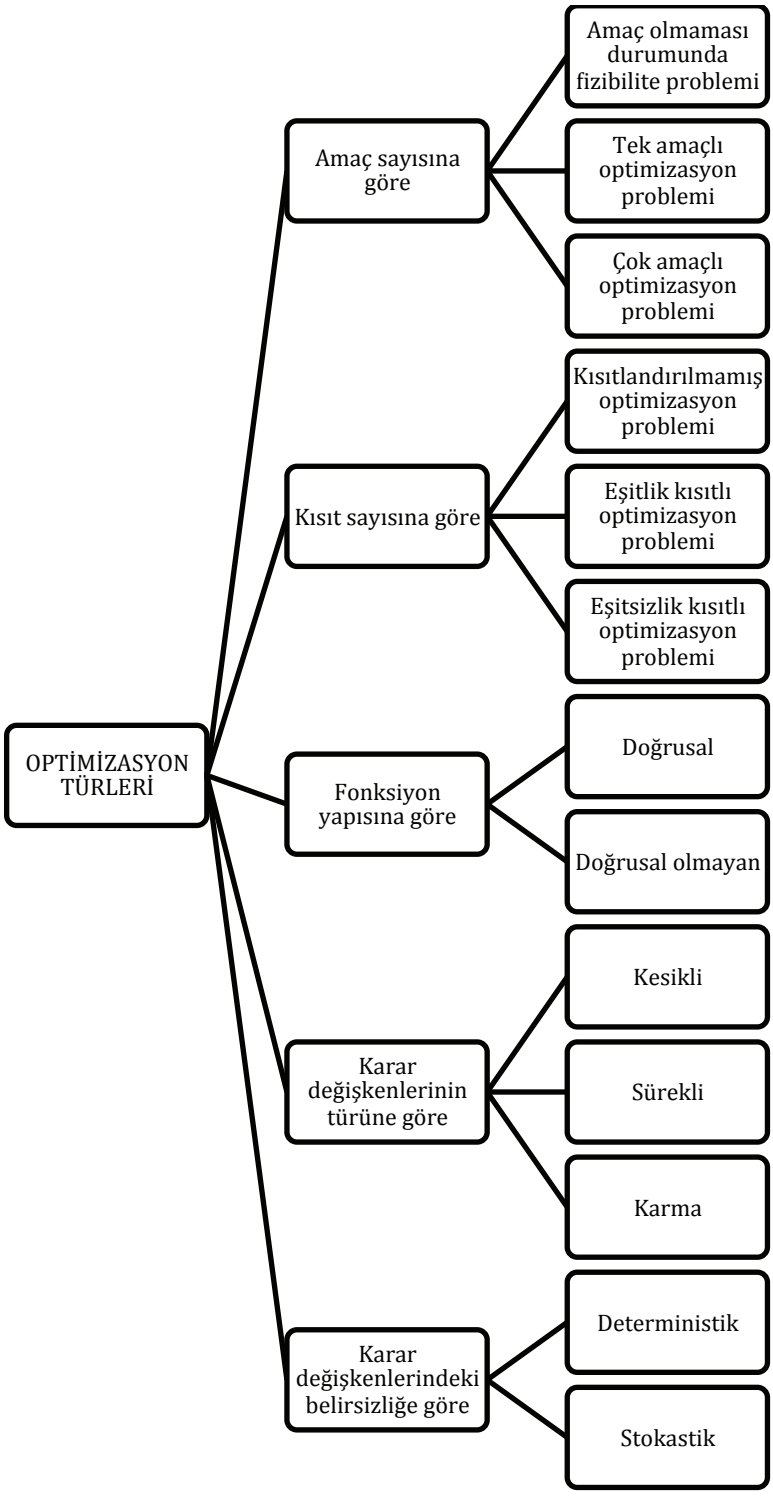
Literatürde optimizasyon ile ilgili yapılmış sınıflandırmalar incelendiğinde, bazı terminolojiler ile ilgili karışıklıklar olduğu görülmektedir. Ancak en çok kullanılan kavramlar ile bir sınıflandırma yapıldığında; amaçların sayısı, kısıtlar, fonksiyon yapıları, amaç fonksiyonlarının durumu, tasarım değişkenlerinin türü, değerlerdeki belirsizlik ve hesaplama süresi gibi başlıkları dikkate alan Şekil 2.1'deki sınıflandırma ortaya çıkmaktadır (Yang, 2010a).

2.2. OPTİMİZASYON ALGORİTMALARININ SINIFLANDIRILMASI

Farklı türdeki optimizasyon problemleri için farklı optimizasyon tekniklerinin kullanılması gerekmektedir. Çünkü bazı algoritmalar belirli türdeki optimizasyonlar için diğer algoritmalara nazaran daha uygundur (Yang, 2010a).

Optimizasyon algoritmalarını odak noktasına ya da karşılaştırılmaya çalışılan özelliklere bağlı olarak çeşitli şekillerde sınıflandırmak mümkündür. Algoritmalar; gradyan tabanlı (ya da türev tabanlı yöntemler) ve gradyansız (veya türev içermeyen yöntemler) olarak sınıflandırılabilmektedir. Dik iniş ve Gauss-Newton yöntemleri, algoritmada türev bilgilerini kullandıklarından gradyan tabanlı yöntemler sayılırken Nelder-Mead yöntemi herhangi bir türev kullanmayıp sadece amaç değerlerini dikkate aldığından gradyansız bir yöntemdir.

Farklı bir bakış açısından ele alındığında ise algoritmalar, yörünge (trajectory) tabanlı veya popülasyon temelli olarak sınıflandırılmaktadır. Bir yörünge tabanlı algoritma genellikle iterasyonlar ve optimizasyon süreci devam ederken yörünge çizen tek bir ajan veya çözüm noktası kullanmaktadır. Tepe tırmanma böyle bir yöntem olup, başlangıç noktası ile son noktayı bir parçalı zikzak yol üzerinden birbirine bağlamaktadır. Bir diğer önemli örnek, yaygın olarak kullanılan tavlama benzetimidir. Öte yandan, parçacık sürü optimizasyonu gibi popülasyon tabanlı algoritmalar, birden fazla yol çizen çoklu etmenle (agent) çalışmaktadır. Popülasyon tabanlı algoritmalara en klasik örnek ise genetik algoritmadır.

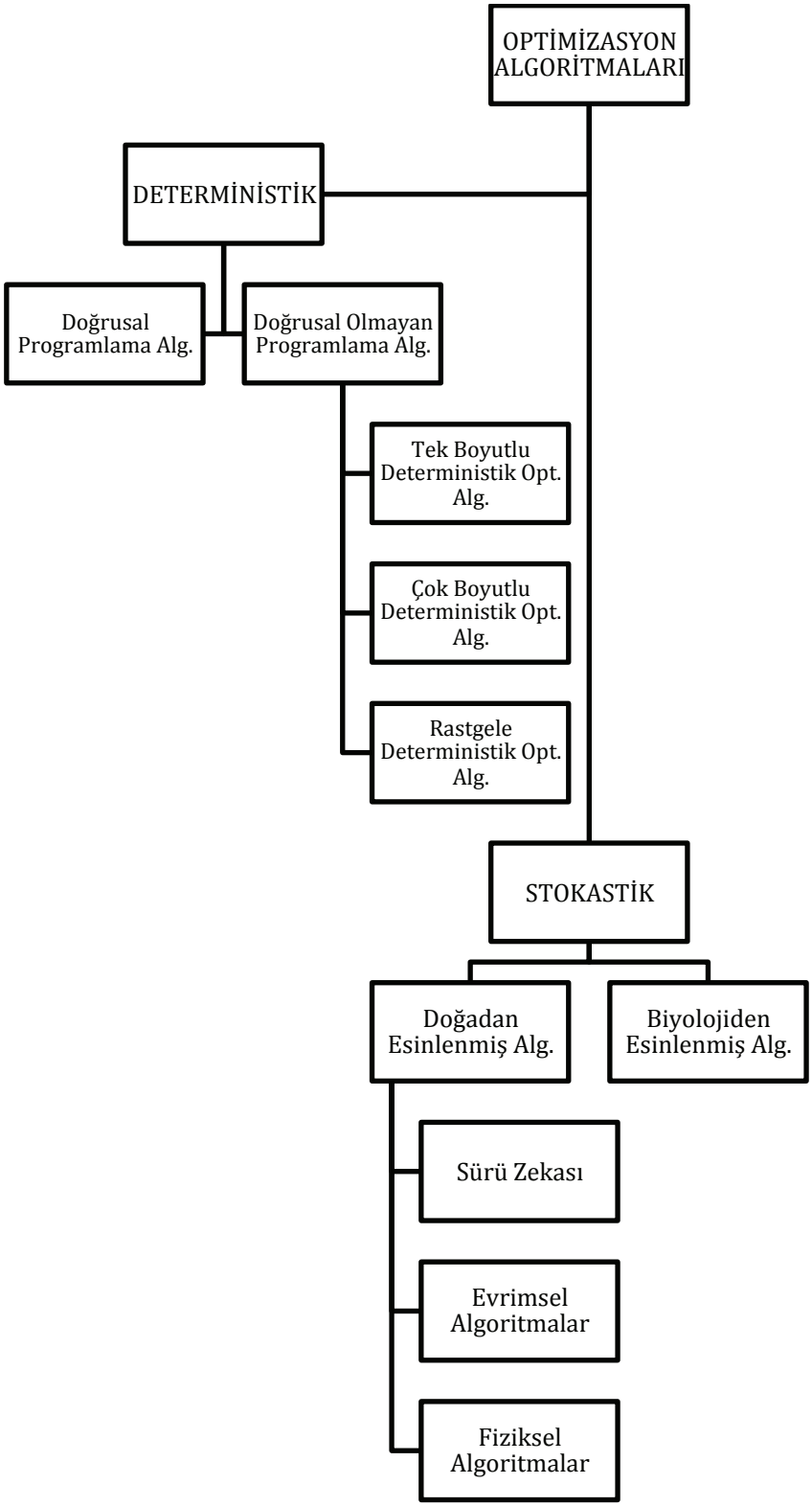


Şekil 2.1: Optimizasyon problemlerinin sınıflandırılması (Yang, 2010a).

Bellek kullanımı da bazı algoritmalar için önemli olabilmektedir. Bu nedenle, optimizasyon algoritmaları belleksiz veya geçmiş bazlı olarak da sınıflandırılabilmektedir. Birçok algoritmada hafıza kullanılmamakta, sadece mevcut en iyi veya mevcut durum kaydedilmekte ve tüm arama geçmişi göz ardı edilebilmektedir. Bu anlamda, bu tür algoritmalar belleksiz olarak kabul edilmektedir. Genetik algoritma ve guguk kuşu arama algoritması bu kategoriye uyan algoritmalardır. Diğer taraftan, bazı algoritmalar bellek (geçmiş) kullanmaktadır. Örneğin tabu aramada, tabu listeleri hareket geçmişlerini kaydetmek için kullanılmakta ve kullanılan çözümler tekrar denenmemektedir. Bu, işlem zamanını önemli ölçüde kısaltacak tamamen farklı yeni çözümleri keşfetmeyi teşvik etmektedir (Koziel ve Yang, 2011).

Genel olarak optimizasyon algoritmaları Şekil 2.2'de görüldüğü gibi deterministik algoritmalar ve stokastik algoritmalar olmak üzere iki kategoriye ayrılabilmektedir Simpleks algoritması, karmarkar algoritması gibi algoritmalar deterministik algoritmalar olup doğrusal programlama algoritmalarının içinde yer alırken; iki seçenekli algoritma, parabolik interpolasyon algoritması, brenth algoritması vb. algoritmalar tek boyutlu deterministik doğrusal olmayan programlama algoritmaları içindedir. Şebek arama algoritması, Hooke–Jeeves algoritması, dik iniş algoritması, birleşik gradyan algoritması gibi algoritmalar ise çok boyutlu deterministik doğrusal olmayan programlama algoritmalarına örnek verilebilir.

Stokastik algoritmalar da kendi içinde doğadan esinlenerek geliştirilmiş ve biyolojiden esinlenerek geliştirilmiş algoritmalar olarak ikiye ayrılmaktadır. Yapay sinir ağları, bağışıklık algoritmaları gibi algoritmalar biyolojik tabanlı algoritmalar olup literatürde birçok uygulamaları bulunmaktadır. Doğadan esinlenerek geliştirilmiş algoritmaları ise üç ana başlıkta toplamak mümkündür. Karınca kolonisi optimizasyonu, parçacık sürü optimizasyonu, bakteriyel besin arama algoritması, arı algoritması, maymun arama algoritması, ateşböceği algoritması, guguk kuşu arama algoritması, yarasa algoritması gibi algoritmalar sürü zekasına dayanan algoritmalarken; genetik algoritma, evrimsel programlama, tabu arama gibi algoritmalar evrimsel algoritmalar olarak literatürde yer almaktadır. Tavlama benzetimi, armoni arama vb. gibi algoritmalar ise fiziksel algoritmalar kategorisi altında bulunmaktadır (Qing, 2009; Yang, 2010a; Wang ve Guo, 2013).



Şekil 2.2: Optimizasyon algoritmalarının sınıflandırılması (Qing, 2009; Yang, 2010a; Wang ve Guo, 2013'den yararlanılarak geliştirilmiştir.).

Deterministik algoritmalarda sıkı bir prosedür takip edilmekte ve karar değişkenleri ile fonksiyonların değerleri tekrarlanabilmektedir. Örneğin, deterministik bir algoritma olan tepe tırmanma algoritması, program ne zaman çalıştırılsa çalıştırılsın aynı başlangıç noktası için aynı yolu takip edecektir. Öte yandan, stokastik algoritmalarda her zaman bir rastgelelik söz konusudur. Buna verilebilecek en iyi örnek genetik algoritmalardır. Genetik algoritmalarda popülasyondaki diziler ya da çözümler algoritmada rastgele sayıların kullanılması sebebiyle program her çalıştırıldığında farklı olacaktır. Nihai sonuçta büyük bir farklılık olmamakla birlikte her bireyin yolları tam olarak tekrarlanabilir değildir (Yang, 2010a).

Stokastik algoritmaları da aralarında küçük fark olmasına rağmen sezgisel ve metasezgisel olarak ayırmak mümkündür. Sezgisel algoritmalar kabul edilebilir bir sürede optimizasyon problemlerinde iyi sonuçlar bulabilmektedir. Dolayısıyla bu, en iyi sonuca ulaşmanın gerekli olmadığı durumlarda yeterli olmaktadır. Metasezgiseller ise sezgisel algoritmalar üzerinde yapılan geliştirmelerle ortaya çıkmış metodolojilerdir. Tüm metasezgisel algoritmalar yerel arama ve rassallık kullanmaktadır. Rassallık yerel aramadan uzaklaşıp daha genel bir ölçekte arama için iyi bir yol sağlamaktadır (Yang, 2010a; Yang, 2010b).

2.2.1. Metasezgiseller

Bir önceki bölümde de değinildiği gibi, stokastik algoritmalar aslında birer metasezgisel algoritmadır. Gerçek yaşam problemlerine bakıldığında çözümler çok sayıda ve bazen de sonsuz sayıda olabilmektedir. Dolayısıyla böyle bir durumda problemin tek bir çözümü sadece özel bir takım parametre değerleri ile gerçekleşeceğinden, geleneksel optimizasyon yaklaşımının uygulanması söz konusu olamamaktadır (Antoniou ve Lu, 2007). Gerçek yaşam problemlerinde karşılaşılan zorluklar şu şekilde özetlenebilir:

- Arama uzayındaki olası çözümlerin sayısının, en iyi çözümü bulmada ayrıntılı arama yapmak için çok büyük olması
- Problemin çok karmaşık olması
- Önerilen çözümün kalitesini tanımlayan değerlendirme fonksiyonunun (evaluation function) zamana bağlı olarak değişmesi ve böylece tek bir çözüme değil bütün çözüm dizilerine ihtiyaç duyulması
- Olası çözümlerin oldukça kısıtlandırılmış olması

- Problemi çözen kişinin yetersiz olması (Brownlee, 2011)

İşte bu tarz problemlerin çözümünde, son yirmi yılda popülerlik kazanmış "yaklaşık optimizasyon teknikleri" ailesini temsil eden *metasezgisellerden* faydalanılmaktadır. Metasezgiseller gelecek vadeden başarılı teknikler arasında olup, bilim ve mühendislik alanındaki zor ve karmaşık problemlere makul bir sürede "kabul edilebilir" çözümler sunmaktadır. Bu nedenle metasezgisellere olan ilgi günden güne artmaktadır. Optimizasyon algoritmalarından farklı olarak, metasezgiseller elde edilen çözümlerin optimalliğini garanti etmemektedir.

"Sezgisel (heuristic)" kelimesi, sorunları çözmek için yeni stratejiler (kurallar) keşfetme sanatı anlamına gelen eski Yunanca "heuriskein" kelimesinden gelmektedir. Eki "meta" ise, yine Yunanca bir kelime olup "üst düzey metodoloji" anlamındadır. Metasezgisel terimi ilk olarak Fred Glover tarafından "Future paths for integer programming and links to artificial intelligence" adlı yayında geçmiştir. Metasezgisel arama yöntemleri, belirli optimizasyon problemlerini çözmek için kullanılan sezgisellerin tasarımında yol gösterici stratejiler olarak kullanılan üst düzey genel metodolojiler (şablon) olarak tanımlanabilir (Talbi, 2009).

Metasezgisellerin özellikleri ana hatlarıyla şu şekildedir:

- ✓ Metasezgiseller arama işlemine "rehberlik" eden stratejilerdir.
- ✓ Amaç, verimli en uygun (yakın) çözümü bulmak için arama uzayını etkin taramaktır.
- ✓ Metasezgisel algoritmaları içeren teknikler, basit yerel arama işlemlerinden karmaşık öğrenme süreçlerine kadar uzanan geniş bir yelpazededir.
- ✓ Metasezgisel algoritmalar yaklaşıktır.
- ✓ Algoritmalar, arama uzayının kapalı alanlarında takılmayı önleyecek mekanizmalar içerebilir.
- ✓ Metasezgiseller temel kavramları, soyut düzeyde açıklamaya izin verir.
- ✓ Metasezgiseller probleme özgü değildir.
- ✓ Metasezgiseller üst düzey strateji ile kontrol edilen sezgisellerin özel bilgisinden yararlanabilirler (Blum ve Roli, 2003; Brownlee, 2011).

Metasezgiseller sezgisel algoritmaların geliştirilmesi ile türetildiklerinden işlevleri de sezgisel algoritmalara benzemektedir. Metasezgisellerde problemin çözümüne genellikle bir veya daha fazla rastgele oluşturulmuş çözümlerle başlanılmaktadır. Daha sonra her iterasyon adımında, mevcut çözüm(ler) üretilen yeni çözüm(ler)le değiştirilmektedir. Yeni çözümler, genellikle varyasyon operatörleri olarak adlandırılan arama operatörleri tarafından oluşturulmaktadır. Ayrıca, problem çözümü sırasında düzenli olarak yoğunlaşma (intensification/exploitation) ve çeşitlendirme (diversification/exploration) aşamaları gerçekleştirilmektedir. Yoğunlaşma aşaması sırasında, mevcut çözümlerin amaç fonksiyonu değeri kullanılmakta ve yüksek kaliteli çözümler için varyasyon operatörlerinin uygulanmasına odaklanılmaktadır. Bu aşama ile aslında maksimum yakınsama için yerel arama sağlanmaktadır. Çeşitlendirme sırasında ise, genellikle amaç fonksiyonu değerleri dikkate alınmamakta fakat sistematik olarak mevcut çözümler değiştirilerek arama uzayındaki yeni alanlar araştırılmaktadır. Böylelikle küresel bir inceleme söz konusu olmakta ve küresel en iyinin maksimum olasılıkla elde edilebilirliği sağlanmaktadır (Rathlauf, 2011; Yang, 2010b, 2012).

2.2.2. Metasezgisellerin Sınıflandırılması

Metasezgisel algoritmaların birçoğu; doğa, topluluk ya da fizik gibi diğer alanlarda bulunan başarılı stratejileri taklit etmektedir. Algoritmaların problemlerden bağımsız çalışması, farklı alanlarda uygulanmalarını kolaylaştırmaktadır (Rathlauf, 2011).

Metasezgisel algoritmaları sınıflandırmak ve tanımlamak için farklı yollar bulunmaktadır. Her biri belirli bir bakış açısının sonucu olmak üzere, aralarında seçilen ayırt edici özelliklerine bağlı olarak birçok sınıflandırma mümkündür. Metasezgisellerin en önemli sınıflandırması kısaca aşağıdaki gibidir (Blum ve Roli, 2003):

- *Doğadan esinlenen veya doğadan esinlenmeyen algoritmalar:* Metasezgiselleri sınıflandırmanın en kolay yolu çıkış noktalarına bakmaktır. Karınca Kolonisi Optimizasyonu ya da Yarasa Algoritması gibi doğadan esinlenen algoritmaların yanında, Tabu Arama ve Yinelemeli Yerel Arama gibi doğadan esinlenmeyen algoritmalar bulunmaktadır. Ancak bu şekilde bir sınıflandırma iki sebepten dolayı çok da anlamlı değildir. Birincisi, birçok güncel hibrid algoritma ya bu iki sınıftan hiçbirine girmemekte ya da her ikisinin kapsamında da yer almaktadır.

İkinci olarak ise, bazen algoritmanın taşıdığı özellikleri açıkça iki sınıftan birine sokmak zordur. Örneğin Tabu arama doğadan esinlenmeyen algoritmalar içinde yer alırken kullandığı hafıza doğa kaynaklı olarak görülebilmektedir.

- *Popülasyon tabanlı veya tek bir nokta arama:* Metasezgisellerin sınıflandırılması için kullanılan bir diğer özellik, aynı anda kullanılabilir çözümlerin sayısıdır. Tek çözüm üzerinde çalışan algoritmalar yörünge tabanlı yöntemler olarak adlandırılmakta ve Tabu Arama, Yinelemeli Yerel Arama, Yakın Komşuluk Arama gibi yerel arama tabanlı metasezgiselleri kapsamaktadır. Bu algoritmaların ortak özelliği, arama işlemi süresince arama uzayında bir yörünge oluşturmalarıdır. Popülasyon tabanlı metasezgiseller ise tam tersine, arama işlemlerini, arama uzayında bir noktalar dizisi evrimi şeklinde gerçekleştirmektedir.
- *Dinamik veya statik amaç fonksiyonu:* Metasezgiseller ayrıca amaç fonksiyonu kullanma şekillerine göre sınıflandırılabilmektedir. Bazı algoritmalar verilen problemi temsil eden amaç fonksiyonunu "olduğu gibi" tutarken, Rehberli Yerel Arama (Guided Local Search) gibi bazı yöntemler, fonksiyonu arama sırasında değiştirmektedir. Bu yaklaşımın temel sebebi, arama bölgesini değiştirerek yerel minimumdan kaçınmaktır. Buna göre, arama işlemi sırasında toplanan bilgiler birleştirmeye çalışılarak amaç fonksiyonu değiştirilmektedir.
- *Bir veya çeşitli komşuluk yapıları:* Metasezgisel algoritmaların birçoğu tek bir komşuluk yapısı üzerinde çalışmaktadır. Değişken Komşuluk Arama (Variable Neighborhood Search) gibi diğer metasezgiseller, arama çeşitlendirmesine imkan veren bir dizi komşuluk yapısı kullanmaktadır.
- *Bellek kullanan veya kullanmayan yöntemler:* Metasezgiselleri sınıflandırmak için dikkate alınan bir diğer önemli özellik, arama geçmişinin kullanılıp kullanılmamasıdır. Bellek kullanmayan algoritmalar bir Markov işlemi gerçekleştirmekte ve bir sonraki adımı belirlemek için, sadece arama sürecinin mevcut durum bilgisini kullanmaktadır. Bellek kullanımı ise birçok farklı şekilde olabilmektedir. Genellikle ayrım, kısa ya da uzun vadeli hafıza kullanımı arasında yapılmaktadır. Kısa vadeli hafıza kullanımında genellikle, son zamanlarda gerçekleştirilen hareketlerin rotaları, kullanılan çözümler veya genel olarak alınan kararlar tutulmaktadır. Uzun vadeli hafıza ise genellikle arama ile

ilgili parametrelerin bir birikimidir. Bellek kullanımı günümüzde güçlü metasezgisellerin temel unsurlarından biri olarak kabul edilmektedir.

2.2.3. Doğadan Esinlenen Algoritmalar

Doğa, yüzyıllardır gelişmesini sürdürmekte ve sürekli değişen ortamlara uyum sağlamak, problemleri çözmek için çeşitli yaratıcı çözümler sunmaktadır. Darwin'in doğal seleksiyon yoluyla evrim teorisi açısından bakıldığında, ortama en iyi uyum gösteren türler nesillerinin devamlılığını sağlayacaktır. Rakipleri karşısında herhangi bir evrimsel avantaj, uzun vadede bireylerin ve türlerin başarısını ve üreme olasılığını artırabilmektedir.

Doğadaki karmaşık sistemlerin başarılı özelliklerini taklit ederek doğadan çeşitli şeyler öğrenmek mümkündür. Doğadan esinlenen algoritmalar da bu amaçla ortaya çıkmıştır. Bu algoritmalar birçok geleneksel, köklü yönteme nazaran çok kısa bir geçmişe sahiptir. Ancak giderek artan çeşitli aralıklardaki uygulamaları ile büyük bir potansiyele, esnekliğe ve etkinliğe sahip olduklarını göstermişlerdir.

Doğal olaylardan ve biyolojik sistemlerden ilham alan yeni algoritmalar hemen hemen her yıl ortaya çıkmaktadır (Yang, 2012). Doğadan esinlenerek geliştirilen bazı algoritmalar aşağıda detaylı bir şekilde incelenmiştir.

2.2.3.1. Tavlama Benzetimi

Tavlama fizikte ve mühendislikte, bir katının ısı banyosunda düşük enerjili durumlarının elde edilmesi için uygulanan bir ısıl süreç olarak tanımlanmaktadır. Eğer katı bir malzeme erime noktasına kadar ısıtılır ve katı hale geçinceye kadar tekrar soğutulursa, soğutulmuş katının yapısal özellikleri soğutma hızına bağlı olarak değişmektedir. Örneğin, büyük kristal yapılar çok yavaş bir soğutma sonucunda elde edilmekte, fakat hızlı bir soğutma, kristal yapısında kusurların oluşmasına neden olabilmektedir. Katının iç özelliklerini kaybetmeden düşük enerji durumuna getirilmesi için kullanılan tavlama süreci ise iki adımı içermektedir:

1. Isı banyosunun sıcaklığının katının eriyebileceği en yüksek değere yükseltilmesi,
2. Katının enerjisiz durumunda (ground state) parçacıkları kendilerini düzenleyene kadar, ısı banyosu sıcaklığının dikkatli bir şekilde azaltılması

Tavlama benzetiminin (TB) temelini oluşturan ilk fikirler, Metropolis ve diğerleri (1953) tarafından sunulan ve ısı banyosundaki bir katının "ısıl dengeye (thermal equilibrium)" ulaşıncaya kadar olan gelişiminin Monte Carlo tekniği ile benzetimini yapan çalışmaya dayanmaktadır. 1983 yılında Kirkpatrick ve diğerleri, bu tür bir benzetimin optimal sonuca yakınsama amacıyla zor kombinatoriyel problemlerin çözümünde kullanılabileceğini söylemişler ve Tavlama Benzetimi algoritmasını ortaya koymuşlardır.

TB algoritması bir başlangıç çözümle ve yerel minimumlardan kaçınmak için yüksek bir sıcaklık (T) değeri ile başlamaktadır. Her bir hesaplama adımında algoritma, bir önceki çözümün komşuları arasından çok sayıda çözümler üretmekte ve sıcaklığı önceden belirlenen bir kurala göre azaltmaktadır. Üretilen komşu çözümler arasından rastgele bir tanesi seçilmektedir. Bu seçilen çözüm ya yeni çözüm olarak kabul edilmekte ya da reddedilmektedir. İşlemler yeni çözümle (eğer kabul edilmişse) veya bir önceki çözümle (eğer kabul edilmemişse) tekrar başlamaktadır. Algoritmayı durdurmak için pek çok farklı kriter (iterasyon sayısı, sıcaklığın minimum değeri vb.) kullanılabilir.

Tavlama sürecinin optimizasyon problemlerine uyarlanması sürecinde yapılan benzetimler Tablo 2.1'de verilmiştir.

Tablo 2.1: Tavlama sürecinin optimizasyon problemlerine uyarlanması.

Tavlama Süreci	***Optimizasyon***
Sistem kararlı bir hal alır	Mümkün çözümler bulunur
Sistemin enerjisi	Amaç fonksiyonu değeri (maliyet)
Durumun Değişmesi	Komşu Çözüm
Sıcaklık	Kontrol Parametresi
Donma Durumu	Sezgisel Çözüm

Algoritmaya ait sözde kod ise şu şekildedir:

TAVLAMA BENZETİMİ ALGORİTMASI

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_p)^T$
Başlangıç sıcaklığını (T_0) ve başlangıç tahminini (x^0) sıfırla.
Son sıcaklığı (T_f) ve maksimum iterasyon sayısı N_{maks}'ı belirle.
Soğutma işlemini tanımla $T \rightarrow \alpha T, (0 < \alpha < 1)$
while $T > T_f$ ve $n < N_{maks}$
    Rastgele olarak yeni konumlara hareket et: $x_{n+1} = x_n + rand$
    $\Delta f = f_{n+1}(x_{n+1}) - f_n(x_n)$ hesapla.
    Yeni sonuç daha iyi ise onu kabul et.
    if daha iyi değilse
        Bir r rassal sayısı türet.
        Eğer $p = exp[-\Delta f/T] > r$ ise kabul et.
    end if
    En iyi x ve f'yi güncelle.
    $n = n+1$
end while

Şekil 2.3: Tavlama benzetimi algoritması sözde kodu (Yang, 2010a).

2.2.3.2. Genetik Algoritma

Genetik algoritmalar (GA), Darwin ve Mendel'in evrim ve genetik teorilerinden esinlenerek arama, optimizasyon ve makina öğrenmesi problemlerine çözümler üreten evrimsel bir algoritmadır.

John Holland 1975 yılındaki çalışmasında, yeni bireyler oluşturmak üzere ebeveynlerden gelen bilgi parçacıklarını farklı kombinasyonlarda birleştirip GA tekniğini geliştirmiştir. Bu tekniğin yeniliği; çaprazlama, mutasyon ve seçme gibi genetik operatörleri kullanmasıdır.

Genetik algoritmanın geleneksel algoritmalara göre iki ana avantajı bulunmaktadır. Bunlardan biri karmaşık problemleri çözme yeteneği diğeri ise paralelliktir. Amaç fonksiyonu sürekli veya süreksiz, sabit veya geçici, doğrusal veya doğrusal olmayan olsun, genetik algoritmalar tarafından rahatça ele alınabilmektedir. Birden fazla gen yapısı paralel uygulamalar için de uygun olabilmektedir (Koziel ve Yang, 2011).

Holland'ın geliştirmiş olduğu genetik algoritma metodunda bir popülasyonda bulunan kromozom, doğal seçilim mekanizması ile çaprazlama ve mutasyon gibi operatörler kullanılarak yeni bir popülasyona aktarılır. Genetik algoritma başlangıç popülasyonunun oluşturulmasıyla başlar. Başlangıç popülasyonundaki her bir birey aday çözümdür ve her bireye "kromozom" adı verilir. Kromozomlar ise bitlerden oluşur. Amaç fonksiyonuna göre uygunluk değerleri en iyi olan kromozomlar bir

sonraki popülasyona (nesile) aktarılır. Uygunluk değerleri düşük olan kromozomların ise yaşamasına izin verilmez. Düşük uygunluk değerli kromozomların yerine iyi kromozomlardan yeni bireyler üretilir. Yeni bireylerin üretilmesinde doğadan esinlenilerek çaprazlama ve mutasyon operatörleri kullanılır. Belirlenen amaç fonksiyonuna göre en iyi çözüme ulaşıncaya kadar bu adımlar tekrar edilir. Buna göre genetik algoritmanın sözde kodu Şekil 2.4'te yer almaktadır:

GENETİK ALGORİTMA

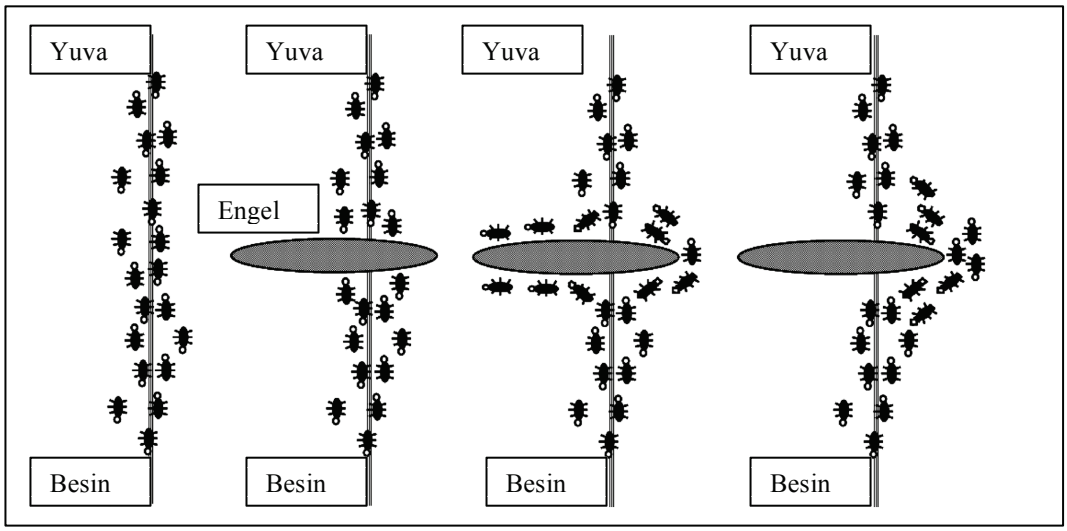
Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_d)^T$
Çözümleri ikili dizi olarak kromozomların içine kodla
F (Maksimizasyon için $F \propto f(x)$) uygunluk değerini tanımla.
İlk popülasyonu oluştur.
Çaprazlama olasılığı (p_c)ve mutasyon olasılığının (p_m) başlangıç değerini ata.
while $t < maksimumüretim$
Çaprazlama ve mutasyon ile yeni çözümler üret.
if p_c<rand ise çaprazla **end if**
if p_m<rand ise mutasyona uğrat **end if**
Eğer uygunluk artarsa yeni çözümleri kabul et.
Gelecek nesil için mevcut en iyiyi seç.
end while
Sonuçları ve görselleri işle.

Şekil 2.4: Genetik algoritma sözde kodu (Yang, 2010b).

2.2.3.3. Karınca Kolonisi Optimizasyonu

Karıncalar, nüfusları yaklaşık 2 milyon ile 25 milyon arasında değişen organize koloniler halinde yaşayan sosyal böceklerdir. Yiyecek arama durumunda, karınca sürüsü bulunduğu yerel ortamda iletişim ve etkileşim içinde olmaktadır. Her karınca diğer karıncalar ile haberleşmek için kokulu ve uçucu bir kimyasal olan *feromon*u geçtiği yola bırakmakta ve her karınca diğer karıncalar tarafından atılmış bu feromon ile işaretlenmiş rotayı takip edebilmektedir. Karıncalar bir besin kaynağı bulduğunda, besin kaynağını ve bu kaynağa giden yolları yine feromon ile belirtmektedir. İlk rastgele yiyecek arama rotasının ardından, feromon konsantrasyonu değişmekte ve karıncalar yüksek feromon konsantrasyonlu rotayı takip etmektedir. Kolonideki karıncaların yaklaşık olarak geçtikleri yola eşit seviyede feromon maddesi bıraktıkları kabul edilirse, feromon miktarı artan karınca sayısı ile çoğalmaktadır. Birçok karıncanın aynı rotayı takip etmesi sonucunda o rota tercih edilen yol haline gelmektedir. Dolayısıyla olumlu bir geribildirim mekanizması sonucunda bu yollar, genelde en kısa ya da daha etkili olarak ortaya çıkacaktır (Yang, 2010a).

Şekil 2.5'te Goss ve diğerleri tarafından gerçek Arjantin karıncalarından oluşan, görme yeteneği zayıf koloni ile yapılmış bir deney gösterilmektedir.



Şekil 2.5: Besin kaynağı ile yuva arasında optimal bir yol arayan bir karınca kolonisinin gösterimi (Goss ve diğ., 1989).

Şekilde görüldüğü gibi, bir karınca sürüsünün besine ulaşma rotası üzerine konan bir cisim (engel) söz konusu olduğunda, karıncalar tercih edilmesi gereken feromonsuz yönlerle karşılaşmakta ve her karıncanın sol veya sağ yolu seçme olasılığı eşit olmaktadır. Ancak karıncaların hızlarının hemen hemen aynı olduğu varsayıldığında, birim zamanda kısa yol üzerinden geçen karıncaların sayısının birim zamanda uzun yoldan geçen karıncaların sayısından daha fazla olacağı da açıktır. Bu durumda birim zamanda kısa yola bırakılan feromon miktarı da daha fazla olacaktır. Dolayısıyla yön tercihinin feromon ile bağlantılı olması, yeni gelen karıncaların kısa yolu daha yüksek olasılıkla seçmesine ve belirli bir süre sonra hepsinin kısa yolu seçerek değişmiş olan çevreye adapte olmasına neden olacaktır (Talbi, 2009)

Karıncaların yukarıda belirtilen davranış özelliklerine dayanarak, bilim adamları bir dizi güçlü karınca kolonisi algoritmaları geliştirmişlerdir. Karınca kolonisi optimizasyonu ile ilgili ilk çalışma Dorigo ve diğerleri tarafından 1991 yılında yapılmış ve yazarların kendi sistemleri "Karınca Sistemi", ortaya çıkardıkları algoritma da "Karınca Algoritması" olarak tanımlanmıştır. Bulunan bu algoritmanın performansını arttırmak ve algoritmanın uygulanabileceği çözüm alanlarını geliştirmek amacıyla birçok değişik karınca kolonisi algoritması ortaya konmuştur (Glover ve Kochenberger, 2003).

Karınca kolonisi optimizasyonun temel adımları Şekil 2.6'da gösterilen sözde kod ile özetlenebilir.

KARINCA KOLONİSİ OPTİMİZASYONU

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_n)^T$
[rotalama problemi için $f(x_{ij}), (i,j) \in \{1,2,\dots,n\}$]
Feromon buharlaşma oranı $\gamma' yi$ belirle.
while
for tüm n boyutları
Yeni sonuçları üret.
Yeni sonuçları değerlendir.
Feromon $\delta\phi_{ij}$'li en iyi lokasyonları/rotaları işaretle.
Feromonu yenile: $\phi_{ij} \leftarrow (1-\gamma)\phi_{ij} + \delta\phi_{ij}$
end for
En iyi sonucu bulana kadar işlemlere devam et.
end while
Sonuçları ve görselleri işle.

Şekil 2.6: Karınca kolonisi optimizasyonunun sözde kodu (Yang, 2010a).

2.2.3.4. Parçacık Sürü Optimizasyonu

Bilim adamları kuş ve balık sürülerindeki organizmaların hareketlerini açıklayan çeşitli bilgisayar simülasyonları geliştirmişlerdir. Reynolds (1987), kuş sürülerinin uçuşlarındaki simetriyi incelerken, Heppner ve Grenander (1990), bir sürüdeki çok sayıda kuşun nasıl senkronize bir biçimde uçup, aniden topluca yön değiştirebildiklerini, aniden dağılıp yine aniden tekrar grup haline gelebildiklerini incelemişlerdir.

Parçacık sürü optimizasyonu (PSO), kuş ve balık gibi hayvanların yukarıda belirtilen sosyal besin bulma hareketinden esinlenerek, sürekli fonksiyonlar için stokastik bir optimizasyon yöntemi olarak Eberhart ve Kennedy (1995) tarafından geliştirilmiştir. Buna göre sürüdeki bireyler en uygun bireyleri takip ederek uçmakta ve genellikle hareketleri geçmişteki iyi bölgelere doğru olmaktadır.

Sürü, birbirleriyle doğrudan ya da dolaylı olarak iletişim içerisinde olan ve kolektif olarak bir problemi çözme amacı taşıyan belirli sayıda bireylerden oluşmuş bir yığın olarak tanımlanmaktadır. Sürüdeki bireylerden her birinin amacı, çevrelerine uyum sağlayacak şekilde hareket etmektir. Matematiksel olarak bu ifade, her bireyin bir amaç fonksiyonu doğrultusunda arama uzayında hareket etmesi olarak tanımlanmıştır. Ayrıca bireyler sosyal varlıklar olup, diğer bireylerle sürekli iletişime açıktır. Sürüdeki her bir

bireye açık olan iki tür bilgi mevcuttur. Bunlardan birincisi kendi deneyimleri, ikincisi ise belirli sayıdaki diğer bireyin (bu sayı sürünün tamamı da olabilir yalnızca diğer bir birey de olabilir) en iyi deneyimleridir. Bu iki bilgi, bireyin öğrenme davranışına etki ederek sürü içerisinde bir tür bilişsel akışı temsil etmektedir.

Algoritmada bireyler parçacıklar olarak tanımlanmıştır. Algoritmanın amacı, tüm parçacıkları çok boyutlu bir hiper-hacimdeki optimum noktaya yerleştirmektir. Bunun için öncelikle küçük rastgele başlangıç hızları ile tüm parçacıklar, rastgele başlangıç konumlarına atanmaktadır. Algoritma; parçacığın hızına, çözüm uzayında bilinen en iyi küresel konuma (*gbest*) ve bir parçacık için bilinen en iyi konumuna (*pbest*) dayalı olarak sırayla her parçacığın konumunu ilerleten bir simülasyon gibi yürütülmektedir. Amaç fonksiyonu her konum güncellemesinden sonra yeniden hesaplanmaktadır. Zamanla, çözüm uzayı içinde bilinen iyi konumların çeşitlendirilme ve yoğunlaştırılma kombinasyonlarıyla parçacıklar kümelenmekte ya da optimum veya birkaç optimum etrafında bir araya gelmektedir.

Her yinelemede bir parçacığın hızı aşağıdaki denklem 2.1 kullanılarak güncellenmektedir:

$$v_i^{(t+1)} = w \times v_i^{(t)} + (c_1 \times rand_1 \times (x_i^{best} - x_i^{(t)})) + (c_2 \times rand_2 \times (x_{gbest} - x_i^{(t)})) \tag{2.1}$$

Burada $v_i^{(t+1)}$ i. parçacık için yeni hızı, w atalet ağırlığını, c_1 ve c_2 sırasıyla bireysel ve küresel en iyi konumlar için öğrenme katsayılarını, $x_i^{(t)}$ i. parçacığın t anındaki konumunu, x_i^{best} i. parçacığın bilinen en iyi konumunu ve x_{gbest} sürünün bilinen en iyi konumunu göstermektedir. $rand$ fonksiyonu, [0,1] aralığında düzgün dağılan rassal sayılar üretmektedir.

Parçacığın konumunun güncellenmesi ise denklem 2.2 kullanılarak şu şekilde yapılmaktadır:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t)} \tag{2.2}$$

PSO'da dikkat edilmesi gereken unsurlar şunlardır:

- Parçacık sayısı düşük (20-40 civarında) olmalıdır.
- Bir parçacığın hızı (yineleme başına konumundaki maksimum değişim) sınırlı olmalıdır (etki alanının belirli bir yüzdesi olacak vb. gibi).
- Öğrenme faktörleri (küresel ve bireysel en iyi konumlara karşı önyargılar), 0 ve 4 arasında, genellikle 2 olmalıdır.
- Yerel komşuluklar, parçacıkların konumları arasındaki öklid uzaklığına dayalı belirlenmelidir.
- Bir atalet veya ivme katsayısı, hız değişikliğini sınırlamak için tanımlanabilir (Brownlee, 2011).

Parçacık sürü optimizasyonu algoritmasının sözde kodu Şekil 2.7'de verilmiştir:

PARÇACIK SÜRÜ OPTİMİZASYONU

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_p)^T$
n adet parçacık için x_i konumlarını ve v_i hızlarını tanımla.
t=0 anında $min\{f(x_1),\ldots,f(x_n)\}$ içinden g^* (küresel en iyiyi) bul.
while
$t = t + 1$
    for n adet parçacık ve tüm p boyutlarındaki döngü
    Yeni hızları $v_i^{(t+1)}$ oluştur.
    Yeni konumları $x_i^{(t+1)}$ hesapla.
    Yeni konumlar için amaç fonksiyonu değerlerini hesapla.
    Parçacıklar arasından en iyiyi (x_i^{best}) bul.
    end for
    Küresel en iyiyi (g^*) bul.
end while
Sonuçları göster.

Şekil 2.7: Parçacık Sürü Optimizasyonu sözde kodu (Yang, 2010a).

2.2.3.5. Armoni Arama

Müzik, insan çabaları tarafından oluşturulan en tatmin edici süreçlerden biridir. Müzikteki uyum, estetik açıdan hoş kabul seslerin bir kombinasyon oluşturmasıyla gerçekleşmektedir. Bir müzik aletinin estetik kalitesi sesin perdesine (veya frekansına), ses rengine (veya sesin kalitesine) ve sesin titreşim genliğine (veya sesin şiddetine) bağlıdır. Ses rengi genel olarak ses sinyallerinin dalga formu olarak tanımlanırken, bu dalgaların oluşturduğu armoni belirli bir enstrümanın frekans ve şiddetine dayanmaktadır. Armoni arama da müzikal performans (örneğin, bir caz üçlüsü) içinde

bulunan yapay fenomenden türetilmiş bir algoritma olup daha iyi uyum için arama süreci olarak tanımlanmıştır. Armoni arama ilk olarak Zong Woo Geem ve diğerleri tarafından 2001 yılında geliştirilmiştir. O günden bu yana da fonksiyon optimizasyonu, su dağıtım şebekeleri, yer altı suları modelleme, yapısal tasarım, araç rotalama gibi birçok optimizasyon probleminde uygulanmıştır.

Armoni arama bir müzisyenin doğaçlama çalma süreci yardımıyla detaylı olarak anlatılabilir. Bir müzisyenin doğaçlama yaparken üç tane seçeneği vardır: ya herhangi ünlü bir müzik eserini ezbere çalacaktır (ses perdeleri uyum içinde olacaktır), ya bilinen bir esere benzer birşey çalacaktır (dolayısıyla ses perdesi çok az ayarlanacaktır), ya da rastgele notalarla yeni bir eser oluşturacaktır. Bu üç seçenek optimizasyon için formüle edilirse, armoni hafızasının kullanımı, frekans ayarlama ve rastgele sıralama olmak üzere üç ilgili bileşen ortaya çıkacaktır.

Armoni hafızasının kullanımı, tıpkı genetik algoritmada en uygun bireylerin seçilmesi gibi çok önemlidir. Çünkü bu, en iyi armonilerin yeni hafızaya devredilmesini sağlayacaktır. Hafızanın etkin bir şekilde kullanımı için armoni hafıza kabul oranı parametresi tanımlanmıştır. Eğer bu oran çok düşük ise, sadece birkaç en iyi armoni seçilecek ve yakınsama çok yavaş olacaktır. Oran çok yüksek olduğunda ise, hafızada kullanılan tüm armoniler kullanılacaktır.

İkinci bileşen olan frekans ayarlama, doğrusal veya doğrusal olmayan şekilde gerçekleştirilebilmektedir. Bu süreç aslında bir rastgele adım olarak görülebilir. Frekans ayarlaması genetik algoritmadaki mutasyon operatörüne de benzemektedir.

Diğer bir bileşen olan rastgele sıralama, çözüm çeşitliliğini artırmaktadır. Frekans ayarlama da benzer bir rol üstlenmesine karşın sınırları olduğundan daha çok yerel aramaya benzemektedir. Rastgele sıralamanın kullanımı, küresel optimumu bulmak için yüksek çözüm çeşitliliğine sahip farklı bölgeleri keşfederek sistemi daha ileriye götürmektedir (Geem ve diğ., 2001; Yang, 2009a; Yang, 2010b).

Armoni arama algoritmasının sözde kodu Şekil 2.8'de verilmiştir:

ARMONİ ARAMA

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_p)^T$
İlk armonileri (gerçel sayı sıralarını) oluştur
Frekans ayarlama oranını (r_{pa}), frekans limitlerini ve bant genişliğini tanımla.
Armoni hafıza kabul oranını (r_{kabul}) tanımla
while t<Maksimum iterasyon sayısı
    En iyi armonileri Kabul ederek yeni armoniler yarat.
    Yeni armoniler (çözümler) alarak frekans ayarla.
    if ($rand > r_{kabul}$),
        Var olan armoniyi rasgele seç.
    else if ($rand > r_{pa}$),
        Frekansı limitler arasında rastgele ayarla.
    else
        Rastgele sıralama ile yeni armoniler oluştur.
    end if
        Yeni armoniler (çözümler) daha iyi ise kabul et.
end while
Mevcut en iyi sonuçları bul.

Şekil 2.8: Armoni arama algoritmasının sözde kodu (Yang, 2010a).

2.2.3.6. Arı Algoritması

Arı algoritması ilk olarak Pham ve diğerleri (2005) tarafından tanıtılarak matematiksel fonksiyonların testinde uygulanmıştır. Bu çalışmada, arı algoritması diğer metasezgisellerden (stokastik tavlama benzetimi, genetik algoritma ve karınca kolonisi sistemi) işlem zamanı ve sonuçların doğruluğu açısından daha üstün çıkmış ve böylece güçlü bir optimizasyon yaklaşımı olduğunu göstermiştir (Parpinelli ve Lopes, 2011)

Arı algoritması, kolonisi aynı anda birden fazla yönde ve 10 km ile 14 km üzerinde yayılan bal arılarının yiyecek arama davranışından ilham almıştır (Zang ve diğ, 2010). Bu algoritmada bir arı, problem değişkenlerini içeren bir d-boyutlu vektördür ve optimizasyon problemi için olası bir çözümü temsil etmektedir. Ayrıca çözüm, ziyaret edilen bölgeyi (yani besin kaynağını) temsil etmektedir ve atanmış bir uygunluk değeri vardır. Uygunluk değeri, optimize edilen amaç fonksiyonuna göre hesaplanmaktadır. Algoritma rastgele yeni bölgeler aramak için izci arılar kullanmakta ve güçlendirme için komşuluk aramaya başvurmaktadır (Parpinelli ve Lopes, 2011).

Arı algoritmasının sözde kodu Şekil 2.9'de verilmiştir:

ARI ALGORİTMASI

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_n)^T$ ve kısıtlar
Başlangıç bitki özü seviyelerini $f(x)$ içine kodla
Dans rotasını tanımla.
while (tanımlanan durdurma kriteri)
    for tüm n boyutlarında döngü
        (rotalama ve çizelgeleme problemleri için düğümler)
    Yeni çözümleri üret.
    Yeni çözümleri değerlendir.
    end for
    İletişim kur ve optimal çözüm kümesini güncelle.
end while
Mevcut en iyi sonuçları ver.

Şekil 2.9: Arı algoritması sözde kodu (Yang, 2010a).

2.2.3.7. Bakteriyel Besin Arama Algoritması

Kevin M. Passino, 2002 yılında yayınladığı "Biomimicry of bacterial foraging for distributed optimization and control" adlı çalışmasıyla bakterilerin besin arama sırasında sergiledikleri düzenli hareketlerin karmaşıklığından, komplike optimizasyon problemlerine yeni bir bakış açısı getirmiştir.

Bakteriler, karmaşık yaşam formlarındaki diğer canlılara göre çok daha basit yapıda, tek hücreli organizmalardır. Basit yapılarına rağmen, bulundukları çevreden bilgi elde etmekte, kendilerini yönlendirmekte ve bu bilgi sayesinde etkili bir şekilde hayatta kalmaktadırlar. Bu yaklaşıma benzer olarak, optimizasyon algoritmalarında da bulunulan çevreden bilgi alınmakta ve bu bilgi optimuma hareket için kullanılmaktadır.

Bunun dışında, bakterilerin birbirleri arasında bilgi paylaşımı olduğu bulunmuş olsa da, haberleşme modelleri günümüzde çok fazla bilinmemektedir. Genel olarak bakteriler bireysel olarak ele alınmakta ve sosyal etkileşimleri modellerde kullanılmamaktadır. Dolayısıyla bu modeller sosyal böceklerin (karınca, arı) davranışları için kullanılan etkileşimli modellerden farklılık göstermektedir (Muller ve diğ., 2002).

Bakteriler sınırlı algı ve hareket kabiliyetlerini kullanarak optimum düzeyde enerji harcayıp, yani beslenme yoluyla elde edilen enerjinin beslenmek için harcanan zamana oranını en yüksek seviyede tutmaya çalışarak beslenme faaliyetlerini gerçekleştirmektedirler. Diğer yaşam formlarına nazaran modellenebilmeleri daha kolay olan bu türdeki canlılardan biri olan *Escherichia coli* (E.coli) bakterisi, yapısı ve çalışma şekli en iyi anlaşılan mikroorganizmalardan birisidir.

Bakteriyel besin arama algoritması (BBAA), insan bağırsağında yaşayan E.coli bakterisinin besin arama davranışlarını ve bakteri popülasyonunun bu davranışlara bağlı olarak geçirdiği evreleri modelleyerek elde edilmiş yeni bir optimizasyon tekniğidir. Aşağıda, bu algoritmanın genel yapısı ile ilgili olarak çeşitli açıklamalara yer verilmektedir.

E.coli Bakterisi:

E. coli bakterisi, sitoplazma ve çekirdeği çevreleyen bir kapsüle, bir plazma zarına ve bir hücre duvarına sahiptir. Ayrıca bu silindirik yapıya bağlı olarak bulunan piluslar, bakteriler arası gen transferi için kullanılırken, kamçılar, hareket yeteneği sağlamaktadır. E. Coli bakteri hücreleri, yaklaşık 1μm kalınlığa ve 2 μm uzunluğa sahiptir. Hücre ağırlıkları yaklaşık 1 piko gram olmakla birlikte bunun %70'i sudur.

E. coli bakterisi büyüdüğü zaman boyca uzar ve sonrasında iki eşit parçaya bölünerek yeni yavrular oluşturur. Eğer yeterli miktarda gıda temini sağlanırsa ve ortam sıcaklığı insan bağırsağında olduğu gibi 37^{o}C'de sabit tutulursa, E. Coli bakterisi her 20 dakikada bir kendi kopyalarını oluşturarak çoğalacaktır. Dolayısıyla bakteri popülasyonunun büyümesi nispeten kısa bir sürede üstel olarak katlanacaktır.

E. coli bakterisi besin aramaya ve zararlı maddelerden kaçınmaya imkan sağlayan bir kontrol sistemine sahiptir. Bakterinin kendisine zarar verebilecek durumdaki alkali ve asidik bir ortamdan daha nötr bir ortama yüzerek geçebilmesi, söz konusu kontrol sisteminin sayesindedir. E. coli bakterisinin hareketinin anlaşılması için, onun işleticilerinin (yani kamçılar), karar verme sensörlerinin ve kapalı döngü yaklaşımının açıklanması gerekmektedir. Çünkü E. Coli bakterisi bir tip sıçrayan arama sergilemektedir.

Kamçılar Aracılığıyla Yüzme ve Yuvarlanma:

E. Coli bakterisinin hareketi nispeten sert kamçı seti ile gerçekleşmekte ve bu kamçıların her birinin aynı yönde saniyede 100-200 devir dönmesiyle bakteri yüzebilmektedir. Kamçılar helezoniktir ve bir nevi pervane gibi düşünülebilir. Eğer kamçılar saat yönünde hareket ederse kamçı, hücreyi çekme yönünde bir kuvvet sergilemektedir. Kamçıların dönmesi için kuvvet oluşturan mekanizma biyolojik motor olarak adlandırılmaktadır. E. Coli bakterisi oldukça yüksek verime sahip bu motorları çalıştırmak için enerjisinin %1'inden daha azını kullanmaktadır.

E. coli bakterisi iki şekilde hareket edebilmektedir: bunlardan biri yüzme diğeri ise yuvarlanmadır. Eğer kamçılar saat yönünde dönerse, her bir kamçı hücreyi göreceli bağımsız bir şekilde çekmekte ve bunun sonucunda "yuvarlanma" adı verilen hareket gerçekleşmektedir. Yuvarlanma hareketi sonucunda bakteri bulunduğu ortamda rasgele bir yöne doğru dönmüştür. Bu hareket boyunca gerçekleşen zaman aralıklarına "yuvarlanma aralığı" denmektedir. Eğer bütün kamçılar saat yönünün tersine doğru dönerse, bu kuvvet bileşik bir pervaneye dönüşmekte ve hücre itilmeye başlamaktadır. Böylece bakteri "yüzme (koşma)" hareketini gerçekleştirmektedir. E. Coli bakterisi saniyede 10-20 μm/s hızda ya da yaklaşık vücut uzunluğunun on katı boyutunda hareket edebilmektedir. Dolayısıyla yaşayan bir organizmanın hareketi için bu oran oldukça hızlıdır. Yüzme hareketi süresince gerçekleşen zaman aralıkları "yüzme aralığı" olarak adlandırılmaktadır.

Kemotaksi:

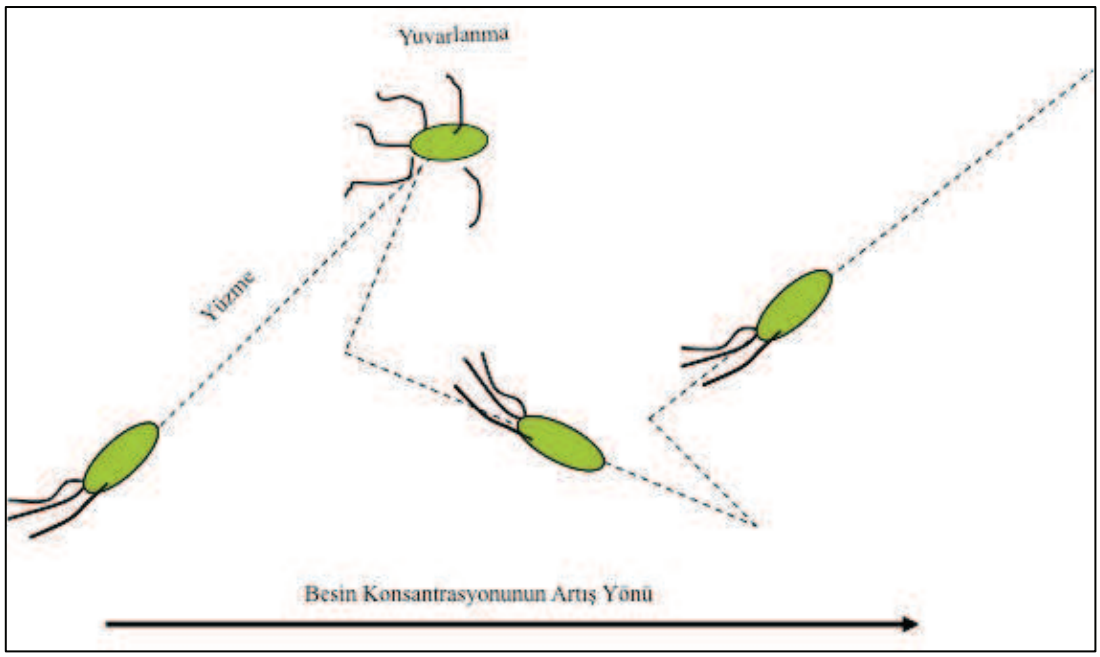
Kemotaksi, kimyasal itici ve çekicilerin olduğu ortamda bakterilerin ürettiği hareket modelleridir. Temel olarak E. Coli bakterisi kendisi için besin anlamına gelen maddeleri bulma ve zararlı maddelerden uzaklaşma çabasındadır. Mikroskop altında incelendiğinde E. Coli bakteri grubu zeki bir davranış sergilemekte ve özellikle grupça hareket etmektedir.

Kemotaksi hareketinin nasıl oluştuğunu anlamak için E. Coli bakterisinin ne kadar yüzeceğine nasıl karar verdiğini açıklamak gerekmektedir. Eğer E. Coli bakterisi besin ya da zararlı madde içermeyen nötr bir madde içindeyse, kamçılar eş zamanlı olarak saat yönünde ve saatin tersi yönünde dönecek ve böylece dönüşümlü olarak yüzme ve yuvarlanma hareketleri meydana gelecektir. İki hareket arasındaki bu değişim bakteriyi rasgele yönlerde hareket ettirir ve dolayısıyla bakteri besin araştırması yapabilir. Besin konsantrasyonunda artış olması durumunda bakteri yüzme işleminin süresini uzatır ve daha az yuvarlanma yapar. Aksi taktirde yüzmeyi kısa kesip yeni bir yuvarlanma hareketi gerçekleştirir. Aynı şekilde bakteri, kendisi için zararlı olan maddelerden uzaklaşmak amacıyla azalan konsantrasyona doğru ilerler.

Bakteri sabit besin konsantrasyonlu bölgeye ulaştığında, belirli bir süre sonra aynı yüzme ve yuvarlanma hareketleriyle besin araştırmasına geri döner. Bakteri etrafındaki besinden hiçbir zaman tatmin olmaz ve sürekli daha yüksek konsantrasyon için arayış

içindedir. Bu arayış sırasında hücre bir önce gözlemlediği konsantrasyon ile bundan üç önce gözlemlediği konsantrasyonu karşılaştırır ve bu veriyi ne kadar yüzmesi gerektiğini saptamak için kullanır (Passino, 2002).

Kemotaksi hareketi genel olarak Şekil 2.10'da gösterilmiştir:



Şekil 2.10: Kemotaksi hareket.

Karar Verme Mekanizması:

E. coli bakterisinin karar verme mekanizması, konsantrasyon farkını tespit eden bir çeşit hafıza sistemine sahiptir. Farkın tespiti, bakteri gövdesinin ön ve arka kısmında bulunan iki farklı sensörün elde ettiği değerlerin farkının alınması şeklinde elde ediliyor gibi görünse de; deneyler, konsantrasyon eğiminin hafızada tutulan bir önceki değerle şimdiki değerin karşılaştırılması sayesinde bulunduğunu göstermektedir. Aslında iç bakteriyel karar verme süreci, bazı integral geri bildirim kontrol mekanizmalarını içermektedir.

Sonuç olarak, hafızaya eklenmiş bir çeşit mekanizma, karşılaştırma yapabilme yeteneği, birkaç basit iç kontrol kuralı ve kimyasal algılamayla ve hareket yeteneğiyle, bakteri karmaşık bir arama ve sakınma hareketi gerçekleştirmektedir.

Eliminasyon ve Dağılma:

Bakterilerin yaşadığı bölgesel alanlarda bazı etkiler sonucu kademeli veya ani değişiklikler olabilmektedir. Bu değişiklikler bir bölgedeki tüm bakterilerin ölmesine ya da bir grup bakterinin başka bölgeye taşınmasına neden olabilir. Böyle bir durum, muhtemelen kemotaksi sürecini bozar; ancak aynı zamanda kemotaksiyi destekler de. Çünkü dağılma, bakterileri daha iyi besin kaynağına yaklaştırabilmektedir.

Tüm bunlar çerçevesinde, bakteriyel besin arama algoritmasının sözde kodu Şekil 2.11'deki gibidir:

BAKTERİYEL BESİN ARAMA ALGORİTMASI

Algoritmanın başlangıç değerlerini belirlemek için, p, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, d_s, ve $C(i)$, i = 1, 2, ..., S değişkenlerinin değerlerinin atanması gerekir. Eğer sürü zekası adımı da kullanılacaksa $\Gamma_{attract}$, $\upsilon_{attract}$, $Y_{repellant}$ ve $\upsilon_{repellant}$ katsayılarının değerleri de belirlenmelidir. BBAA optimizasyon süreci, bakterileri çözüm uzayına rastgele dağıtarak başlar. Döngülerde sayaç olarak kullanılan değişkenler, $j = k = l = 0$ başlangıç değerlerini alırlar.

1) Eliminasyon ve dağılma döngüsü: $l = l+1$
2) Çoğalma döngüsü: $k = k+1$
3) Kemotaksis döngüsü: $j = j+1$
   a) i = 1, 2, ..., S, için, i. bakteri için bir kemotaktik adım başlat.
   b) $J(i, j, k, l)$'yi hesapla. $J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j, k, l), P(j, k, l))$
   (hücreden hücreye çekici etkisini besin konsantrasyonuna ekle.)
   c) $J_{last} = J(i, j, k, l)$ atamasını gerçekleştir ve yüzme işlemi aracılığı ile daha iyi bir sonuç elde edinceye kadar bu değeri sakla.
   d) Yuvarlanma işlemini gerçekleştir: $\Delta_m(i)$, m = 1, 2, ..., p olmak üzere [-1, 1] aralığında rasgele bir $\Delta(i) \in R^p$ vektörü oluştur.
   e) Bir adım hareket et:

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

   Atamasını gerçekleştir. Bu sonuç, i. bakteri için yuvarlanma yönünde C(i) adım büyüklüğü kadardır.
   f) $J(i, j+1, k, l)$'yi hesapla. $J(i, j+1, k, l) = J(i, j+1, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l))$
   g) Yüzme işlemini gerçekleştir.
      i) Yüzme uzunluğunun sayacı olmak üzere $m = 0$ ilk değerini ata.
      ii) $m < N_s$ iken
         - $m = m+1$ atamasını gerçekleştir.
         - Eğer $J(i, j+1, k, l) < J_{last}$ ise, $J_{last} = J(i, j+1, k, l)$ ve

$$\theta^i(j+1, k, l) = \theta^i(j+1, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

         atamalarını gerçekleştir. Yeni $J(i, j+1, k, l)$ değerini hesaplamak için bu $\theta^i(j+1, k, l)$ değerini kullan.

Şekil 2.11: Bakteriyel besin arama algoritması sözde kodu (Abraham ve diğ., 2008).

- $m = N_s$ ise bu bakteri için yüzme sürecini durdur.

h) Son bakteri değilse ($i \neq S$) bir sonraki bakteriyi seç ($i=i+1$) ve b maddesine git.

4) Eğer $j<N_c$ ise bakterilerin yaşam süresi bitmemiştir 3. adıma tekrar dön ve kemotaktik adımlara devam et.

5) Çoğaltma işlemi:

a) Mevcut k ve l değerleri ve her bir $i = 1, 2, ..., S$, değeri için,

$$J_{health}^{i} = \sum_{j=1}^{N_{c+1}} J(i,j,k,l)$$

değerlerini hesapla ve atamalarını yap. Sonuçta elde edilen Jhealth değerlerine göre bakterileri ve kemotaktik parametrelerini küçükten büyüğe doğru sırala.

b) Sıralamanın alt yarısındaki sağlık durumları (J_{health}^{i} değerleri) diğerlerine göre daha kötü olan $S_r = S/2$ adet bakteri ölmüş olarak kabul edilir. Bu durumda daha iyi olan ve sıralamanın üst kısmında kalan Sr adet bakteri bölünerek çoğaltılır. Ebeveynleri ile aynı konumlara yerleştirilecek şekilde, yeni nesil bakteri çiftlerinden her bir tanesi listenin alt kısmındaki ölen bakterilerin yerine geçirilir.

6) Eğer $k<N_{re}$ ise belirlenen çoğaltma üst sınırına ulaşılmamıştır. 2. adıma geri dön ve sonraki nesil ile kemotaktik adımları yerine getir.

7) Eliminasyon ve Dağılma işlemi: $i = 1, 2, ..., S$, dizisi içerisindeki bütün bakterileri p_{ed} olasılığına bağlı kalarak eliminasyon ve dağılma işlemine tabi tut. Bu işlem, popülasyon adedini sabit tutmak şartıyla bir bakterinin basit bir şekilde bulunduğu yerden rasgele yeni bir yere atılması ile gerçekleştirilir.

8) Eğer $l<N_{ed}$ ise 1. adıma git, aksi takdirde programı sonlandır.

Şekil 2.11 (devam): Bakteriyel besin arama algoritması sözde kodu (Abraham ve diğ., 2008).

2.2.3.8. Maymun Arama Algoritması

Maymun arama algoritması etmen (agent) bazlı bir arama algoritması olup, çözüm olarak ağacın dalları etmen tarafından tespit edilmektedir. Bu yaklaşım maymunların davranışlarının birebir kopyası olmamakla birlikte, ajanların hareketi maymunların besin aramak için ağaçlarda dolaşmasına benzemektedir.

Algoritmaya göre, bir maymun ağaca ilk tırmandığında rastgele olarak dalları seçmekte ve bu şekilde dalları keşfetmektedir. Besinlerin olduğu bir dal bulunduğunda, maymun bu dalı köke bağlayan tek bir yoldan aşağıya inmekte ve bu yolu, sonuç (iyi besinlerin olması durumu) ile birleştirmektedir. Daha sonra maymun, yüksek olasılıklı önceden işaretlenmiş dalları rastgele seçip deneyimlerinden faydalanarak ağaca tekrar tırmanmakta ve iyi değerlerin olduğu (besinlerin olduğu) dallara doğru ilerlemektedir. Bu süreç, maymun ağacın keşfedilmemiş sınırına ulaşıncaya kadar devam etmektedir. Böylelikle maymun iyi besin kaynağına bağlı diğer dalları keşfedebilmekte ve böylece global çözümü bulma şansı yükselmektedir.

Ağacın dalları olası tüm çözümleri içermektedir. Kökteki herhangi bir daldan başlayarak, aynı dalın ucundaki yeni komşu sonuç verilir. İki sonuç arasındaki işlevsel uzaklık, rastgele karışıklık (random perturbation) ile tanımlanmaktadır. Ağaçlar ikili ağaç yapısındadır, yani her bir daldan komşu çözümlü iki yeni dal çıkmaktadır. Maymun, daha iyi bir amaç fonksiyonu değeri bulduğu her seferde tırmanmayı durdurmakta ve bu değeri o anki en iyi sonuç olarak saklamaktadır. Aramanın durmasına neden olan bir diğer durum da maymunun ağacın en uç noktasına ulaşmış olmasıdır. Herhangi bir durma durumu gerçekleştiğinde maymun dal dal aşağıya inmekte ve bu sırada diğer dallarda olabilecek iyi sonuçlara bakmaktadır. Aşağı inme sırasında ayrıca sonraki keşiflerinde kullanmak üzere, geçtiği tüm dalları güncellemekte ve işaretlemektedir.

Algoritmanın her bir adımında, var olan çözümden başlayarak ve var olan çözüme uygulandığında yeni çözümler üreten rastgele karışıklık fonksiyonu kullanılarak yeni olası çözümler yaratılmaktadır. Karışıklık fonksiyonunu tanımlamak için diğer metasezgisel yaklaşımlardan faydalanılmıştır. Bunlardan biri genetik algoritmalarda kullanılan çaprazlama işlemidir. Bununla birlikte karışıklık fonksiyonu, optimizasyon probleminin kendi yapısına bağlı olarak da türetilebilmektedir (Mucherino ve Seref, 2007).

2.2.3.9. Ateş Böceği Algoritması

Ateş böceği algoritması, Xin-She Yang (2007) tarafından geliştirilen ve tropikal iklim bölgelerindeki ateş böceklerinin sosyal davranışlarını baz alan bir metasezgisel optimizasyon algoritmasıdır.

Ateş böceklerinin yaklaşık iki bin türü bulunmakta ve bunların çoğu kısa ve ritmik ışıklar yaymaktadır. Bir ateş böceği ışıklarını yakıp söndürmesinin iki amacı bulunmaktadır. Bunlardan biri eş bulmak amacıyla diğer ateş böceklerini çekmek, diğeri ise olası avlarını etkileyip avcılarından da kendilerini korumaktır. Ritmik ışık, yanıp sönme oranı ve sinyal sisteminin zaman miktarı her iki cinsi bir araya getirmektedir. Dişiler, aynı tür içinde bir erkeğin benzersiz yanıp sönmesine cevap verirken; photuris gibi bazı türlerde, kadın ateş böcekleri diğer türlerin yanıp sönmelerini taklit ederek onları cezbetmekte ve uygun eş bulduğu düşüncesine kapılan erkek ateş böceklerini yemektedir.

Bir ışık kaynağından belirli bir r mesafedeki ışık yoğunluğunun (I) ters kare yasasına uyduğu bilinmektedir. Yani başka bir deyişle, $I \propto 1/r^2$ 'ye göre; r mesafesi arttıkça ışık yoğunluğu azalmaktadır. Ayrıca hava, mesafe arttıkça daha da zayıflayan ışığı emmektedir. Bu iki bileşik faktör çoğu ateş böceklerini sadece sınırlı bir mesafede, genellikle geceleri yüzlerce metrede görünür yapmakta ve bu ateş böceklerinin iletişimi için yeterli olmaktadır.

Yanıp sönen ışık, bir şekilde ilişkili olduğu amaç fonksiyonunu optimize etmek için formüle edilebilmektedir. Ateş böceği algoritmasında da, verimli optimal çözümler elde etmek için, verilen bir optimizasyon probleminin amaç fonksiyonu, ateş böceği sürüsüne parlak ve daha çekici yerlere gitmede yardım eden yanıp sönen ışık ya da ışık şiddeti ile ilişkili olmaktadır.

Algoritma üç varsayıma dayanmaktadır:

1. Bütün ateş böcekleri tek cinstir (unisex), dolayısıyla erkek ve dişi ayrımı olmadığı gibi, bütün ateş böcekleri diğer ateş böcekleri tarafından cezbedilebilir.

2. Ateş böceklerinin çekiciliği, parlaklıkları ile doğru orantılıdır. Aynı zamanda mesafe, parlaklığı azalttığı için cazibeyi de azaltmaktadır. Şayet bir ateş böceğinden daha parlak ateş böceği varsa, bu ateş böceği parlak olana doğru hareket edecektir. Olmaması durumunda ise ateş böceği rastgele yönlerde hareket edecektir.

3. Bir ateş böceğinin parlaklığı amaç fonksiyonu tarafından etkilenir veya belirlenir. Bir maksimizasyon problemi için, parlaklık sadece amaç fonksiyonu değerine orantılı olabilir. Parlaklığın diğer formları ise genetik algoritmadaki uygunluk fonksiyonuna benzer şekilde tanımlanabilir (Yang, 2009b; Yang, 2010c).

Bu üç kural doğrultusunda, ateş böceği algoritması temel adımları Şekil 2.12'de gösterilen sözde kod ile özetlenebilir:

ATEŞ BÖCEĞİ ALGORİTMASI

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_d)^T$
Ateşböcekleri için ilk popülasyon x_i (i =1,2,...,n)'yi oluştur
x_i böceğindeki parlama yoğunluğu I_i'yi $f(x_i)$ ile tanımla.
Işık emme katsayısı γ belirle.
while t < *maksimumüretim*
for i = 1 : n
    for j=1: i
        if $I_i < I_j$, i ateşböceğini j ateşböceğine doğru hareket ettir.
        end if
        Uzaklık r 'ye bağlı olarak exp [-γ r] yoluyla çekicilik değerini değiştir.
        Yeni sonuçları değerlendir ve ışık yoğunluğunu güncelleştir.
    end for j
end for i
Ateşböceklerini sırala ve en iyi sonucu bul.
end while
Sonuçları ve görselleri işle.

Şekil 2.12: Ateşböceği algoritmasının sözde kodu (Yang, 2010b).

2.2.3.10. Guguk Kuşu Arama Algoritması

Guguk kuşu arama algoritması, 2009 yılında Xin-She Yang ve Suash Deb tarafından geliştirilmiş doğadan esinlenen bir metasezgisel algoritmadır (Yang ve Deb, 2010). Bu algoritma belirli türdeki guguk kuşlarının türlerini iyileştirmek/üretmek için geliştirdikleri kuluçka asalaklığı hareketinden esinlenerek geliştirilmiştir. *"Ani"* ve "*Guira"* gibi bazı türdeki guguk kuşları, yumurtalarını ortak yuvalara koymakta ve yumurtaların kuluçkadan çıkma olasılığını böylece artırmaktadır. Ayrıca birçok tür, genellikle başka türlerdeki ev sahibi kuşların yuvalarına yumurtalarını yatırmakta ve kuluçka asalaklığını mecbur kılmaktadır.

Üç tip kuluçka asalaklığı söz konusudur: Kendi türündeki kuşların yuvalarına yumurtalarını yatırma (intraspecific brood-parasitism), farklı türdeki kuşların yuvalarını kullanma (interspecific brood-parasitism/cooperative breeding) ya da yuvaları ele geçirme (nest take over)

"The New World brood-parasitic Tapera" gibi bazı guguk kuşu türlerinde, dişi asalak guguk kuşu, seçilen birkaç ev sahibi türün yumurtalarının rengini ve dokusunu taklit etme açısından gelişmiş ve evrimleşmiştir. Bu, yumurtaların terkedilme olasılığını düşürmekte ve üretkenliği artırmaktadır.

Bunların dışında, bazı türlerin yumurtalarını yatırma zamanı da ilginçlik göstermektedir. Asalak guguk kuşları, genellikle ev sahibi kuş kendi yumurtalarının üzerine yattığı zaman yuvaları seçmektedirler. Genellikle guguk kuşlarının yavruları, ev sahibi kuşun yavrularına göre daha önce yumurtadan çıkmaktadır. İlk guguk kuşu yumurtadan çıktığında içgüdüsel ilk hareketi ev sahibi kuşun yumurtalarını yuvadan iterek atmaktır. Böylece yiyecek paylaşım oranını artırmış olur. Ayrıca yapılan araştırmalar göstermektedir ki guguk kuşu yavruları, ev sahibi kuşun yavrularının ötüşünü taklit edebilmekte ve daha fazla beslenme şansı elde edebilmektedir. (Yang ve Deb, 2009).

Guguk kuşu arama algoritması aşağıdaki şekilde kurulmaktadır:

1. Her bir guguk kuşu belirli bir zamanda bir yumurta yumurtlamakta ve bu yumurtayı rastgele seçilen bir yuvaya bırakmaktadır.

2. Bir nesil, ev sahibi yuvaların kümesi olarak tanımlanmaktadır.

3. Yüksek kalitedeki yumurtalarla en iyi olan yuva gelecek nesilleri oluşturacaktır. Yani her bir nesilden en iyi yumurtalarla gelecek nesiller meydana getirilir.

4. Yumurtaların kalitesi, rasgele seçilen bir yuvadan alınan yumurta ile yeni bir yumurta üreterek geliştirilir. Yeni yumurta, seçilen yumurtanın rastgele hareketi ile meydana gelir. Eğer yeni yumurta diğer rastgele seçilen yuvadaki yumurtadan daha üstün ise, eski yumurta yenisi ile yer değiştirir.

5. Mümkün olan tüm ev sahibi yuvaların sayısı her bir nesilde sabittir ve guguk kuşu tarafından yumurtlanan yumurtanın ev sahibi kuş tarafından bulunma olasılığı $p_a \in [0, 1]$'dir. Bu durumda ev sahibi kuş ya yumurtayı yuva dışına atabilir ya da yuvayı terkedip baştan yeni bir yuva oluşturabilir. Bu son varsayım yeni yuvalar (yeni rasgele çözümler) ile yerleştirilen n adet yuvanın p_a kesiri ile tahmin edilebilir. p_a olasılığı ile, rasgele yeni üretilen yumurta, en düşük sıralanmış yuvadaki yumurta ile yer değiştirir. Böylece yerel minimumlardan kaçınılır (Kumar ve Chakarverty, 2011).

Guguk kuşu arama algoritmasının sözde kodu Şekil 2.13'te yer almaktadır.

GUGUK KUŞU ARAMA ALGORİTMASI

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_d)^T$
n adet ev sahibi yuva için ilk popülasyon x_i (i =1,2,...,n)'yi oluştur
while $t <$ *maksimumüretim* ya da $t <$ *durdurma kriteri*
    Rastgele bir guguk kuşu (i) al
    Bunun kalitesini/uygunluğunu (F_i) hesapla
    n adet yuva arasından rastgele birini (j) seç.
        if $F_i < F_j$ ise,
        j'yi yeni çözüm ile değiştir.
        end
    Kötü yuvaların bir bölümünü bırak ve yenileri oluştur.
    En iyi sonuçları (ya da kaliteli sonuçlara sahip yuvaları) sakla.
    Sonuçları sırala ve mevcut en iyiyi bul.
end while
Sonuçları ve görselleri işle.

Şekil 2.13: Guguk kuşu arama algoritmasının sözde kodu (Yang, 2010b).

2.2.3.11. Kedi Sürüsü Optimizasyonu

Kedilerin Davranışları:

Biyolojik sınıflandırmaya göre kedigiller içinde; aslan, leopar, kaplan, kedi vb. gibi yaklaşık 30 tane farklı tür bulunmaktadır. Çoğunun farklı yaşam alanı olmasına rağmen, kedigiller benzer davranış modellerini sergilemektedir.

Kedigillerin avlanma becerisi kalıtsal değildir, alıştırmalar aracılığıyla kazanılmaktadır. Bu avlanma becerisi yaban kedileri için hayatta kalmalarını sağlarken, evcil kedilerde benzer doğal avlanma becerisi hareketli nesnelere güçlü bir merakı uyandırmaktadır. Aslında bu güçlü merak bütün kedilerde olmasına rağmen, kediler zamanlarının çoğunu hareketsiz olarak geçirmektedirler.

Avlanma becerilerinin yanında kediler çok yüksek seviyede atikliğe de sahiptirler. Bu atiklik dinlenme zamanlarında bile kedilerin etraflarını sürekli gözetlemelerine neden olmaktadır. Ayrıca kediler, çok zeki ve planlı yaratıklar oldukları halde tembel gibi görünmektedirler (Chu ve diğ, 2006). Kedilerin dokuz canlı olduğu söylenerek, kedilerin güçlü canlılığına gönderme yapılmaktadır. Evcil kediler sık sık alçak frekansta ses çıkarmaktadırlar. Kediler hoşnut oldukları, tehlike hissettikleri veya hasta oldukları zaman mırlarlar. Alçak frekans mırlamanın, hücre onarımına yardım ettiğine inanılmaktadır ve bu kedilerin canlılık nedeni olarak görülmektedir (Chu, Tsai, 2007).

Kedi Algoritmasının İçeriği:

Kedi Sürü Optimizasyonu, kedilerin başlıca iki davranışsal özelliği olan "arama modu" ve "izleme modu"nun birleşimi olarak modellenmiştir.

Arama modu, dinlenmekte olan fakat sonraki hareketi için çevresinde aranıp tetikte olan bir kedinin modellenmesi için kullanılmaktadır. Arama modunda dört gerekli faktör yer almaktadır. Bunlar arama hafızası havuzu (*AHH*), seçilen boyutun arama aralığı (*SBAA*), değişen boyutların sayısı (*DBS*) ve kendi pozisyonunu değerlendirmedir (*KPD*). *AHH* her bir kedinin arama hafızasının boyutunu tanımlamak için kullanılır ve kedi tarafından aranan noktaları belirtir. *SBAA* seçilen boyutlar için mutasyon oranını bildirir. Arama modu süresince, eğer bir boyut mutasyon için seçilmişse, yeni ve eski değerler arasındaki farklılık *SBAA* tarafından tanımlanan aralık dışında olmamalıdır. *DBS* boyutlardan kaç tanesinin değişime uğrayacağını ifade eder. *KPD ise* bilinen bir ikili değerdir, ve kedilerin bulunduğu noktanın hareket için aday noktalardan biri olup olmayacağını bildirir. *KPD*, *AHH* değerini etkilemez.

Arama modunun çalışma adımları aşağıda yer almaktadır:

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

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

Adım 3: Bütün aday noktaların uygunluk değerini (*UD*) hesapla.

Adım 4: Eğer bütün uygunluk değerleri tam olarak aynı değilse, her bir aday noktanın seçilme olasılığını denklem 2.3'e göre hesapla. Aksi takdirde her bir aday noktanın seçilme olasılığını 1 olarak ata.

$$P_i = \frac{|UD_i - UD_b|}{UD_{maks} - UD_{min}} \quad ; \qquad 0 < i < j \tag{2.3}$$

Adım 5: Hareket noktasını aday noktalardan rastgele çıkart ve $kedi_k$' nin pozisyonuyla değiştir.

Eğer uygunluk fonksiyonunun amacı minimum çözümü bulmaksa, $UD_b=UD_{maks}$, aksi takdirde $UD_b=UD_{min}$ *olmalıdır.*

İzleme modu, kedinin hedefi izlemedeki durumunu modellemek için bir alt modeldir. Kedi izleme moduna girdiğinde, her bir boyutun kendi hızlarına göre kedi hareket etmektedir. İzleme modundaki çalışma aşağıdaki gibidir:

Adım 1: Bütün boyutlar için hızları ($v_{k,d}$) olarak denklem 2.4'ü kullanarak güncelle.

$$v_{k,d} = v_{k,d} + r_1 \times c_1 \times \left(x_{best,d} - x_{k,d}\right) \quad ; \quad d = 1,2,\dots,M \tag{2.4}$$

Adım 2: Hızların, maksimum hız aralığında olduğunu kontrol et. Yeni hızın aralığın üzerinde olması durumunda limite eşitle (*limit=sınır*).

Adım 3: $kedi_k$'nin pozisyonunu denklem 2.5'i kullanarak güncelle.

$$v_{k,d} = v_{k,d} + v_{k,d} \tag{2.5}$$

$x_{best,d}$, en iyi uygunluk değerine sahip kedinin pozisyonu; $x_{k,d}$, $kedi_k$'nin pozisyonu, c_1 bir sabiti, r_1 ise gelişigüzel bir değeri göstermektedir (Chu ve diğ,2006; Chu ve Tsai, 2007).

Kedi Sürüsü Optimizasyonunun Çalışması:

Daha önce de belirtildiği gibi kedi sürüsü optimizasyonunun arama modu ve izleme modu olarak iki alt modeli bulunmaktadır. Bu iki modu algoritma şeklinde birleştirmek için, bir karışım oranı (*KO*) tanımlanmaktadır. Kedilerin, zamanlarının çoğunu dinlenmeye ve gözetlemeye (mesela zamanlarının çoğu arama modunda geçmektedir) harcadığını garantilemek için *KO'*ya çok küçük bir değer atanmaktadır. Buna göre optimizasyon sürecinin genel adımları şu şekildedir:

Adım 1: Süreçte *N* tane kedi yarat.

Adım 2: Kedileri *M* boyutlu çözüm uzayına rastgele serpiştir ve her kedinin hızına, maksimum hız aralığında olan rastgele değerler ver. Ardından bir kısım kediyi gelişigüzel ayır ve bunları *KO*'ya göre izleme modunun içine ata. Diğerlerini ise arama modunun içine ata.

Adım 3: Kedilerin konumlarını amaç kriterini yansıtan fonksiyona koyarak uygunluk değerini hesapla ve en iyi kediyi hafızada sakla.

Adım 4: Kedileri hareket ettir. Eğer $kedi_k$ arama modundaysa, kediye arama modu sürecini uygula, aksi takdirde izleme modu sürecini uygula.

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

Adım 6: Sonlandırma koşullarını kontrol et. Eğer yeterliyse programı sonlandır, aksi durumda Adım 3'ten Adım 5'e kadar tekrar et.

2.2.3.12. Kurbağa Sıçrama Algoritması

Kurbağa sıçrama algoritması (KSA), sezgisel fonksiyon kullanarak bilinçli bir sezgisel arama yapan ve küresel en iyi çözümü aramak için tasarlanmış bir memetik metasezgiseldir. Çok yeni olarak ele alınan KSA, ilk olarak Eusuff ve Lansey'in (2003) ortaya koydukları "Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm" adlı çalışma ile literatüre kazandırılmıştır. Algoritma, interaktif bireyler tarafından taşınan memlerin (kültürel iletim birimi-sosyolojik gen) evrimine ve bilginin popülasyon arasında küresel değişimine dayanmaktadır (Eusuff ve diğ., 2006).

Kurbağa Sıçrama Algoritması, kurbağaları memetik evrimde dönüştürerek ilerlemektedir. Bu algoritmada bireysel kurbağalar önemli olmamakta, bunun yerine memler için ev sahibi olarak görülmekte ve bir memetik vektör olarak açıklanmaktadır. Her mem bir dizi mem-bilgisi (memotype) içermektedir. Mem-bilgisi, genetik algoritmada bir kromozomdaki özelliği temsil eden bir gen ile benzer amacı temsil etmektedir. KSA bir bireyin fiziksel özelliklerini değiştirmemekte, tam tersine artan bir biçimde sanal popülasyondaki her kurbağa tarafından desteklenen düşünceleri geliştirmektedir. Sanal popülasyon, genetik algoritmadaki kromozom havuzunu temsil eden popülasyona benzer bir şekilde, bir çok farklı kurbağadan oluşan mem havuzunu modellemek için kullanılmaktadır.

KSA'nın genel benzetimi şu şekilde yapılabilmektedir: Bir göl üzerinde bir grup kurbağanın bulunduğunu ve gölde farklı noktalarda kurbağaların sıçrayabileceği bir dizi taş olduğunu varsayalım. Kurbağaların amacı, memlerini geliştirerek, olabildiğince hızlı

bir şekilde en yüksek miktarda besin sağlayacakları taşı bulmaktır. Kurbağalar yalnızca besine olan uzaklık değerlerini bilmekte ve sahip oldukları bu mesafe bilgisi, onların mem-bilgilerini oluşturmaktadır. Kurbağalar birbirleriyle iletişimleri sayesinde, kendi mem-bilgilerini geliştirebilmekte ve bunun sonucunda bir taş üzerinden imkanlar dahilinde diğer taşa zıplayabilmektedir. Mem-bilgisinin bu şekilde iyiye doğru güncellenmesi de besine yaklaşma anlamına gelmektedir (Eusuff ve Lansey, 2003).

KSA bünyesinde hem deterministik hem de rastlantısal yaklaşımları bir arada bulundurmaktadır. Deterministik yaklaşım, sezgisel arama yapabilmek için tepki yüzeyi bilgilerini kullanırken; rastlantısal yaklaşım, esnekliği ve yerel en iyi noktalarda toplanmayı önleyerek arama sağlamlığını sağlamaktadır.

KSA'nda arama rastgele bir kurbağa popülasyonu seçilmesiyle başlamaktadır. Daha sonra bu kurbağa popülasyonu, farklı yönlerde arama yapmak için bağımsızca yayılmaya izinli birçok paralel topluluğa (mempleklere) ayrılmaktadır. Her bir mempleks içerisindeki kurbağalar, diğer kurbağalardan etkilenmekte ve böylece memetik evrime tabi olmaktadır. Memetik evrim, bireylerin sahip olduğu memlerin kalitesini yükseltmekte ve hedefe doğru her bireyin performansını artırmaktadır. Etkileşim sürecinin rekabetçi olduğundan emin olmak için, daha iyi memlere sahip bir kurbağanın zayıf fikirli kurbağaya göre yeni fikirlerin gelişmesine daha fazla katkıda bulunması gerekmektedir. Bu nedenle, kurbağaların seçiminde üçgen olasılık dağılımının kullanılması bir avantaj sağlamaktadır. Evrim süresince kurbağalar memlerini, en iyi mempleks bilgisine veya popülasyonun en iyisi bilgisine göre değiştirebilmektedir. Mem-bilgisi içindeki artan değişiklikler, sıçrama adım boyutuna ve yeni mem de kurbağanın yeni konumuna karşılık gelmektedir. Bir birey, konumunu geliştirdikten sonra, tekrar popülasyona döndürülmektedir. Konumdaki bir değişiklikten elde edilen bilgi daha fazla geliştirilmek için hemen kullanılabilmektedir.

Mempleksler içinde yapılan, belirli sayıdaki memetik evrimden sonra, mempleksler birleştirilmekte ve bir karıştırma süreciyle yeni mempleksler oluşturulmaktadır. Bu karıştırma işlemi, farklı bölgelerindeki kurbağaların katılması nedeniyle mem kalitesini artırmaktadır. Kurbağa göçü (fikir ve/veya tasarımların çapraz döllenmesi), deneyimleri paylaşarak etkileme şeklinde arama işlemi hızlandırmakta ve herhangi bir bölgesel eğilimden arınmış kültürel evrimi sağlamaktadır (Eusuff ve diğ., 2006).

2.2.3.13. Yarasa Algoritması

Yarasa algoritması 2010 yılında "A New Metaheuristic Bat-Inspired Algortihm" adlı çalışmasıyla Xin-She Yang tarafından ortaya atılmıştır (Yang, 2010d). Algoritmanın temeli küçük yarasaların sahip olduğu ekolokasyon özelliğine dayanmaktadır. Bu tez çalışmasında yöntem olarak kullanılacak yarasa algoritmasının varsayımları, akış şeması ve işleyişi tezin malzeme ve yöntem bölümünde detaylı anlatılacaktır. Ancak algoritmanın temelini oluşturan yarasaların genel davranışları ve kullandıkları akustik özelliğine bu bölümde yer verilecektir. Ayrıca tezin, literatürde sıkça kullanılan belge setlerini yarasa algoritması ile kümeleyen ilk çalışma olduğunu vurgulamak adına, yarasa algoritması ile yapılmış literatürde yer alan diğer çalışmalar kapsamlı bir şekilde bölüm 2.3'te verilecektir.

Yarasaların davranışı:

Yarasalar kanatlı hayvanlar içinde tek memeli tür olup, ses yankısıyla arama yetenekleri gelişmiş hayvanlardır. Yarasaların yaklaşık 996 farklı türü olduğu tahmin edilmekte ve bu sayı, tüm memeli türlerinin %20'sine denk gelmektedir. Yarasaların boyutu küçük yaban arısı yarasalardan (yaklaşık 1,5 ile 2 g.) kanat açıklığı yaklaşık 2 m. ve ağırlığı yaklaşık 1 kg kadar olan dev yarasalara kadar değişmektedir. Ancak yarasaların genellikle ön kol uzunluğu 2,2 cm ile 11 cm arasındadır.

Birçok yarasa türü böcekçildir. Yarasalar; avlarını ve avlarının bulundukları yeri tespit etmek, avlarını sınıflandırmak, engellerden kaçınmak ve karanlıkta kendi tüneme yarıklarını bulmak için *ekolokasyon* (*echolocation*) adı verilen bir tür sonar kullanmaktadırlar. Bu özellik sayesinde yarasalar, çok yüksek frekanslı ses sinyalleri yaymakta ve çevredeki nesnelerden geri yansıyan ses yankılarını dinlemektedirler. Ekolokasyon sırasında beynin görsel kısmı çalışmaktadır. Buna göre yarasalar cisimlerin şekil ve uzaklıklarını, hareketli veya sabit olduklarını, yankının dal, yeşillik gibi istenmeyen hedeflerden gelip gelmediğini tespit etmektedirler (Schnitzler ve Kalko, 2001).

Birçok yarasa türü, ekolokasyon özelliğini belli bir dereceye kadar kullanırken; kullandıkları yaklaşım farklı olabilmektedir. Bazı türler dil tıklamalarını sinyal olarak kullanırken diğerleri ses telleri yardımıyla sinyal yaymaktadırlar. Uçan ya da su yüzeyinde hareket eden böcekleri avlayan türler yüksek şiddetli (110 dB'den fazla) ses

sinyallerini kullanırken, diğer türler alçak şiddetli (60-80 dB) sinyaller üretmektedirler (Fenton, 2004).

Yarasaların sinyalleri, türlerine dayalı olarak değişkenlik göstermekte ve onların avcılık stratejileri ile ilişkili olabilmektedir. Birçok yarasa türü kısa, frekans-ayarlı sinyalleri kullanırken; diğerleri daha çok sabit frekanslı sinyalleri kullanmaktadır.

Akustik:
Birçok yarasa türü aynı andaki sinyal ve yankıyı ayırt edebilmekte ancak giden sinyal ile gelen yankı arasındaki örtüşmeyi tolere edememektedir. Bu türler zamanın yaklaşık %10'u kadar sinyal oluşturmakta, yani çağrılar arasındaki aralıklar oldukça kısa olmaktadır. Diğer türler ise zamanın %40-80'i kadar sinyal oluşturarak sinyal ile yankıyı frekans olarak ayırabilmektedir (Fenton, 2004).

Yarasaların birçoğu için tipik frekans aralığı 25kHz ile 100kHz arasındaki bölgedeyken, bazı türler 200 kHz'e kadar yüksek frekanslarda sinyal yayabilmektedirler. Her bir ultrasonik patlama, genellikle 5 ile 20 ms. sürebilmekte ve yarasalar her bir saniyede bu ses patlamalarından 10 ile 20 tane yaymaktadırlar. Avlanma sırasında ise, yarasalar avlarına yakın uçarken sinyal emisyon oranı saniyede yaklaşık 200 sinyale kadar hızlandırılabilmektedir. Bu kısa ses patlamaları, yarasaların inanılmaz sinyal işleme yeteneğini belirtmektedir.

Havadaki sesin hızı v = 340 m/sn olduğuna göre, sabit frekans f ile ultrasonik ses patlamalarının dalga boyu λ, denklem 2.6 ile hesaplanabilmektedir:

$$\lambda = \frac{v}{f} \tag{2.6}$$

25 kHz ile 150 kHz frekans aralığı için dalga boyu 2 mm ile 14 mm arasında olacaktır. Bu dalga boyları aynı zamanda yarasaların avlarının boyutları ile orantılıdır.

Küçük dalga boyları yüksek frekanstan ötürü av hakkında daha fazla bilgi verirken; yüksek frekans, atmosferik zayıflamadan ötürü bir dezavantaj olabilmektedir. Atmosferik zayıflama ve yayılma kaybı, havada ses dalgalarının hareketini sınırlamak üzere birleşerek ekolokasyon etkililik aralığını azaltmaktadır.

Yarasalar tarafından yayılan sinyal 110 dB kadar yüksek olabilmekte ve bu değer ultrasonik bölgede bulunmaktadır. Ses yüksekliği (loudness) av ararken ne kadar yüksekse ava doğru yönelirken o kadar düşüktür. Bu tür kısa sinyallerin hareket aralığı gerçek frekanslara bağlı olarak bir kaç metredir. Dolayısıyla yarasalar insan kılı kadar ince ve küçük engellerden kaçmayı başarabilmektedirler.

Çalışmalar, yarasaların yankı yayma ve algılamadan kaynaklanan zaman gecikmesini, iki kulakları arasındaki zaman farkını ve çevrelerinin üç boyutlu senaryolarını oluşturmak amacıyla yankıların ses yüksekliği varyasyonlarını kullandıklarını göstermektedir. Böylece yarasalar hedeflerinin mesafe ve yönünü, avlarının türünü ve hatta avları olan küçük böceklerin hareket hızını algılayabilmektedirler. Ayrıca hedef böceklerin kanat çırpma oranları ile oluşturulan Doppler etkisi varyasyonları ile yarasaların hedefleri ayırt etmesi mümkün görünmektedir.

Bazı yarasaların görme duyuları iyi olmakla birlikte birçok yarasanın çok hassas koku duyusu bulunmaktadır. Gerçek hayatta yarasalar tüm bu duyularının birleşimini, etkin algılama ve düzgün yön tayinini üst düzeye çıkarmak için kullanmaktadırlar. Ancak, yarasa algoritmasında sadece ekolokasyon ve bu özellikle ilişkilendirilmiş davranış biçimi dikkate alınmıştır.

2.3. YARASA ALGORİTMASININ UYGULAMA ALANLARI

Standart yarasa algoritmasının pek çok avantajı vardır ve en önemli avantajlarından biri, başlangıç aşamasında çok hızlı yakınsama sağlayabilir olmasıdır. Bu nedenle yarasa algoritması, hızlı çözüme ihtiyaç duyulan sınıflandırma ve benzeri uygulamalarda etkili bir algoritma olarak ortaya çıkmaktadır. Geliştirilmeye oldukça açık olan bu algoritma; optimizasyon, sınıflandırma, kümeleme, görüntü işleme, özellik seçimi, çizelgeleme, veri madenciliği gibi hemen hemen her alanda uygulanmıştır. Literatürde algoritmanın başlıca uygulama alanları aşağıdaki gibi özetlenebilir:

2.3.1. Sürekli Optimizasyon

Yarasa algoritmasının ilk uygulamaları arasında, mühendislik tasarım optimizasyonu bağlamında sürekli optimizasyon yaygın olarak incelenmiştir. Yarasa algoritması doğrusal olmayan problemlerin çözümünde son derece etkili sonuçlar sergilemiş ve uygun sonuçları en doğru şekilde bulabilmiştir. Yang (2010d) çalışmasında yarasa algoritmasını literatürde bilinen sürekli test fonksiyonlarına uygulamış ve sonuçları

değerlendirmiştir. Ayrıca algoritmanın performansını parçacık sürü optimizasyonu ve genetik algoritma ile karşılaştırmıştır.

Daha sonra Yang (2011), çok amaçlı optimizasyon problemlerini çözmek için yarasa algoritmasını yeniden formüle etmiştir. Yang önerdiği bu çalışmayı konveks, konveks olmayan ve süreksiz olmak üzere üç farklı fonksiyon ve literatürde en fazla bilinen "kaynaklı kiriş tasarımı" problemi üzerinde test ederek; çok amaçlı yarasa algoritmasının fonksiyonlar üzerinde verimli olduğunu göstermiştir.

Yang ve Gandomi (2012), yarasa algoritmasının performansını görmek için; Himmelblau problemi, üç barlı kafes tasarım problemi, hız azaltıcı tasarım problemi gibi bilinen bazı zor tasarım problemlerinde algoritmayı test etmişlerdir. Böylece gerçek hayatta karşılaşılan karmaşık kısıtlı, doğrusal olmayan mühendislik problemlerinin çözümünde algoritmanın etkin olabileceği belirtilmiştir.

Bora ve diğerleri (2012) tek ve çok amaçlı fırçasız DC tekerlekli motor optimizasyon problemleri üzerinde çalışmışlar ve çok amaçlı yarasa algoritması geliştirerek sonuçları değerlendirmişlerdir. Ayrıca sonuçlar literatürde önerilen diğer optimizasyon yaklaşımları ile karşılaştırılarak, elektromanyetik alanında doğrusal olmayan problemler için tanıtılan bu yeni tekniğin uygulanabilirliği gösterilmiştir.

Gandomi ve diğerleri (2013) kısıtlı küresel optimizasyon problemlerinde karşılaşılabilecek birçok türde zorluğu içeren çeşitli matematiksel test problemlerine ve bilinen üç tasarım problemine (basınçlı kap tasarımı, kaynaklı kiriş tasarımı, gerilim sıkıştırma yay tasarımı) yarasa algoritmasını uygulayarak çeşitli karşılaştırmalar yapmışlardır.

Fister ve diğerleri (2013), yarasa algoritmasının yerel arama aşamasına diferansiyel evrim algoritmasını entegre etmişlerdir. Beş farklı fonksiyon üzerinde 10, 20 ve 30 boyutlu olmak üzere yapılan analizler sonucunda, diferansiyel evrim yaklaşımının yarasa algoritması sonuçlarını geliştirdiği görülmüştür.

Tsai ve diğerleri (2013), yarasa algoritmasını baz alarak geliştirdikleri algoritmanın duyarlılığını ve hızını üç farklı test fonksiyonu üzerinde incelemişlerdir. Sonuçlar,

geliştirilen algoritmanın Yarasa Algoritmasına göre yakınsama hızını % 99,42 geliştirdiğini ayrıca hesaplama maliyetini % 6,07 azalttığını göstermiştir.

Wang ve diğerleri (2013), yarasa algoritmasının beklenilenden erken yakınsama sorununun üstesinden gelmek için, yarasa algoritmasının eksiklikleri üzerinde durarak algoritmayı geliştirmişlerdir. Yaklaşımlarına göre, yarasa algoritması avın konumunun zamanla birlikte rastgele değişimini dikkate almamakta, ayrıca frekans ve hız değişimi formülleri yarasanın uçuş hareketini sınırlandırarak arama yeteneğinin düşmesine neden olmaktadır. Dolayısıyla formüllere zamanla değişen atalet ağırlık faktörü ve en iyi konuma yakınlaşmasını sağlayan kontrol bileşeni eklemişlerdir. Geliştirilen yarasa algoritması, altı yüksek boyutlu fonksiyon üzerinde test edilerek orijinal yarasa algoritması ile karşılaştırılmış ve geliştirilen algoritmanın lokal optimumlara takılmadığı görülmüştür.

Yılmaz ve Kucuksille (2013), yarasa algoritmasının yerel (exploitation) ve küresel (exploration) aramadaki yetersizliğini gidermek amacıyla algoritmaya üç adet modifikasyon uygulamışlardır. Geliştirilen algoritmanın performansı farklı boyuttaki 10 fonksiyon üzerinde test edilerek standart yarasa algoritması ile karşılaştırılmış ve geliştirilen algoritmanın oldukça etkili olduğu görülmüştür.

Yılmaz ve diğerleri (2014) yarasa algoritmasının küresel arama yeteneğini, sinyal emisyon oranı ve ses şiddeti hesaplamalarında yapmış oldukları değişiklik ile geliştirmişlerdir. Algoritmanın etkinliği 15 farklı kısıtsız fonksiyon üzerinde test edilmiş ve standart yarasa algoritması ile yapılan karşılaştırmalar sonucunda geliştirilen algoritmanın daha iyi olduğu sonucuna varılmıştır.

2.3.2. Kesikli Optimizasyon ve Çizelgeleme

Hesaplama karmaşıklığı açısından kesikli optimizasyon problemleri sürekli optimizasyon problemlerine göre daha zor kabul edilmektedir. Kombinatoryal problemler, genellikle belirli olmayan polinom zamanlı olduklarından çözümleri oldukça çaba gerektirmektedir.

Ramesh ve diğerleri (2013) yarasa algoritmasını kullanarak ekonomik yük ve emisyon dağıtım problemi hakkında detaylı bir çalışma sunmuşlardır. Çalışmalarında yarasa algoritmasını karınca kolonisi algoritması, melez genetik algoritma ve diğer yöntemlerle

karşılaştırmışlardır. Yapılan karşılaştırmalar neticesinde yarasa algoritmasının kolay uygulandığını tespit etmişler, doğruluk ve verimlilik açısından diğer algoritmalardan çok daha üstün olduğu sonucuna varmışlardır.

Musikapun ve Pongcharoen (2012) çok aşamalı, çok makineli, çok ürünlü çizelgeleme problemini yarasa algoritmasını kullanarak çözmüşler ve bu çalışmaları ile, belirli olmayan polinom zamanlı zor problem sınıfını ayrıntılı bir parametrik çalışma ile ele almışlardır. Ayrıca algoritma için en uygun parametre setini kullanarak performansı yaklaşık %8,4 daha geliştirmişlerdir.

Marichelvam ve Prabaharan (2012) hibrid akış tipi çizelgeleme problemlerinde tamamlanma zamanını ve ortalama akış süresini en aza indirmek için yarasa algoritmasını kullanmışlardır. Elde ettikleri sonuçlar doğrultusunda yarasa algoritmasının hibrid akış tipi çizelgeleme problemlerinin çözümü için etkin bir yaklaşım olduğunu ileri sürmüşlerdir.

Fister ve diğerleri (2015), antrenörler için çok zor bir görev olan uygun spor eğitimi planlama problemini yarasa algoritması ile ele almışlardır. Çalışmalarında, sporculara takılan saatlerden elde edilen güvenilir verilere göre, yarasa algoritmasını kullanarak digital bilgisayarlarda eğitim planları oluşturan yeni bir akıllı planlama yöntemi tanıtmışlardır. Amatör bisikletçilerin gerçek eğitim süreçleri boyunca üzerlerindeki saatlerden alınan 1000 adet dosya, antrenörler tarafından belirlenen on temel eğitimin planlanması için kullanılmıştır.

2.3.3. Ters (Inverse) Problemler ve Parametre Tahmini

Yang ve diğerleri (2012b) mikroelektronik uygulamalarda topolojik şekil optimizasyonunu incelemek için yarasa algoritmasını kullanmışlardır. Böylece farklı termal özelliklerdeki malzemeleri sıkı kısıtlamalar altında ısı transferini en verimli hale getirecek şekilde yerleştirebilmişlerdir.

Lin ve diğ. (2012) lineer olmayan dinamik biyolojik sistemde parametreleri tahmin etmek için, yarasa algoritmasına kaotik dizi ve Lévy uçuş (Lévy Flight) yapılarını dahil etmişler ve algoritmanın etkinliğini kanıtlamışlardır.

2.3.4. Sınıflandırma, Kümeleme ve Veri Madenciliği

Komarasamy ve Wahi (2012) K-ortalamalar kümeleme algoritması ile yarasa algoritmasını birleştirmişler ve diğer algoritmalara göre bu kombinasyondan daha yüksek performans elde etmişlerdir. Çalışmalarında öbek sayısını ve merkezlerini yarasa algoritması ile belirleyerek K-ortalamalar algoritması sonuçlarında giriş parametresinin etkililiğini ortadan kaldırmaya çalışmışlardır.

Khan ve diğerleri (2011) bulanık yarasa algoritması kullanarak gerçekleştirdikleri vaka çalışması ile ofis işyerleri için kümelenme sorununu çalışmalarında sunmuşlardır. Khan ve Sahari (2012a) bulanık c-ortalamalar algoritmasının eksikliklerinin üstesinden gelmek için, yarasa algoritmasının bir versiyonu olarak geliştirdikleri bi-sonar optimizasyon yöntemi ile bulanık c-ortalamalar algoritmasını bilinen üç veri seti üzerinde entegre çalıştırmışlardır.

Öte yandan, Mishra ve diğerleri (2012) çalışmalarında yapay sinir ağlarının ağırlıklarını güncelleştirmek için yarasa algoritmasından faydalanmışlar ve uygulamada mikrodizi verileri sınıflandırmışlardır. Damodaram ve Valarmathi (2012) internet sitelerindeki kimlik hırsızlığı ve dolandırıcılık gibi olayların tespiti için değiştirilmiş yarasa algoritmasını kullanarak veri madenciliği çalışması yapmışlar ve iyi sonuçlar elde etmişlerdir. Rui ve diğerleri (2012) web verilerini kümelemede çeşitli doğadan esinlenen algoritmaların performanslarını değerlendirmişlerdir.

Sood ve Bansal (2013), bölümlemeli kümeleme yöntemlerinden biri olan K-Medoids algoritmasında temsili küme merkezlerini belirlemek için yarasa algoritmasını kullanmışlardır. Geliştirilen bu hibrid yaklaşımın büyük veri setlerinde daha iyi sonuçlar verdiği belirtilmiştir.

Taha ve Tang (2013), veri madenciliği ve makine öğrenmesi alanlarında sınıflandırma algoritmalarının tahmin doğruluğunu geliştirmek, özellik alanı boyutunu azaltmak ve bağlı kavramların görselleştirilmesini sağlamak için kullanılan öznitelik azaltma (attribute reduction) işlemini yarasa algoritması ile gerçekleştirmişlerdir. 13 farklı veri seti üzerinde; yarasa algoritması, tavlama benzetimi, karınca kolonisi optimizasyonu, genetik algoritma, tabu arama ve dağılım arama ile yapılan öznitelik azaltma işlemi sonucunda yarasa algoritması yüksek performans sergilemiştir.

2.3.5. Görüntü İşleme

Akhtar ve diğerleri (2012) laboratuvar ortamında kaydedilen çok-görünümlü video verilerinden tam vücut hareket izleme sorununu gidermeye çalışmışlardır. Bunun için yarasa algoritmasını kullanmışlar ve algoritmanın parçacık sürü optimizasyonu (PSO), partikül filtresi (PF) ve tavlanmış partikül filtresi (APF) uygulamalarından daha iyi bir performans sergilediği sonucuna varmışlardır.

Zhang ve Wang (2012) yarasa algoritmasının mutasyon ile birleştirilmiş bir versiyonunu sunarak görüntü eşleştirme problemini ele almışlardır. Çalışmada yarasa-tabanlı modelin, diferansiyel evrim ve genetik algoritma gibi diğer modellere göre daha etkili ve uygun olduğu belirtilmiştir.

2.3.6. Bulanık Mantık ve Yapay Zeka Uygulamaları

Reddy ve Manoj (2012), elektrik dağıtım sistemlerinde kaybın azaltılması için en uygun kondansatör yerleşimini bulanık mantık kullanarak ele almışlardır. Ayrıca çalışmalarında, kayıpları en aza indirecek şekilde en uygun kondansatör boyutlarını bulmak için yarasa algoritmasından yararlanmışlardır. Sonuçlar gerçek güç kaybının önemli ölçüde azaltılabileceğini göstermiştir.

Bunun dışında Lemma ve Hashim (2011) bir endüstriyel gaz tribünündeki ana bileşenlerin ekserji değişimlerini yakalamak için, yarasa algoritmasını bulanık sistemlerin içine katarak modellemişlerdir.

Khan ve Sahai (2012b) yapay sinir ağlarının eğitiminde yarasa algoritmasının üstünlüğünü göstermek için; genetik algoritma, parçacık sürü optimizasyonu, Levenberg-Marquard algoritması ve geri yayılma (back propagation) algoritması ile karşılaştırmışlardır. Tıp alanındaki 5 farklı veri seti üzerinde yapılan karşılaştırmada ortalama hata bakımından yarasa algoritması en iyi sonuçları vermiştir.

Nawi ve diğerleri (2014), geri yayılımlı sinir ağlarında ağırlıkların belirlenmesi için yarasa algoritmasından faydalanmışlardır. Önerdikleri yarasa-geri yayılım algoritmasının performansını iki girdili tek çıktılı ve üç girdili tek çıktılı veri seti ile mantıksal operatörlü bir diğer veri seti üzerinde değerlendirmişlerdir. Yapay arı kolonisi kullanarak geri yayılımlı sinir ağları (Artificial Bee Colony using Back-propagation neural network, ABC-BPNN) ve basit geri yayılımlı sinir ağları

algoritmaları ile yapılan karşılaştırma sonucunda, önerilen algoritmanın iki girdili tek çıktılı ve mantıksal operatörlü veri setlerinde 0 ortalama hata karesi ve % 100 ortalama doğrulukla çalıştığı sonucuna varılmıştır. Ayrıca, tüm veri setlerinde hesaplanan CPU zamanı da karşılaştırma yapılan iki algoritmaya göre oldukça düşüktür.

2.3.7. Diğer Uygulamalar

Zhou ve diğerleri (2013) kapasite kısıtlı araç rotalama problemini, açgözlü rastgele uyarlamalı arama süreci (greedy randomized adaptive search procedure-GRASP) ile yarasa algoritmasını birleştiren hibrid bir modelle ele almışlardır. Yol birleştirme (path relinking) bir yoğunlaşma stratejisi olarak, önerilen algoritma ile elde edilen iyi çözümlerle bağlantılı yerel yörüngeleri keşfetmek için kullanılmıştır. Yapılan simülasyon çalışmaları sonucunda hibrid modelin oldukça kaliteli sonuçlar sağladığı görülmüş ve araç rotalama probleminin farklı türlerine bu modelin kolayca uygulanabileceği sonucuna varılmıştır.

Goyal ve Patterh (2013), çok-modelli ve çok boyutlu optimizasyon problemlerinden biri olan kablosuz algılayıcı ağı probleminde yarasa algoritmasından faydalanmışlardır. Kablosuz algılayıcı ağı probleminin ana amacı; referans düğümleri kullanarak, en fazla sayıdaki hedef düğümlerin koordinatını öğrenmektir.

Alihodzic ve Tuba (2013), sürekli optimizasyon problemleri için yarasa algoritması tabanlı, nesneye yönelik yazılım sistemi geliştirmişlerdir. BATapp adı verilen ve C# ile oluşturulan yazılım, literatürdeki beş farklı test fonksiyonu üzerinde uygulanarak başarısı kanıtlanmıştır.

Büyüksaatçı (2015), tek sıra tesis içi yerleşim problemini yarasa algoritması ile ele alarak, yarasa algoritmasının performansını etkilediği düşünülen beş algoritma parametresini, deney tasarımı ile optimize etmiştir. Ayrıca Büyüksaatçı, literatürde bu problem tipi için geliştirilmiş beş farklı sezgisel algoritma ile yarasa algoritmasının performansını 6 veri seti üzerinde karşılaştırmış ve yarasa algoritmasının ortalama amaç fonksiyonu değerlerinin optimum sonuçtan problem boyutuna göre ne kadar değiştiğini göstermiştir.

2.4. BELGE KÜMELEME

Bölüm 2.3'te de belirtildiği gibi, yarasa algoritmasının birçok farklı alanda uygulaması bulunmaktadır. Ancak literatürde sıkça kullanılan belge setlerini yarasa algoritması ile kümeleyen; yarasa algoritmanın kümeleme performansı belge kümeleme analizinde en çok kullanılan algoritmalar ile karşılaştırılaran herhangi bir çalışma literatürde henüz yer almamaktadır. Bu tez çalışması kapsamında, yarasa algoritması ile belge kümeleme uygulaması ele alınacağından, tezin bu bölümünde belge kümeleme konusunda detaylı bilgi verilecektir.

Son yıllarda bilgisayar donanım teknolojisindeki sürekli ve şaşırtıcı ilerleme, güçlü ve uygun fiyatlı bilgisayarların, veri toplama ekipmanlarının ve depolama araçlarının artan sayıda tedarik edilmesine sebep olmuştur. Bu gelişmeler bilişim sektöründe de büyük bir teşvik uyandırmış; işlem yönetimi, bilgi alma ve veri analizi uygulamalarında kullanılmak üzere büyük miktarda veri tabanlarının ve bilgi depolarının oluşturulması sağlanmıştır. Böylece internet üzerindeki metin belgelerinin hacmi, dijital kütüphaneler ve depoları, haber kaynakları, şirket çapında intranet, blog makaleler ve e-postalar gibi dijital kişisel belgeler büyük bir büyüme göstermiştir.

Günümüzde hala devam eden elektronik belgelerin sayısındaki artış, bu belgelerin elle verimli bir şekilde organize ve analiz edilmesini zorlaştırmaktadır. Bu zorluklar, metin belgelerinin otomatik olarak etkin ve verimli bir şekilde organize edilmesini beraberinde getirmiştir.

Veri madenciliği; verilerden örtülü, daha önce bilinmeyen ve potansiyel olarak yararlı bilgilerin çekilmesi işlemidir. Belge kümeleme ise, veri kümeleme alanının içinde yer alan, bilgi alma, doğal dil işleme ve makine öğrenme alanlarındaki kavramları içeren bir veri madenciliği tekniğidir. Belge kümeleme, belgeleri küme olarak adlandırılan farklı gruplar halinde düzenlemektedir. Bu düzenlemede her küme içindeki belgeler, tanımlanmış benzerlik ölçüsüne göre bazı ortak özellikleri paylaşmaktadır (Shah ve Mahajan, 2012).

Belge kümelemenin en önemli noktalarından biri; bilgileri etkili taramak, özetlemek ve düzenlemek için kullanıcılara yardımcı hızlı ve kaliteli belge kümeleme algoritmalarının kullanılmasıdır. Literatürde önerilmiş birçok kümeleme algoritması olmasına rağmen,

bu algoritmaların çoğu, belgelerin kümelenmesinde önemli olduğu düşünülen aşağıdaki gereksinimleri karşılamamaktadır:

- **Yüksek Boyutluluk:** Bir belge kümesindeki ilgili terimlerin sayısı genellikle binleri bulmaktadır. Bu terimler her bir belge vektöründe de bir boyutu oluşturmaktadır. Doğal kümeler, genellikle uzayın tam boyutunda yer almayıp bağlantılı boyutlarda bir dizinin oluşturduğu alt uzayda bulunmaktadırlar. Ancak bu kümeleri alt uzay içine yerleştirmek zor olabilmektedir.
- **Ölçeklenebilirlik:** Gerçek hayatta karşılaşılan veri setleri, yüz binlerce belge içerebilmektedir. Birçok kümeleme algoritması küçük veri setleri üzerinde verimli çalışırken, büyük veri setlerini verimli işlemek konusunda başarısızdır.
- **Doğruluk:** En çok istenen özelliktir. İyi bir kümeleme çözümünde küme içi benzerlik yüksek, kümeler arası benzerlik ise düşük olmalıdır; yani, aynı küme içindeki belgeler birbirine benzerken diğer kümelerdeki belgelerle aralarında fark vardır.
- **Anlamlı Küme Açıklamaları ile Tarama Kolaylığı:** Başarılı bir küme etiketleme, kısa ama anlamlı bir küme açıklamasını sağlayarak kümeleme sürecinde kullanıcıya rehberlik edebilir. Çıkan konu hiyerarşisi interaktif taramayı desteklemek için, anlamlı küme açıklamaları ile birlikte mantıklı bir yapı sağlamalıdır.
- **Önceki Alan Bilgisi:** Birçok kümeleme algoritması küme sayısı vb. gibi bazı giriş parametrelerinin kullanıcı tarafından belirtilmesini gerektirmektedir. Ancak, kullanıcılar çoğunlukla bu tür alan bilgisine önceden sahip değildirler. Bir algoritma bu tip giriş parametrelerine çok hassas ise kümeleme doğruluğu büyük ölçüde düşebilmektedir (Fung ve diğ., 2006; Bisht ve Paul, 2013)

Belge kümeleme denetimsiz öğrenme yöntemlerinden biri olup, iş dünyasının ve bilimin birçok alanında uygulanmaktadır. Başlangıçta bilgi alma sistemlerindeki hassasiyeti geliştirmek amacıyla kullanılan belge kümeleme işlemi, günümüzde farklı uygulamalara sahiptir. Bu uygulamalar genel hatlarıyla aşağıdaki gibi özetlenebilir:

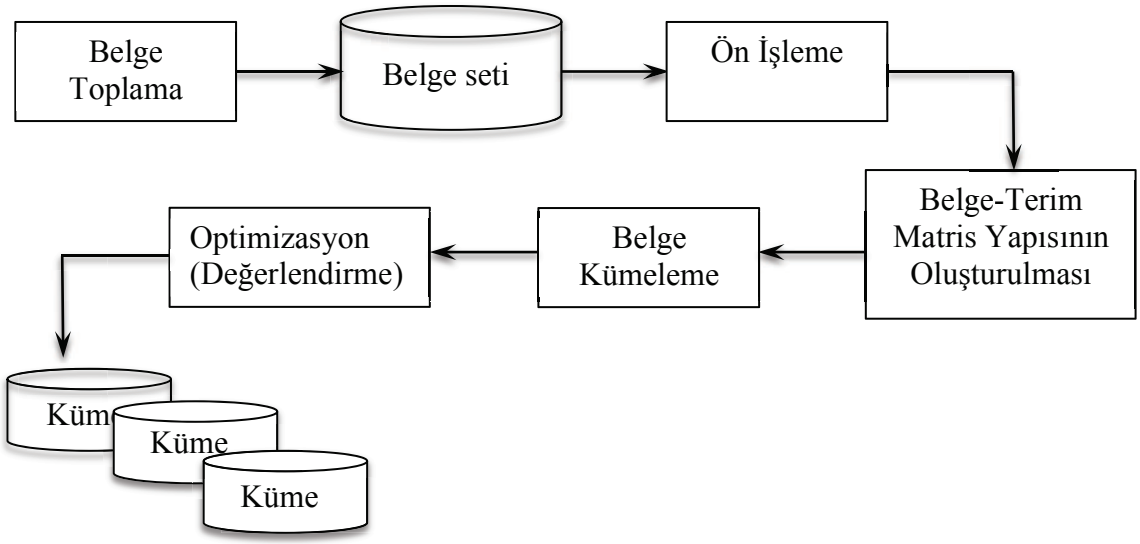
- Benzer Belgeler Bulma: Arama sonucu belge ile eşleşen benzer belgeleri bulmak için belge kümelemeden yararlanılmaktadır. Kümeleme, aynı kelimelerin

çoğunu paylaşan belgeleri bulan arama tabanlı yaklaşımlara nazaran, kavramsal olarak benzer belgeleri bulabilmektedir.

- Büyük Belge Setlerini Düzenleme: Sınıflandırılmamış çok sayıda belge, belge kümeleme uygulamaları sayesinde, birçok insanın kolay erişim için yaratacağı düzen ile aynı şekilde sınıflandırılabilmektedir.
- İçerik Algılama: Birçok uygulamada, çok sayıda belge içindeki kopyaları bulmaya ihtiyaç vardır. Kümeleme, ilgili haberlerin gruplanması, intihal tespiti ve arama sonuçlarının sıralamasını yeniden düzenlemek için kullanılmaktadır.
- Öneri Sistemleri: Bu uygulamalarda, kümeleme işlemi ile makaleler gruplanarak, kullanıcının okuduğu makalelere dayanan farklı makaleler önerilmektedir (örneğin Science Direct).
- Arama Optimizasyonu: Kümeleme işlemi, arama motorlarının kalite ve etkinliğinin artırılmasına çok yardımcı olmaktadır. Kümeleme sayesinde, bir kullanıcının sorgusuna cevaben bir arama motoru tarafından döndürülen sonuçlar düzenlenmektedir. Böylece kullanıcı sorgusu, doğrudan belgelerle karşılaştırılmak yerine ilk olarak kümelerle kıyaslanabilmektedir (Shah ve Mahajan, 2012).

Belge kümeleme işlemi yalnızca tek bir işlem gibi görünse de, birçok aşamadan oluşan bir süreç niteliğindedir. Bu aşamalar filtreleme, ağırlıklandırma gibi geleneksel bilgi erişim işlemlerini kapsamakta ve çoğu kümeleme algoritmasının kalitesini ve performansını etkilemektedir. Dolayısıyla kümeleme algoritmasının gerçek potansiyeli bu aşamalar ile birlikte ele alınmalıdır.

Genel hatları ile belge kümeleme süreci Şekil 2.14'te verilmiştir:



Şekil 2.14: Belge kümeleme süreci (Jensi ve Jiji, 2013).

Belge toplama aşaması, kümelenecek belge setinin web sayfaları, e-postalar, dijital kütüphaneler vb. gibi çeşitli kaynaklardan oluşturulmasıdır. Bu işlem sonrasında belge seti, ön işlemeye tabi tutulmaktadır. Ön işleme temelde; harf, noktalama işaretleri, etiketler, grafikler gibi tüm biçimlendirmelerin yok edilmesidir. Ön işlemede ayrıca zamir, bağlaç, edat gibi durak kelimeler, bu kelimelerin önceden hazırlanmış bir listesi kullanılarak ortadan kaldırılmaktadır. Durak kelimelerin yok edilmesi ile hem belge seti içindeki gürültü azaltılmakta hem de sadece anlamlı kelimelerin kalması sağlandığı için kümeleme hesaplamaları kolaylaştırılmaktadır.

Ön işleme aşamasında uygulanan bir diğer işlem, kelimelerin kök hallerine getirilmesidir. İngilizce belge setlerinde bu aşamada kullanılan en popüler algoritma "*Porter Stemmer*"dır. Açık kaynak kodlu, tüm Türki diller için kullanılan Türkçe doğal dil işleme kütüphanesi ise "*Zemberek*"tir (Jensi ve Jiji, 2013).

Ön işlemenin ardından belgelerin kümeleme yapılabilmesi için sayısal forma getirilmesi gerekmektedir. Bunun için birçok yöntem olmakla birlikte, en fazla kullanılan yöntem "Vektör Uzay Modeli"dir. Bu modelin detayları bölüm 2.3.4'te anlatılacaktır.

Belge setinin sayısal forma dönüştürülmesi sonrasında, istenen kümeleme türü ile, belirlenen ölçüt fonksiyonu ve benzerlik ölçüsünü kullanılarak kümeleme işlemi gerçekleştirilmektedir.

Kümeleme işleminin ardından, kümeleme sonuçlarının gerçek veriyi ne kadar iyi yansıttığı değerlendirilmelidir. Bu aşamada çeşitli içsel ve dışsal göstergeler kullanılarak sonuçlar analiz edilmektedir. Değerlendirme aşamasında kullanılan göstergeler bölüm 2.4.7'de detaylı sunulacaktır.

2.4.1. Belge Kümeleme Türleri

Belge kümeleme konusu ele alınırken, verilen bir uygulama için hangi kümeleme türünün gerekli ve mümkün olduğunu bilmek her zaman önemlidir. Bu bağlamda, kümeleme türlerini aşağıdaki şekilde gruplamak mümkündür:

- *Keskin ve Bulanık Kümeleme:* İncelenen bilgilerin birkaç kategori içinde geçerli olması mümkündür. Örneğin "biyomedikal mühendislik" alanında yer alan bir belge, "kimya mühendisliği", "biyoloji" ve "ilaç" gibi farklı kategoriler içinde de yer alabilir. Belgelerin çeşitli kategorilerde yer almasını sağlayan bu kümeleme türüne bulanık kümeleme denmektedir. Keskin kümelemede ise bir belge sadece bir kümeye atanmaktadır ve kümeler içinde birbirine yakın belgeler yer almaktadır.
- *Hiyerarşik ve Bölümlemeli (partitional) Kümeleme*: Bilgileri düzenlerken, ne tür bir yapının gerekli olacağını belirlemek tabii ki her zaman önemlidir. Bazı uygulamalar için sadece, tüm verilerin bir bellek alanına (bucket) konulması ve istek üzerine bu alana dönülmesi kabul edilebilir bir çözümdür. Bu yaklaşım; kümeler arası mesafelerin yüksek, küme içi mesafelerin düşük ve tüm kümelerin aynı boyutta olduğu durumlarda pratik ve gerçekçidir. Belge uzayındaki tüm belgelerin farklı bellek alanlarına ayrılmasıyla sonuçlanan kümeleme yöntemleri bölümlemeli kümeleme yöntemleri olarak adlandırılmaktadır.

  Bununla birlikte, bilgi genellikle birçok farklı açılardan kabul edilebilir, kümeler değişen büyüklükte olabilir ve her küme ayrı ayrı kendi kümeleri ile bir alt uzay olarak görülebilir. Bölümlemeli bir kümeleme işlemi ile bu yapı gerçekleştirildiğinde, ilgisiz bilgiler aynı küme içine atanmak durumunda kalabilir. Bunu önlemek için, bilgiler hiyerarşik olarak da organize edilmelidir. Böyle bir hiyerarşide en üst düzeyler en büyük kümeleri içermeli ve koleksiyonunun en genel konuları burada olmalıdır. Daha sonra hiyerarşide aşağı doğru ilerledikçe büyük kümeler birkaç küçük kümeye ayrılmalıdırlar. Bu

şekilde, ağaçlar yapısında organize edilmiş ve iç içe geçmiş alt kümelerden oluşan kümeleme işlemine hiyerarşik kümeleme denmektedir.

Buradan da anlaşılacağı üzere, bir hiyerarşik kümeleme, bölümlemeli kümelemelerin bir dizisi olarak görülebilir. Ayrıca bir bölümlemeli kümeleme, bu dizinin herhangi bir üyesi alınarak yani hiyerarşik ağacı belirli bir düzeyde keserek elde edilebilir.

Hiyerarşik kümeleme bulanık kümelemenin özel bir durumu olarak da görülebilir. Ancak, hiyerarşik kümeleme ile bağlantılı olarak keskin ve bulanık küme arasında ayrım yaparken; hiyerarşinin aynı seviyesindeki kümeler arasında örtüşme olduğunda, hiyerarşik kümeleme bulanık hiyerarşik kümeleme olarak tanımlanır. Keskin hiyerarşik kümelemede ise aynı düzeydeki kümeler arasında çakışma yoktur. Hiyerarşik bir belge kümeleme yaratmak için çeşitli yaklaşımlar bulunmaktadır.

- *Tam ve Kısmi Kümeleme*: Tam kümelemede veri setindeki her veri bir kümeye atanırken, kısmi kümelemede veri seti içerisindeki bazı veriler hiçbir kümeye ait olmaksızın dışlanırlar. Bu tür veriler, veri seti içindeki gürültüler (noise) ya da aykırı değerler (outliers) olarak bilinirler. Örnek olarak, bazı gazeteler küresel ısınma gibi özel bir başlığı işlerken bazı gazeteler sıradan konuları içerebilir. Son ayın önemli konularını bulmak isteyen bir kişi, küresel ısınma başlığı ile ilgili belgeleri araştırmalı ve kümelemelidir.
- *Çevrimiçi ve Çevrimdışı Kümeleme*: Kümelemenin ne zaman gerçekleştirileceği bir başka önemli hususdur. Bu husus; küme terim uzayının boyutu, kullanılacak algoritmanın karmaşıklığı, kümelemenin ne için kullanılacağı ve aynı anda ne kadar işlem gerçekleştirileceği gibi birçok faktöre bağlıdır.

  Kümeleme işlemleri için büyük bir terim uzayı gerekiyorsa, çevrimdışı kümeleme muhtemelen en uygun yaklaşımdır. Bu, hızlı bir veri tabanında kümeleri oluşturma ve saklama anlamına gelmektedir. Veri tabanının sorgulanması sırasında sadece basit işlemler gerçekleştirilecektir. Çevrimdışı kümelemenin dezavantajı ise, tek bir belge değiştirilmesi veya eklenmesi gerektiğinde en kısa sürede güncellemenin olmamasıdır. Bu nedenle sık sık güncelleme yapılması gerekmektedir.

  Diğer taraftan çevrimiçi kümeleme, bu tarz güncellemeleri gerektirmeyen, tüm işlemlerin talep üzerine gerçekleştirildiği bir yapıdır. Dolayısıyla çok hızlı

algoritmalar ve sınırlı bir veri kümesi gerektirir. Çevrimiçi kümeleme, kümeleme arama sonuçları için pratik olabilir (Vester ve Martiny, 2005; Kumar ve diğ., 2006).

2.4.2. Belge Kümeleme Algoritmaları

Bu bölümde en önemli kümeleme algoritmaları üzerinde durulacaktır. İncelenecek algoritmalar hiyerarşik kümeleme ve hiyerarşik olmayan kümeleme kategorilerinden birinde yer almaktadır.

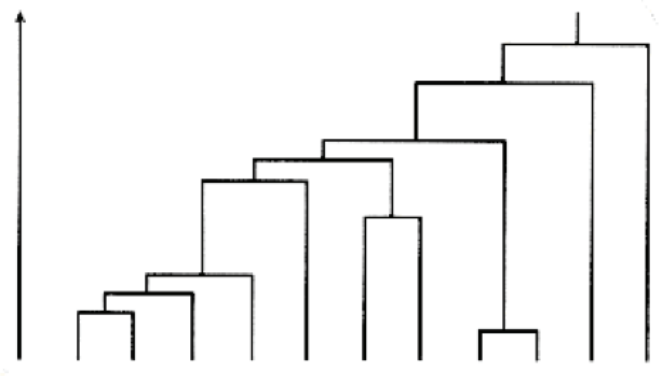
Hiyerarşik olmayan kümeleme algoritmaları genellikle düz (flat) yöntemler veya bölümlemeli (partitional) yöntemler olarak bilinmektedirler (Steinbach ve diğ., 2000). Bu algoritmalar küme sayısı, küme boyutu, küme üyeliği için kriter ve küme yapısı hakkında önsel kararlar gerektirdiğinden doğalarında sezgisellik vardır. N öğenin olası bölünmelerinin büyük sayıdaki K küme halinde olması, optimal bir çözümü imkansız hale getirdiğinden, hiyerarşik olmayan algoritmalar yaklaşık bir sonuç bulmaya çalışmaktadırlar. Bunun için genellikle veriyi bölümlemekte ve daha sonra bazı kriterler optimize edilinceye kadar öğeleri yeniden ayırmaktadırlar (Frakes ve Baeza-Yates, 1992).

Hiyerarşik kümeleme algoritmaları için ise iki temel yaklaşım vardır:

a) Birleştiricilik (Agglomerative): Her bir nokta birer bireysel küme olarak düşünülerek bu algoritmalara başlanır ve her adımda kümelerin en benzer veya yakın çifti birleştirilir. Bu tip algoritmalarda küme benzerliğinin veya uzaklığın tanımlanması gerekmektedir.
b) Ayırıcılık (Divisive): Tüm noktaları içeren bir küme ile algoritmaya başlanır. Ardından her bir adımda, bireysel noktaların tekil kümeleri kalıncaya kadar küme bölünür. Bu tip algoritmalarda her aşamada hangi kümenin bölüneceğine ve bölme işleminin nasıl gerçekleştirileceğine karar verilmesi gerekmektedir (Steinbach ve diğ., 2000).

Hiyerarşik birleştirici kümeleme algoritmalarından elde edilen küme yapısı genellikle, Şekil 2.15'te gösterilen bir dendrogram olarak gösterilmektedir. Bu yapıda veri kümesindeki nesnelerin ikili bağlantısının sırası görülmekte ve her birleşmede benzerlik fonksiyonu değeri oluşmaktadır. Bir kümelenmiş belge dizisinden bilgi elde etmek

istenildiğinde, takip edilecek erişim yollarını göstermesi nedeniyle dendrogram yararlı bir gösterimdir (Frakes ve Baeza-Yates, 1992).



Şekil 2.15: Hiyerarşik kümelemede dendrogram.

Birleştirici algoritmaların adımları aşağıdaki gibi özetlenebilir:

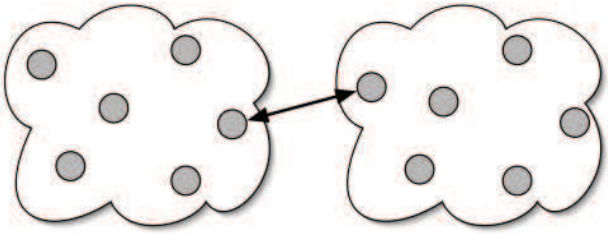
1) i-nci ve j-nci küme arasındaki benzerliği gösteren $[a_{ij}]$ elemanlarından oluşan benzerlik matrisi A'yı hesapla.
2) En benzer iki kümeyi birleştir.
3) Benzerlik matrisini güncelle (yeni kümeyi temsil eden sütunun güncelleştirilmesi gerekir).
4) Tüm belgeler tek bir kümede birleşene kadar 2. ve 3. adımları tekrarla (Steinbach ve diğ., 2000; Tarczynski, 2011).

Birleştirici algoritmalar genellikle kullandıkları kümeler arası benzerlik ölçümlerine göre sınıflandırılmaktadırlar. Bunlardan en popüler olanları *tek-bağlantı*, *tam bağlantı* ve *grup ortalaması*dır.

Tek-bağlantı benzerlik ölçümüne dayalı kümeleme algoritmaları, Şekil 2.16'da gösterilmektedir. Bu algoritmalar, aynı küme içinde olmayan en yakın iki belgeyi, belirli bir mesafe ölçüsünü kullanarak değerlendirmekte; kümelerin genel yapısı ve kümeler içinde yer alan diğer uzak belgeler dikkate alınmamaktadır. Bu ölçüm, diğer iki benzerlik ölçümüne (tam bağlantı ve grup ortalaması) göre daha etkili bir şekilde uygulanabilmektedir. Bununla birlikte bu değerlendirme elips kümeleri betimlemek için uygun; küresel ya da zayıf biçimde ayrılmış kümelerin oluşturulması için ise uygun değildir.

Tek-bağlantı kümeleme algoritmasının benzerlik ölçümü aşağıdaki gibi tanımlanmaktadır:

$$\mathrm{Sim}(C_j, C_k) = \min_{d_j \in C_j, d_k \in C_k} \left(dist(d_j, d_k) \right) \tag{2.7}$$

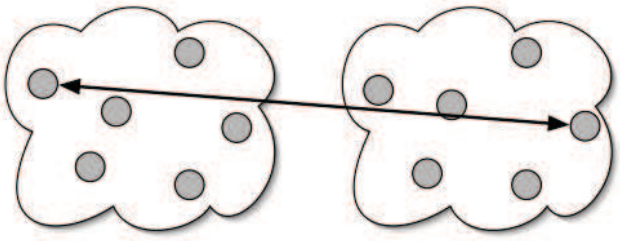


Şekil 2.16: Tek-bağlantı benzerlik ölçümü.

Tam-bağlantılı benzerlik ölçümünü kullanan kümeleme algoritmaları, Şekil 2.17'deki gibi "en uzak" belge çiftini değerlendirmektedir. Bu algoritmalarda dikkat edilmesi gereken en önemli nokta, kümenin genel yapısına uymayan aykırı belgelerdir. Dolayısıyla bu benzerlik ölçümünü kullanan kümeleme algoritmalarının hesaplama karmaşıklığı tek bağlantı tabanlı algoritmalardan normal olarak daha yüksektir.

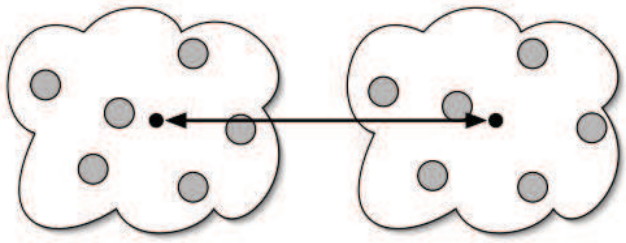
Tam-bağlantılı benzerlik ölçümünü kullanan kümeleme algoritması denklem 2.8'yi kullanmaktadır.

$$\mathrm{Sim}(C_j, C_k) = \max_{d_j \in C_j, d_k \in C_k} \left(dist(d_j, d_k) \right) \tag{2.8}$$



Şekil 2.17: Tam-bağlantı benzerlik ölçümü.

Şekil 2.18'de gösterildiği gibi, grup ortalaması benzerlik ölçümünü kullanan kümeleme algoritmaları, minimum ortalama belge mesafeli iki kümeyi birleştirmektedir.



Şekil 2.18: Grup ortalaması benzerlik ölçümü.

2.4.2.1. K-ortalamalar

K-ortalamalar kümeleme algoritması, en yaygın kullanılan bölümlemeli kümeleme algoritmasıdır. Algoritmanın popülerliği, sadeliği ve iyi sonuçlarından kaynaklanmaktadır. K-means algoritmasının genel adımları aşağıdaki şekildedir:

1) *k* küme sayısını belirle.
2) Durdurma kriterini belirle.
3) *k* adet noktayı kümelerin merkezi olarak belirle.
4) Her bir belgeyi en yakın kümeye ata.
5) Küme merkezlerini yeniden hesapla.
6) Eğer durdurma kriteri sağlandıysa algoritmayı durdur. Sağlanmadıysa 4. Adıma geri dön.(Tarczynski, 2011).

Birinci adımda yer alan *k* küme sayısının seçimi veri kümesine bağlı olmakla birlikte, bu sayının belirlenmesi için çeşitli sistematik yaklaşımlar geliştirilmiştir (Ray ve Turi, 1999; Tibshirani ve diğ., 2001; Sugar and James, 2003; Chiang ve Mirkin, 2007). İkinci adımdaki durdurma kriteri, maksimum iterasyon sayısı, belirlenen en küçük değişim değeri vb. olabilmektedir. Üçüncü adımda noktaların seçimi genellikle rastgele gerçekleştirilmektedir. Bir belgenin en yakın kümeye atanması için herhangi bir benzerlik ölçüsü kullanılmaktadır. Eğer belgelerin kümeler arasındaki hareketi sonlanmışsa, durdurma kriterine ulaşılmış demektir.

Standart *k*-ortalamalar algoritmasında bir kümenin merkezi, kümedeki belgelerin ağırlık merkezi olarak hesaplanmaktadır. Ancak bu konuda birçok değişiklik önerilmiştir. Bunlardan biri, bu adımda ortalamanın yerine medyan hesaplayan k-medyan algoritmasıdır. Hesaplanan medyan gerçek bir belge olmak zorunda değildir. Çünkü her terimin medyanı ayrı ayrı hesaplanmaktadır. Diğer bir değişiklik, belgelerden birini

küme merkezi olarak seçen k-medoid algoritmasıdır. Seçilen bu belge genellikle düşük ikili farklılığa sahiptir.

2.4.2.2. İkiye Bölmeli (Bisecting) K-ortalamalar

İkiye bölmeli k-ortamalar algoritması hiyerarşik ayırıcı bir algoritmadır ve bölümlemeli k-ortamalar algortiması ile karıştırılmamalıdır. Algoritmanın başında tüm belgeleri içeren tek bir küme vardır. Her bir iterasyonda seçilen küme iki kümeye bölünmektedir. Her küme tek bir belge içerene kadar algoritma çalışmaya devam eder. Algoritmanın adımları genel olarak aşağıda verilmiştir:

1) Bölmek için bir küme seç.
2) $k = 2$ için standart k-ortalamalar kümeleme algoritmasını uygula (ikiye bölme adımı).
3) İkiye bölme adımını iterasyon sayısı kadar tekrarla ve en yüksek benzerliğe sahip bölme işlemini sonuç olarak seç.
4) İstenilen sayıda kümeye ulaşıncaya kadar ilk üç adımı yinele.

Hangi kümeden bölme işlemine başlanacağının seçimi için farklı yollar mevcuttur. Örneğin, her aşamada en büyük küme veya en az genel benzerliğe sahip bir küme seçilebilmektedir; ya da her ikisine de dayalı bir ölçü kullanmak mümkündür.

İkiye bölme adımı ise uyarlanabilir bir adımdır ve bu adımda bölümlemeli kümeleme algoritmalarından herhangi bir kullanılabilmektedir. Ancak k-ortalamalar algoritması basit ve etkili bir algoritma olduğu için, bu aşamada kullanmak daha mantıklıdır.

İkiye bölmeli k-ortamalar algoritması, belgelerin sayısıyla doğrusal bir zaman karmaşıklığına sahiptir. Ancak küme sayısının büyük ve herhangi bir belgeyi elemenin mümkün olmadığı durumlarda, ikiye bölmeli k-ortamalar algoritması standart k-ortamalar algoritmasından daha etkilidir (Steinbach ve diğ., 2000; Tarczynski, 2011).

2.4.3. Metasezgisel Algoritmalar ile Belge Kümeleme

Geleneksel kümeleme yöntemlerinin mevcut sınırlamaları, çok boyutlu birçok optimizasyon probleminde olduğu gibi belge kümeleme alanında da metasezgisel algoritmalara yönelinmesine neden olmuştur. Özellikle daha esnek ve sağlam yapıdaki

sürü zekası tabanlı algoritmalar, K-ortalamalar gibi en yaygın kullanılan bölümlemeli algoritmalara göre daha iyi sonuçlar elde etmişlerdir.

PSO, literatürde bazı optimizasyon problemlerini çözmek için hem etkili hem de hızlı olduğunu kanıtlayan sürü zekası tabanlı algoritmalardan biridir. Belge kümeleme başlığı da optimal küme merkezlerini bulmayı amaçlayan bir optimizasyon problemi olarak ele alınırsa, PSO algoritmasının kümeleme çözümüne uygulanması söz konusudur. Cui ve diğerleri (2005), PSO tabanlı hibrid belge kümeleme algoritması önermişlerdir. hızlı Önerdikleri hibrid algoritmada, PSO algoritmasının küresel arama yeteneği ile K-ortalamalar algoritmasının yakınsama özelliğini birleştirmişler ve böylelikle her iki algoritmanın da dezavantajlarını önlemişlerdir. Algoritmayı literatürdeki dört belge seti üzerinde deneyerek, PSO ve K-ortalamalar algoritmaları ile karşılaştırmışlardır. Performans göstergesi olarak küme merkezleri ile belgeler arasındaki ortalama uzaklığı kullanmışlar ve denemeler sonucunda hibrid PSO algoritmasının daha anlamlı kümelemeler oluşturduğunu ortaya çıkarmışlardır. Cui ve Potok (2005) daha sonra, Cui ve diğerlerinin çalışmalarını genişleterek K-ortalamalar algoritması ile elde ettikleri kümeleme sonucunu PSO algoritmasının başlangıç çözümü olarak kullanarak yine aynı belge setleri üzerinde denemeler yapmışlardır. Bu çalışmalarındaki amaçları, hibrid PSO algoritmasında kullandıkları algoritmaların sırasının önemli olup olmadığını belirlemektir.

Machnik (2007), karınca kolonisi optimizasyonu tabanlı kümeleme yöntemi geliştirmiştir. Yöntemin belge kümeleme alanında uygulanabilirliğini kanıtlamak adına, literatürde en fazla kullanılan ve uygulanan üç kümeleme algoritması; K-ortalamalar, tek bağlantı ve grup ortalaması birleştirici algoritmalar; ile geliştirdiği algoritmanın karşılaştırmasını yapmıştır. Deneysel çalışmalarında, 20 haber grubu (20 newsgroup) ve Reuter-21578 belge kümelerini kullanmış ve zaman, saflık açısından algoritmaları değerlendirmiştir.

Cui ve diğerleri (2006), kuş ve balık gibi doğada sürü halinde hareket eden canlıların bu hareket yaklaşımına bağlı olarak, belge kümeleme algoritması oluşturmuşlardır. Bir yapay bir de 12 başlıktan oluşan 100 adetlik gerçek belge seti üzerinde yaptıkları deneyler sonucunda, geliştirdikleri algoritmanın performansını F-ölçüsü göstergesi ile

değerlendirerek; K-ortalamalar ve karınca kümeleme algoritmasına nazaran oluşturulan yeni algoritmanın daha iyi olduğunu görmüşlerdir.

Mahdavi ve Abolhassani (2009), k-ortalamalar algoritması ile armoni arama optimizasyon yöntemini birleştirerek belge kümelemeye uygulamışlardır. 5 belge seti için, tüm belgelerin küme merkezlerine uzaklıklarının ortalaması dikkate alınarak yapılan kümeleme sonuçları; k-ortalamalar, genetik k-ortalamalar, PSO ve Mises-Fisher üretken model tabanlı algoritma ile karşılaştırılmıştır.

Premalatha ve Natarajan (2010b), belge kümeleme problemi için hibrid PSO ve genetik algoritmalar yaklaşımı üzerine çalışmışlardır. Bu hibrid yaklaşım ile, genetik algoritmanın erken yakınsamayı önlemek için önceki çözümlerden yararlanma yeteneği PSO algoritması ile birleştirilmiştir. Yaklaşım, 3 farklı belge seti üzerinde denenerek Cui ve Potok (2005) tarafından öne sürülen K-ortalamalar ve PSO+K-ortalamalar algoritmaları ile kıyaslanmıştır ve başarısı ortaya konmuştur.

Gao ve Lu (2012), küme içi dağılımı minimize ederken kümeler arası ayrımı maksimum kılan, PSO tabanlı otomatik belge kümeleme algoritması geliştirmişlerdir. Uzaklık olarak kosinüs uzaklığını kullandıkları bu algoritmayı, k-ortalamalar, ikiye bölmeli k-ortalamalar, birleştirici kümeleme algoritmaları gibi çeşitli algoritmalarla karşılaştırmışlardır. Deneyler, 20 haber grubu (20 newsgroup) belge setinden alınan 10 farklı set ile TREC (Text Retrieval Conference) koleksiyonundan elde edilen 5 alt set üzerinde gerçekleştirilmiştir. Entropi, kesinlik, anma ve F-ölçüsü göstergelerine göre, PSO tabanlı otomatik belge kümeleme algoritması diğer algoritmalara göre çok daha iyi sonuçlar elde etmiştir.

Zaw ve Mon (2013), bazı guguk kuşu türlerinin zorunlu davranışlarından esinlenerek geliştirilen guguk kuşu arama optimizasyon algoritması ile web belgelerini kümelemişlerdir. Rastsal seçilen 300 web sayfası kosinüs uzaklıkları kullanılarak 100 iterasyon sonucunda 3 farklı kümeye ayrılmış ve kesinlik, anma ve f-ölçüsü göstergeleri hesaplanmıştır.

Karol ve Mangat (2013), hibrid PSO tabanlı belge kümeleme uygulamasını sunmuşlardır. İki farklı bölümlemeli kümeleme algoritmasını -bulanık c-ortalamalar ve k-ortalamalar- parçacık sürü optimizasyonu algoritması ile birleştirmişlerdir. Hibrid

algoritmaların performansını sırasıyle 2000 ve 1000 belgelik 20 haber grubu (20 newsgroup) ve Reuters-21578 belge setleri üzerinde deneyerek sonuçları farklı küme sayılarına göre, klasik bulanık c-ortalamalar ve k-ortalamalar algoritmaları ile f-ölçüsü ve entropi göstergeleri doğrultusunda karşılaştırmışlardır.

Akter ve Chung (2013), genetik algoritma tabanlı evrimsel bir algoritma geliştirerek belge kümeleme problemine çözüm geliştirmişlerdir. Tüm belge setine genetik algoritmayı uygulamak yerine, veri setini alt gruplara bölerek her birine genetik algoritma uygulamış ve daha sonra elde ettikleri sonuçlardan oluşan tüm set için yine genetik algoritma kullanmışlardır. 5 farklı başlıktan oluşan 1000 belgelik Reuters-21578 belge seti üzerinde, Davies-Bouldin indeksini ölçüt fonksiyonu olarak kullanıp yaptıkları deneme sonucunda, geliştirilen evrimsel algoritma k-ortalamalar ve genetik algoritmadan daha iyi sonuçlar vermiştir.

Azaryuon ve Fakhar (2013) standart karınca kümeleme algoritmasında tamamen rastgele olan karıncaların hareketini değiştirerek belge kümeleme için yeni bir algoritma önermişlerdir. Kümeleme işlemi sırasında her bir karıncanın, taşıdığı elemana benzer elemanların bulunduğu bölgeye doğru hareket etmesini, taşıyıcı olmayan karıncaların ise benzer olmayan elemanlar tarafından çevrili bir elemanın bulunduğu bölgeye doğru hareket etmesini sağlamışlardır. Bu hareket için, her bir karıncanın belgelerin konumunu gösteren evrensel bir haritaya sahip olduğunu varsaymışlardır. Önerilen modeli Reuters-21578 belge setinden alınan 50 belge üzerinde deneyerek algoritmanın performansını k-ortalamalar ve standart karınca kümeleme algoritmasına göre değerlendirmişlerdir.

Aly ve Kelleny (2014), guguk kuşu arama algoritması parametrelerinden yuva sayısını değişken tutarak, arama uzayını farklı sayıda kümelerle incelemişler ve bu algoritmayı belge kümeleme için kullanmışlardır. Algoritmanın performansını Çağdaş Arapça dizisinden alınan 12 farklı kategorideki belge üzerinde test ederek K-ortalamalar algoritması ile karşılaştırmışlardır.

Forsati ve diğerleri (2015), klasik arı kolonisi optimizasyonu algoritmasının bazı temel karakteristiklerini değiştirerek algoritmayı geliştirmişler ve hem kümeleme hem de belge kümelere veri setlerinde uygulamışlardır. Belge sayısı ve küme sayısı açısından farklı özellikteki 5 farklı belge seti üzerinde yapılan analizlerde, performans göstergesi

olarak F-ölçüsü kullanılmıştır. K-ortalamalar, genetik algoritma ve armoni arama tabanlı kümeleme algoritmaları ile yapılan karşılaştırmalar sonucunda, geliştirilmiş arı kolonisi optimizasyonu algoritması başarılı performans sergilemiştir.

2.4.4. Bir Belgenin Gösterimi

Belge kümelemede her bir verinin metin formundan standart sayısal formlara dönüştürülmesi gerekmektedir. Belgelerin sayısal formlarda en yaygın gösterimi Salton, Wong ve Yang (1975) tarafından sunulan "*Vektör Uzay Modeli*"dir. Terim vektör modeli olarak da adlandırılan vektör uzay modeli, aslında metin belgelerinin vektör olarak temsilini sağlayan cebirsel bir modeldir. Bu modelde, bir belgenin içeriği çok boyutlu uzayda biçimlendirilmekte ve bir d vektörü tarafından temsil edilmektedir. $i = 1,2, \dots, M$ ve $j = 1,2, \dots, N$ olmak üzere w_{ij} bir j belgesi içindeki i teriminin terim ağırlığını göstermekte ve belge kümesindeki j belgesi $d_j = \{w_{1j}, w_{2j}, \dots, w_{Mj}\}$ şeklinde ifade edilmektedir.

Terim ağırlık değeri, ele alınan bir terimin bir belge içinde ne kadar önemli olduğunu göstermektedir. Terim ağırlığını hesaplamak için en yaygın olarak Terim Frekansı-Ters Belge Frekansı (Term Frequency-Inverse Document Frequency, TD-IDF) yöntemi kullanılmaktadır. Bu yöntemde her bir ağırlık, *terim frekansı* ve *ters belge frekansı* olarak adlandırılan iki bileşenin ürünüdür.

Bir terimin bir belgede yer alma sayısına *terim frekansı* (tf) denmektedir. Terim frekansı, o terimin belirtilen belgede kaç kez bulunduğunun o belgedeki toplam terimlerin sayısına oranı olarak hesaplanmaktadır. Dolayısıyla bir belge içinde sıkça görülen bir terim, terim ağırlığının da artmasına neden olmaktadır. j belgesi içindeki i teriminin terim frekansı yani görülme sıklığı denklem 2.9'daki gibi tanımlanmaktadır:

$$tf_{ij} = {n_{ij}} \Big/ {\sum_{i=1}^{m} n_{ij}} \tag{2.9}$$

Ters belge frekansı (idf) ise ele alınan bir terimin tüm belgeler arasında yaygın veya nadir olup olmadığının bir ölçüsüdür. Ters belge frekansı, toplam belge sayısının incelenen terimi içeren belgelerin sayısına bölünmesi ve daha sonra bu bölümün logaritmasının alınması ile elde edilmektedir. Birçok belge setinde belge sayısının fazla

olması nedeniyle, logaritma fonksiyonu sayesinde bu sayı sıkıştırılmaktadır. i. terimin ters belge frekansı denklem 2.10'da gösterilmiştir.

$$idf_i = \log_2 \left({}^{N}/_{df_i} \right) = \log_2 \left({}^{N}/_{\left| \{ j \in N : t_i \in d_j \} \right|} \right) \tag{2.10}$$

Burada df_i i teriminin belge kümesi içindeki sıklığı, N ise belge kümesindeki toplam belge sayısıdır.

Ters belge frekansı aslında, kümeleme süreci üzerinde herhangi bir etkiye sahip olmayan ortak kelimeler ile ilgili bir sorunu çözmektedir. Bir kelimenin tüm belgelerde görülmesi durumunda, ters terim frekansında hem pay hem de payda eşit olmakta ve bu durumda terim ağırlığı otomatikman "0" değerini almaktadır. Bu durum "ve", "ama", "bir" vb. kelimelerin etkisini ortadan kaldırmaktadır (Tarczynski, 2011; Shah ve Mahajan, 2012). Dolayısıyla, ters belge frekansı sayesinde sadece birkaç belge içinde yer alan terimler, bulundukları belgeleri tüm set içinden rahatlıkla ayırabilmektedirler (Premalatha ve Natarajan, 2010a).

j belgesi içindeki i teriminin ağırlık değeri ise aşağıdaki gibi hesaplanmaktadır:

$$w_{ij} = tf_{ij} \times idf_i \tag{2.11}$$

Denklem 2.11 göstermektedir ki bir belge içinde sıkça görülen ancak tüm belge seti içinde nadir rastlanan terimler, belgeleri kümelemede yüksek güce sahiptir (Bisht ve Paul, 2013).

Buradaki ortak sorun, benzer denklemlerde de görülen 0 ile bölme işlemidir. Bu sorunun üstesinden gelmek için, iki yaklaşım sunulmuştur. İlk yaklaşım paydaya 1 ekleme fikridir, ancak bu sonucu değiştirecektir. Daha yaygın olarak kullanılan ikinci yaklaşımda ise herhangi bir belge içerisinde yer almayan kelimelerin kaldırıldığı kabul edilmektedir (Tarczynski, 2011).

Son olarak, farklı uzunluklardaki belgelerin hesabı için, her belge vektörün uzunluğu birim uzunluğa normalleştirilir; yani, her bir belge birim hiperkürenin bir vektördür. (Anastasiu ve diğ., 2013)

2.4.5. Belge Kümelemede Benzerlik Ölçüleri

Küme analiz yöntemleri genel olarak, bir nesne çifti arasındaki benzerliğin ölçülmesine dayanmaktadır. Bir nesne çifti arasındaki benzerliğin belirlenmesi, üç ana aşamayı içermektedir: nesneleri tanımlamak için kullanılacak değişkenlerin seçimi, bu değişkenler için bir ağırlık şemasının seçimi ve iki belge vektörü arasındaki benzerlik derecesini belirlemek için bir benzerlik ölçüsünün seçimi.

Benzerlik ölçüleri hemen hemen her kümeleme algoritmasının önemli parçalarından biridir. Bu ölçüler, sadece iki belge arasındaki benzerliği belirlemekle kalmayıp aynı zamanda bir belge ile bir küme arasındaki ya da iki küme arasındaki benzerliği hesaplamak için de kullanılabilmektedirler (Shah ve Mahajan, 2012). Benzerlik aslında hedef nesnelerin yakınlık ve ayrılma derecesini yansıtmakta ve verilerin içine gömülü kümeleri ayırt edeceğine inanılan özelliklere karşılık gelmektedir.

Bir D uzaklık ölçüsünün metrik olarak sayılabilmesi için aşağıdaki dört durumu sağlaması gerekmektedir. x ve y bir küme içindeki iki nesne ve $D(x,y)$, x ve y nesneleri arasındaki uzaklık olmak üzere;

1. Herhangi iki nesne arasındaki uzaklık negatif olmamalıdır. ($D(x,y) \geq 0$)
2. Özdeş iki nesne arasındaki uzaklık 0 olmalıdır. ($x \equiv y$ ise $D(x,y) = 0$)
3. Uzaklık simetrik olmalıdır. ($D(x,y) = D(y,x)$)
4. Ölçüm üçgen eşitsizliğini sağlamalıdır. Yani ($D(x,z) \leq D(x,y) + D(y,z)$) olmalıdır (Huang, 2008).

Kosinüs benzerliği vektör uzayında iki belge arasındaki benzerliği hesaplamak için en çok kullanılan katsayıdır. Bu katsayı, d_j ve d_k belge vektörleri olmak üzere, vektörler arasındaki kosinüs açısına bağlı olarak denklem 2.12'deki gibi ölçülmektedir.

$$\cos(d_j, d_k) = \frac{d_j \cdot d_k}{\|d_j\| \times \|d_k\|} = \frac{\sum_{i=1}^{M} w_{ij} \times w_{ik}}{\sqrt{\sum_{i=1}^{M} w_{ij}^2} \times \sqrt{\sum_{i=1}^{M} w_{ik}^2}} \tag{2.12}$$

Bu formül birim vektörler için $\cos(d_j, d_k) = d_j \cdot d_k$ şeklinde ifade edilmektedir. Eğer iki belge birbirinin özdeşi ise kosinüs benzerliği 1 değerini, eğer iki belge ortagonal ise

yani ortak herhangi bir terim içermiyorlarsa kosinüs benzerliği 0 değerini alır. Sonuç olarak kosinüs benzerliği, [0,1] arasında negatif olmayan bir değerdir.

Belgeleri karşılaştırmak için kullanılan diğer popüler katsayılar Minkowski, Öklid, Manhattan ve Chebyshev uzaklıkları ile Jaccard katsayısıdır.

M boyutlu vektör uzayında Minkowski uzaklığı aşağıdaki şekilde hesaplanmaktadır.

$$\text{Distance}(d_j, d_k) = \left(\sqrt{\sum_{t=1}^{M} |d_{t,j} - d_{t,k}|^n} \right)^{1/n} \tag{2.13}$$

$n = 2$ için bu uzaklık Öklid uzaklığına dönüşmektedir. Öklid uzaklığı geometrik problemler için kullanılan standart bir ölçüdür.

$$\text{Distance}_2(d_j, d_k) = \sqrt{\sum_{t=1}^{M} (d_{t,j} - d_{t,k})^2} \tag{2.14}$$

Burada $d_{t,j}$, d_j belgesindeki t-inci elemanı göstermektedir. Öklid uzaklığının 0 olması durumunda belgelerin birbirleri ile aynı olduğu sonucuna varılmaktadır. Belgeler arasında ortak bir nokta olmaması durumunda ise uzaklık $\sqrt{2}$ olarak hesaplanmaktadır.

Minkowski uzaklığı esas alınarak hesaplanan Manhattan ve Chebyshev uzaklıkları ise sırasıyla denklem 2.15 ve 2.16 ile hesaplanmaktadır:

$$\text{Distance}_1(d_j, d_k) = \sum_{t=1}^{M} |d_{t,j} - d_{t,k}| \tag{2.15}$$

$$\text{Distance}_\infty(d_j, d_k) = \max_{t=1} |d_{t,j} - d_{t,k}| \tag{2.16}$$

Jaccard katsayısı, ortak terimlerin toplam ağırlığı ile iki belgede bulunan ancak ortak olmayan terimlerin toplamını karşılaştırır. Buna göre D_j ve D_k, d_j ve d_k belgelerinin terim setlerini göstermek üzere, Jaccard katsayısı aşağıdaki şekilde hesaplanmaktadır (Anastasiu ve diğ., 2013):

$$\mathrm{J}(d_j, d_k) = \frac{|D_j \cap D_k|}{|D_j \cup D_k|} \tag{2.17}$$

Belge benzerlik ölçülerini, bir kümedeki tüm belgelerin benzerlik ölçüsü olarak genişletebilmek mümkündür. Bunlara *küme içi benzerlik (ICS) ölçüleri* denmektedir. En yaygın olarak kullanılan küme içi benzerlik ölçüsü grup ortalama benzerliğidir. Bu, kümedeki belgelerin ikili benzerliğinin ortalaması olarak tanımlanmakta ve aşağıdaki şekilde ifade edilmektedir.

$$\mathrm{ICS} = \frac{\sum_{d_j, d_k \in C_i} \mathrm{Distance}(d_j, d_k)}{n_i^2} \tag{2.18}$$

Burada C_i, i-inci kümeyi; n_i ise bu kümedeki elemanların sayısını ifade etmektedir. Küme içi benzerlik ölçüsünü hesaplarken kosinüs benzerliği kullanıldığında 2.18 eşitliğinin basitleştirilmesi mümkündür. Bunun için bir ağırlık merkezinin tanımlanmasına ihtiyaç duyulmaktadır. Bir kümenin ağırlık merkezi, tüm kümeyi ifade eden bir vektördür ve bir kümedeki belgeleri temsil eden tüm vektörlerin ortalaması olarak tanımlanmaktadır. i-inci kümenin ağırlık merkezi c_i olarak ifade edilirse;

$$c_i = \frac{1}{n_i} \cdot \sum_{d_j \in C_i} d_j \tag{2.19}$$

şeklinde hesaplanmaktadır. Buradan küme içi benzerlik değerinin, ağırlık merkezi vektörünün uzunluğunun karesine eşit olduğu kolayca kanıtlanabilmektedir.

$$ICS = \|c_i\|^2 \tag{2.20}$$

Dolayısıyla, iki küme arasındaki benzerliğin hesaplamasını, kendi ağırlık merkezlerinin benzerliğinin hesaplaması olarak ifade etmek mümkündür (Tarczynski, 2011).

2.4.6. Kümeleme Ölçüt Fonksiyonları

Kümeleme problemi, belirli bir kümeleme ölçüt fonksiyonunu optimize edecek şekilde, k-yönlü küme çözümünün hesaplanmasına dayanmaktadır. Kümeleme ölçüt fonksiyonlarının gösterimine geçmeden önce kullanılacak notasyonlar ile ilgili kısa hatırlatmalar yapılması gerekmektedir. N belgelerin sayısını, M terimlerin sayısını, K kümelerin sayısını göstermekle birlikte; $C_1, C_2, \ldots, C_K$ K kümeden her birini,

$n_1, n_2, \ldots, n_K$ ise ilgili kümenin boyutunu belirtmektedir. Ayrıca bu başlık altında kullanılan d_j, d_k, d_q ... gibi ifadeler belge vektörlerini göstermektedirler.

Verilen bir A belge setine ait bileşik vektör D_A ve ağırlık merkezi vektörü c_A sırasıyla Denklem 2.21 ve denklem 2.22 kullanılarak hesaplanmaktadır.

$$D_A = \sum_{d \in A} d \tag{2.21}$$

$$c_A = \frac{D_A}{|A|} \tag{2.22}$$

Görüldüğü üzere bileşik vektör, A belge setindeki tüm belge vektörlerinin toplamıdır. Ağırlık merkezi vektörü ise A belge setindeki belgelerde mevcut olan çeşitli terimlerin ağırlıkları ortalamasıdır.

Bu bilgiler ışığında kümeleme ölçüt fonksiyonları aşağıdaki şekilde ifade edilmektedir.

2.4.6.1. İçsel Ölçüt Fonksiyonları

Bu tip kümeleme ölçüt fonksiyonları, her kümenin parçası olan tüm belgeler üzerinde tanımlanmış belirli bir kriter fonksiyonunu optimize edecek şekilde kümeleme çözümü üretmeye odaklanırlar. Bu fonksiyonlar farklı kümelere atanmış belgeleri dikkate almazlar. Kümeleme sürecinin bu küme içi görüşü nedeniyle bu ölçüt fonksiyonlarına içsel denmektedir.

İlk içsel ölçüt fonksiyonu, her kümeye atanan belgeler arasındaki ikili benzerliklerin toplamının ortalamasını, her kümenin büyüklüğüne göre ağırlıklı olarak maksimize etmektedir. Başka bir ifade ile bu ölçüt, her bir kümede yer alan belgelerin ikili karşılaştırmalarının ortalamasının ağırlıklı toplamıdır. Özellikle belgelerin arasındaki benzerliği ölçmek için kosinüs fonksiyonu kullanıldığında, aşağıda belirtilen bu ölçüt fonksiyonunu optimize edecek şekilde kümeleme çözümü aranmaktadır:

$$\text{maks } \mathrm{I}_1 = \sum_{r=1}^{K} n_r \left(\frac{1}{n_r^2} \sum_{(d_j, d_k) \in C_r} cos(d_j, d_k) \right) \tag{2.23}$$

Belgeler arasındaki benzerliğin ölçülmesinde kosinüs fonksiyonu kullanıldığında, belge setlerine ait bileşik ve ağırlık merkezi vektörlerini içeren çeşitli özelliklerden yararlanmak mümkündür. C_j ve C_k, sırasıyla n_j ve n_k belgelerini içeren birim uzunluktaki belgelerin iki kümesi olmak üzere; D_j ve D_k bu setlerin bileşik vektörleri, c_j ve c_k ise bu setlerin ağırlık merkezi vektörleridir. Bu durumda C_j kümesindeki belgeler arasındaki ikili benzerliklerin toplamı $\left\|D_j\right\|^2$ olarak ifade edilebilmektedir. Şöyle ki;

$$\sum_{d_q,d_r\in D_j} \cos\left(d_q, d_r\right) = \sum_{d_q,d_r\in D_j} d_q^T d_r = D_j^T D_j = \left\|D_j\right\|^2 \tag{2.24}$$

Buradan 2.23 eşitliği aşağıdaki şekilde yeniden yazılabilmektedir:

$$\text{maks } \mathrm{I}_1 = \sum_{r=1}^{K} \frac{\|D_r\|^2}{n_r} \tag{2.25}$$

İkinci içsel ölçüt fonksiyonu, k-ortalamalar algoritmasının vektör-uzay biçimlerinde kullanılmaktadır. Bu algoritmada her bir küme, kendisinin ağırlık merkezi vektörü ile ifade edilmekte ve her bir belge ile atandığı kümenin ağırlık merkezi arasındaki benzerliği maksimum kılacak kümeleme çözümünü bulma amaçlanmaktadır. Belge ile kümenin ağırlık merkezi birbirine ne kadar benzerse kosinüs benzerlik değeri artacaktır. Dolayısıyla ölçüt fonksiyonu maksimize edilmektedir. Belge ile ağırlık merkezi arasındaki benzerliğin ölçülmesinde kosinüs fonksiyonu kullanıldığında ölçüt fonksiyonu denklem 2.26 ile ifade edilmektedir:

$$\text{maks } \mathrm{I}_2 = \sum_{r=1}^{K} \sum_{d_j\in C_r} cos\left(d_j, c_r\right) \tag{2.26}$$

Denklem 2.26'yı aşağıdaki şekilde yazmak mümkündür:

$$\text{maks } \mathrm{I}_2 = \sum_{r=1}^{K} \sum_{d_j\in C_r} \frac{d_j^T c_r}{\|c_r\|} = \sum_{r=1}^{K} \frac{D_r^T c_r}{\|c_r\|} = \sum_{r=1}^{K} \frac{D_r^T D_r}{\|D_r\|} = \sum_{r=1}^{K} \|D_r\| \tag{2.27}$$

Diğer bir içsel ölçüt fonksiyonu, geleneksel k-ortalamalar algoritmasında kullanılmaktadır. Bu fonksiyon, hangi belgelerin birlikte kümelenmesi gerektiğini

belirlemek için öklid uzaklığını kullanmakta ve kümeleme sonucunun kalitesini hata karelerinin toplamı fonksiyonunu kullanarak hesaplamaktadır. Bu ölçüt fonksiyonu Denklem 2.28'de belirtilmiştir:

$$\min \mathrm{I}_3 = \sum_{r=1}^{K} \sum_{d_j \in C_r} \left\| d_j - c_r \right\|^2 \tag{2.28}$$

Bazı basit cebirsel hileler ile denklem 2.29 aşağıdaki şekilde yeniden yazılabilmektedir:

$$\min \mathrm{I}_3 = \sum_{r=1}^{K} \frac{1}{n_r} \sum_{d_j, d_k \in C_r} \left\| d_j - d_k \right\|^2 \tag{2.29}$$

2.4.6.2. Dışsal Ölçüt Fonksiyonları

Dışsal ölçüt fonksiyonları içsel ölçüt fonksiyonlarının aksine, çeşitli kümelerin birbirinden ne kadar farklı olduğuna dayanan bir fonksiyon optimizasyonuna odaklanarak kümeleme çözümü türetmektedir.

Anlamlı kümeleme çözümlerine götüren dışsal ölçüt fonksiyonlarını tanımlamak oldukça zordur. Farklı kümelerin ağırlık merkezi vektörlerinin mümkün olduğu kadar karşılıklı ortogonal olması durumunda; yani, kümelerin çok az terimi paylaşan belgeleri içerdiği durumda dışsal ölçüt fonksiyonu türetilebilmektedir. Bununla birlikte, birçok problem için bu ölçüt fonksiyonu önemsiz çözümler vermektedir. Bu tip çözümlere; diğer belgelerle çok az sayıda ortak terimi olan tek bir belgenin ilk $k-1$ adet kümeye atanması ve sonrasında diğer belgelerin k'ıncı kümeye atanması durumunda ulaşılmaktadır.

Dolayısıyla ele alınacak dışsal fonksiyonlar, belgeleri farklı kümeler arasında ayırmanın aksine, her bir kümedeki belgeleri bütün belge setinden ayırmaktadır. İlk dışsal ölçüt fonksiyonu $\mathcal{E}_1$ aşağıdaki şekilde ifade edilmektedir:

$$\min \mathcal{E}_1 = \sum_{r=1}^{K} n_r \cos\left(c_r, c\right) \tag{2.30}$$

Burada c, tüm belge setinin ağırlık merkezi vektörüdür. Denklem 2.30'dan her bir kümenin ağırlık merkezi vektörü ile, tüm belge setinin ağırlık merkezi vektörü

arasındaki kosinüs uzaklığının minimize edilmeye çalışıldığı görülmektedir. Uzaklığın minimizasyonu ile aslında aradaki açının olabildiğince yükseltilmesi denenmektedir. Ayrıca her bir kümenin katkısının, küme boyutu ile ağırlıklandırıldığı; dolayısıyla büyük kümelerin ağırlığının tüm kümeleme sonucunda daha etkili olacağı unutulmamalıdır.

Denklem 2.30'u çoklu diskriminant analizi ile denklem 2.31 şeklinde yazmak mümkündür.

$$\min \mathcal{E}_1 = \sum_{r=1}^{K} n_r \frac{c_r^T c}{\|c_r\|\|c\|} = \sum_{r=1}^{K} n_r \frac{D_r^T D}{\|D_r\|\|D\|} = \frac{1}{\|D\|}\left(\sum_{r=1}^{K} n_r \frac{D_r^T D}{\|D_r\|}\right) \tag{2.31}$$

$1/\|D\|$ sabitine bakılmaksızın ölçüt fonksiyonu aşağıdaki şekilde yeniden ifade edilebilmektedir:

$$\min \mathcal{E}_1 = \sum_{r=1}^{k} n_r \frac{D_r^T D}{\|D_r\|} \tag{2.32}$$

Başka bir dışsal ölçüt fonksiyonu ($\mathcal{E}_2$), öklid uzaklık fonksiyonu ve ağırlık merkezi vektörlerinin hata karelerine göre aşağıdaki gibi tanımlanabilmektedir:

$$\min \mathcal{E}_2 = \sum_{r=1}^{K} n_r \|c_r - c\|^2 \tag{2.33}$$

Görüldüğü üzere, $\mathcal{E}_2$ dışsal ölçüt fonksiyonunu minimize etmek aslında I_3 içsel ölçüt fonksiyonunu maksimize etmeye benzemektedir.

2.4.6.3. Hibrid Ölçüt Fonksiyonları

İçsel ölçüt fonksiyonları her bir küme içindeki belgelerin çeşitli benzerliklerini maksimize etmeye çalışırken, dışsal ölçüt fonksiyonları küme içindeki belgelerle toplamdaki tüm belgeler arasındaki benzerliği miminum kılmaya çalışmaktadır. Bununla birlikte, çeşitli ölçüt fonksiyonlarını birleştirip hibrid ölçüt fonksiyonları olarak tanımlayarak aynı anda birden fazla ölçüt fonksiyonunun optimize edilmesi mümkündür. Hibrid ölçüt fonksiyonlarına aşağıdaki örnekler verilebilir:

$$\text{maks } \mathcal{H}_1 = \frac{\mathrm{I}_1}{\mathcal{E}_1} = \frac{\sum_{r=1}^{K} n_r \left(\frac{1}{n_r^2}\sum_{(d_j,d_k)\in C_r} cos\left(d_j, d_k\right)\right)}{\sum_{r=1}^{K} n_r \cos\left(c_r, c\right)} = \frac{\sum_{r=1}^{K} \|D_r\|^2/n_r}{\sum_{r=1}^{K} n_r\, D_r^T D/\|D_r\|} \tag{2.34}$$

$$\text{maks } \mathcal{H}_2 = \frac{\mathrm{I}_2}{\mathcal{E}_1} = \frac{\sum_{r=1}^{K}\sum_{d_j\in C_r} cos\left(d_j, c_r\right)}{\sum_{r=1}^{K} n_r \cos\left(c_r, c\right)} = \frac{\sum_{r=1}^{K}\|D_r\|}{\sum_{r=1}^{K} n_r\, D_r^T D/\|D_r\|} \tag{2.35}$$

$\mathcal{E}_1$ ölçütü minimize edilmeye çalışıldığından ve paydada yer aldığından; $\mathcal{H}_1$ ve $\mathcal{H}_2$ değerlerinin maksimum olması istenmektedir (Zhao ve Karypis, 2001; Zhao ve Karypis, 2002; Zhao ve Karypis, 2004).

2.4.7. Kümeleme Algoritmasının Değerlendirilmesi

Küme analizinin en önemli konularından biri, kümeleme sonuçlarının değerlendirilmesidir. Değerlendirme, çıktıların orijinal veri yapısını ne kadar iyi temsil ettiğini anlamak için yapılan analizdir.

Kümeleme sonuçlarının değerlendirilmesi iki bölümden oluşmaktadır:

- İç kalite göstergesi: Burada, belge çiftleri arasındaki benzerliğe dayanan genel bir kalite göstergesi kullanılmaktadır ve dışarıdan bir bilgiye ihtiyaç duyulmamaktadır. Dunn indeksi, Davies-Bouldin indeksi, Calinski-Harabasz indeksi, Silhouette indeksi bu göstergelerden bazılarıdır.
- Dış Kalite göstergesi: Veri için bazı dış bilgiler gerekmektedir. Entropi, F-ölçüsü, saflık (purity), doğruluk vb. bu kapsamda incelenen göstergelerdir.

2.4.7.1. Dunn indeksi

Dunn indeksi kümeler arasındaki uzaklıkları maksimum, küme içi uzaklıkları ise minimum yapmaya çalışmaktadır. Eğer bir belge seti iyi ayrılmış kümeler içeriyorsa, kümeler arasındaki uzaklıkların genellikle büyük ve küme çaplarının küçük olması beklenmektedir. Dolayısıyla büyük bir Dunn indeksi değeri, daha iyi küme yapılandırmasını ifade etmektedir.

Literatürde çeşitli Dunn benzeri indeksler ileri sürülmüştür. Bu indeksler küme uzaklığı ve küme çapı için farklı tanımlama kullanmaktadırlar.

Dunn indeksi denklem 2.36 ile hesaplanmaktadır:

$$D = \min_{i=1,2,\ldots,K} \left\{ \min_{j=1,2,\ldots,K} \left\{ \frac{\text{Distance}(c_i, c_j)}{\max_{r=1,2,\ldots,K} d(C_r)} \right\} \right\} \tag{2.36}$$

Burada $d(c_i, c_j)$, $i.$ ve $j.$ kümeler arasındaki uzaklıkları ifade ederken; $d(C_r)$, $r.$ kümenin küme içi uzaklıkları toplamını göstermektedir.

2.4.7.2. Davies-Bouldin indeksi (DBI)

Bu indeks, David L. Davies ve Donald W. Bouldin (1979) tarafından kümeleme algoritmalarını değerlendirmek için geliştirilmiştir. İndeks, küme içi saçılmaların (scatter) toplamının kümeler arası ayrılmaya oranının bir fonksiyonudur ve hem kümeleri hem de bunların örnek ortalamalarını kullanmaktadır.

Buna göre DBI denklem 2.37 ile hesaplanmaktadır.

$$\text{DBI} = \frac{1}{K} \sum_{i=1}^{K} \max_{i \neq j,} \left\{ \frac{\overline{S_i} + \overline{S_j}}{\text{Distance}(c_i, c_j)} \right\} \tag{2.37}$$

$\overline{S_i}$ ve S_j sırasıyla $i.$ ve $j.$ kümelerinin iç dağılımlarını göstermektedir. Başka bir deyişle herhangi bir kümenin iç dağılımı, belirtilen kümeye atanan belgelerin küme merkezine uzaklıklarının ortalamasıdır ve denklem 2.38'e göre hesaplanmaktadır.

$$\overline{S_i} = \frac{1}{n_i} \sum_{r=1}^{n_i} \text{Distance}(d_r, c_i) \tag{2.38}$$

n_i, $i.$ kümenin belge sayısını göstermektedir. Optimal küme çözümü, küçük DBI değerine sahiptir.

2.4.7.3. Calinski-Harabasz indeksi (CHI)

Varyans oran kriteri olarak da isimlendirilen bu indeks, denklem 2.39 ile hesaplanmaktadır:

$$\text{CHI} = \frac{SS_B}{SS_w} \times \frac{N - K}{K - 1} \tag{2.39}$$

Burada SS_B kümeler arası varyansların SS_w ise küme içi varyansların tüm toplamıdır. K küme sayısı, N ise tüm belgelerin sayısıdır.

Kümeler arası varyansların toplamı şu şekildedir:

$$SS_B = \sum_{i=1}^{K} n_i(\text{Distance}_2(c_i, c))^2 \tag{2.40}$$

c_i, i. küme merkezini ifade ederken c tüm belgelerin ortalamasıdır. $\text{Distance}_2(c_i, c)$ ise öklid uzaklığını ifade etmektedir.

Küme içi varyansların toplamı ise denklem 2.41 ile hesaplanmaktadır.

$$SS_w = \sum_{i=1}^{K} \sum_{d_j \in C_i} (\text{Distance}_2(d_i, c_i))^2 \tag{2.41}$$

İyi bir kümeleme sonucunda kümeler arası varyans büyük, küme içi varyans ise küçük bir değer olmalıdır. Dolayısıyla doğru yapılmış bir kümeleme işlemi sonucunda CHI'nın büyük olması istenmektedir.

2.4.7.4. Silhouette indeksi (SI)

Silhouette indeksi, herhangi bir kümeye atanmasına bağlı olarak her bir belge için genişlik hesaplamakta ve tüm belgeler üzerinden bir ortalama almaktadır. i-inci belgenin silhouette indeksi aşağıdaki gibi hesaplanmaktadır:

$$SI_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{2.42}$$

Burada a_i, i'inci belgenin aynı küme içindeki diğer belgelere uzaklıklarının ortalaması, b_i ise i'inci belgenin diğer kümelerdeki belgelere en küçük ortalama uzaklığıdır.

Silhouette indeksi -1 ile +1 arasında değer almaktadır. Yüksek bir değer, o belgenin kendi kümesi ile iyi örtüştüğünü ve diğer kümeler ile uyuşmadığını göstermektedir. Dolayısıyla, birçok belgenin yüksek SI değerine sahip olması, kümeleme çözümünün uygun olduğunu göstermektedir (Das ve diğ., 2009; Rendón ve diğ., 2011).

2.3.7.5. Doğruluk (Accuracy)

Doğruluk, doğru kümelenmiş belgelerin toplam belge sayısına oranından denklem 2.43'teki gibi hesaplanmaktadır.

$$Doğruluk = \sum_{i=1}^{K} n_i^+ \Big/ n \tag{2.43}$$

Burada n_i^+, i. kümedeki doğru atanmış belgelerin sayısıdır.

2.4.7.6. Entropi

Her bir küme için öncelikle belgelerin kategori dağılımı hesaplanmakta ve j-inci kümedeki bir belgenin i-inci kategoriye ait olma olasılığı p_{ij} olarak gösterilmektedir. Buna göre, her bir j kümesinin entropisi aşağıdaki şekilde hesaplanmaktadır:

$$E_j = -\sum p_{ij} \log (p_{ij}) \tag{2.44}$$

Toplam entropi ise, her bir kümenin boyutu doğrultusunda ağırlıklandırılan küme entropilerinin toplamıdır ve denklem 2.45 ile hesaplanmaktadır:

$$E_{en} = \sum_{j=1}^{K} \frac{n_j}{n} E_j \tag{2.45}$$

Burada k toplam küme sayısını, n_j j'inci kümenin boyutunu, n ise toplam belge sayısını göstermektedir.

2.4.7.7. F-ölçüsü

F-ölçüsü, bilgi alma kapsamındaki *kesinlik* (precision) ve *anma* (recall) kavramlarının harmonik ortalamasıdır. Kesinlik; doğru kümelenmiş pozitif belge sayısının, pozitif olarak kümelenmiş toplam belge sayısına oranıdır. Anma ise doğru olarak sınıflandırılmış pozitif belgelerin sayısıdır. Örnek üzerinde bu kavramları daha net bir şekilde ifade edecek olursak; A, B ve C şeklinde üç kümeye ayrılan bir belge setini düşünelim. Kümeleme sonucunda elde edilen karışıklık matrisi (confusion matrix) Tablo 2.2'de verildiği gibi olsun.

Tablo 2.2: Karışıklık matrisi (confusion matrix).

		Kümeleme Sonucu Elde Edilen Atamalar		
Kümelerin Gerçek Atamaları		A	B	C
	A	tp_A	e_{AB}	e_{AC}
	B	e_{BA}	tp_B	e_{BC}
	C	e_{CA}	e_{CB}	tp_C

Karışıklık matrisi, kümeleme işleminin gerçek duruma göre nasıl yapıldığını göstermektedir. Dolayısıyla diyagonal elemanlar, her küme için doğru yapılan atamaların sayısını ifade ederken, diyagonal olmayan elemanlar hatalı küme atamalarını belirtmektedir. Bu bilgiler doğrultusunda örneğin A kümesi için kesinlik, anma ve F-ölçüsü değerleri aşağıdaki gibi hesaplanmaktadır.

$$Kesinlik\ (A) = \frac{tp}{tp + fp} = \frac{tp_A}{tp_A + e_{BA} + e_{CA}} \tag{2.46}$$

$$Anma(A) = \frac{tp}{tp + fn} = \frac{tp_A}{tp_A + e_{AB} + e_{AC}} \tag{2.47}$$

$$F - \text{ölçüsü}: F\ (A) = \frac{2 \times Kesinlik\ (A) \times Anma(A)}{Kesinlik\ (A) + Anma(A)} \tag{2.48}$$

Basit bir ifadeyle yüksek kesinlik, bir algoritmanın oldukça doğru atama yaptığını; yüksek anma ise bir algoritmanın doğru atamaların çoğunu bulduğunu ifade etmektedir.

Tüm kümeleme için F-ölçüsü ise, tüm F-ölçüsü değerlerinin ağırlıklı ortalaması alınarak aşağıdaki şekilde hesaplanmaktadır:

$$F = \sum_{i=1}^{K} \frac{n_i}{n} \{F\ (i)\} \tag{2.49}$$

Yüksek F-ölçüsü kümelemenin doğru yapıldığının göstergesidir (Tarczynski, 2011; Shah ve Mahajan, 2012)

2.4.8. Belge Kümelemede Karşılaşılan Zorluklar

Belge kümeleme sürecinde karşılaşılan zorluklardan bazıları şunlardır:

- Kümeleme işleminde kullanılmak üzere belgelerin uygun özelliklerinin seçimi
- Belgeler arasında uygun bir benzerlik ölçüsünün seçimi
- Seçilen benzerlik ölçüsünü kullanan uygun bir kümeleme yönteminin seçimi
- Gerekli bellek ve CPU kaynakları açısından kümeleme algoritmasının etkin bir şekilde uygulanması.
- Elde edilen kümelerin kalitesini değerlendirme yollarının bulunması.
- Yeni belgelerin eklenmesi durumunda kümeleme işleminin güncellenmesi için mümkün yolların bulunması

Ayrıca, büyük belge koleksiyonlarında (10.000 belge ve üzeri), terim-belge ilişkilerinin sayısı oldukça yüksek (milyon üzeri) olduğundan ve uygulanan algoritmanın hesaplama karmaşıklığı söz konusu olacağından; gerçek hayat problemleri için belge kümeleme işleminin uygun olup olmadığı önemli bir faktördür. Terim-belge ilişkilerini temsil eden yoğun bir matris oluşturulursa, bu matrisin bellekte tutulması için çok büyük alan gerekecektir. Örneğin $100.000\ belge \times 100.000\ terim = 10^{10}\ giriş$ için yaklaşık 32 bit kullanılarak 40 GB hafıza kullanılacaktır. Vektör modeli uygulanırsa, elde edilen vektör uzayı aynı şekilde oldukça yüksek boyut olacaktır. Dolayısıyla, vektör uzayında iki belge arasındaki öklid uzaklığını bulma gibi basit işlemler, zaman alıcı görevler haline gelecektir. (Vester ve Martiny, 2005)

3. MALZEME VE YÖNTEM

Bu bölümde, belge kümeleme probleminin çözümü için kullanılacak metasezgisel yöntem olan yarasa algoritmasının varsayımları, akışı ve formülasyonu üzerinde durularak, algoritmanın deneneceği belge setleri detaylı bir şekilde anlatılacaktır.

3.1. YARASA ALGORİTMASININ YAPISI

Yarasa algoritmasının temelini oluşturan yarasaların ekolokasyon özelliği bölüm 2.2.3.13'te sunulmuştur. Bu bölümde ise algoritmanın işleyişine yer verilecektir.

Yarasaların ses yankısına dayalı arama hareketlerini, optimize edilmek istenen problemlerin amaç fonksiyonu ile ilişkilendiren Yang (2010d), yarasa algoritmasını oluşturarak bunun için çeşitli varsayımlar yapmıştır.

i. Tüm yarasalar mesafe tespiti için ses yankısını kullanmakta ve besin kaynağı (av) ile çevredeki engeller arasındaki farkı sezgisel güçleriyle algılayabilmektedirler.
ii. Yarasalar av aramak için rastgele olarak x_i pozisyonunda v_i hızıyla sabit f_{min} frekansı (ya da λ değişken dalga boyu) ve A_0 ses yüksekliği ile uçmaktadırlar. Yarasalar otomatik olarak, yaydıkları sinyallerin dalga boyunu (veya frekansını) ve hedeflerinin yakınlığına bağlı olarak sinyal emisyon oranını ($r \in [0,1]$ olmak üzere) ayarlayabilmektedirler.
iii. Ses yüksekliği birçok yönden farklılık göstermesine rağmen ses yüksekliğinin büyük bir (pozitif) A_0 değerinden minimum sabit A_{min} değerine değiştiği varsayılmaktadır.
iv. Besin kaynağının (avın) sabit olduğu, yarasaların hareketi süresinde yer değiştirmediği farz edilmektedir.

Algoritmayı oluştururken yapılan başka basitleştirmelerden biri de kullanılan frekans yaklaşımıdır. Genel olarak, f frekansına ait $[f_{min}, f_{maks}]$ aralığı, $[\lambda_{min}, \lambda_{maks}]$ dalga boyu aralığına karşılık gelmektedir. Örneğin $[20kHz, 500kHz]$ frekans aralığı 0,7 mm den 17 mm'ye uzanan bir dalga boyu aralığına uymaktadır.

Verilen bir problem için, uygulama kolaylığı açısından herhangi bir dalga boyu kullanılabilmektedir. Uygulama esnasında, dalga boyları (veya frekanslar) düzeltilerek aralık ayarlanabilmektedir. Tespit edilen aralık (ya da en büyük dalga boyu), ele alınan problemin boyutu ile karşılaştırılabilir şekilde seçilmelidir ve daha sonra küçük aralıklara alçaltılmalıdır. Ayrıca, dalga boyu λ'yı sabit tutarak frekans değişikliği yapmak da mümkündür. λf değerinin sabit olması nedeniyle λ ve f değerleri birbirleriyle bağlantılıdır.

Kolay olması açısından $f \in [0, f_{maks}]$ varsayılmıştır. Yüksek frekansların kısa dalga boyuna sahip olduğu ve daha kısa mesafede hareket ettiği bilinmektedir. Yarasalar için, aralıklar birkaç metredir ve sinyal oranı sadece [0, 1] aralığında olabilmektedir. Burada 0 hiçbir sinyal olmadığı anlamına gelirken, 1 maksimum sinyal emisyon oranı anlamına gelmektedir.

Bu varsayımlar ışığında, yarasa algoritmasının temel adımları Şekil 3.1'de yer alan sözde kod ile özetlenebilir:

YARASA ALGORİTMASI

Amaç fonksiyonu $f(x)$, $x = (x_1,..., x_d)^T$
x_i (i =1,2,...,n) yarasa popülasyonunu ve v_i hızını belirle.
x_i konumundaki f_i frekansını tanımla.
r_i sinyal oranı ve A_i ses yüksekliğini belirle.
while $t <$ *maksimum iterasyon sayısı*
    Frekansı ayarlayarak yeni sonuçlar üret.
    Hızları ve konumları(çözümleri) güncelleştir.
        if $rand > r_i$,
        En iyi sonuçlar arasından bir sonuç seç.
        Seçilen bu sonuç etrafından yerel bir sonuç türet.
        end if
        Rastgele uçarak bir sonuç üret.
            if ($rand < A_i$) ve ($f(x_i) < f(x_*)$)
            Yeni sonuçları kabul et.
            r_i değerini arttır ve A_i değerini düşür.
            end if
        Yarasaları sırala ve mevcut en iyiyi (x_*) bul.
end while
Sonuçları ve görselleri işle.

Şekil 3.1: Yarasa algoritmasının sözde kodu (Yang, 2010d).

3.1.1. Sanal Yarasaların Hareketi

Yarasaların *d* boyutlu bir arama uzayında x_i konumlarının ve v_i hızlarının nasıl güncelleştirileceğinin belirlenmesi gerekmektedir. *t* zamanındaki çözümler x_i^t ve hızlar v_i^t olmak üzere;

$$f_i = f_{min} + (f_{maks} - f_{min})\beta \tag{3.1}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i \tag{3.2}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{3.3}$$

hesaplanır. Burada;

β: rassal bir vektör ($\beta \in [0,1]$)

x_*: tüm *n* adet yarasa arasındaki çözümlerin karşılaştırılması sonucunda bulunan mevcut küresel en iyi konumdur.

$\lambda_i f_i$ çarpımı hız artışına neden olduğundan, problemin tipine göre λ_i (ya da f_i) değeri sabit tutulurken f_i (veya λ_i) değeri hız değişimini ayarlamak için kullanılabilmektedir. Her bir yarasaya başlangıçta $[f_{min}, f_{maks}]$ aralığında düzgün dağılmış bir frekans rassal olarak atanmaktadır.

Algoritmanın yerel arama kısmı için, mevcut en iyi çözümler arasından bir çözüm seçildiğinde, her yarasa için yeni bir çözüm rastgele yürüyüş ile lokal olarak denklem 3.4'teki gibi oluşturulur.

$$x_{yeni} = x_{eski} + \varepsilon A^t \tag{3.4}$$

Burada $\varepsilon \in [-1,1]$ olmak üzere bir rassal sayıyı gösterirken, $A^t = < A_i^t >$ tüm yarasaların *t* zamanındaki ortalama ses yüksekliğidir.

Yarasaların hız ve konumlarının güncelleştirilmesi işlemi, aslında standart parçacık sürü optimizasyonundaki prosedürle bazı benzerlikler taşımaktadır. f_i parçacıkların hareket hızını ve aralığını kontrol etmektedir. Bu nedenle yarasa algoritması bir dereceye kadar,

standart parçacık sürü optimizasyonu ile ses yüksekliği ve sinyal oranı ile kontrol edilen yerel aramanın dengeli bir birleşimi olarak kabul edilebilmektedir.

3.1.2. Ses Yüksekliği ve Sinyal Emisyonu

Algoritmada ses yüksekliği (A_i) ve sinyal emisyon oranı (r_i) değerlerinin iterasyon ilerlemesine bağlı olarak güncellenmesi gerekmektedir. Bu güncelleme işlemi denklem 3.5'e göre yapılmaktadır. Ses yüksekliği genellikle yarasa avını bulduğunda azalacağından, sinyal emisyon oranı artarken ses yüksekliği için uygun herhangi bir değer seçilebilir. Örneğin, A_0 = 100 ve A_{min}= 1 kullanabilir. Basitlik için, A_{min}= 1 ve A_0 = 0 alınarak A_0 = 0 durumunda yarasanın avını bulduğu ve herhangi bir ses yaymayı geçici olarak bıraktığı varsayılmaktadır.

$$A_i^{t+1} = \alpha A_i^t \qquad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{3.5}$$

Burada α ve γ sabit değerlerdir. Hatta α tavlama benzetimi sürecindeki soğutma faktörüne benzerlik göstermektedir. Herhangi bir $0 < \alpha < 1$ ve $\gamma > 0$ için;

$$A_i^t \to 0, \qquad r_i^t \to r_i^0, \qquad t \to \infty \tag{3.6}$$

olmaktadır (Yang, 2010d.; Yang, 2011, Yang ve Gandomi, 2012)

3.2. BELGE SETLERİ

Belge kümeleme algoritmalarının etkinliğini değerlendirmek için, literatürde birçok belge seti (metin koleksiyonu) mevcuttur. Bu setler bilgi alma, doğal dil işleme, sayısal dilbilim araştırmaları ile diğer belge seti tabanlı araştırmalar için yararlıdır. Gerçek metinlerden seçilmiş, kümeleme amacıyla bu tez çalışmasında da kullanılacak bazı veri setleri aşağıda detaylı olarak açıklanmaktadır.

3.2.1. Reueters-21578

Bu koleksiyonundaki belgeler 1987 yılında Reuters online gazetesinde çıkmıştır. Belgeler daha sonar toplanarak Reuters Ltd. personeli Sam Dobbins, Mike Topliss, Steve Weinstein ve Carnegie Group Anonim Şirketi (CGI) personeli Peggy Andersen, Monica Cellio, Phil Hayes, Laura Knecht, Irene Nirenburg tarafından kategorilere ayrılmıştır.

1990 yılında Reuters ve CGI tarafından, Amherst'teki Massachusetts Üniversitesi'nde Bilgi Erişim Laboratuvarı ile Bilgisayar ve Enformasyon Bilimi Bölümü'nde araştırma amaçlı hazır hale getirilen belgelerin biçimlendirilmesi ve ilişkili veri dosyalarının üretimi, yine aynı yıl Bilgi Erişim Laboratuvarı'nda David D. Lewis ve Stephen Harding tarafından yapılmıştır.

Bu veri setinin literatürde çeşitli araştırmacılar tarafından hazırlanmış birçok versiyonu bulunmaktadır. Orjinal Reuters-21578 belge seti, 135 kategoride 21578 tane belge içermektedir. ModApte sürümünde ise birden fazla kategori etiketleri olan belgeler atılarak 65 kategoride 8293 belge bırakılmıştır. (Reuters-21578)

3.2.2. BBC Sport

BBC Sport belge seti, BBC Spor web sitesinden 2004-2005 yılları arasında toplanan sporla ilgili 5 farklı kategorideki haber makalelerinden meydana gelmektedir. 737 belge ve 4613 terimden oluşan veri seti; atletizm, kriket, futbol, rugby ve tenis kümelerine ayrılmaktadır.

BBC Sport belge setinin durak kelimeleri çıkarılmış, Porter algoritması ile kök haline getirilmiş ve düşük frekanslı terimlere (terim görülme sayısı<3) filtre uygulanmış hali bu tez çalışmasında kullanılmıştır (BBC Sport).

3.2.3. Classic 4

Metin madenciliği uygulamalarında sıkça kullanılan ve iyi bilinen referans veri kümesi Classic koleksiyonu, CACM, CISI, CRAN ve MED olmak üzere 4 farklı belge kümesinden oluşmaktadır. Toplam 7095 belgeden oluşan Classic 4 seti aşağıdaki kümelere ayrılmaktadır:

- CACM: 3204 adet belge
- CISI: 1460 adet belge
- CRAN: 1398 adet belge
- MED: 1033 adet belge

Bu tez çalışmasında kullanılan Classic 4 veri seti, Porter algoritması ile ön işlenmiştir. Ayrıca terimler tek kelimeden oluşup, her bir terim en az 3 belgede görünmektedir (Classic 4).

4. BULGULAR

Bölüm 3.2'de verilen belge setleri üzerinde yarasa algoritması ile kümeleme çalışmaları, MATLAB R2014b programında yazılan kod ile gerçekleştirilmiştir. Ayrıca yarasa algoritmasının performansını karşılaştırmak adına, yine aynı programda yazılan parçacık sürü zekası optimizasyon algoritması ve MATLAB R2014b programının kendi içinde yer alan k-ortalamalar algoritması ile veri setleri farklı benzerlik ölçülerine ve farklı ölçüt fonksiyonlarına göre kümelenmiştir. Tüm algoritmalar 2,83 GHz Intel® Core™2 Duo işlemcili ve 4,00 GB RAM'li bilgisayarlarda çalıştırılmıştır.

Algoritmalarla kümeleme işlemine geçmeden önce her bir belge setinde durak kelimelerin çıkarılması, kelimelerin kök hallerine getirilmesi vb. işlemlerin yapılması gerekmektedir. Ancak kullanılan veri setleri ön işlemden geçirilmiş hazır setler olduğundan doğrudan kullanılmışlar, herhangi bir ön işlemden geçirilmemişlerdir.

Belge setlerinin boyutları oldukça büyük olduğu için, tez çalışmasında kolaylık sağlaması açısından belge setleri içinden rastgele alt setler seçilmiştir. Seçilen alt setlerde, belgeler içinde hiç geçmeyen terimler çıkarılmıştır. Bu alt setlerin belge, terim ve küme sayıları Tablo 4.1'de özetlenmektedir.

Tablo 4.1: Tez çalışmasında kullanılan belge setleri.

Belge Alt Seti #	Belge Alt Seti Adı	Belge Sayısı	Terim Sayısı	Küme sayısı
1	Classic4-1	245	2486	4
2	Classic4-2	507	3330	4
3	Reuters-1	514	6562	9
4	Reuters-2	248	4274	9
5	BBCSport-1	737	4613	5
6	BBCSport-2	245	4113	5

Alt setler içinde yer alan her bir kümeye atanması gereken belge sayıları ve bunların küme içindeki oranı ise Tablo 4.2'de verilmektedir.

Tablo 4.2: Her bir kümede yer alan belge sayıları ve oranları.

CLASSIC4_1			**CLASSIC4_2**		
Küme	**Belge Sayısı**	**Oran (%)**	**Küme İndisi**	**Belge Sayısı**	**Oran (%)**
1	106	43,27	1	243	47,93
2	63	25,71	2	105	20,71
3	42	17,14	3	83	16,37
4	34	13,88	4	76	14,99
REUTERS_1			**REUTERS_2**		
Küme İndisi	**Belge Sayısı**	**Oran (%)**	**Küme İndisi**	**Belge Sayısı**	**Oran (%)**
1	259	50,39	1	133	53,63
2	137	26,65	2	63	25,40
3	25	4,86	3	12	4,84
4	28	5,45	4	11	4,44
5	18	3,50	5	7	2,82
6	21	4,09	6	7	2,82
7	11	2,14	7	6	2,42
8	7	1,36	8	6	2,42
9	8	1,56	9	3	1,21
BBCSPORT_1			**BBCSPORT_2**		
Küme	**Belge Sayısı**	**Oran (%)**	**Küme İndisi**	**Belge Sayısı**	**Oran (%)**
1	101	13,70	1	40	16,33
2	124	16,82	2	27	11,02
3	265	35,96	3	97	39,59
4	147	19,95	4	50	20,41
5	100	13,57	4	31	12,65

Belge setlerinin boyutlarının küçültülmesinin ardından oluşturulan alt belge setleri, doğrudan Terim Frekansı - Ters Belge Frekansı (TD-IDF) yöntemi kullanılarak ağırlıklandırılmış ve setler içinde yer alan her bir belge sayısal vektör haline getirilmiştir. Bu yöntemde her bir ağırlık, terim frekansı ve ters belge frekansı olarak adlandırılan iki bileşenin ürünüdür.

Kümeleme işlemi başlatılmadan önce algoritmalara ait bazı parametrelerin belirlenmesi gerekmektedir. Analizler süresince her bir veri seti üzerinde kümeleme işlemi yaparken algoritmalar için Tablo 4.3'te verilen parametreler kullanılmıştır.

Tablo 4.3: Kümeleme işlemi sırasında üç algoritma için kullanılan parametreler.

	K-Ortalamalar Algoritması	**Yarasa Algoritması**	**PSO Algoritması**
İterasyon sayısı	100	100	100
Popülasyon sayısı	-	80	80
Bireysel öğrenme parametresi (c_1)	-	-	1,49
Sosyal öğrenme parametresi (c_2)	-	-	1,49
Atalet ağırlığı (ω)	-	-	0,90
Sinyal emisyon oranı (r_i)	-	0,5	-
Ses yüksekliği (A_i)	-	0,5	-
Minimum frekans (f_{min})	-	0	-
Maksimum frekans (f_{maks})	-	0,02	-
α	-	0,1	-
γ	-	0,9	-

K-ortalamalar algoritması ile karşılaştırma yapabilmek için; ilk olarak bu algoritmanın amaç fonksiyonu olan, tüm noktaların atandıkları küme merkezine olan uzaklıklarının karelerinin toplamı fonksiyonu (I_3) yarasa algoritmasında ölçüt fonksiyonu olarak alınmıştır. Benzerlik ölçüsü olarak da yine k-otalamalar algoritmasının genel yapısında bulunan öklid uzaklığı kullanılmıştır. Hem yarasa algoritması hem de k-ortalamalar algoritması için, belirtilen ölçüt fonksiyonu ve benzerlik ölçüsü doğrultusunda 30'ar koşum sonucunda elde edilen amaç fonksiyonu (uzaklık) değerleri Tablo 4.4'te verilmiştir.

Tablo 4.4'te görüldüğü gibi, yarasa algoritması tüm alt belge setlerinde k-ortalamalar algoritmasından daha küçük ortalama uzaklık sonuçları bulmuştur.

Tablo 4.4: I_3 ölçüt fonksiyonu ve öklid benzerlik ölçüsü ile yapılan kümeleme işlemi sonucunda k-ortalamalar algoritması ve yarasa algoritması ile hesaplanan uzaklıklar.

Belge Alt Seti #	Belge Alt Seti Adı		K-Ortalamalar Algoritması	Yarasa Algoritması
1	Classic4-1	En iyi değer	229,4157	229,3214
		Ortalama	233,1924	**229,4630**
		Standart Sapma	17,9775	0,1241
2	Classic4-2	En iyi değer	482,4371	482,3879
		Ortalama	482,9313	**482,9140**
		Standart Sapma	0,7524	0,1456
3	Reuters-1	En iyi değer	463,9562	463,9139
		Ortalama	465,0456	**465,0251**
		Standart Sapma	1,3478	0,8091
4	Reuters-2	En iyi değer	217,1459	217,5413
		Ortalama	217,4998	**217,0805**
		Standart Sapma	0,3051	0,3833
5	BBCSport-1	En iyi değer	678,8397	678,8193
		Ortalama	680,6478	**679,8984**
		Standart Sapma	1,5080	1,0974
6	BBCSport-2	En iyi değer	222,9805	222,8823
		Ortalama	223,7706	**223,4409**
		Standart Sapma	0,5680	0,4345

K-ortalamalar ve yarasa algoritmaları ile belge kümeleme sonuçlarını hesaplama süreleri ise Tablo 4.5'te gösterilmiştir. Yarasa algoritması sürü tabanlı bir algoritma yapısında olduğundan, algoritmanın içindeki döngüler popülasyon sayısına bağlı olarak artmaktadır. Ancak k-ortalamalar algoritmasında böyle bir durum söz konusu değildir. Bu nedenle, yarasa algoritmasının belge kümeleme sonuçlarını hesaplama süreleri tüm alt belge setleri için k-ortalamalar algoritmasından daha yüksektir. Ayrıca k-ortalamalar algoritması MATLAB R2014b programının kendi içinde yer alan hazır bir kod iken, yarasa algoritmasının kodu belge kümeleme problemi için tarafımdan uyarlanmıştır. Ticari bir programın içindeki kod ile tarafımdan hazırlanan kodun hesaplama hızlarını karşılaştırmanın çok doğru olmadığını düşünmekle birlikte sonuçlar yine de tablo olarak verilmiştir.

Tablo 4.5: I_3 ölçüt fonksiyonu ve öklid benzerlik ölçüsü ile yapılan kümeleme işleminin k-ortalamalar algoritması ve yarasa algoritması ile hesaplama süreleri (saniye).

Belge Alt Seti #	**Belge Alt Seti Adı**		**K-Ortalamalar Algoritması**	**Yarasa Algoritması**
1	Classic4-1	En iyi değer	2,5200	480,2920
		Ortalama	4,3912	533,6671
		Standart Sapma	1,0714	22,0005
2	Classic4-2	En iyi değer	3,6359	1156,0980
		Ortalama	10,2630	1499,0166
		Standart Sapma	5,7319	277,2068
3	Reuters-1	En iyi değer	28,6128	4640,6117
		Ortalama	39,9654	6937,5072
		Standart Sapma	8,3900	1181,0278
4	Reuters-2	En iyi değer	6,1770	1271,6856
		Ortalama	9,3146	2472,5226
		Standart Sapma	1,4551	501,5802
5	BBCSport-1	En iyi değer	5,3855	4223,8639
		Ortalama	15,0030	3766,9763
		Standart Sapma	8,1860	421,5231
6	BBCSport-2	En iyi değer	4,5346	1093,3555
		Ortalama	6,6922	1205,3161
		Standart Sapma	1,5097	128,9366

Algoritmaların ne kadar iyi kümeleme yaptıklarını ölçmek için birçok gösterge bulunmaktadır. Ancak literatürdeki belge kümeleme yayınlarda en fazla incelenen göstergeler doğruluk ve F-ölçüsü göstergeleri olduğundan, bu tez çalışmasında da algoritmaların kümeleme performansını değerlendirmek için bu iki gösterge kullanılmıştır. Buna göre, I_3 ölçüt fonksiyonu ve öklid benzerlik ölçüsü kullanılarak k-ortalamalar ve yarasa algoritmalarının yaptığı belge kümeleme işlemlerinin 30'ar koşum sonucundaki doğruluk değerleri Tablo 4.6'da sunulmuştur:

Tablo 4.6: K-ortalamalar ve yarasa algoritmalarının belge setleri üzerinde kümeleme sonuçlarının "doğruluk" göstergesi değerleri.

Belge Alt Seti #	Belge Alt Seti Adı		K-Ortalamalar Algoritması	Yarasa Algoritması
1	Classic4-1	En iyi değer	0,6735	0,6939
		Ortalama	0,6338	0,6329
		Standart Sapma	0,0478	0,0148
		En iyi değer oranı	%3,34	%3,34
2	Classic4-2	En iyi değer	0,6568	0,6391
		Ortalama	0,5927	**0,5971**
		Standart Sapma	0,0645	0,0425
		En iyi değer oranı	%10	%6,67
3	Reuters-1	En iyi değer	0,5370	0,5584
		Ortalama	0,4617	**0,4794**
		Standart Sapma	0,0467	0,0386
		En iyi değer oranı	%3,34	%3,34
4	Reuters-2	En iyi değer	0,6089	0,5887
		Ortalama	0,4577	**0,4910**
		Standart Sapma	0,0662	0,0502
		En iyi değer oranı	%3,34	%3,34
5	BBCSport-1	En iyi değer	0,9607	0,9661
		Ortalama	0,7974	**0,8352**
		Standart Sapma	0,1250	0,1038
		En iyi değer oranı	%16,67	%6,67
6	BBCSport-2	En iyi değer	0,9592	0,9551
		Ortalama	0,7899	**0,8185**
		Standart Sapma	0,1163	0,0898
		En iyi değer oranı	%16,67	%6,67

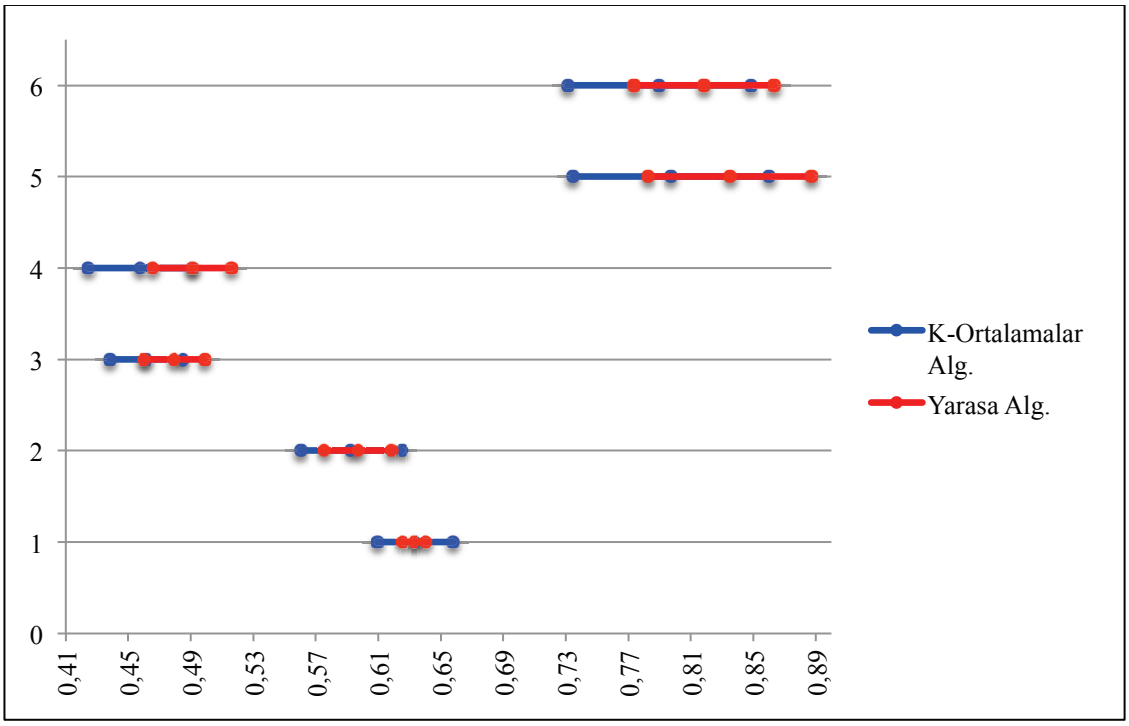
Tablo 4.6'dan görüldüğü üzere, yarasa algoritmasının Classic4-1 veri seti dışında diğer tüm veri setlerinde yer alan belgeleri daha doğru bir şekilde kümelere atama ortalaması k-ortalamalar algoritmasına göre daha yüksektir. Bu da bize, yarasa algoritmasının kümeleme sonuçlarının I_3 ölçüt fonksiyonu ve öklid benzerlik ölçüsü ile k-ortalamalar algoritmasına nazaran daha başarılı olduğunu göstermektedir.

30'ar koşum sonucuna bağlı olarak yaptığımız bu değerlendirmeden yola çıkarak hem k-ortalamalar algoritması hem de yarasa algoritması için genel bir kestirim yapmak adına güven aralıkları belirlenmiştir. Örneklem ortalaması ($\bar{x}$) ve standart sapması (s)

dikkate alınarak anakütle ortalamasının $\%100(1-\alpha)$ güven aralıkları denklem 4.1'deki gibi hesaplanmaktadır (Montgomery ve diğ., 2011).

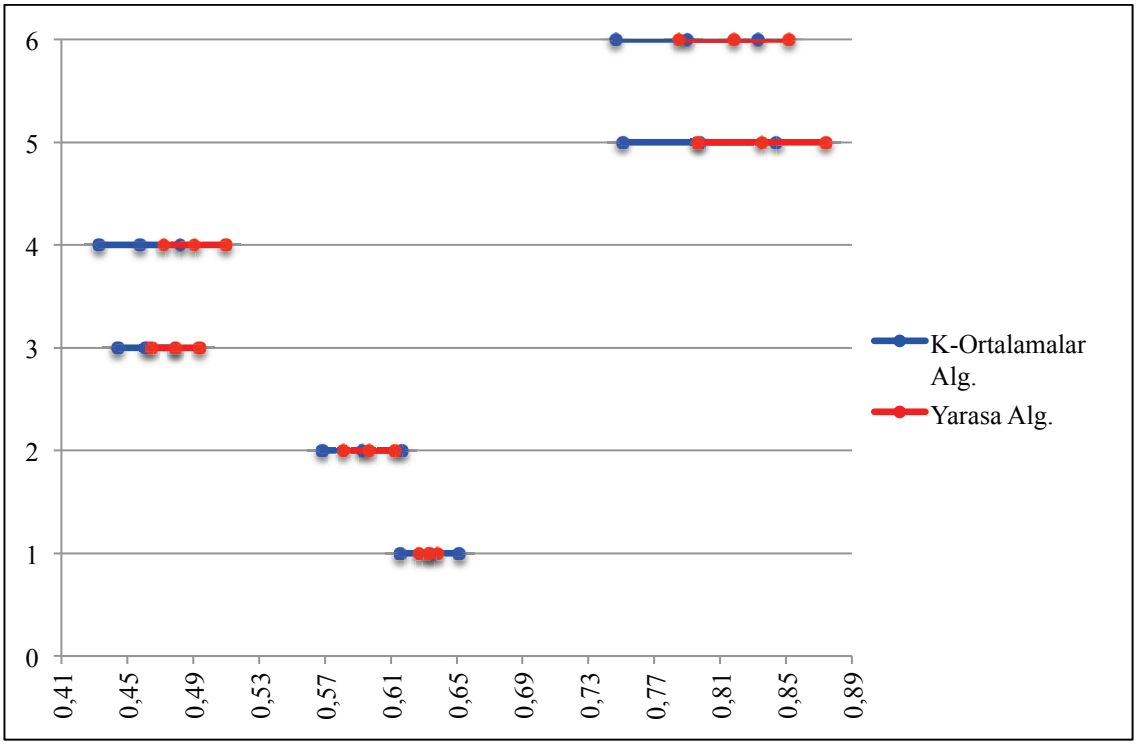
$$\bar{x} - \frac{t_{\frac{\alpha}{2},n-1}s}{\sqrt{n}} < \mu < \bar{x} + \frac{t_{\frac{\alpha}{2},n-1}s}{\sqrt{n}} \tag{4.1}$$

Buna göre, $n = 30$ için algoritmaların doğruluk sonuçlarının %99 ($\alpha = 0{,}01$) güven aralıkları Şekil 4.1'de verilmiştir.

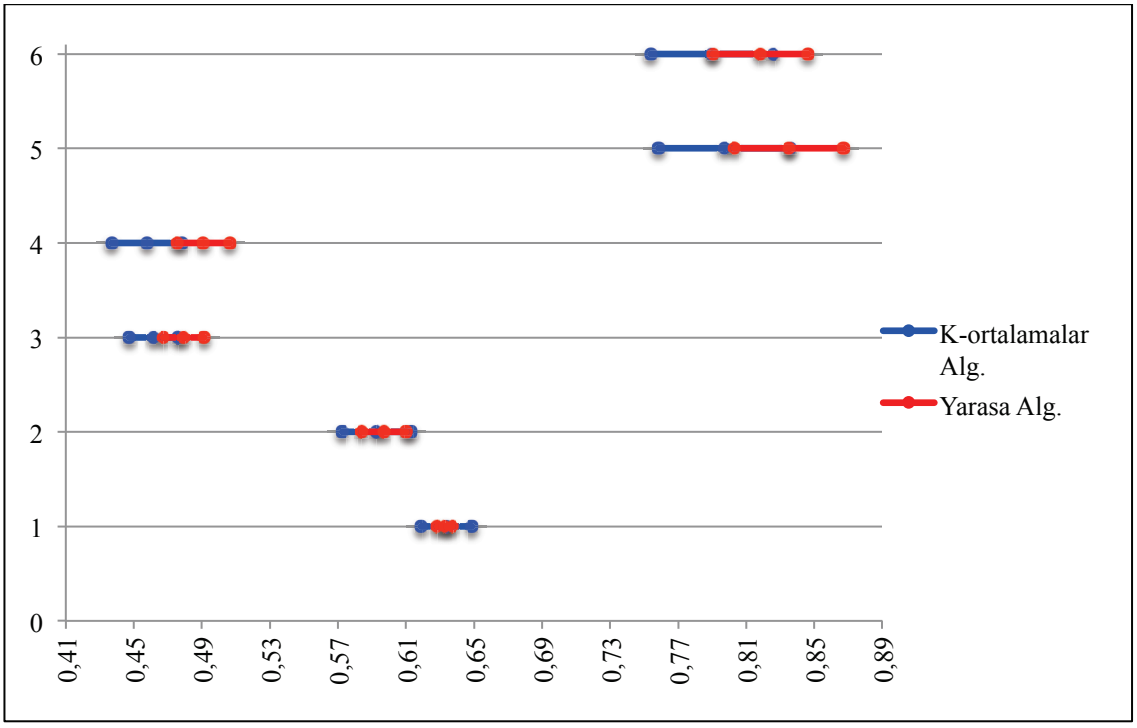


Şekil 4.1: K-ortalamalar ve yarasa algoritmalarının "doğruluk" göstergesi değerlerinin %99 güven aralıkları.

$n = 30$ için algoritmaların doğruluk sonuçlarının %95 ($\alpha = 0{,}05$) ve %90 ($\alpha = 0{,}1$) güven aralıkları ise sırasıyla Şekil 4.2 ve Şekil 4.3'te gösterilmiştir.



Şekil 4.2: K-ortalamalar ve yarasa algoritmalarının "doğruluk" göstergesi değerlerinin %95 güven aralıkları.



Şekil 4.3: K-ortalamalar ve yarasa algoritmalarının "doğruluk" göstergesi değerlerinin %90 güven aralıkları.

Şekil 4.1, 4.2 ve 4.3'te yatay eksen doğruluk değerlerini, düşey eksen ise her bir alt belge setini göstermektedir. Bu şekillerden görüldüğü üzere, yarasa algoritması k-ortalamalar algoritması ile aynı sonucu vermektedir ve yarasa algoritmasının yayılmaları k-ortalamalar algoritmasına göre daha düşüktür.

Yarasa algoritmasının kümeleme sonuçlarının karşılaştırıldığı bir diğer algoritma parçacık sürü optimizasyonu algoritmasıdır. Parçacık sürü optimizasyonu algoritması, k-ortalamalar algoritması ile yapılan karşılaştırmada kullanılan I_3 ölçüt fonksiyonu ve öklid benzerlik ölçüsü kullanıldığında tüm belgeleri tek bir kümenin içine atamış, dolayısıyla anlamlı bir kümeleme işlemi gerçekleştirmemiştir. Bu nedenle, her bir belge ile atandığı kümenin ağırlık merkezi arasındaki benzerliği maksimum kılan ölçüt fonksiyonu (I_2) ve kosinüs uzaklığı benzerlik ölçüsü alınarak yarasa algoritması ile parçacık sürü optimizasyonu algoritması karşılaştırılmıştır. Buna göre, her iki algoritma için 30'ar koşum sonucunda elde edilen amaç fonksiyonu (uzaklık) değerleri Tablo 4.7'de verilmiştir. Algoritmaların kümeleme sonuçlarını hesaplama süreleri ise Tablo 4.8'de gösterilmiştir.

Tablo 4.7'de görüldüğü üzere, yarasa algoritması tüm alt belge setlerinde parçacık sürü optimizasyonu algoritmasından daha küçük ortalama uzaklıklarla belge kümeleme işlemini gerçekleştirmiştir. Ancak yarasa algoritmasının hesaplama süreleri parçacık sürü optimizasyonu algoritmasına göre daha uzundur. Bunun başlıca nedeni, algoritmaların kodlarının yazım farklılığıdır. Yarasa algoritmasında matrislerle yapılan işlemler parçacık sürü zekası optimizasyonu algoritmasına göre daha fazladır. Bu da hesaplama süresinin uzamasına neden olmuştur.

Tablo 4.7: I_2 ölçüt fonksiyonu ve kosinüs benzerlik ölçüsü ile yapılan kümeleme işlemi sonucunda parçacık sürü optimizasyonu ve yarasa algoritması ile hesaplanan uzaklıklar.

Belge Alt Seti #	**Belge Alt Seti Adı**		**PSO Algoritması**	**Yarasa Algoritması**
1	Classic4-1	En iyi değer	210,5503	196,9135
		Ortalama	219,0603	**197,2559**
		Standart Sapma	1,6757	0,1214
2	Classic4-2	En iyi değer	463,3221	425,1682
		Ortalama	463,7243	**425,7529**
		Standart Sapma	0,2664	1,8650
3	Reuters-1	En iyi değer	475,6181	409,3071
		Ortalama	475,9437	**410,2745**
		Standart Sapma	0,1519	0,2471
4	Reuters-2	En iyi değer	224,4363	185,0415
		Ortalama	224,7351	**185,5096**
		Standart Sapma	0,1453	0,1493
5	BBCSport-1	En iyi değer	639,5699	580,1442
		Ortalama	639,8883	**580,4217**
		Standart Sapma	0,2627	0,1165
6	BBCSport-2	En iyi değer	210,6336	185,1082
		Ortalama	210,8222	**185,4428**
		Standart Sapma	0,0994	0,1353

Tablo 4.8: I_2 ölçüt fonksiyonu ve kosinüs benzerlik ölçüsü ile yapılan kümeleme işleminin parçacık sürü optimizasyonu algoritması ve yarasa algoritması ile hesaplama süreleri (saniye).

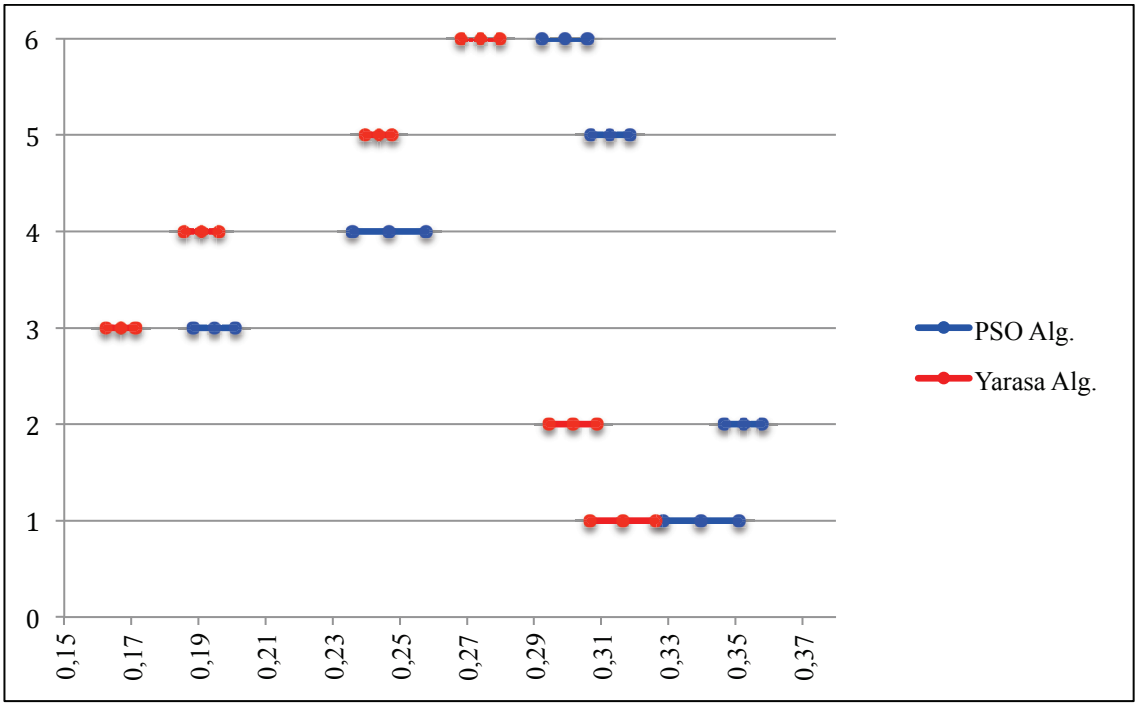
Belge Alt Seti #	**Belge Alt Seti Adı**		**PSO Algoritması**	**Yarasa Algoritması**
1	Classic4-1	En iyi değer	195,3602	1056,4937
		Ortalama	203,3882	1128,7831
		Standart Sapma	12,8829	24,1331
2	Classic4-2	En iyi değer	569,0110	2756,9069
		Ortalama	608,0979	2942,7466
		Standart Sapma	38,1624	84,7619
3	Reuters-1	En iyi değer	745,2753	3389,7211
		Ortalama	775,0141	3643,6866
		Standart Sapma	9,9876	103,7553
4	Reuters-2	En iyi değer	546,8657	2023,0079
		Ortalama	559,2908	2723,2153
		Standart Sapma	5,5327	237,3907
5	BBCSport-1	En iyi değer	984,6321	5343,1349
		Ortalama	1092,4061	5740,3937
		Standart Sapma	104,6478	140,2953
6	BBCSport-2	En iyi değer	400,0210	1930,5399
		Ortalama	497,8286	2099,1984
		Standart Sapma	39,0871	155,7209

Parçacık sürü optimizasyonu ve yarasa algoritmalarının kümeleme performansını değerlendirmek için kullanılan doğruluk göstergeleri Tablo 4.9'da sunulmuştur. Bu değerlere bakılacak olursa; yarasa algoritması Classic4-1 ve BBCSport-2 alt belge setlerinde yaklaşık %2'lik bir belgeyi doğru kümeye atayamamıştır. Reuters-1 ve Classic4-2 alt belge setlerinde bu miktar sırasıyla %3 ve %5 olurken, Reuters-2 ve BBCSport-1 alt belge setlerinde sırasıyla %6 ve %7'yi bulmuştur.

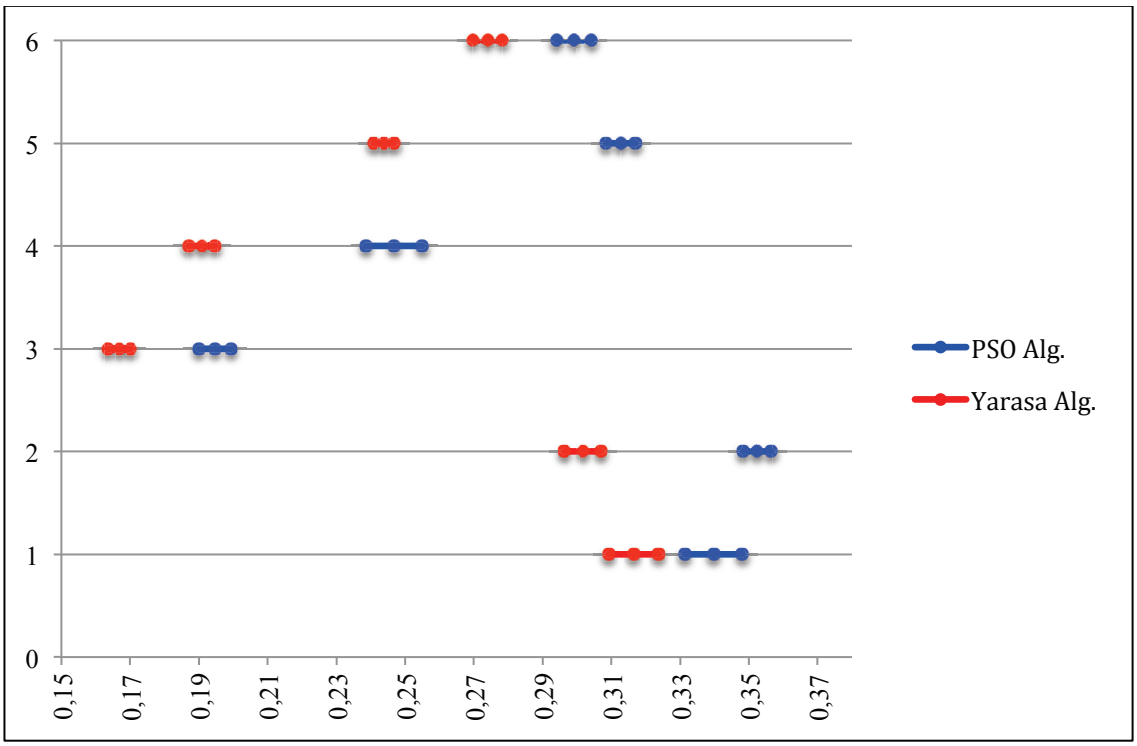
Tablo 4.9: Parçacık sürü optimizasyonu ve yarasa algoritmalarının belge setleri üzerinde kümeleme sonuçlarının "doğruluk" göstergesi değerleri.

Belge Alt Seti #	Belge Alt Seti Adı		PSO Algoritması	Yarasa Algoritması
1	Classic4-1	En iyi değer	0,3775	0,3551
		Ortalama	0,3398	0,3165
		Standart Sapma	0,0224	0,0196
		En iyi değer oranı	%26,67	%26,67
2	Classic4-2	En iyi değer	0,3688	0,3294
		Ortalama	0,3524	0,3017
		Standart Sapma	0,0110	0,0142
		En iyi değer oranı	%13,33	%26,67
3	Reuters-1	En iyi değer	0,2276	0,1907
		Ortalama	0,1947	0,1669
		Standart Sapma	0,0126	0,0086
		En iyi değer oranı	%30	%30
4	Reuters-2	En iyi değer	0,2863	0,2177
		Ortalama	0,2469	0,1909
		Standart Sapma	0,0219	0,0103
		En iyi değer oranı	%60	%60
5	BBCSport-1	En iyi değer	0,3338	0,2605
		Ortalama	0,3127	0,2438
		Standart Sapma	0,0115	0,0079
		En iyi değer oranı	%16,67	%16,67
6	BBCSport-2	En iyi değer	0,3265	0,2980
		Ortalama	0,2992	0,2741
		Standart Sapma	0,0134	0,0115
		En iyi değer oranı	%16,67	%16,67

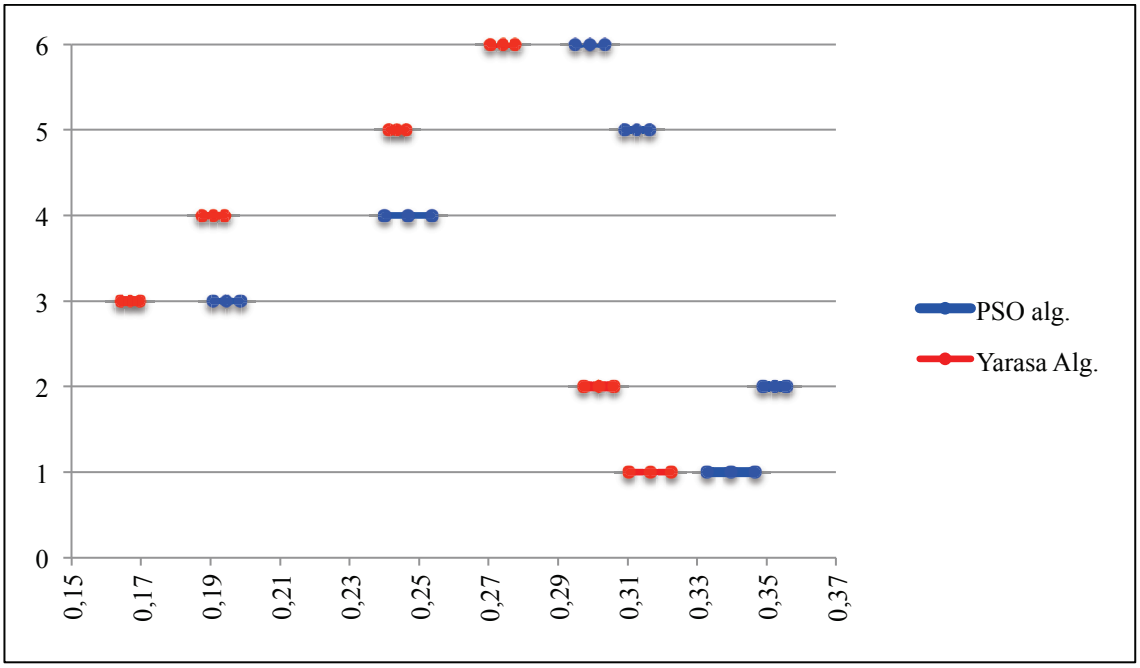
Parçacık sürü optimizasyonu ve yarasa algoritmalarında $n = 30$ için algoritmaların doğruluk sonuçlarının %99 $(\alpha = 0{,}01)$, %95 $(\alpha = 0{,}05)$ ve %90 $(\alpha = 0{,}1)$ güven aralıkları sırasıyla Şekil 4.4, Şekil 4.5 ve Şekil 4.6'da sunulmuştur.



Şekil 4.4: Parçacık sürü optimizasyonu ve yarasa algoritmalarının "doğruluk" göstergesi değerlerinin %99 güven aralıkları.



Şekil 4.5: Parçacık sürü optimizasyonu ve yarasa algoritmalarının "doğruluk" göstergesi değerlerinin %95 güven aralıkları.



Şekil 4.6: Parçacık sürü optimizasyonu ve yarasa algoritmalarının "doğruluk" göstergesi değerlerinin %90 güven aralıkları.

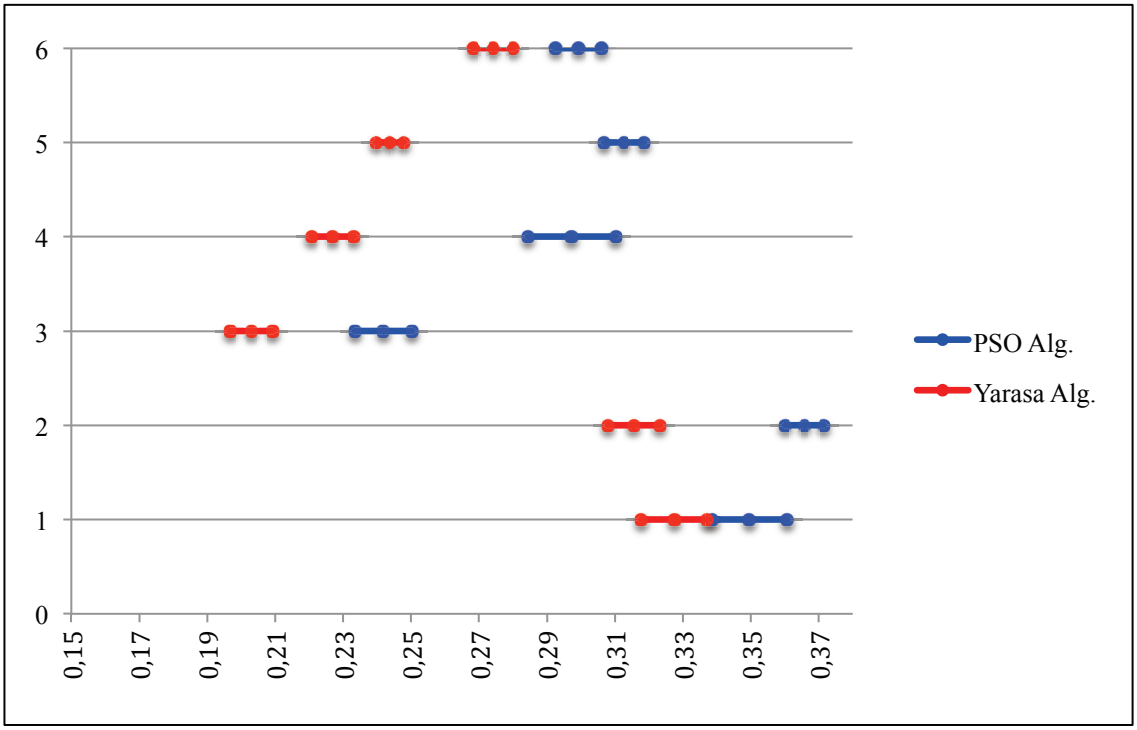
Şekil 4.4, 4.5 ve 4.6'dan görüleceği gibi, yarasa algoritmasının tüm belge setlerinde güven aralıkları ya daha küçüktür ya da parçacık sürü optimizasyonu algoritmasına yakındır.

Parçacık sürü optimizasyonu ve yarasa algoritmalarının kümeleme performansını değerlendirmek için kullanılan bir diğer gösterge de F-ölçüsüdür ve Tablo 4.10'da verilmiştir. F-ölçüsü değerlerine göre, yarasa algoritmasının performansı parçacık sürü optimizasyonu algoritmasına göre yine %2-%7 oranında daha düşük çıkmıştır.

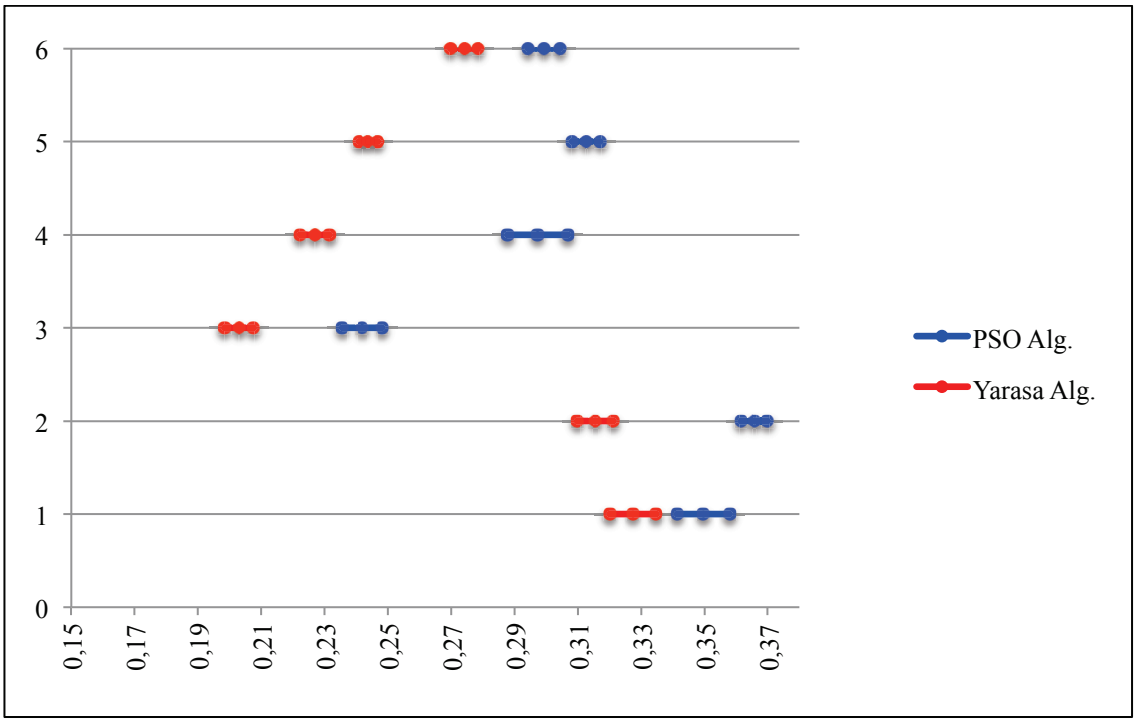
Tablo 4.10: Parçacık sürü optimizasyonu ve yarasa algoritmalarının belge setleri üzerinde kümeleme sonuçlarının "F-ölçüsü" göstergesi değerleri.

Belge Alt Seti #	**Belge Alt Seti Adı**		**PSO Algoritması**	**Yarasa Algoritması**
1	Classic4-1	En iyi değer	0,3866	0,3715
		Ortalama	0,3497	0,3275
		Standart Sapma	0,0220	0,0194
2	Classic4-2	En iyi değer	0,3903	0,3419
		Ortalama	0,3658	0,3156
		Standart Sapma	0,0111	0,0152
3	Reuters-1	En iyi değer	0,2838	0,2348
		Ortalama	0,2419	0,2030
		Standart Sapma	0,0167	0,0124
4	Reuters-2	En iyi değer	0,3569	0,2536
		Ortalama	0,2973	0,2269
		Standart Sapma	0,0256	0,0124
5	BBCSport-1	En iyi değer	0,3338	0,2605
		Ortalama	0,3127	0,2438
		Standart Sapma	0,0117	0,0080
6	BBCSport-2	En iyi değer	0,3265	0,2979
		Ortalama	0,2993	0,2742
		Standart Sapma	0,0136	0,0116

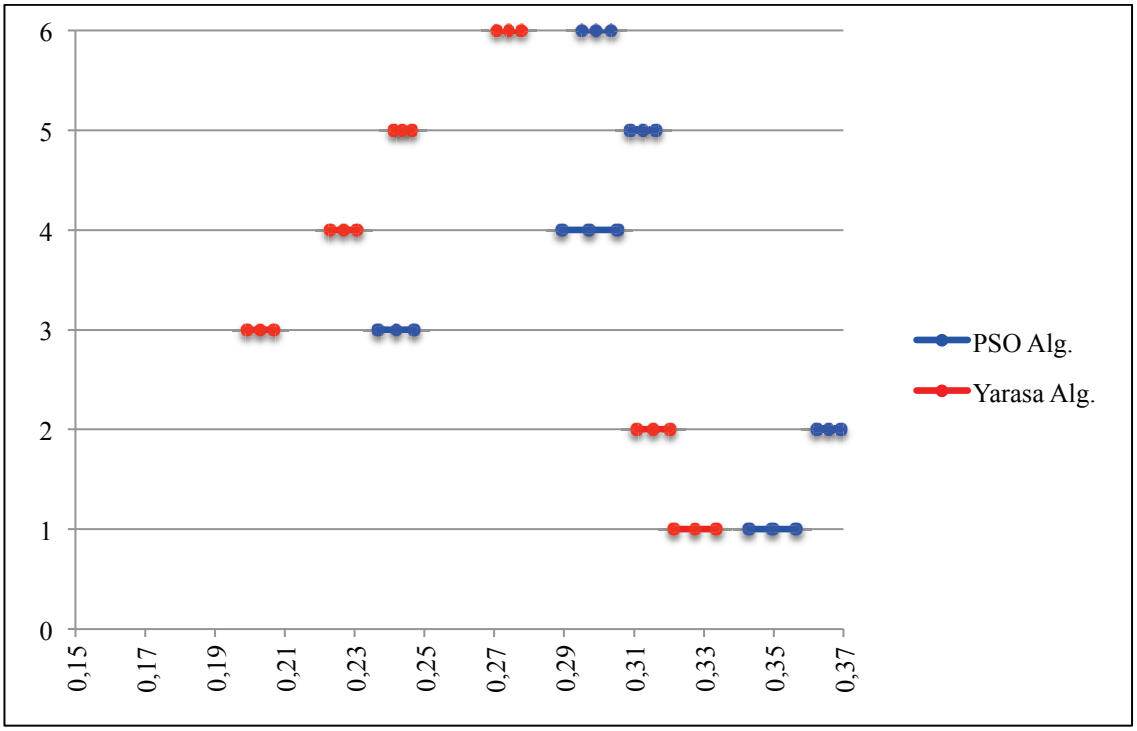
Parçacık sürü optimizasyonu ve yarasa algoritmalarının F-ölçüsü sonuçlarının $n = 30$ için %99 ($\alpha = 0{,}01$), %95 ($\alpha = 0{,}05$) ve %90 ($\alpha = 0{,}1$) güven aralıkları ise sırasıyla Şekil 4.7, Şekil 4.8 ve Şekil 4.9'da sunulmuştur.



Şekil 4.7: Parçacık sürü optimizasyonu ve yarasa algoritmalarının "F-ölçüsü" göstergesi değerlerinin %99 güven aralıkları.



Şekil 4.8: Parçacık sürü optimizasyonu ve yarasa algoritmalarının "F-ölçüsü" göstergesi değerlerinin %95 güven aralıkları.



Şekil 4.9: Parçacık sürü optimizasyonu ve yarasa algoritmalarının "F-ölçüsü" göstergesi değerlerinin %90 güven aralıkla**rı.**

Şekil 4.7, 4.8 ve 4.9'dan görüleceği üzere, yarasa algoritmasının güven aralıkları parçacık sürü optimizasyonu algoritmasına göre küçük çıkmakta fakat ortalamalar parçacık sürü optimizasyonu algoritması tarafından hesaplanan düzeylere ulaşamamaktadır. Dolayısıyla yarasa algoritmasında kullanılan parametrelerin değerleri bir kez de deney tasarımı yardımıyla ele alınıp tekrar denenmelidir. Bu çalışmada elde edilen sonuçlara göre, parametre optimizasyonu ve farklı örnek hacimleri ile denemeler yapılması sonucunda yarasa algoritmasının, parçacık sürü optimizasyonu algoritması tarafından hesaplanan sonuçlara benzer hatta daha iyi sonuçlar elde edileceği düşünülmektedir.

5. TARTIŞMA VE SONUÇ

1980'li yıllardan günümüze kombinatoryal optimizasyon teorisi alanında büyük gelişmeler katedilmiş ve sezgisel algoritmalar olarak adlandırılan yaklaşık algoritmalar araştırma ve uygulamalarda önemli bir hale gelmiştir. Bununla birlikte, klasik sezgisel algoritmaların karmaşık optimizasyon problemleri üzerinde yeterince etkin olmadığı durumlar için metasezgisel algoritmalar olarak adlandırılan ve yine yaklaşık algoritmalar sınıfında yer alan algoritmalara yönelim başlamıştır.

Bir metasezgisel algoritma biçimsel olarak; arama uzayını keşfetmek ve kuvvetlendirmek için farklı akıllı kavramları birleştirerek ve optimuma yakın çözümler bulmak amacıyla bilgi yapılandırmasında öğrenme stratejileri kullanarak sezgisel algoritmalara rehberlik eden iteratif bir oluşum süreci olarak tanımlanabilmektedir. Metasezgisel algoritmalar "yeterince küçük" işlem zamanında "yeterince iyi" bir çözüm bulmak amacıyla özel olarak geliştirilmiştir.

Metasezgisel algoritmaların temelleri önemli ölçüde değişkenlik göstermektedir. Bazı algoritmalar optimizasyon sürecini, görünüşte optimizasyon ile alakasız olan hayvan sürülerinin davranışları, doğal evrim, fiziksel değişimler vb. gibi yaklaşımlar kullanarak açıklarken, bazıları biyolojik temellere dayanmaktadır. Ancak genel olarak tüm metasezgisel algoritmalar yapılarında rastgelelik bulundurmaktadır.

Bu doktora tezi kapsamında doğadan esinlenerek geliştirilmiş ve literatürde yeni sayılabilecek metasezgisellerden biri olan yarasa algoritması ile belge kümeleme uygulaması incelenmiştir. Literatürde sıkça kullanılan belge setlerinden oluşturulan alt belge setlerine farklı ölçüt fonksiyonları ve farklı benzerlik ölçüleri ile yapılan analizler sonucunda yarasa algoritmasının performansı, kümeleme işlemlerinde çoğunlukla kullanılan k-ortalamalar ve parçacık sürü optimizasyonu algoritması ile karşılaştırılmıştır. Çalışma, literatürde sıkça kullanılan belge setlerini yarasa algoritması ile kümeleyen ilk çalışma olması yönünden önem taşımaktadır.

Yapılan analizler, k-ortamalamalar algoritmasının amaç fonksiyonunun (hata karelerinin toplamı) ölçüt fonksiyonu ve öklid uzaklığının benzerlik ölçüsü olarak kullanıldığı belge kümeleme işlemlemlerinde, yarasa algoritmasının k-ortalamalar algoritması kadar etkin olduğunu ortaya koymuştur. Yarasa algoritması k-ortalamalar algoritmasından hem daha küçük amaç fonksiyonu değerleri hesaplamış hem de belgeleri daha doğru bir şekilde kümelere atamıştır.

Parçacık sürü optimizasyonu ile yapılan karşılaştırmada ise yarasa algoritması yine daha küçük amaç fonksiyonu değerleri ile kümeleme işlemini gerçekleştirmiştir. Bu kümeleme işlemi sırasında benzerlik ölçüsü olarak kosinüs uzaklığı, ölçüt fonksiyonu olarak ise belgelerin küme merkezlerine uzaklıklarını dikkate alan I_2 ölçüt fonksiyonu kullanılmıştır. Ancak, yarasa algoritmasının kümeleme kalitesinin arttırılmaya ihtiyacı vardır. Kümeleme kalitesinin değerlendirilmesi için hesaplanan doğruluk değerlerine göre yarasa algoritması Classic4-1 ve BBCSport-2 alt belge setlerinde yaklaşık %2'lik bir belgeyi doğru kümeye atayamamıştır. Reuters-1 ve Classic4-2 alt belge setlerinde bu miktar sırasıyla %3 ve %5 olurken, Reuters-2 ve BBCSport-1 alt belge setlerinde sırasıyla %6 ve %7'yi bulmuştur. Aynı şekilde, kümeleme kaltesinin değerlendirilmesinde önemli bir gösterge olan F-ölçüsüne göre, yarasa algoritmasının performansı parçacık sürü optimizasyonu algoritmasına göre yine %2-%7 oranında daha düşük çıkmıştır.

Yarasa algoritmasının kümeleme performansını arttırmak için algoritmanın farklı metasezgisellerle birleştirilerek geliştirilmesi gelecekteki araştırma konularından biri olarak hedeflenmektedir. Ayrıca algoritmada kullanılan parametrelerin deney tasarımı yaklaşımı ile analizi yapılarak, daha iyi sonuçların elde edilmesi için parametre optimizasyonu yapılması gelecek araştırma konularından bir diğeri olarak düşünülmektedir.

Tez çalışması kapsamında, belgeleri kümeleme sırasında belirli ölçüt fonksiyonları kullanılmıştır. Tüm ölçüt fonksiyonları kullanılarak yapılacak bir değerlendirme yine gelecekteki başka bir çalışmanın konusudur.

Bu tez çalışmasında, literatürde yer alan İngilizce belge setleri kullanılarak belge kümeleme işlemi gerçekleştirilmiştir. Türkçe online gazetelerden ya da web sitelerinden oluşturulacak belge setlerine, yarasa algoritmasını uygulayarak algoritmanın

performansının bir de bu setler üzerinde değerlendirilmesi gelecekte yapılması muhtemel başka bir çalışmadır.

Literatürde yarasa algoritması gibi, doğadan esinlenerek geliştirilmiş ve sürü zekası tabanlı birçok algoritma bulunmaktadır. Tüm bu algoritmaların belge kümeleme konusunda birbirlerine göre performanslarının incelenmesi gelecekteki araştırma konularından bir başkası olarak görülmektedir.

KAYNAKLAR

Abraham, A., Biswas, A., Dasgupta, S., Das, S., 2008, Analysis of reproduction operator in bacterial foraging optimization algorithm, *IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence),* 1476-1483.

Akhtar, S., Ahmad, A. R., Abdel-Rahman, E. M., 2012, A Metaheuristic Bat-Inspired Algorithm for Full Body Human Pose Estimation, *Ninth Conference on Computer and Robot Vision (CRV),* IEEE, 369-375.

Akter, R., Chung, Y., 2013, An Evolutionary Approach for Document Clustering, *IERI Procedia*, *4*, 370-375.

Alihodzic, A., Tuba, M., 2013, Framework for Bat Algorithm Optimization Metaheuristic, *Recent Researches in Medicine, Biology and Bioscience,* Recent Advances in Biology and Biomedicine Series 4 , WSEAS Press, 157-162, ISBN: 978-960-474-326-1

Aly, W. M., Kelleny, H. A., 2014, Adaptation Of Cuckoo Search For Documents Clustering, *International Journal Of Computer Applications*, *86*(1), 4-10.

Anastasiu, D. C., Tagarelli, A., Karypis, G., 2013, *Document Clustering: The Next Frontier*, Technical Report, University of Minnesota.

Antoniou, A., Lu, W-S., 2007, *Practical Optimization-Algorithms and Engineering Applications,* Springer Science and Business Media, LLC, ISBN: 0-387-71106-6.

Azaryuon, K., Fakhar, B., 2013, A Novel Document Clustering Algorithm Based on Ant Colony Optimization Algorithm, *Journal of mathematics and computer Science,* 7, 171-180.

BBC Sport, http://mlg.ucd.ie/datasets/bbc.html

Bisht, S., Paul, A., 2013, *Document Clustering: A Review*. International Journal of Computer Applications, 73(11), 26-33.

Blum, C., Roli, A., 2003, Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM Computer Survey*, 35 (3), 268–308.

Brownlee, J., 2011, *Clever Algorithms: Nature-Inspired Programming Recipes*, Lulu, ISBN: 978-1-4467-8506-5.

Bora, T. C., Coelho, L. S., Lebensztajn, L., 2012, Bat-Inspired optimization approach for the brushless DC wheel motor problem, *IEEE Transactions on Magnetics*, 48 (2), 947-950.

Büyüksaatçı, S., 2015, Bat Algorithm Application for the Single Row Facility Layout Problem, *Recent Advances in Swarm Intelligence and Evolutionary Computation,* 101-120, Springer International Publishing, ISBN:978-3-319-13825-1.

Chiang, M. M. T., Mirkin, B., 2007, Experiments for the number of clusters in k-means. In *Progress in Artificial Intelligence*, Springer Berlin Heidelberg, 395-405.

Chu, S-C., Tsai, P-W., Pan, J-S., 2006, Cat swarm optimization., *PRICAI 2006: Trends in Artificial Intelligence*, Springer Berlin Heidelberg, 854-858.

Chu, S-C., Tsai, P-W., 2007, Computational intelligence based on the behavior of cats, *International Journal of Innovative Computing, Information and Control*, *3* (1), 163-173.

Classic 4, http://www.dataminingresearch.com/index.php/tag/dataset-2/

Cui, X., Potok, T. E., Palathingal, P., 2005, Document clustering using particle swarm optimization, *Proceedings of IEEE* in *Swarm Intelligence Symposium, SIS 2005,* 185-191.

Cui, X., Potok, T. E., 2005, Document clustering analysis based on hybrid PSO+K-means algorithm, *Journal of Computer Sciences (special issue)*, *27*-33.

Cui, X., Gao, J., Potok, T. E., 2006, A flocking based algorithm for document clustering analysis, *Journal of systems architecture*, *52*(8), 505-515.

Damodaram, R., Valarmathi, M. L., 2012, Phishing website detection and optimization using Modified bat algorithm, *International Journal of Engineering Research and Applications*, 2 (1), 870-876.

Das, S., Abraham, A., Konar, A., 2009, *Metaheuristic clustering* (Vol. 178). Springer Science & Business Media, e-ISBN 978-3-540-93964-1.

Davies, D.L., Bouldin, D.W., 1979, A cluster separation measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*, 224–227.

Dorigo, M., Maniezzo, V., Colorni, A., 1991, *The Ant System: An autocatalytic optimizing process,* Technical Report (No:91-016) Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Eberhart, R., Kennedy, J., 1995, A new optimizer using particle swarm theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS'95),* 39-43.

Eusuff, M. M., Lansey, K. E., 2003, Optimization of water distribution network design using the shuffled frog leaping algorithm, *Journal of Water Resources Planning and Management*, 129 (3), 210-225.

Eusuff, M., Lansey, K., Pasha, F., 2006, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, *Engineering Optimization*, 38 (2), 129-154.

Fenton, M. B., 2004, Bat natural history and echolocation, *Bat Echolocation Research: tools, techniques and analysis*, 2-5.

Fister, Jr, I., Fister, D., Yang, X. S., 2013, A hybrid bat algorithm, *arXiv preprint arXiv:1303.6310*.

Fister, I., Rauter, S., Yang, X. S., Ljubič, K., 2015, Planning the sports training sessions with the bat algorithm, *Neurocomputing*, *149*, 993-1002.

Forsati, R., Keikha, A., Shamsfard, M., 2015, An improved bee colony optimization algorithm with an application to document clustering, *Neurocomputing*, *159*, 9-26.

Frakes, W. B., Baeza-Yates, R., 1992, Information retrieval: data structures and algorithms, Englewood Cliffs, NJ: Prentice Hall.

Fung, B.C., Wang, K., Ester, M., 2006, Hierarchical document clustering, *Encyclopedia of Data Warehousing and Mining*, 555-559.

Gandomi, A. H., Yang, X. S., Alavi, A. H., Talatahari, S., 2013, Bat Algorithm for constrained optimization tasks, *Neural Computing and Applications*, 1-17.

Gao, X., Lu, Y., 2012, Automatic text clustering via particle swarm optimization, *JDCTA: International Journal of Digital Content Technology and its Applications*, 6(23), 12-21.

Geem, Z. W., Kim, J. H., Loganathan, G. V., 2001, A new heuristic optimization algorithm: harmony search, *Simulation*, *76* (2), 60-68.

Glover, F., 1986, Future paths for integer programming and links to artificial intelligence, *Computers & operations research*, 13(5), 533-549.

Glover, F., Kochenberger, G.A., 2003, *Handbook of Metaheuristics*, Kluwer Academic Publishers, ISBN: 0-306-48056-5.

Goss, S., Aron, S., Deneubourg, J. L., Pasteels, J. M., 1989, Self-organized shortcuts in the Argentine ant, *Naturwissenschaften*, 76(12), 579-581.

Goyal, S., Patterh, M. S., 2013, Performance of BAT Algorithm on Localization of Wireless Sensor Network, *International Journal of Computers & Technology*, *6*(3), 351-358.

Heppner, F., Grenander, U., 1990, A stochastic nonlinear model for coordinated bird flocks, *The Ubiquity* of *Chaos,* AAAS Publications, Washington DC, 233-238.

Holland, J. H., 1975, *Adaptation in natural and artificial system: An introductory analysis with applications to biology, control, and artificial intelligence*, MI: University of Michigan Press.

Huang, A., 2008, Similarity measures for text document clustering, *In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand, 49-56.

Jensi, R., Jiji, G. W., 2013, A Survey on optimization approaches to text document clustering, *International Journal of Computational Sciences & Applications (IJCSA),* 3(6), 31-44.

Karol, S., Mangat, V., 2013, Evaluation of text document clustering approach based on particle swarm optimization, *Central European Journal of Computer Science*, *3*(2), 69-90.

Khan, K., Nikov, A., Sahai, A., 2011, A fuzzy bat clustering method for ergonomic screening of office workplaces, *Third International Conference on Software, Services and Semantic Technologies (S3T 2011)*, Springer Berlin Heidelberg, 59-66.

Khan, K., Sahai, A., 2012a, A fuzzy c-means bi-sonar-based metaheuristic optimization algorithm, *International Journal of Artificial Intelligence and Interactive Multimedia,* 1 (7), 26-32.

Khan, K., Sahai, A., 2012b, A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context, *International Journal of Intelligent Systems and Applications (IJISA)*, 4 (7), 23-29.

Kırkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983, Optimization by simulated annealing, *Science*, 220.4598, 671-680.

Koziel, S., Yang, X-S. (Eds.), 2011, *Computational Optimization, Methods and Algorithms*, Springer, Verlag Berlin Heidelberg, ISBN: 978-3-642-20858-4.

Komarasamy, G., Wahi, A., 2012, An Optimized K-Means Clustering Technique Using Bat Algorithm, *European Journal of Scientific Research*, 84 (2), 263-273.

Kumar, V., Tan, P. N., Steinbach, M., 2006, Cluster analysis: basic concepts and algorithms, *Introduction to data mining*, 487-586.

Kumar, A., Chakarverty, S., 2011, Design optimization for reliable embedded system using Cuckoo Search, *3rd International Conference on Electronics Computer Technology (ICECT),* 264-268.

Lemma, T. A., Bin Mohd Hashim, F., 2011, Use of fuzzy systems and bat algorithm for exergy modeling in a gas turbine generator, *Colloquium on Humanities, Science and Engineering (CHUSER)*, IEEE, 305-310.

Lin, J.-H., Chou ,C.-W., Yang, C-H., Tsai, H-L., Lee, I-H., 2012, A Bio-inspired Optimization Algorithm for Modeling the Dynamics of Biological Systems, *In Third International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA)*, 206-211.

Machnik, Ł., 2007, A document clustering method based on ant algorithms, *Task Quarterly*, *11*(1-2), 87-102.

Mahdavi, M., Abolhassani, H., 2009, Harmony K-means algorithm for document clustering, *Data Mining and Knowledge Discovery*, *18*(3), 370-391.

Marichelvam, M. K., Prabaharan, T., 2012, A Bat Algorithm For Realistic Hybrid Flowshop Scheduling Problems to Minimize Makespan and Mean Flow Time, *ICTACT Journal of Soft Computing*, 3 (1), 428-433.

Metropolis; N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., 1953, Equaiton of state calculations by fast computing machines, *Journal of Chemical Physics*, 21, 1087-1092.

Mishra, S., Shaw, K., Mishra, D., 2012, A New Meta-heuristic Bat Inspired Classification Approach for Microarray Data, *Procedia Technology*, 4, 802-806.

Montgomery, D. C., Runger, G. C., Hubele, N. F., 2011, *Engineering statistics*, Fifth Edition, John Wiley & Sons, 196, ISBN-13 978- 0-470-63147-8.

Mucherino, A., Seref, O., 2007, Monkey Search: A Novel Meta-Heuristic Search for Global Optimization, *Data Mining, Analysis and Optimization in Biomedicine,* 953, 162-173.

Muller, S. D., Marchetto, J., Airaghi, S., Kournoutsakos, P., 2002, Optimization based on bacterial chemotaxis, *IEEE Transactions on Evolutionary Computation*, *6* (1), 16-29.

Musikapun, P., Pongcharoen, P., 2012, Solving multi-stage multi-machine multi-product scheduling problem using Bat algorithm, *Proceedings of 2nd International Conference on Management and Artificial Intelligence (IPEDR)*, 98-102.

Nawi, N. M., Rehman, M. Z., Khan, A., 2014, A New Bat Based Back-Propagation (BAT-BP) Algorithm, *Advances in Systems Science*, Springer International Publishing, 395-404.

Parpinelli, R. S., Lopes, H. S., 2011, New inspirations in swarm intelligence: a survey, *International Journal of Bio-Inspired Computation*, *3* (1), 1-16.

Passino, K.M., 2002, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Systems Magazine*, 22 (3), 52-67.

Premalatha, K., Natarajan, A. M., 2010a, A literature review on document clustering, *Information Technology Journal*, *9* (5), 993-1002.

Premalatha, K., Natarajan, A. M., 2010b, Hybrid PSO and GA models for Document Clustering. *Int. J. Advance. Soft Comput. Appl*, *2* (3), 302-320.

Qing, A., 2009, *Differential Evolution: Fundamentals and Applications in Electrical Engineering*, John Wiley and Sons, Singapore, ISBN: 978-0-470-82392-7.

Ramesh, B., Mohan, V., Reddy, V., 2013, Application of bat algorithm for combined economic load and emission dispatch, *International Journal of Electrical and Electronic Engineering & Telecommunications*, 2, 1-9.

Ray, S., Turi, R. H., 1999, Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*,137-143.

Reddy, V. U., Manoj, A., 2012, Optimal Capacitor Placement for Loss Reduction in Systems Using Bat Algorithm, *IOSR Journal of Engineering*, 2 (10), 23-27.

Rendón, E., Abundez, I., Arizmendi, A., Quiroz, E. M., 2011, Internal versus External cluster validation indexes, *International Journal of computers and communications*, *5*(1), 27-34.

Reuters-21578, http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

Reynolds, C. W., 1987, Flocks, herds and schools: a distributed behavioral model, *Computer Graphics,* 21 (4), 25-34.

Rothlauf, F., 2011, *Design of Modern Heuristics Principles and Application*, Springer, Verlag Berlin Heidelberg, ISBN: 978-3-540-72961-7.

Rui, T., Fong, S., Yang, X. S., Deb, S., 2012, Nature-inspired Clustering Algorithms for Web Intelligence Data. *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT),* Vol. 3, 147-153.

Salton, G., Wong, A., Yang, C-S., 1975, A vector space model for automatic indexing, *Communications of the ACM*, 18.11, 613-620.

Schnitzler, H-U., Kalko, E.K.V., 2001, Echolocation by Insect-Eating Bats, *Bioscience*, 51 (7), 557-569.

Shah, N., Mahajan, S., 2012, Document Clustering: A Detailed Review, *Int'l Journal of Applied Information Systems*, 4 (5), 30-38.

Sood, M., Bansal, S., 2013, K-Medoids Clustering Technique using Bat Algorithm, *International Journal of Applied Information Systems (IJAIS)*, 5 (8), 20-22.

Steinbach, M., Karypis, G., Kumar, V., 2000, A comparison of document clustering techniques, In *KDD workshop on text mining*, Vol. 400, No. 1, 525-526.

Sugar, C. A., James, G. M., 2003, Finding the number of clusters in a dataset. *Journal of the American Statistical Association*, *98* (463).

Taha, A. M., Tang, A. Y., 2013, Bat algorithm for rough set attribute reduction, *Journal of Theoretical and Applied Information Technology*, *51*(1).

Talbi, E-G., 2009, *Metaheuristics from Design to Implementation*, John Wiley & Sons, Inc., Hoboken, New Jersey, ISBN: 978-0-470-27858-1.

Tarczynski, T., 2011, Document Clustering-Concepts, Metrics and Algorithms. *International Journal of Electronics and Telecommunications*, 57.3: 271-277.

Tibshirani, R., Walther, G., Hastie, T., 2001, Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411-423.

Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., Istanda, V., 2012, Bat algorithm inspired algorithm for solving numerical optimization problems, *Applied Mechanics and Materials*, 148, 134-137.

Vester, K. L., Martiny, M. C., 2005, *Information retrieval in document spaces using clustering* (Doctoral dissertation, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark).

Wang, G., Guo, L., 2013, A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization, *Journal of Applied Mathematics*.

Wang, W., Wang, Y., Wang, X., 2013, Bat algorithm with recollection, *Intelligent Computing Theories and Technology*, Springer Berlin Heidelberg, 207-215

Yang, X-S., 2009a, Harmony search as a metaheuristic algorithm, *Music-inspired harmony search algorithm*, Springer Berlin Heidelberg, 1-14.

Yang, X-S., 2009b, Firefly algorithms for multimodal optimization, *Stochastic algorithms: foundations and applications*, Springer Berlin Heidelberg, 169-178.

Yang, X-S., 2010a, *Engineering Optimization An Introduction with Metaheuristic Applications*, John Wiley & Sons Inc., Hoboken, New Jersey, ISBN: 978-0-470-58246-6.

Yang, X-S., 2010b, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, ISBN: 1-905986-28-9.

Yang, X-S., 2010c, Firefly algorithm, Levy flights and global optimization, *Research and Development in Intelligent Systems XXVI*, Springer London, 209-218.

Yang, X-S., 2010d, A new metaheuristic bat-inspired algorithm, *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer Berlin Heidelberg, 65-74.

Yang, X-S., 2011, Bat algorithm for multi-objective optimisation, *International Journal of Bio-Inspired Computation*, 3 (5), 267-274.

Yang, X-S., 2012, Nature-Inspired Mateheuristic Algorithms: Success and New Challenges*, Journal of Engineering & Information Technology*, 1(1), 1-3.

Yang, X-S., Deb, S., 2009, Cuckoo search via Lévy flights, *World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, 210-214.

Yang, X-S., Deb, S., 2010, Engineering optimisation by cuckoo search, *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4), 330-343.

Yang, X-S., Gandomi, A.H., 2012, Bat algorithm: a novel approach for global engineering optimization, *Engineering Computations*, 29 (5), 464-483.

Yang, X. S., Karamanoglu, M., Fong, S., 2012b, Bat algorithm for topology optimization in microelectronic applications, *IEEE International Conference on Future Generation Communication Technology (FGCT2012), British Computer Society*, 150–155.

Yılmaz, S., Kucuksille, E. U., 2013, Improved bat algorithm (IBA) on continuous optimization problems, *Lecture Notes on Software Engineering*, 1(3), 279-283.

Yılmaz, S., Kucuksille, E. U., CENGIZ, Y., 2014, Modified Bat Algorithm. *Electronics and Electrical Engineering*, 20(2), 71-78.

Zang, H., Zhang, S., Hapeshi, K., 2010, A review of nature-inspired algorithms, *Journal of Bionic Engineering*, *7*, 232-237.

Zaw, M. M., Mon, E. E., 2013, Web document clustering using cuckoo search clustering algorithm based on levy flight, *International Journal of Innovation and Applied Studies*, *4*(1), 182-188.

Zhang, J., Wang, G., 2012, Image Matching Using a Bat Algorithm with Mutation, *Applied Mechanics and Materials*, 203, 88-93.

Zhao, Y., Karypis, G., 2001, *Criterion functions for document clustering: Experiments and analysis*, Technical report, 01-40.

Zhao, Y., Karypis, G., 2002, Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, ACM, 515-524.

Zhao, Y., Karypis, G., 2004, Empirical and theoretical comparisons of selected criterion functions for document clustering, *Machine Learning*, *55*(3), 311-331.

Zhou, Y., Xie, J., Zheng, H., 2013, A hybrid bat algorithm with path relinking for capacitated vehicle routing Problem, *Mathematical Problems in Engineering*, Hindawi Publishing Corporation

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı	Sinem BÜYÜKSAATÇI
Uyruğu	TC
Doğum tarihi, Yeri	11/04/1984, Fatih
Telefon	0 537 437 77 04
E-mail	sinemb@istanbul.edu.tr

Eğitim

Derece	Kurum/Anabilim Dalı/Programı	Yılı
Doktora	İ.Ü. Fen Bilimleri Enstitüsü / Endüstri Mühendisliği ABD.	2015
Yüksek Lisans	İ.Ü. Fen Bilimleri Enstitüsü / Endüstri Mühendisliği ABD.	2009
Lisans	İ.Ü Endüstri Mühendisliği	2006
Lise	Uzunköprü Muzaffer Atasay Anadolu Lisesi	2002

Makaleler / Bildiriler

Kitap Bölümü

BÜYÜKSAATÇI, S., 2015, "Bat Algorithm Application for the Single Row Facility Layout Problem", *Recent Advances in Swarm Intelligence and Evolutionary Computation,* Xin-She Yang, Eds., Springer International Publishing, 101-120

Uluslararası Hakemli Dergilerde Yayınlanan Makaleler

BÜYÜKSAATÇI, S., ESNAF, Ş., 2014, "Carbon Emission Based Optimization Approach for the Facility Location Problem", *The Online Journal of Science and Technology*, 4, 9-20.

Uluslal Hakemli Dergilerde Yayınlanan Makaleler

BÜYÜKSAATÇI, S., KÜÇÜKDENİZ, T., ESNAF, Ş., 2008, "Geri Dönüşüm Tesislerinin Yerinin Gustafson-Kessel Algoritması - Konveks Programlama Melez Modeli Tabanlı Simülasyon İle Belirlenmesi", *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, Yıl:7, Sayı:13, Bahar 2008/1, 1-20

Uluslararası Bilimsel Toplantılarda Sunulan ve Bildiri Kitabında (Proceedings) Yayınlanan Bildiriler

BÜYÜKSAATÇI, S., TÜYSÜZ, F., BİLEN, K., 2015, "Balancing and Simulation of Assembly Line in an LCD Manufacturing Company", *6th International Conference on Modeling, Simulation and Applied Optimization*, İstanbul.

BÜYÜKSAATÇI, S., BARAY, A., 2014, "A Comparison Of Three Nature-Inspired Metaheuristics For The Single Row Facility Layout Problem", *International Conference on Engineering and Applied Sciences Optimization (OPT-i),* Kos Island, YUNANISTAN, 1032-1043.

KÜÇÜKDENİZ, T., **BÜYÜKSAATÇI, S.,** 2014, "Imperialist Competitive Algorithm Compared With Particle Swarm Optimization And K-Means On Document Clustering", *International Conference on Engineering and Applied Sciences Optimization (OPT-i),* Kos Island, YUNANISTAN, 1206-1216.

BÜYÜKSAATÇI, S., ESNAF, Ş., 2013, "Carbon Emission Based Optimization Approach for the Facility Location Problem", *International Science and Technology Conference (ISTEC*), Roma-Italya, 70-81.

Uluslararası Bilimsel Toplantılarda Sunulan ve Özeti Yayınlanan Bildiriler

ESNAF, Ş., KÜÇÜKDENİZ, T., **BÜYÜKSAATÇI, S.,** 2008, "Fuzzy C-Means and Center of Gravity Combined Model for a Capacitated Planar Multiple Facility Location Problem", *International Conference on Multivariate Statistical Modelling & High Dimensional Data Mining*, Kayseri, 131.

Ulusal Bilimsel Toplantılarda Sunulan ve Bildiri Kitabında (Proceedings) Yayınlanan Bildiriler

BÜYÜKSAATÇI, S., BARAY, A., ESNAF, Ş., 2012, "Tabanca Kabzası Tasarımı İçin El Ölçülerinin K-Ortalamalar ve Bulanık C-Ortalamalar Algoritmaları ile Sınıflandırılması", *12. Ulusal Üretim Araştırmaları Sempozyumu*, İzmir-Menemen, 911-924.

SÖNMEZ, A., **BÜYÜKSAATÇI, S.,** TUNÇBİLEK, N., GENÇYILMAZ, G., 2008, "İMKB'de İşlem Gören Bankaların Performanslarının Değerlendirilmesi", *10. Ulusal Üretim Araştırmaları Sempozyumu*, Girne-KKTC, 99-114.

TUNÇBİLEK, N., **BÜYÜKSAATÇI, S.,** ESNAF, Ş., 2010, "Üretim Senaryolarının Analizi İçin Kısıtlar Teorisi Tabanlı Bir Karar Destek Sistemi", *10. Ulusal Üretim Araştırmaları Sempozyumu*, Girne-KKTC, 753-763.

BÜYÜKSAATÇI, S., KÜÇÜKDENİZ, T., ESNAF, Ş., 2008, "Asfalt Geri Dönüşüm Tesislerinin Optimum Yer ve Kapasitelerinin Belirlenmesi İçin Bulanık Öbekleme Esaslı Simülasyon Uygulaması", *VIII. Ulusal Üretim Araştırmaları Sempozyumu*, İstanbul, 237-247