T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KARGA VE YARASA TABANLI ALGORİTMALARIN YENİ VERSİYONLARININ GELİŞTİRİLMESİ VE PERFORMANSLARININ DEĞERLENDİRİLMESİ

Hazırlayan
Zaher AKHDIR

Danışman
Dr. Öğr. Üyesi Mustafa DANACI

Yüksek Lisans Tezi

Temmuz 2020
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KARGA VE YARASA TABANLI ALGORİTMALARIN YENİ VERSİYONLARININ GELİŞTİRİLMESİ VE PERFORMANSLARININ DEĞERLENDİRİLMESİ

Hazırlayan
Zaher AKHDIR

Danışman
Dr. Öğr. Üyesi Mustafa DANACI

Yüksek Lisans Tezi

Temmuz 2020
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Zaher AKHDIR

"Karga ve Yarasa Tabanlı Algoritmaların Yeni Versiyonlarının Geliştirilmesi ve Performanslarının Değerlendirilmesi" adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi' ne uygun olarak hazırlanmıştır.

Hazırlayan
Zaher AKHDIR

Danışman
Dr. Öğr. Üyesi Mustafa DANACI

Bilgisayar Mühendisliği ABD Başkanı
Prof. Dr. Veysel ASLANTAŞ

Dr. Öğr. Üyesi Mustafa DANACI danışmanlığında **Zaher AKHDIR** tarafından hazırlanan **"Karga ve Yarasa Tabanlı Algoritmaların Yeni Versiyonlarının Geliştirilmesi ve Performanslarının Değerlendirilmesi"** adlı bu çalışma jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği** Anabilim Dalında **Yüksek Lisans** tezi olarak kabul edilmiştir.

29/07/2020

JÜRİ:

Danışman : Dr. Öğr. Üyesi Mustafa DANACI .....

Üye : Prof. Dr. DERVİŞ KARABOĞA .....

Üye : Dr. Öğr. Üyesi HASAN BADEM .....

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun ……...… tarih ve ……..……sayılı kararı ile onaylanmıştır.

29/07/2020

Prof. Dr. Mehmet AKKURT

Enstitü Müdürü

ÖNSÖZ/TEŞEKKÜR

“Karga ve Yarasa Tabanlı Algoritmaların Yeni Versiyonlarının Geliştirilmesi ve Performanslarının Değerlendirilmesi” isimli yüksek lisans tez çalışmam kapsamında bilgi, tecrübe ve desteğini esirgemeyen kıymetli danışman hocam sayın Dr. Öğr. Üyesi Mustafa DANACI’ya teşekkür ederim.

Ayrıca her zaman olduğu gibi yüksek lisans çalışmalarımda da beni destekleyen sevgili aileme en içten teşekkürlerimi sunarım.

Zaher AKHDIR

Temmuz 2020, KAYSERİ

KARGA VE YARASA TABANLI ALGORİTMALARIN YENİ VERSİYONLARININ GELİŞTİRİLMESİ VE PERFORMANSLARININ DEĞERLENDİRİLMESİ

Zaher AKHDIR

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Temmuz 2020
Danışman: Dr.Öğr. Üyesi, Mustafa DANACI

ÖZET

Parçacık Sürü Optimizasyonu (PSO), Genetik Algoritma (GA) ve diğer algoritmalar gibi meta-sezgisel algoritmalar, birçok zorlu optimizasyon problemini çözmek için güçlü ve ünlü yöntemlerdir. Bu çalışmada iki ayrı algoritma geliştirilmiş ve popüler test problemlerinden Benchmark ve gezgin satıcı problemlerine uygulanmıştır.

Bu tezdeki ilk çalışmada, Hibrit Karga Arama Yarasa Algoritması (Hybrid Crow Search BAT Algorithm, HCSBA) adlı yeni bir hibrit algoritması önerilmiştir. Bu çalışmada, Karga Arama Algoritması (Crow Search Algorithm CSA)na ait farkındalık faktörü ve rakip kargayı izleme özellikleri ve Yarasa Algoritması (Bat Algorithm BA)'na ait ses yüksekliği ve yerel arama özelliklerine ek olarak sürünün bireylerinin konumlarını güncellemek için kuantum davranışlı PSO denklemleri kullanılarak Hibrit Karga Arama Yarasa Algoritması (Hybrid Crow Search BAT Algorithm, HCSBA) adlı yeni bir hibrit model geliştirilmiştir. Burada geliştirilen hibrit model ile hem yerel optimumdan kaçınma hem de global optimumu yakalamak için algoritmanın keşif ve sömürü yetenekleri arasında uygun bir denge kurulmuştur. Geliştirilen model benchmark problemleriyle test edilerek literatürde yer alan Genetik Algoritma (GA), Parçacık Sürü Optimizasyon (PSO), Diferansiyel Gelişim (DE), Guguk Arama (CS) ve Ateşböceği Algoritması (FA) vb. ile kıyaslanmış ve literatürde yer diğer literatürdeki tanınmış meta-sezgisel yöntemlere kıyasla çok başarılı ve rekabetçi bir çözüm sağladığı görünmüştür.

Bu tezdeki ikinci temel çalışmada, gezgin satıcı problemini (Travelling Salesman Problem, TSP) çözmek için Karga Arama Algoritması (CSA) kullanılmıştır. Karga Arama Algoritması genelde süreklilik içeren optimizasyon problemlerini çözmek için kullanılmakta iken bu çalışmada Hamming mesafesi ve 2-opt algoritması kullanılarak süreklilik içermeyen gezgin satıcı gibi kombinatorial problemlerin çözümüne uyarlamak

için Ayrık Karga Algoritması geliştirilmiştir. İlk defa TSP çözümleri için uygulanan Ayrık Karga Algoritması ile yapılan çalışmada 15 adet farklı TSP problemine çözüm üretilmiştir. Elde edilen sonuçlar ve diğer iyi bilinen algoritmalardan Tepe Tırmanışı (Hill Climbing HC), Yerel Arama (Local Search LS), Tavlama Algoritması (Simulated Annealing SA), Tabu Arama (Tabu Search TS), Kanguru Algoritması (Kangro Search KA), Yapay Arı Kolonisi (Artificial Bee Colony ABC), Genetik Algoritma (Genetic Algorithm GA) ve Karınca Kolonisi Opytimizasyonu Algoritması (Ant Colony Optimization ACO) ile kıyaslanmış ve çok başarılı sonuçlar elde edilmiştir.

Anahtar Kelimeler: Meta-Sezgisel, Karga Arama Algoritması, Yarasa Algoritması, Benchmark Fonksiyonları, Gezgin Satıcı Problemi.

DEVELOPMENT OF NEW VERSIONS OF CROW AND BAT BASED ALGORITHMS AND EVALUATION OF THEIR PERFORMANCE

Zaher AKHDIR

Erciyes University, The Graduate School Of Natural And Applied Scıences
Department of Computer Engineering
Master Thesis, July 2020
Advısor: Dr. Mustafa DANACI

ABSTRACT

Meta-heuristic algorithms such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and other algorithms are powerful and famous methods to solve many difficult optimization problems. In this thesis two main algorithms were suggested and developed and applied to solve Benchmark problems and Travelling Salesman Problem.

In the first study a new hybrid algorithm called Hybrid Crow Search BAT Algorithm, HCSBA is proposed. In this study, using the quantum behavior PSO equations to update the positions of the flock individuals, in addition to pursuing other swarm members feature and the awareness factor of Crow Search Algorithm (Crow Search Algorithm CSA) and the loudness and local search features of Bat Algorithm (Bat Algorithm BA). With the hybrid model developed in this thesis, an appropriate balance has been established between the exploration and exploitation capabilities of the algorithm to avoid local optimum and to achieve the global optimum. The developed model has been tested with benchmark problems and compared with Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Development (DE), Cuckoo Search (CS) and Firefly Algorithm (FA) etc. algorithms in the literature, and it has been found to provide a very successful and competitive solution compared to other well-known meta-heuristic methods in the literature.

The second study in this thesis deals with the Travelling Salesman Problem (TSP) with metaheuristics which is the most general class of the stochastic optimization techniques. The TSP is a NP-hard problem in the optimization field. District Crow Search Algorithm (DCSA) was developed and used to solve TSP. To achieve this goal, Hamming distance and 2-opt algorithm were used. DCSA was applied for the first time for 15 TSP problems and the results were compared with other well-known algorithms (Hill Climbing HC, Local Search LS, Simulated Annealing SA, Tabu Search (TS),

Kangaroo Algorithm (KA), Artificial Bee). The new developed DSCA appears to provide a very successful and competitive results.

Keywords: Meta-heuristic, Crow Search Algorithm, Bat Algorithm, Benchmark Functions, Traveling Salesman Problem.

İÇİNDEKİLER

KARGA VE YARASA TABANLI ALGORİTMALARIN YENİ VERSİYONLARININ GELİŞTİRİLMESİ VE PERFORMANSLARININ DEĞERLENDİRİLMESİ

KISALTMALAR

ABC	Yapay Arı Kolonisi
ACO	Karınca Kolonisi Optimizasyon
BA	Yarasa Algoritması
CS	Guguk Kuşu Arama Algoritması
CSA	Karınca Optimizasyon Algoritması
DCSA	Ayrık Karga Arama Algoritması
DE	Diferansiyel Gelişim
EA	Evrimsel Algoritma
GA	Genetik algoritma
HC	Tepe Tırmanışı
HCSBA	Hibrit Karga Arama Yarasa Algoritması
PSO	Parçacık Sürü Optimizasyonu
SA	Benzetimli tavlama
TS	Tabu Arama

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

1. BÖLÜM

GENEL BİLGİLER ve LİTERATÜR ÇALIŞMASI

1.1. Optimizasyon Nedir?

Optimizasyon, bir problemin ya iyileştirilmesi ya da en iyi sonucunun bulunması anlamına gelir. Gerçek dünya problemleri çözülmesi zor problemlerdir. Bu problemleri çözmek için çeşitli yöntemler geliştirilmiştir [1, 2]. Son yirmi yıla bakıldığında meta-sezgisel teknikler gerçek mühendislik problemlerinin çözülmesinde daha popüler hale gelmiştir. Çünkü oldukça basit konsepte sahip olup uygulaması kolay, yerel optimumdan kaçabilen özellikleriyle farklı alanlara ait çok çeşitli problemlere uyarlanmaktadır [3].

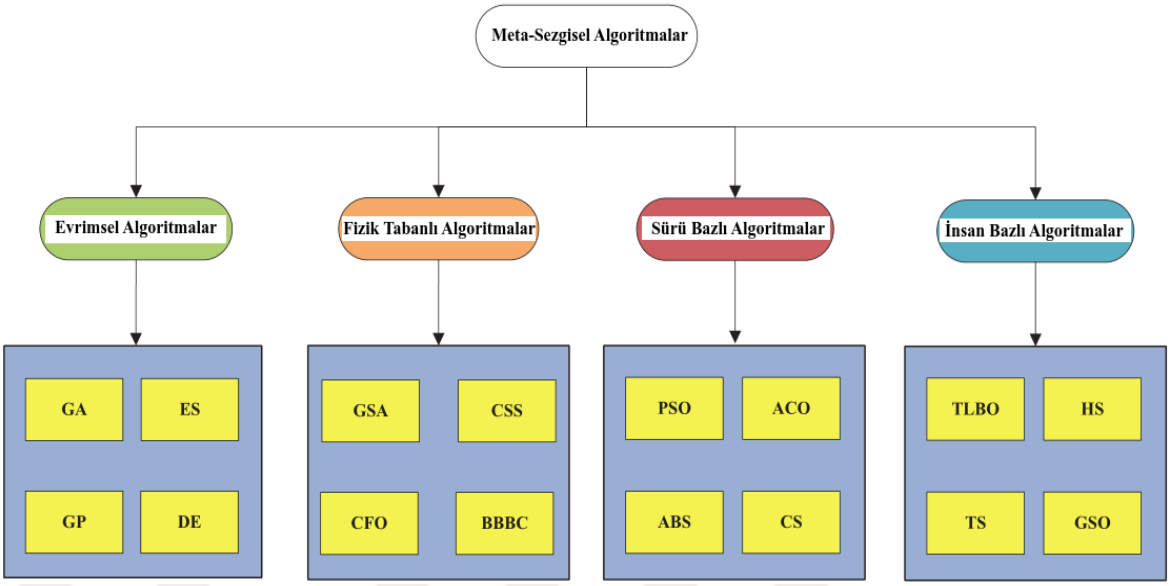
Meta-sezgisel optimizasyon algoritmaları:

Meta-sezgisel optimizasyon algoritmaları mühendislik uygulamalarında gittikçe daha popüler hale geliyor, çünkü [1][3]:

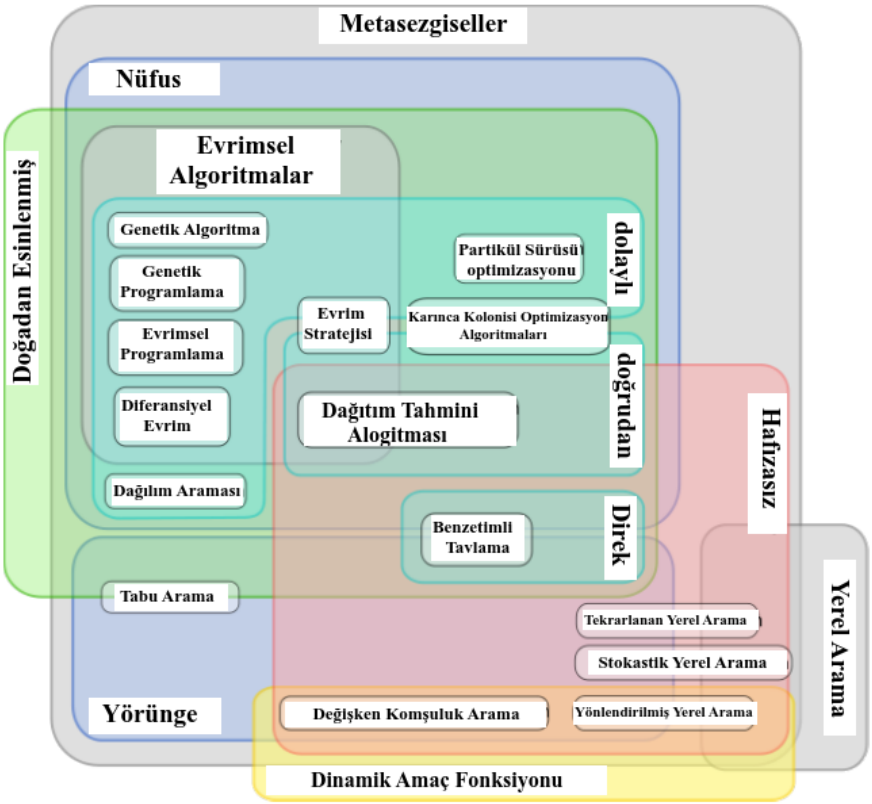
(i) Oldukça basit kavramlara sahip olup uygulaması kolaydır,
(ii) Yerel optima'yı atlayabilir;
(iii) Farklı disiplinleri kapsayan çok çeşitli problemlerde kullanılabilir.

Doğadan ilham alan meta-sezgisel algoritmalar biyolojik veya fiziksel olayları taklit ederek optimizasyon problemlerini çözer. Bunlar Şekil 1.2. görüldüğü gibi dört ana kategoride gruplandırılabilir; evrim temelli, fizik temelli, sürü temelli ve insan temelli yöntemler.

Şekil 1.1. [4] ve Şekil 1.2 [5]. ilgili algoritmaları ve özelliklerini göstermektedir.



Şekil 1.1. Meta-Sezgisel Algoritmalar Şeması



Şekil 1.2. Meta-Sezgisel Algoritmaların Çeşitleri ve Özellikleri

Evrim tabanlı algoritmalar

Evrime dayalı yöntemler, doğal evrim yasalarından esinlenmiştir. Arama süreci, sonraki nesiller boyunca gelişen rastgele oluşturulmuş bir popülasyonla başlar. Bu yöntemlerin gücü, en iyi bireylerin daima yeni nesil bireyleri oluşturmak için bir araya getirilmeleridir. Bu, popülasyonun nesiller boyunca optimize edilmesini sağlar [1][2]. En popüler evrim esinli teknik, Darwinci evrimi simüle eden Genetik Algoritmalardır (GA). Diğer popüler algoritmalar Evrim Stratejisi (Evolution Strategy) (ES), Olasılık Temelli Artan Öğrenme (Probability-Based Incremental Learning) (PBIL), Genetik Programlama (Genetic Programming) (GP) ve Biyocoğrafyaya Dayalı Optimize Edici (Biogeography-Based Optimizer) (BBO) dir [3][4][5].

Fizik tabanlı algoritmalar

Fizik tabanlı yöntemler evrendeki fiziksel kuralları taklit eder. En popüler algoritmalar Simüle Tavlama (Simulated Annealing) (SA), Yerçekimi Yerel Arama (Gravitational Local Search) (GLSA), Big-Bang Big-Crunch (BBBC), Yerçekimi Arama Algoritması (Gravitational Search Algorithm) (GSA), Yüklü Sistem Arama (Charged System Search) (CSS), Merkezi Kuvvet Optimizasyonu (Central Force Optimization) (CFO), Yapay Kimyasal Reaksiyon Optimizasyonu Algoritması (Artificial Chemical Reaction Optimization Algorithm) (ACROA), Kara Delik (Black Hole) (BH) algoritması, Ray Optimizasyonu (Ray Optimization) (RO) algoritması, Küçük Dünya Optimizasyon Algoritması (Small-World Optimization Algorithm) (SWOA), Galaxy Tabanlı Arama Algoritması (Galaxy-based Search Algorithm) (GbSA) ve Kavisli Alan Optimizasyonu (Curved Space Optimization)(CSO) [3][4][5].

Sürü tabanlı algoritmalar

Doğadan ilham alan üçüncü yöntem grubu, hayvan gruplarının sosyal davranışlarını taklit eden sürü tabanlı teknikleri içerir. En popüler algoritma, başlangıçta Kennedy ve Eberhart tarafından geliştirilen Parçacık Sürü Optimizasyonu ‘dur (Particle Swarm Optimization PSO), kuş sürüsünün sosyal davranışından esinlenmiştir [5][6][7]. En iyi çözümü (yani en uygun konumu) bulmak için arama alanında dolaşan bir dizi parçacık (aday çözümler) kullanır. Bu arada, hepsi yollarındaki en iyi yeri (en iyi çözümü) izler. Başka bir deyişle, parçacıklar kendi en iyi çözümlerini ve sürünün şimdiye kadar elde

ettiği en iyi çözümü göz önünde bulundururlar. Bir başka popüler sürü bazlı algoritma, ilk olarak Dorigo ve ark. tarafından önerilen Ant Koloni Optimizasyonu'dur (Ant Colony Optimization ACO) [5]. Bu algoritma, karınca kolonisindeki karıncaların sosyal davranışlarından esinlenmiştir. Aslında, karıncaların yuvadan en yakın yolu ve yiyecek kaynağını bulmadaki sosyal zekası bu algoritmanın ana ilham kaynağıdır. Bir feromon matrisi, aday çözümlerle yineleme boyunca evrilir [5][6].

Meta-sezgisel yöntemler problemleri çözmede başarılı olduğu için ilgi çekmeye başladı. Ayrıca PSO'nun evrim temelli ve fiziksel temel algoritmalar ile çok rekabetçi olduğu kanıtlanmıştır. Genel olarak, sürüye dayalı algoritmalar evrim temelli algoritmalara göre bazı avantajlara sahiptir [7]. Örneğin, sürüye dayalı algoritmalar, sonraki yinelemelere göre arama alanı bilgilerini korurken, evrim tabanlı algoritmalar yeni bir popülasyon oluşur oluşmaz tüm bilgileri atarlar. Genellikle evrimsel yaklaşımlara (seçim, çaprazlama, mutasyon vb.) kıyasla daha az operatör içerirler ve bu nedenle uygulanması daha kolaydır [7, 8].

Burada, literatürde insan davranışlarından esinlenen başka meta sezgisel yöntemlerin de olduğunu belirtmek gerekir. En popüler algoritmalardan bazıları Öğrenme Tabanlı Optimizasyon (TLBO), Uyum Arama (HS), Tabu Arama (TS), Grup Arama Optimize Edici (GSO), Emperyalist Rekabet Algoritması (ICA), Lig Şampiyonası Algoritması (LCA), Havai Fişek Algoritması, Çarpışma Organları Optimizasyonu (CBO), İç Arama Algoritması (ISA), Mayın Patlama Algoritması (MBA), Futbol Ligi Yarışması algoritması (SLC), Arayıcı Optimizasyon Algoritması (SOA), Sosyal Tabanlı Algoritma (SBA), Değişim Pazarı Algoritması (EMA) ve Grup Danışmanlığı Optimizasyonu algoritması (GCO) [8][9][10].

Popülasyon tabanlı meta-sezgisel optimizasyon algoritmaları, doğası ne olursa olsun ortak bir özelliği paylaşır. Arama süreci iki aşamaya ayrılmıştır: araştırma ve uygulanma aşamasıdır. Optimize edici, arama alanını global olarak keşfetmek için operatörler içermelidir: bu aşamada, hareketler (yani tasarım değişkenlerinin bozulması) mümkün olduğunca rastgele seçilmelidir. Sömürü aşaması, araştırma aşamasını takip eder ve arama alanının ümit verici alanlarını ayrıntılı olarak inceleme süreci olarak tanımlanabilir [5, 7, 9, 10]. Bu nedenle sömürü, keşif aşamasında bulunan tasarım alanının gelecek vaat eden bölgelerinde yerel arama kabiliyeti ile ilgilidir. Keşif ve

sömürü arasında uygun bir denge bulmak, optimizasyon sürecinin stokastik doğası nedeniyle herhangi bir meta sezgisel algoritmanın geliştirilmesinde en zorlu görevdir [6, 7]. Önerilen ve önemli olan meta-sezgisel algoritmalar Tablo 1.1'de gösterilmektedir [5].

Tablo 1.1. Önerilen ve Tanınmış Meta-Sezgisel Algoritmalar

No.	Yıl	Algoritma
1	1975	Holland introduced the Genetic Algorithm (GA).
2	1977	Glover proposed Scatter Search (SS).
3	1980	Smith elucidated genetic programming.
4	1983	Kirkpatrick et al. proposed Simulated Annealing (SA).
5	1986	Glover and McMillan offered Tabu Search (TS).
6	1986	Farmer et al. suggested the Artificial immune system (AIS).
7	1988	Koza registered his first patent on genetic programming.
8	1989	Evolver provided the first optimization software using the GA.
9	1989	Moscato presented Memetic Algorithm.
10	1992	Dorigo proposed the Ant Colony Algorithm (ACO).
11	1993	Fonseca and Fleming provided Multi-Objective GA (MOGA).
12	1994	Battiti and Tecchiolli introduced Reactive Search Optimization (RSO) principles for the online self-tuning of heuristics.
13	1995	Kennedy and Eberhart proposed Particle Swarm Optimization (PSO).
14	1997	Storn and Price suggested Differential Evolution (DE).
15	1997	Rubinstein presented the Cross Entropy Method (CEM).
16	1999	Taillard and Voss proposed POPMUSIC.
17	2001	Geem et al. provided Harmony Search (HS).
18	2001	Hanseth and Aanestad offered Bootstrap Algorithm (BA).
19	2004	Nakrani and Tovey presented Bees Optimization (BO).
20	2005	Krishnanand and Ghose introduced Glowworm Swarm Optimization (GSO).
21	2005	Karaboga proposed Artificial Bee Colony Algorithm (ABC).
22	2006	Haddad et al. suggested Honey-bee Mating Optimization (HMO).
23	2007	Hamed Shah-Hosseini offered Intelligent Water Drops (IWD).
24	2007	Atashpaz-Gargari and Lucas introduced Imperialist Competitive Algorithm (ICA).
25	2008	Yang presented Firefly Algorithm (FA).
26	2008	Mucherino and Seref suggested Monkey Search (MS).
27	2009	Husseinzadeh- Kashan provided League Championship Algorithm (LCA).
28	2009	Rashedi et al. introduced Gravitational Search Algorithm (GSA)
29	2009	Yang and Deb offered Cuckoo Search (CS).
30	2010	Yang developed Bat Algorithm (BA).
31	2011	Shah-Hosseini introduced the Galaxy-based Search Algorithm (GbSA) .
32	2011	Tamura and Yasuda designed Spiral Optimization (SO).
33	2011	Rao et al. presented Teaching-Learning-Based Optimization (TLBO) algorithm.
34	2012	Gandomi and Alavi proposed the Krill Herd (KH) Algorithm.
35	2012	Çivicioglu introduced Differential Search Algorithm (DSA).

1.2. Çok Amaçlı Optimizasyon için Popülasyon Bazlı Algoritmalar

Son yıllarda, optimizasyon problemlerini ele almak için kesin ve yaklaşık algoritmalar dahil olmak üzere birçok optimizasyon algoritması önerilmiştir. Optimizasyon algoritmaları sınıfında, algoritmaların tasarımı ve uygulanması genellikle dinamik programlama, geri izleme ve dal ve sınır yöntemleri gibi yöntemlere dayanır [5]. Bununla birlikte, bu algoritmalar birçok problemde iyi bir performansa sahip iken daha büyük ölçekli kombinasyonel ve oldukça doğrusal olmayan optimizasyon problemlerini çözmede problem büyüklüğüne göre arama alanının katlanarak artması ve ayrıntılı arama bu problemlerde pratik olmaması nedeniyle etkili değildir. Ayrıca, açgözlü algoritmalar gibi geleneksel yaklaşık yöntemler genellikle birçok durumda doğrulanması kolay olmayan birkaç varsayım yapılmasını gerektirir [6]. Bu nedenle, bu sınırlamaların üstesinden gelmek için bir dizi daha uyarlanabilir ve esnek algoritmalar gereklidir. Bu motivasyona dayanarak, literatürde genellikle doğal fenomenlerden esinlenen çeşitli algoritmalar önerilmiştir. Bunlar arasında, popülasyon tabanlı çerçeveye sahip bazı meta-sezgisel arama algoritmaları, yüksek boyutlu optimizasyon problemlerini ele almak için tatmin edici özellikler göstermiştir [6, 7].

Popülasyon temelli optimizasyon teknikleri (Population-based optimisation techniques) (PBOT) son 30 yılda çok dikkat çekmiştir. Bu, yerel optima'dan kaçma konusundaki doğal yetenekleri, çok-amaçlı problemlere (MOP'ler) genişletilebilirlikleri ve doğrusal ve doğrusal olmayan eşitsizlik ve eşitlik kısıtlamalarının doğrusal bir şekilde ele alınabilme kabiliyetlerine atfedilebilir [8][9]. Çok çeşitli problemlere uygulanabilir ve problem yapısına çok az kısıtlama getirir. Örneğin, süreklilik ve farklılaşma gerekli değildir. PBOT'lar ayrıca, kapsamlı aramanın pratik olmadığı veya basitçe imkânsız olduğu NP zor problemlerde de iyi performans gösterir. Bir karar vektörleri popülasyonu olduğundan, Pareto cephesi (Pareto front) (PF) keyfi hassasiyete yaklaşabilir (teorik olarak). Yani, PF'nin sürekli olduğu ve seçilen algoritmanın en uygun noktalara yakınlaşabildiği varsayılarak, PF yaklaşımının çözünürlüğünün iradesiyle istenildiği gibi denetlenebilir [3][10].

Yapay Bağışıklık Sistemi (Artificial Immune System) (AIS), Genetik Algoritma (GA), Karınca Kolonisi Optimizasyonu (Ant Colony Optimization) (ACO), Parçacık Sürü Optimizasyonu (Particle Swarm Optimization) (PSO), Stokastik Difüzyon Arama

(Stochastic Diffusion Search) (SDS), Yapay Arı Kolonisi (Artificial Bee Colony) (ABC), Akıllı Su Damlaları (Intelligent Water Drops) (IWD), Nehir Oluşumu Dinamik (River Formation Dynamics) (RFD), Yerçekimi Arama Algoritması (Gravitational Search Algorithm) (GSA) ve Yüklü Sistem Arama (Charged System Search) (CSS) bu algoritmalar sınıfındadır [7][8][10]. Bu algoritmalar ve onların geliştirilmiş şeması, sinir ağı eğitimi, görüntü tanıma, fonksiyon optimizasyonu, görüntü işleme, veri madenciliği, kombinatorial optimizasyon problemleri ve benzeri gibi çok çeşitli problemlerde iyi bir performans göstermiştir.

1.3. Akıllı Sürülerin Genel Özellikleri

Dünyada çok fazla sürü var. Hepsini zeki olarak adlandırmak mümkün değildir veya zeka düzeyleri sürülerden sürülere değişebilir. Kendi kendini örgütleme, basit ajanlar arasındaki yerel etkileşimler yoluyla toplu davranışla sonuçlanan bir sürü sisteminin önemli bir özelliğidir. Bonabeau ve ark., (1999) sürülerdeki öz-örgütlenmeyi dört özellik üzerinden yorumlamışlardır [5][6]:

- Pozitif geri bildirim: uygun yapıların oluşturulmasını teşvik etmek. Bazı karınca türlerinde iz bırakma ve takip etme gibi görevlendirme ve pekiştirme, pozitif geribildirime örnek olarak gösterilebilir.

- Negatif geri bildirim: Pozitif geri bildirimi dengelemek ve kolektif kalıbı dengelemeye yardımcı olmak. Mevcut yem arama maddelerinde oluşabilecek doygunluğu önlemek için negatif bir geri besleme mekanizması gereklidir.

- Dalgalanmalar: yaratıcı yürüyüşler için hayati önem taşıyan sürü bireyleri arasında rastgele yürüyüşler, hatalar, rastgele görev değişimi. Yeni çözümlerin keşfedilmesini mümkün kıldığı için rastgele ortaya çıkan yapılar için genellikle önemlidir.

- Çoklu etkileşimler: Sürüdeki ajanlar, diğer ajanlardan gelen bilgileri kullanırlar, böylece bilgi ağ boyunca yayılır.

Bu özelliklere ek olarak, iş bölümü olarak adlandırılan özel ajanlar tarafından eşzamanlı olarak görevlerin yerine getirilmesi, zekanın ortaya çıkması için bir sürünün yanı sıra kendi kendine örgütlenmenin de önemli bir özelliğidir. Millonas'a göre, sürünün zeki olarak adlandırılması için sürünün aşağıdaki ilkeleri karşılaması gerekir [5][6][7]:

- Sürü basit uzay ve zaman hesaplamaları yapabilmelidir (yakınlık ilkesi).

- Sürü ortamdaki kalite faktörlerine cevap verebilmelidir (kalite prensibi).

- Sürü, aşırı dar kanallar boyunca faaliyetlerini yürütmemelidir (farklı tepki ilkesi).

- Sürü, ortamın her dalgalanmasında davranış biçimini değiştirmemelidir (kararlılık ilkesi).

- Sürü gerektiğinde davranış modunu değiştirebilmelidir (uyarlanabilirlik ilkesi).

1.4. Levy Uçuşları

Doğada, hayvanlar yiyecekleri rastgele veya yarı rastgele bir şekilde arar. Genel olarak, bir hayvanın yemleme yolu etkili bir şekilde rastgele bir yürüyüştür, çünkü bir sonraki hareket mevcut konuma / duruma ve bir sonraki yere geçiş olasılığına dayanır. Hangi yönü seçtiği, örtük olarak matematiksel olarak modellenebilecek bir olasılığa bağlıdır [1][5][8]. Örneğin, çeşitli çalışmalar birçok hayvan ve böceğin uçuş davranışının Levy uçuşlarının tipik özelliklerini gösterdiğini göstermiştir. Reynolds ve Frye (2007) tarafından yapılan yeni bir araştırma, meyve sineklerinin veya Drosophila Melanogaster'ın, 90° ani dönüş ile noktalanan bir dizi düz uçuş yolunu kullanarak manzaralarını keşfettiğini ve Levy-uçuş tarzı aralıklı ölçeksiz arama desenine yol açtığını gösteriyor. Avcı-toplayıcı yem arama kalıpları gibi insan davranışları üzerine yapılan çalışmalar da Levy uçuşlarının tipik özelliğini göstermektedir. Daha sonra, bu tür davranışlar optimizasyona ve optimal aramaya uygulanmıştır ve ön sonuçlar umut verici yeteneğini göstermektedir [3, 4].

Parçacık Sürü Optimizasyonu (PSO)

PSO ilk olarak 1995 ve 1996 yıllarında Kennedy ve Eberhart tarafından geliştirilmiştir. PSO, en iyi çözümü bulmak için kuş sürüsü veya böcek hareketini simüle eden sürü tabanlı bir meta-sezgisel algoritmadır. Algoritmada, kuşlar veya böcekler parçacıklar olarak adlandırılır ve rastgele konumlar ve hızlarla başlatılır [7].

Her parçacık, d boyutlu bir arama alanında (Xi, Vi, Pi) olarak belirtilen bir vektör grubu tarafından tanımlanır; burada, Xi ve Vi, aşağıdaki gibi tanımlanan i parçacığının konumu ve hızıdır [7]:

$$\vec{X}_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \quad for \quad i = 1,2,\ldots,N . \tag{1.1}$$

$$\vec{V}_i = (v_{i1}, v_{i2}, \ldots, v_{id}) \quad for \quad i = 1,2,\ldots,N . \tag{1.2}$$

Pi, ith parçacığı tarafından bulunan en iyi kişisel konumdur:

$$\vec{P}_i = (p_{i1}, p_{i2}, \ldots, p_{id}) \quad for \quad i = 1,2,\ldots,N . \tag{1.3}$$

Ayrıca, tüm popülasyon (Pg) tarafından elde edilen en iyi pozisyon, parçacık hızını güncellemek için hesaplanır:

$$\vec{Pg} = (p_{g1}, p_{g2}, \ldots, p_{gd}). \tag{1.4}$$

$$v_{id}(t+1) = w(t) \times v_{id}(t) + C_1 \times rand \times (p_{id}(t) - x_{id}(t)) + C_2 \times rand \times (p_{gd}(t) - x_{id}(t)), \tag{1.5}$$
$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1),$$

Burada, Vid (t + 1) ve Vid (t) sırasıyla parçacıkların bir sonraki ve mevcut hızıdır. w atalet ağırlığı, Cı ve C2 ivme katsayılarıdır, rand [0, 1] aralığında eşit olarak rasgele sayıdır ve N parçacık sayısıdır. Xid (t + 1) ve Xid (t), partikülün bir sonraki ve mevcut pozisyonunu gösterir [7].

Denk. (6), ikinci ve üçüncü terim sırasıyla bilgi ve sosyal terim olarak adlandırılmaktadır. Ayrıca, | Vid | <Vmax dikkate alınır ve v max, kullanıcılar tarafından çözüm alanının sınırlarına bağlı olarak sabit olarak ayarlanır. Daha büyük bir w değeri, global bir araştırmayı teşvik eder (yeni alanlar arar), daha az atalet ağırlığı yerel bir sömürüyü kolaylaştırır. Genellikle 0,9'dan 0,4'e düşürülür [7].

PSO algoritmasında, Pg'yi seçmek için iki model gbest (veya global topoloji) ve lbest (veya yerel topoloji) modelleri olarak kabul edilir. Global modelde, her parçacığın konumu, arama alanındaki tüm popülasyonun en iyi zindelik parçacığından etkilenirken, yerel modelde, her parçacık, mahallesinden seçilen en iyi zindelik parçacığından etkilenir. Bratton ve Kennedy'ye göre, lbest modeli birçok problemde gbest modelinden daha iyi sonuçlar verebilir; gbest modeline göre daha düşük yakınsama oranına sahip olabilir. PSO aşamaları aşağıdaki kaba kodda gösterilmektedir [7].

Başlangıç P parçacığı oluştur.

P parçacığın kendi en iyisini ata. Pbesti
P parçacığın global en iyisini ata. Gbest
While
    i=1'den P parçacık sayısına kadar tekrar et
    Eğer i. Parçacığın uygunluğu Gbest'ten daha iyi ise Gbest olarak ata.
    Eğer i. Parçacığın uygunluğu daha önceki değerlerinin en iyisi ise Pbesti olarak ata.
    Hız formülünü güncelle
    Konum formülünü güncelle
End
Durma koşulu gerçekleşti ise dur

PSO'nun uygulanması kolay olsa da, yavaş yakınsama oranı, parametre seçim problemi ile karşı karşıya kalabilir ve karmaşık multimodal problemleri çözerken zayıf keşfi nedeniyle kolayca yerel bir optimumda sıkışabilir. Bir parçacık yerel bir optimumun içine düşerse, bazen bu konumdan kendiliğinden kurtulamaz. PSO, optimizasyon probleminin çözümünde yaygın olarak uygulanan en popüler optimizatörlerden biridir. Böylece, algoritmanın performans ve teorik çalışmalarının artırılması cazip hale gelir. Ayrıca, PSO'nun performansı hakkında topolojik yapılar, parametre çalışmaları ve yardımcı operasyonlarla kombinasyon açısından bazı araştırmalar yapılmıştır [7, 9, 10].

2. BÖLÜM

TEZDE KULLANILAN ALGORİTMALAR

2.1. Benchmark Test Problemlerinin Çözümünde Kullanılan Algoritmalar

2.1.1. Yapay Arı Kolonisi (ABC) Algoritması

Yapay Arı Kolonisi algoritması (ABC) ilk olarak Karaboga tarafından 2005 yılında sayısal optimizasyon problemleri için teknik bir rapor olarak yayınlandı [11]. Bal arılarının kolonilerindeki akıllı yemleme davranışını simüle ederek gelişimi motive edildi ve performansı başlangıçta benchmark optimizasyonu fonksiyonlarını kullanarak ölçülmüştür. Daha sonra farklı sektörlerde üzerinde çok fazla çalışma kullanılmıştır. ABC algoritmasının yeni kombinatorial versiyonu Karaboga tarafından 2011 yılında önerildi [11][12]. Yapay Arı Kolonisi algoritması, gezgin satıcısı problemini çözmek için çoklu güncelleme kuralları ve K-opt operasyonu ile Indadul Khan a ve Manas Kumar Maiti tarafından 2018 yılında önerilmiştir [13]. 2019'da hızlı CABC (qCABC) algoritması adı verilen geliştirilmiş CABC algoritması sürümü karaboga ve arkadaşları tarafından önerilmiştir [14].

Yapay arı kolonisi (ABC) algoritması, bal arılarının yemleme davranışını simüle eden bir optimizasyon tekniğidir ve çeşitli pratik sorunlara başarılı bir şekilde uygulanmıştır. ABC, sürü zekâ algoritmaları grubuna aittir. Bal arısı sürülerinin kolektif zekasının ortaya çıkmasına neden olan minimal yem seçimi modeli üç temel bileşenden oluşur: gıda kaynakları, istihdam edilen yem ve işsiz yemleyiciler. Modelde iki modu tanımlanır: zengin bir nektar kaynağına işe alım ve fakir bir kaynağın terk edilmesi [11, 15].

2.1.2. Guguk Arama (Cuckoo Search CS):

Guguk arama Xin-she Yang ve Suash Deb tarafından 2009 yılında geliştirilen bir optimizasyon algoritmasıdır [16]. 2011 yılında Geliştirilmiş Guguk Arama Algoritması Global Optimizasyon Ehsan, Valian ve ark., tarafından sunulmuştur [17]. Bu çalışmada algoritma çalışması sırasında parametreler değiştirilmektedir. Aziz Ouaarab ve arkadaşları 2013'te gezgin satıcı problemini (TSP) çözmek için Guguk Arama algoritmasının geliştirilmiş ve ayrık bir versiyonunu sunmuştur [18].

Guguk Arama aşağıdaki üç idealize edilmiş kuralı kullanır [16][19][20]:

- Her guguk kuşu her seferinde bir yumurta bırakır ve rastgele seçilen bir yuvaya atar.
- Yüksek kaliteli yumurta (solüsyon) içeren en iyi yuvalar gelecek nesillere taşınır.
- Mevcut konakçı yuvalarının sayısı sabittir ve bir konakçı p a $\in$ [0,1] olasılığı olan bir yabancı yumurta bulunabilir. Bu durumda ev sahibi kuş, yumurtayı atabilir veya yeni bir yerde tamamen yeni bir yuva inşa etmek için yuvayı terk edebilir.

2.1.3. Diferansiyel Gelişim-DG (Differential Evolution-DE):

Price ve Storn ilk önce DE algoritmasını 1997 yılında tanıtmıştır. DE, evrimsel bir optimizasyon algoritması olarak sınıflandırılabilir [21][22]. Algoritma başlangıçta sürekli değişkenler üzerinde çalışmak üzere tasarlanmıştır. Son zamanlarda, karma tamsayı-kesikli-sürekli değişkenlerin işlenmesi için DE de genişletilmiştir. Şu anda, DE'nin çeşitli varyantları vardır. 2004'te, Diferansiyel Evrim algoritmasının uyarlanabilir kontrol parametreleriyle yeni bir versiyonunu tanıtmak için bir çalışma önerildi. Mutasyon operasyonu ve Çaprazlama operasyonu için arama parametrelerini uyarlamak için bulanık (fuzzy) mantık kontrolörleri kullanan bulanık adaptif diferansiyel evrim algoritması sunulmuştur [23]. 2009 yılında A. K. Qin ve ark., Kendi kendini uyarlayan bir DE (SaDE) algoritması önermiştir. Hem deneme vektörü oluşturma stratejileri hem de ilişkili kontrol parametresi değerleri, bir sonraki çözümler üretmedeki önceki deneyimlerinden öğrenerek kendi kendine uyarlanır [24][25].

2.1.4. Ateş Böceği Algoritması (Firefly Algorithm):

Ateşböceği algoritması, 2009 yılında Xin-She Yang tarafından önerilen ve yanıp sönen davranışlardan esinlenen bir meta-sezgiseldir [26]. Xin-She Yang, 2010'da Firefly Algoritmasını diğer meta-sezgisel algoritmalarla karşılaştıran bir makale yazdı [26]. Bu makale, önerilen ateş böceği algoritmasının mevcut meta-sezgisel algoritmalardan üstün olduğunu göstermektedir. 2013 yılında Shuhao Yu ve arkadaşları, Kendinden Uyarlamalı Adım Ateşböceği Algoritması önermiştir [27]. Lokal optimumun içine düşmekten kaçınmak ve maksimum yinelemenin etkisini azaltmak için, makalede kendinden uyarlamalı bir adım ateşböceği algoritması önerilmektedir. 2018'de, karınca kolonisi sisteminin parametrelerini ayarlamak için kendi kendini ayarlayan bir ateşböceği algoritması başlıklı bir makale M.K.A. Ariyaratne ve ark tarafından sunulmuştur [28]. Bu makalenin amacı, simetrik TSP problemlerini çözen Karınca kolonisi sistemi (ACS) parametrelerini ayarlamak için ateşböceği algoritması (FA) ile birlikte kendi kendini ayarlayan optimizasyon algoritmaları önermektir.

2.1.5. Genetik Algoritma (Genetic Algorithm):

Genetik algoritma, John Holland (1975) ve öğrencileri tarafından tanıtılan ve araştırılan modele dayanmaktadır [29, 30, 31]. Genetik algoritma, bir arama alanında yeni örnek noktaları oluşturmak için seçim ve rekombinasyon operatörlerini kullanan popülasyon tabanlı herhangi bir modeldir. 2004 yılında, Zhong ve ark. tarafından Sayısal Optimizasyon için Çok Ajanlı Genetik Algoritma başlıklı bir makale önerilmiştir [32]. Bu makalede, çoklu ajan sistemleri ve genetik algoritmalar entegre edilerek, sayısal optimizasyon problemini çözmek için çoklu ajan genetik algoritması (MAGA) adlı yeni bir algoritma oluşturulmuştur. 2013 yılında Seng Poh Lim ve ark. GA, DE ve PSO'nun performansını aynı parametre belirleme ve optimizasyon problemlerini kullanarak karşılaştırmak için bir çalışma yapmıştır [33]. GA'nın DE ve PSO'ya kıyasla daha yüksek performans gösterdiği ve en iyi minimum uygunluğu elde etmenin her iki diğer algoritmadan daha hızlı olduğu kanıtlanmıştır. 2018'de Kin-Ming Lo ve arkadaşları genetik algoritma ve TSP problemi hakkında bir makale yayınlamıştır [34]. Bu makalede Çoklu Gezgin Satıcısı Problemini (Multiple Traveling Salesman Problem MTSP) çözmek için yeni ve etkili Genetik Algoritma (GAL) önerilmiştir. Sonuçlar, GAL'ın daha iyi bir yol kümesi bulduğunu göstermektedir.

2.1.6. Parçacık Sürü Optimizasyonu (Particle Swarm Optimization PSO):

Parçacık sürüsü optimizasyonu hakkında bilgiler 1.3.2. önceki bölümde detaylı olarak verilmiştir.

2.1.7. Karga Algoritması (Crow Search Algorithm):

Karga Arama Algoritması Alireza Askarzadeh tarafında 2016 yılında geliştirilmiştir. Kargaların akıllı davranışlarına dayanan, karga arama algoritması (CSA) adlı yeni bir meta-sezgisel algoritma geliştirilmiştir [35, 36]. CSA, kargaların fazla yiyeceklerini sakladığı ve yiyecek gerektiğinde geri getirdiği fikrine dayanan sürü temelli bir tekniktir [37, 38]. 2018'de Deepak Gupta ve ark., Karga Arama Algoritma ile ilgili yeni bir çalışma sunmuştur [39]. Bu çalışmada, Parkinson hastalığının erken teşhisini iyileştirmek için, karga arama algoritmasının yeni geliştirilmiş ve optimize edilmiş bir versiyonunu sunmuştur (Optimized Version of Crow Search Algorithm OCSA). Önerilen OCSA, Parkinson hastalığını%100 doğrulukla tahmin etmek ve bireyin erken evrede uygun tedaviye sahip olmasına yardımcı olmak için kullanılabilir. 2018'de Farid Mohammadi ve ark., Karga Arama Algoritma ile ilgili yeni bir çalışma sundukları makalede ekonomik yük dağıtımı (Economic Load Dispatch ELD) probleminin çözümü için değiştirilmiş Crow arama algoritması (MCSA) önerilmiştir [40]. 2019 yılında Ko-Wei Huang ve ark., tarafından permütasyon Akış Tipi Çizelgeleme ve Atölye Tipi Çizelgeleme problemlerinin (Permutation Flow Shop Scheduling Problems PFSP) yapısını en aza indirmek için hibrit Karga Arama Algoritması önerilmiştir [41].

2.1.8. Yarasa Algoritması (Bat Algorithm):

Yarasa algoritması Xin-She Yang tarafından 2010 yılında global optimizasyon için geliştirilen bir meta-sezgisel algoritmadır. Değişen nabız emisyon ve ses yüksekliği oranlarına sahip olan yarasaların ekolojik yerleştirme davranışından esinlenmiştir. Yarasalar büyüleyici hayvanlardır [42][43][44][45]. Kanatlı tek memelilerdir ve ayrıca gelişmiş ekolokasyon yeteneğine sahiptirler. 2012 yılında Nakamura ve ark., yarasaların davranışına dayanan yeni bir özellik seçimi tekniği önermiştir [46]. Bu çalışma, bir doğrulama setinde yüksek doğrulukta özellikler çıkartmak için yarasaların keşif gücünü kullanmıştır. 2017 yılında Asma Chakri ve ark., tarafından Yarasa Algoritmasını iyileştirmek için yeni bir yaklaşım önerilmiştir [47]. Keşif ve sömürü yeteneklerini

geliştirmek için standart yarasa algoritmasına yönlü yankı eklenmiştir. Yeni önerilen yaklaşım, yani yönlü Bat Algoritması (dBA), daha sonra birkaç kıyaslama seti kullanılarak test edilmiştir. İstatistiksel test sonuçları iyi sonuçlar vermiştir. 2019 yılında Deepak Gupta ve ark. makalede, farklı tipte beyaz kan hücrelerinin sınıflandırılması için optimize edilmiş ikili yarasa algoritması önermiştir [48].

2.2. Gezgin Satıcı Problemine Çözümler

2.2.1. Kesin Çözücüler

Dallanma ve Sınırlandırma Yaklaşımı (Branch and Bound):

Dallanma ve Sınırlandırma Yaklaşımı yöntemi, sorunların tamsayı kısıtlamalarının gevşetildiği hesaplamaları kullanarak tüm uygulanabilir çözümleri örtülü olarak numaralandırır [49][50]. Diğer bir deyişle, Dallanma ve Sınırlandırma stratejisi, çözülmesi gereken bir problemi bir dizi alt probleme böler. Her biri birden fazla olası çözüme sahip olabilen ve bir alt problem için seçilen çözümün sonraki alt problemlerin olası çözümlerini etkileyebileceği bir dizi alt problemi çözmek için bir sistemdir. Tüm kısmi ağaçların tam hesaplanmasını önlemek için, önce pratik bir çözüm bulmaya çalışılır ve değeri optimum için bir üst sınır olarak not edilir. Mesafe üst sınırın mesafesini aştıkça hesaplamalar yapılır. Yeni daha ucuz bir çözüm bulunursa, değeri yeni üst sınır olarak kullanılır. Bu yöntem 40 ila 60 düğüm (şehirler) için uygundur [49][50][51].

Kesen Düzlem (The Cutting Plane):

Kesen düzlem algoritmaları ilk olarak Gomory tarafından 1950 yılında önerilmiştir. Gomory kesitleri büyük ölçekli problemlerde zayıf sonuçlar verdiğinden dolayı uzun yıllar ilgi görmemiştir [52, 53]. Yakın geçmişte ise Gomory kesitlerinin aslında kullanılabilir olduğunu göstermişlerdir. Son zamanlarda kesen düzlem algoritmalarında yeni kavramlara yer verilmiştir. Bunlardan ikisi lift-and-project kesitleri ve Fenchel kesitleridir. Ayrıca Marchand tarafından ele alınan kesen düzlem çalışması ise birleşimsel eniyileme problemlerini çözen bir algoritmadır [53, 54].

Dallanma ve Kesme Yaklaşımı (Branch and Cut):

Bu fikrin ilkel bir versiyonu Hong ve Miliotis tarafından TSP'ye uygulanmıştı.Grotschel, Junger ve Reinelt bunu Doğrusal Sipariş Problemine uyguladı. Dal ve kes terimi Padberg ve Rinaldi tarafından sunulmuştur [55][56]. Çok büyük TSP örneklerini çözmek için kullanmışlardır (2000 şehre kadar). Dal ve kesme yöntemi, normal simpleks algoritmasını kullanarak tamsayı kısıtı olmadan doğrusal programı çözer. Optimal bir çözüm elde edildiğinde ve bu çözüm, tamsayı olması gereken bir değişken için tamsayı olmayan bir değere sahipse, bir kesen düzlem algoritması kullanılır [56, 57].

Dinamik programlama:

Kısmi sonuçları saklayarak ve gerektiğinde tekrar kullanarak tekrarlanmaları hesaplamak için kullanılan bir tekniktir [58].

Kaba kuvvet yöntemi:

TSP'yi çözmeyi düşünüldüğünde, akla gelebilecek ilk yöntem kaba kuvvet yöntemidir. Kaba kuvvet yöntemi, mümkün olan tüm turları oluşturmak ve mesafelerini hesaplamaktır. Bu nedenle en kısa tur en uygun turdur [59].

2.2.2. Tam Olmayan Çözümler

Bu çözücüler potansiyel olarak optimal olmayan ancak daha hızlı çözümler sunar. Kesin olmayan çözücüler aşağıdaki bölümlere ayrılabilir [58]:

- **Yaklaşık Algoritmalar**

TSP'yi çözmek için iki geleneksel yöntem, 2-opt algoritması yaklaşımı ve birleşik MST ve Minimum Eşleştirme Sorunu (MMP) tabanlı algoritma sağlayan saf bir MST tabanlı algoritmadır.

- **Sezgisel Algoritmalar**

Bu algoritmalar sadece uygulanabilir bir çözüm vaat eder. En Yakın Komşu, Clarke-Wright ve Çoklu Fragman1 gibi basit tur yapım yöntemlerinden Tabu Arama ve Lin-Kernighan gibi daha karmaşık tur geliştirme algoritmalarına kadar çeşitlilik gösterir.

Son zamanlarda, yaklaşık çözümler ve büyük çalışma sürelerini birleştiren popüler algoritmalar vardır. Bunlar Benzetimli Tavlama, Genetik Algoritmalar, Karınca Kolonisi Algoritmaları ve Yapay Sinir Ağları gibi makine öğrenme algoritmaları gibi yöntemlerdir [58].

2.2.3. TSP Çözmek İçin Tanınmış Algoritmalar

TSP problemini çözmek için kullanılan popüler algoritmalardan kısaca bahsedilecektir.

Christofides Algoritması

Christofides algoritmasının amacı (Nicos Christofides'in adını almıştır), üçgen kenar eşitsizliğini kullanarak gezgin satıcı problemine bir çözüm bulmaktır [60][61].

Clarke-Wright Algoritması

Clarke-Wright algoritması bir araç yönlendirme (Vehicle Routing Algorithm) algoritmasından türetilmiştir [62][63].

En Yakın Komşu (Nearest Neighbour)

Bu yöntem TSP için doğal bir stratejidir, çünkü gezgin satıcının bir seyahat rotasını seçme şeklini taklit eder. Bir başlangıç noktası seçer ve ardından her zaman tura eklenecek en yakın şehri seçer, daha sonra o şehre gider ve tüm şehirler turda olana kadar seçilmemiş yeni bir şehir seçerek tekrar eder. Turu tamamlamak için son seçilen şehir ile başlangıç şehri arasına bir kenar eklenir [64] [65] [66].

Ekleme Sezgisel (Insertion Heuristics)

Yerleştirme sezgiselinin temelleri, tüm şehirlerin alt turuyla başlamak ve daha sonra geri kalanını biraz sezgisel olarak yerleştirmektir. Ayrıca alt tur olarak tek bir kenarla da başlanabilir.

Açgözlü Sezgisel (The Greedy Heuristic)

Açgözlü sezgisel tarama, sonuçta bir tur sonuçlanıncaya kadar, bir T anına kadar en düşük maliyetli bir kenarı ekleyerek tekrarlayan bir tur oluşturur. Açgözlü Yaklaşım'ı kullanarak TSP'yi çözmek için şehirden (düğüm) çıkan tüm yolara bakılır ve en kısa yol

seçilir. Bu en kısa yol Hamilton çevrimi oluşturuyorsa, optimal bir çözüm bulundu demektir [59].

Tepe Tırmanışı (Hill Climbing)

Bilgisayar biliminde, tepeye tırmanma yerel arama ailesine ait bir matematiksel optimizasyon tekniğidir. Soruna rastgele bir çözümle başlayan, daha sonra çözümün tek bir öğesini değiştirerek daha iyi bir çözüm bulmaya çalışan yinelemeli bir algoritmadır. Değişiklik daha iyi bir çözüm üretiyorsa, yeni çözümde değişiklik yapılır. Başka iyileştirme bulunamayana kadar bu tekrarlanır [67].

Benzetimli Tavlama (Simulated Annealing)

Benzetimli tavlama, birçok kombinatoryal optimizasyon probleminin çözümünde başarıyla kullanılan iyi bilinen bir meta-sezgisel arama yöntemidir [68]. İyi çözümler üretmesine rağmen, basit bir tepe tırmanma prosedürüne kıyasla çok yavaştır. Benzetimli tavlama terimi, atomların istikrarlı bir duruma gelene kadar yavaş soğutma kullanarak sistemin enerjisini en aza indirmeye çalıştığımız katıların tavlanmasından esinlenmiştir. Yavaş soğutma tekniği, metal atomlarının kendilerini hizalamasına ve yüksek yoğunluklu ve düşük enerjiye sahip düzenli bir kristal yapı oluşturmasına izin verir. Başlangıç sıcaklığı ve sıcaklığın düşürüldüğü hıza tavlama programı denir [68][69].

Tabu Arama (Tabu Search)

Kombinatorial optimizasyon problemleri çözmek için yaygın olarak kullanılmaktadır. Adı yasak veya kısıtlı anlamına gelen "tabu" kelimesinden türetilmiştir. Yaklaşımın temel özelliği, çözüm ararken bellek kullanımıdır. En basit düzeyde, tabu arama etkileşimli iyileştirmeye çok benzer, ancak bazı çözümlerin lokal çözümlerin ziyaret edilebileceği ek kısıtlamalar kavramı ile çalışır. Kavramsal düzeyde, kısıtlamalar bir çözüm dizisi oluşturulur (yasak çözüm listesi) ve ancak belirtilen bir süre sonra bu çözümler tekrar alınabilir [70][71].

Karınca Kolonisi Optimizasyonu (Ant Colony Optimization ACO)

Karınca algoritmaları ilk olarak Dorigove meslektaşları tarafından 1992 yılında geliştirilmiştir. Karınca Kolonisi Optimizasyonu, gerçek karıncaların davranışlarından esinlenen meta-sezgisel bir tekniktir. İlkeleri Dorigo ve arkadaşlar tarafından sunulmuştur. Bir yol üzerinde mevcut olan feromon miktarına bağlı olarak, yeni karıncalar, yüksek bir olasılıkla, aynı yolu izlemeye teşvik edilir ve bu yol üzerine daha fazla feromon yerleştirilir. Gıda kaynaklarına giden daha kısa yolların feromon miktarı daha yüksektir. Böylece, zamanla, karıncaların çoğu en kısa yolu kullanmaya yönlendirilir [72][73].

Genetik Algoritmalar (Genetic Algorithms)

Genetik algoritma, John Holland (1975) ve öğrencileri tarafından tanıtılan ve araştırılan modele dayanmaktadır [29]. Bir GA'nın teması, biyolojik evrim, doğal seleksiyon ve canlı organizmalarda en uygun olanın hayatta kalma süreçlerini simüle etmektir. Genetik özellikleri açısından daha iyi veya daha zinde olan bireyler, yavrularını yetiştirmek ve üretmek için hayatta kalırlar. Bu şekilde, daha iyi turlara sahip genler kalacak, kötü turlara sahip genler ortadan kalkma eğilimi gösterecektir [74].

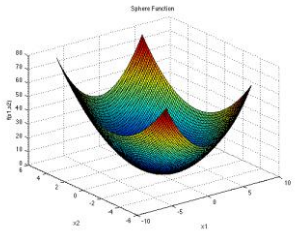
3. BÖLÜM

TEST PROBLEMLERİ

Literatürde, meta-sezgisel algoritmaların performansını değerlendirmek için Benchmark Gezgin Satıcı vb. test fonksiyonları kullanılmıştır. Bir dizi sayısal optimizasyon probleminde iyi performans gösteren algoritmalar, gerçek dünya problemlerini çözmek için etkili yöntemler olarak kabul edilir [75, 76, 83].

3.1. Benchmark problemleri

Sphere: Global fonksiyon dışında Sphere fonksiyonunun d yerel minimumu vardır. Sürekli, dışbükey ve tek modludur. Çizim iki boyutlu şeklini gösterir.



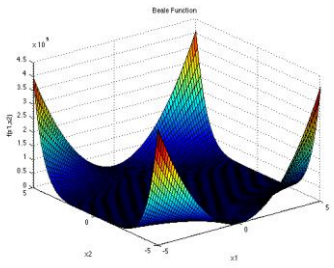
$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2$$

Şekil 3.1. Sphere fonksiyonu grafiği ve denklemi

Girdi Alanı: Fonksiyon genellikle tüm i = 1, …, d için hiperküp xi ∈ [-5.12, 5.12] üzerinde değerlendirilir.

Global Minimum: $f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0, \ldots, 0)$

Beale: Boyutlar: 2. Beale işlevi, giriş alanının köşelerinde keskin tepeler bulunan multimodal işlevidir.

	$f(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$

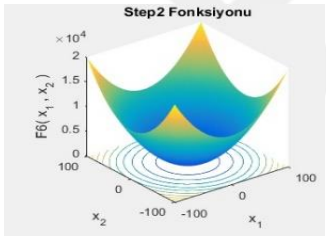
Şekil 3.2 Beale fonksiyonu grafiği ve denklemi

Girdi Alanı: Fonksiyon genellikle tüm i = 1, 2 için xi ∈ [-4.5, 4.5] karesinde değerlendirilir.

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (3, 0.5)$$

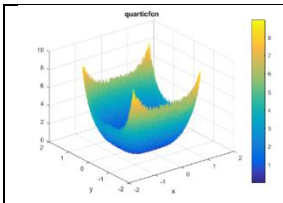
Global Minimum:

Step: Step2 fonksiyonunun arama alanı $-100 \leq x_i \leq 100$ olarak tanımlanmıştır.

	F6(x)=$\sum_{i=1}^{n}([x_i + 0.5])^2$

Şekil 3.3. Step fonksiyonu grafiği ve denklemi

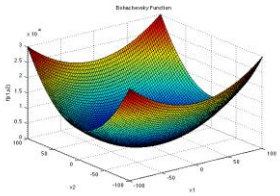
Quartic:

	$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{n} ix_i^4 + \text{random}[0, 1)$

Şekil 3.4. Quartic fonksiyonu grafiği ve denklemi

Global Minimum: Fonksiyonun bir global minimum f (x ∗) = 0 + x ∗ = (0, …, 0) 'da rastgele gürültüsü vardır.

Bohachevsky: Boyutlar: 2. Bohachevsky işlevlerinin hepsi aynı benzer kase şekline sahiptir. Yukarıda gösterilen, ilk işlevdir.

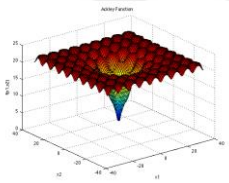


$$f_1(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

Şekil 3.5. Bohachevsky fonksiyonu grafiği ve denklemi

Global Minimum: $f_j(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0,0)$, for all $j = 1,2,3$

Ackley: Ackley işlevi, optimizasyon algoritmalarını test etmek için yaygın olarak kullanılır. Yukarıdaki çizimde gösterildiği gibi iki boyutlu formunda, neredeyse düz bir dış bölge ve merkezde büyük bir delik ile karakterizedir. İşlev, optimizasyon algoritmalarının, özellikle tepe tırmanma algoritmalarının, birçok yerel minimumdan birinde sıkışıp kalma riski taşır.

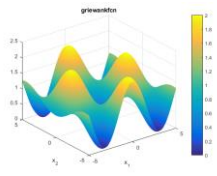


$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Şekil 3.6. Ackley fonksiyonu grafiği ve denklemi

Global Minimum: $f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0,\ldots,0)$

Griewank:

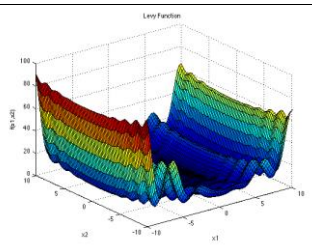


$$f(\mathbf{x}) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Şekil 3.7. Griewank fonksiyonu grafiği ve denklemi

Global Minimum: f(x∗) = 0 at x∗ = (0, …,0)

Levy:



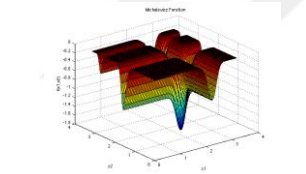
$$f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 \left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2 \left[1 + \sin^2(2\pi w_d)\right],$$

$$\text{Where } w_i = 1 + \frac{x_i - 1}{4}, \text{ for all } i = 1, \ldots, d$$

Şekil 3.8. Levy fonksiyonu grafiği ve denklemi

Global Minimum: $f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (1, \ldots, 1)$

Michalewiz (Michalewicz): Michalewicz fonksiyonu yerel minima ve multimodal özelliğine sahiptir. M parametresi vadilerin ve sırtların dikliğini tanımlar; daha büyük bir m daha zor bir aramaya yol açar. M'nin önerilen değeri m=10'dur. Fonksiyonun iki boyutlu formu yukarıdaki grafikte gösterilmiştir.

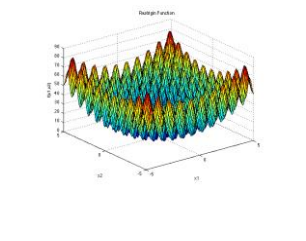


$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}\left(\frac{i x_i^2}{\pi}\right)$$

Şekil 3.9. Michalewiz fonksiyonu grafiği ve denklemi

Global Minimum: at $d = 2$: $f(\mathbf{x}^*) = -1.8013$, at $\mathbf{x}^* = (2.20, 1.57)$
at $d = 5$: $f(\mathbf{x}^*) = -4.687658$
at $d = 10$: $f(\mathbf{x}^*) = -9.66015$

Rastrigin: Rastrigin fonksiyonunun birkaç yerel minimi vardır. Oldukça multimodal, ancak minima yerleri düzenli olarak dağıtılır. Yukarıdaki grafikte iki boyutlu olarak gösterilmektedir.

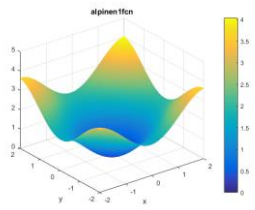


$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Şekil 3.10. Rastrigin fonksiyonu grafiği ve denklemi

Global $f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, \ldots, 0)$ Minimum:

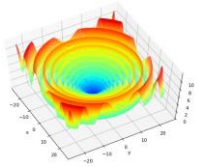
Alpine: Tanım ve Özellikler: İşlev dışbükey değil. İşlev n-boyutlu uzayda tanımlanır. İşlev ayrılamaz. İşlev farklıdır.

	$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \lvert x_i sin(x_i) + 0.1x_i \rvert$

Şekil 3.11. Alpine fonksiyonu grafiği ve denklemi

Global minimum: x*= (0, …, 0) konumunda global bir min. f (x ∗) = 0 değeri vardır.

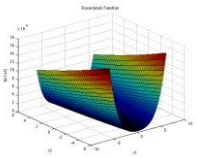
Schaffer:

	$f(\mathbf{x}) = \sum_{i=1}^{N-1} (x_i^2 + x_{i+1}^2)^{0.25} \cdot \left[\sin^2(50 \cdot (x_i^2 + x_{i+1}^2)^{0.10}) + 1.0\right]$

Şekil 3.12. Schaffer fonksiyonu grafiği ve denklemi

Global Minimum: xi = 0, ∀i ∈ {1… N}, f (x) = 0

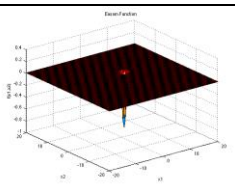
Rosenbrock: Valley (Vadi) veya Banana (Muz) işlevi olarak da adlandırılan Rosenbrock işlevi, gradyan tabanlı optimizasyon algoritmaları için popüler bir test sorunudur. Yukarıdaki grafikte iki boyutlu olarak gösterilmektedir. İşlev unimodal ve global minimum dar, parabolik bir vadide yatıyor. Ancak, bu vadiyi bulmak kolay olsa da minimuma yakınsamak zordur.

	$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$

Şekil 3.13. Rosenbrock fonksiyonu grafiği ve denklemi

Global Minimum: $f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (1, \ldots, 1)$

Easom: Easom işlevinin birkaç yerel minimi vardır. Tek modlu bir işlevdir ve global minimumun arama alanına göre küçük bir alanı vardır.

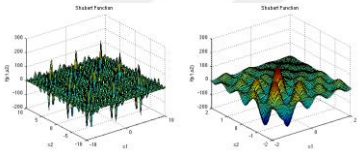
	$f(\mathbf{x}) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$

Şekil 3.14. Easom fonksiyonu grafiği ve denklemi

Boyutlar: 2

Global Minimum: $f(\mathbf{x}^*) = -1$, at $\mathbf{x}^* = (\pi, \pi)$

Shubert:

	$f(\mathbf{x}) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1+i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2+i)\right)$

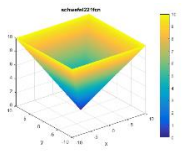
Şekil 3.15. Shubert fonksiyonu grafiği ve denklemi

Boyutlar: 2

Shubert fonksiyonunun birkaç yerel minimi ve birçok global minimi vardır. İkinci grafik, daha kolay görüntüleme için işlevi daha küçük bir giriş alanındaki gösterir.

Global Minimum: $f(\mathbf{x}^*) = -186.7309$

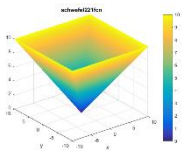
Schwefel 2.21: Fonksiyon süreklidir. İşlev dışbükeydir. Fonksiyon n boyutlu uzayda tanımlanır. Fonksiyon unimodal. İşlev ayırt edilemez. Fonksiyon ayrılabilir.

	$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \max_{i=1,\ldots,n} \lvert x_i \rvert$

Şekil 3.16. Schwefel 2.21 fonksiyonu grafiği ve denklemi

Global Minimum: İşlevin x*= (0, …, 0) konumunda global bir minimum f (x ∗) = 0 değeri vardır.

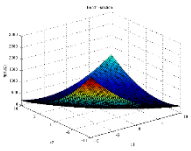
Schwefel 2.22: Fonksiyon süreklidir. İşlev dışbükeydir. İşlev n-boyutlu uzayda tanımlanır. Fonksiyon unimodal. İşlev ayırt edilemez. Fonksiyon ayrılabilir.

	$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$

Şekil 3.17. Schwefel 2.22 fonksiyonu grafiği ve denklemi

Global Minimum: İşlevin x*= (0, …, 0) konumunda global bir minimum f (x ∗) = 0 değeri vardır.

Booth:

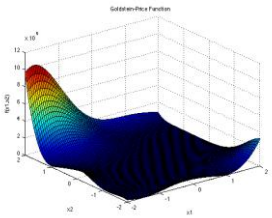
	$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$

Şekil 3.18. Booth fonksiyonu grafiği ve denklemi

Boyutlar: 2

Global Minimum: $f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (1, 3)$

Goldstein price: Goldstein-Price işlevinin birkaç yerel minimi vardır.

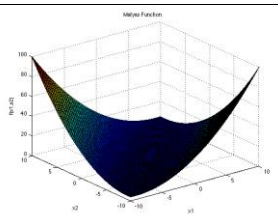
	$f(\mathbf{x}) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$

Şekil 3.19. Goldstein price fonksiyonu grafiği ve denklemi

Boyutlar: 2

Global Minimum: $f(\mathbf{x}^*) = 3, \text{ at } \mathbf{x}^* = (0, -1)$

Matyas: Matyas işlevinin global işlev dışında yerel bir minimi yoktur.

	$f(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$

Şekil 3.20. Matyas fonksiyonu grafiği ve denklemi

Boyutlar: 2

Global Minimum: $f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, 0)$

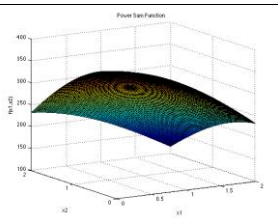
Powell:

$$f(\mathbf{x}) = \sum_{i=1}^{d/4} \left[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4\right] \tag{3.21}$$

Boyutlar: d

Global Minimum: $f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, \ldots, 0)$

Power sum: Güç Toplamı işlevi. Yukarıda iki boyutlu olarak gösterilmiştir. D-4 için b-vektörünün önerilen değeri: b = (8, 18, 44, 114).

	$f(\mathbf{x}) = \sum_{i=1}^{d} \left[\left(\sum_{j=1}^{d} x_j^i \right) - b_i \right]^2$

Şekil 3.21. Power sum fonksiyonu grafiği ve denklemi

Boyutlar: d

Global Minimum: x* = (1,2,3,4), f (x *) = 0'da b = (8,18,44,114)

3.2. Gezgin Satıcı Problemi (Travelling Salesman Problem TSP)

3.2.1. Tanım

Gezgin Satıcı Problemi, her şehri tam olarak bir kez ziyaret eden ve daha sonra başlangıç noktasına geri dönen en kısa rotayı belirlemek için bir şehir koleksiyonu verilir. Daha matematiksel olarak TSP aşağıdaki gibi tanımlanabilir [77]:

N> = 3 tamsayısı ve n x n matrisi C= (cij) verildiğinde, burada her cij negatif olmayan bir tamsayıdır.1 ile n arasındaki tamsayıların döngüsel permütasyonu π toplamı en aza indirir.

$$\sum_{i=1}^{n} c_{i\pi(i)} \tag{3.23}$$

Gezgin Satıcı Sorunu nispeten eski bir sorundur: 1759'da Euler tarafından belgelenmiştir. 'Gezgin Satıcı' terimi ilk olarak 1932'de deneyimli bir gezgin satıcı tarafından yazılmış bir Alman kitabında kullanılmıştır. TSP, RAND Corporation tarafından 1948'de tanıtıldı. Kurumun itibarı, TSP'nin bilinen ve popüler bir sorun haline gelmesine yardımcı oldu. TSP aynı zamanda yeni doğrusal programlama konusu ve kombinatoryal problemleri çözme girişimleri nedeniyle popüler oldu [77]. Yıllar boyunca Gezgin Satıcısı Problemi birçok araştırmacının düşüncelerini işgal etti. Bunun birkaç nedeni vardır.

İlk olarak, TSP'nin tanımlanması çok kolaydır, ancak çözülmesi çok zordur. Çözülebileceği hiçbir polinom zaman algoritması bilinmemektedir. Herhangi bir polinom zaman algoritmasının olmaması, TSP'nin klasik bir örnek olduğu NP-tam problemler sınıfının bir özelliğidir. NP (Non-deterministic-polynomial Time), polinomsal zamanda doğrulanabilen karar problemlerinin karmaşıklık sınıfı demektir. Burada problemin bir çözümü olup olmadığı ile ilgilenilmiyor. NP-tam problem demek ki her adımdaki çözümleme zamanı kendinden önceki adımdaki çözümleme zamanlarından daha fazla olduğu için bu problem tiplerinin çokterimli zamanda (polynomial time) çözülmesi mümkün değildir.

İkinci olarak, TSP çeşitli yönlendirme ve çizelgeleme problemlerine geniş çapta uygulanabilir.

Üçüncüsü, TSP hakkında zaten çok fazla bilgi bilindiği için bir tür “test” sorunu haline gelmiştir; yeni kombinatoryal optimizasyon yöntemleri TSP'ye sıklıkla uygulanır, böylece yararlılıkları hakkında bir fikir oluşturulabilir. Son olarak, Yapay Zeka'da sezgisel tekniklerle gerçekten tedavi edilen çok sayıda sorun, bir sonraki paragrafta açıklanacağı gibi, n elementlerin en iyi permütasyonunun araştırılması ile ilgili olup, TSP için çok sayıda sezgisel algoritma geliştirilmiştir [77].

3.2.2. Gezgin Satıcı Probleminin Çeşitleri

TSP'nin geleneksel modeli, birçok araştırmacı tarafından sorunu birçok gerçek yaşam problemine etkili bir şekilde uygulamak için değiştirilmiştir. TSP'nin iki temel varyantı vardır; simetrik TSP (sTSP), asimetrik TSP (aTSP) ve çoklu TSP (mTSP) TSP'nin bu temel versyonları aşağıdaki gibi tanımlanabilir [77]:

- sTSP: dij'in, i ve j şehirleri arasındaki Öklid mesafesi olduğunu varsayarsak. Dij = dji ise, tur aynı maliyetle her iki yönde de değerlendirilebilir. STSP, her şehri bir kez ziyaret eden minimum uzunlukta kapalı bir tur bulma sorunudur.

- aTSP: En az bir kenar (i, j) için dij ≠ dji ise, TSP bir aTSP olur.

- mTSP: Her şehir için ziyaretleri tam olarak bire sınırlamak için temel model varsayımı da atlanabilir; Çok ziyaretli TSP'ye mTSP denir.

4. BÖLÜM

TSP İÇİN AYRIK KARGA ARAMA ALGORİTMASI

4.1. Karga Arama Algoritması (Crow Search Algorithm-CSA):

Karga Arama Algoritması Alireza Askarzadeh tarafında 2016 yılında geliştirilmiştir[35]. Kargalar akıllılıkları ile bilinir ve sofistike bir şekilde iletişim kurabilir, yüzleri hatırlayabilir ve araçlar kullanabilir. Bu yüzden gezegende bulunan en akıllı hayvanlardan biri olarak tanınırlar. Algoritmanın arkasındaki temel kavram, kargaların fazlalık yiyeceklerini gizlice saklandığı yere gizleyerek saklamasıdır. Birkaç ay sonra bile yiyecek saklama yerlerini hatırlayabilirler [35][36]. Kargalar, diğer kuşların yiyecek sakladığı yerleri gözlemler ve yiyeceklerini çalar. Kargaların yiyecek saklama yerlerini bulmak zor bir iştir, çünkü birisinin takip ettiğini bilmeleri durumunda başka bir yere giderek yanlış yere götürürler. CSA aşağıdaki varsayımlarda formüle edilmiştir [35, 36]:

- Kargalar genellikle sürülerde bulunur.
- Yiyecek saklama yerlerini ezberlerler.
- Yiyecekleri çalmak için birbirlerini takip ediyorlar.
- Gıda depolarını olasılık kullanarak soyulmaktan korurlar.

Kargaların yiyeceklerini n boyutlu bir arama ortamında sakladıkları düşünülmektedir. N (c) karga sayısı dikkate alındığında, sürü büyüklüğü N (c) 'dir. Karga k'nın i (th) yinelemesinde mevcut konumu vektör x (k, i) olarak tanımlanabilir [35]:

$$x^{k,i} = [x_1^{k,i}, x_2^{k,i}, x_3^{k,i} \ldots, x_n^{k,i}] \tag{2.11}$$

Burada k = 1,2, ..., Nc, i = 1,2 ..., i (m) ve i (m) maksimum yineleme sayısıdır. Her karga, yemek saklandığı yer hakkındaki bilgilerin saklandığı ilişkili bir belleğe sahiptir.

Karga k'nın i (th) yinelemesinde gıda gizleme yeri M (k, i) ile temsil edilir. Bu, şimdiye kadar karga k tarafından bulunan en iyi yerdir. Aslında, her karganın saklandığı yerin en iyi deneyimi hakkında bilgi saklanır. Kargalar, daha iyi yiyecek saklama yerleri bulmak için n boyutlu arama alanındaki farklı yerleri ziyaret eder ve gözlemler.

Kargaların konum güncelleme mekanizması:

I(th) yinelemesinde, karga k gıda saklama yerini, (Mk), i ziyaret etmek gerekiyor varsayalım. Aynı zamanda (tekrarlama) karga q, karga k'ın gıda saklanma yerine erişmek için karga k'yi kovalamaya karar verir. Bu durumda, iki ihtimal düşünülebilir:

İhtimal 1: Karga k, onu karga q'nun izlediğinin farkında olmayabilir. Sonuç olarak, karga q, karga k'ın gıda saklama yerini görecek ve karga q'nun yeni pozisyonu aşağıdaki denklemle elde edilecektir [35]:

$$x^{q,i+1} = x^{q,i} + r_q \times FL^{q,i} \times (M^{k,i} - x^{q,i}) \tag{2.12}$$

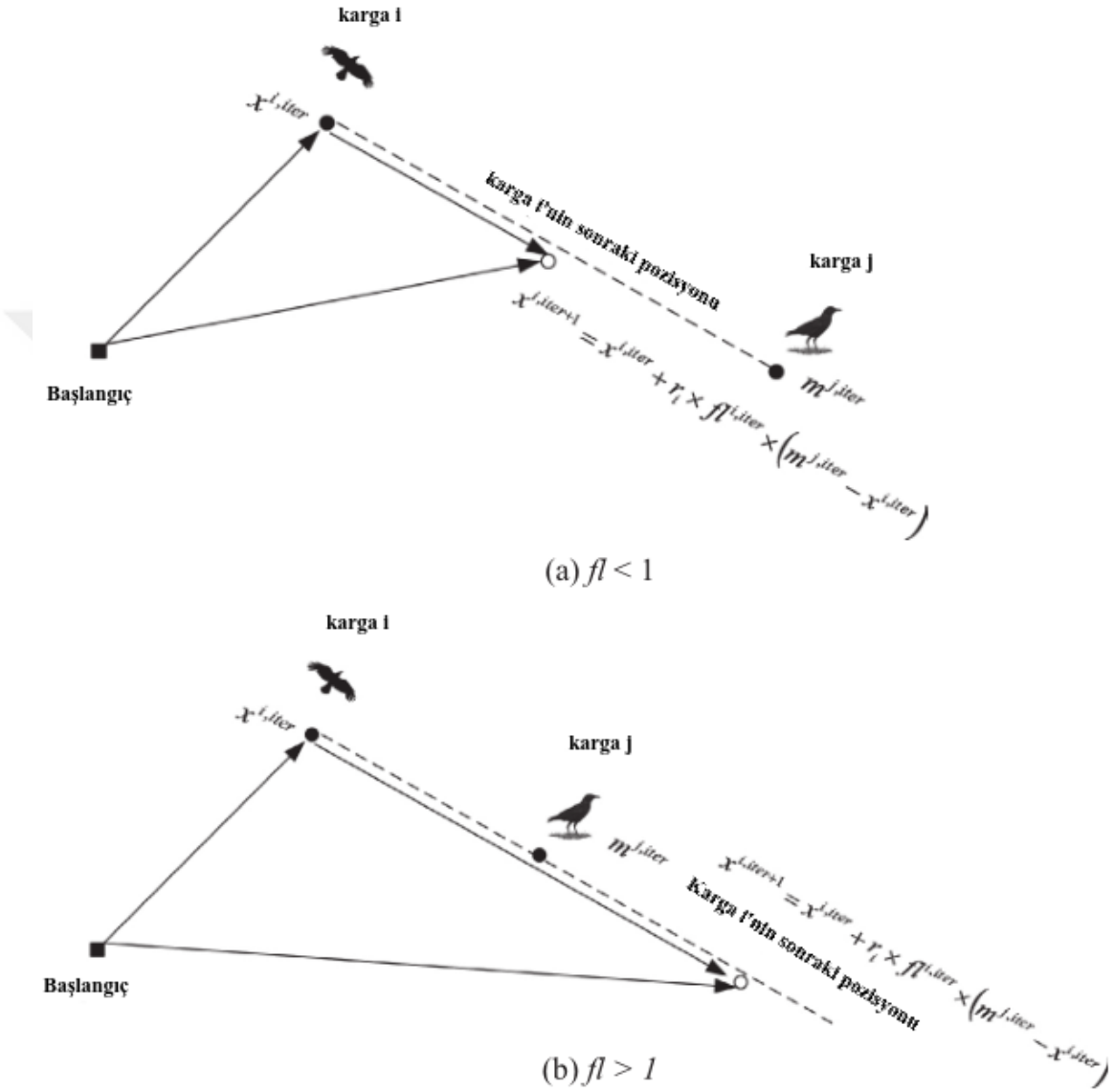
Burada r(q), 0 ile 1 arasında homojen olarak dağıtılmış rasgele bir sayıdır ve FL (q, i), i(th) yinelemesinde karga q'nun uçuş adımını temsil eder. FL (q, i) algoritmanın arama yeteneğini önemli ölçüde etkiler, yani FL'nin daha düşük değerleri yerel aramayı artırır, daha yüksek FL değerleri global aramayı artırır.

İhtimal 2: Karga k, karga q'nun takip ettiğinin farkındadır. Sonuç olarak, karga k ayrılmış yiyecek kaynağını soyulmaya karşı savunmaya çalışacak ve karga q'yu yanıltmak için arama alanında rastgele bir yere gidecektir.

Bu ihtimallere dayanarak, kargaların konum güncelleme mekanizması aşağıdaki gibi formüle edilebilir [35]:

$$x^{q,i+1} = \begin{cases} x^{q,i} + r_q \times FL^{q,i} \\ \times (M^{k,i} - x^{q,i}) & r_k \geq ap^{k,i} \\ \text{a random location} & \text{otherwise} \end{cases} \tag{2.13}$$

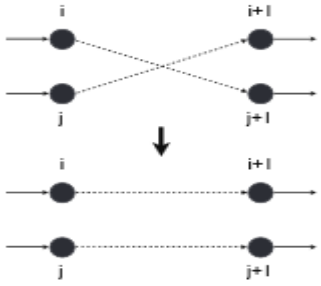
Burada r(k), [0,1] ve ap(k, i) aralığındaki rasgele sayıların eşit dağıldığı durumlarda, i(th) yinelemesinde karga k'nın farkındalık olasılığını temsil eder. Şekil 4.1'de [35] bu denklemlerin anlamı gösterilmektedir.



Şekil 4.1. Karga Takip Etme Davranışı

4.2. 2-opt Algoritması

2-opt algoritması temel olarak turdan iki kenarı kaldırıp oluşturulan iki yeni alt turu yeniden bağlar. Bu genellikle 2-opt hareketi olarak adlandırılır. Turun geçerli kalması için iki alt turu yeniden bağlamanın tek bir yolu vardır. Adımlar yalnızca yeni tur daha kısa olduğu sürece tekrarlanır. Turun çıkarılması ve yeniden bağlanması en uygun rotayı sağlar (minimum yerel) [78]. Şekil 4.2 2-opt'un çalışmasını göstermektedir [78].



Şekil 4.2. 2-opt Algoritmasının çalışması

4.3. Yeni Geliştirilen Ayrık Karga Arama Algoritması (District Crow Search Algorithm DCSA)

Bu çalışmada, Gezgen Satıcı Problemini çözmek için klasik Karga Arama Algoritması (Crow Search Algorithm CSA) kullanılmıştır. Karga Arama Algoritması başlangıçta matematiksel karşılaştırma problemleri gibi sürekli problemleri çözmek için uygulanmıştır. Bu yüzden ilk önce Karga Arama Algoritması, TSP problemine uyacak şekilde düzeltilir. Bunun için Karga Arama Algoritmasının bazı bölümleri değiştirilip, algoritmaya başka teknikler de eklenerek ayrık problem çözümlerine uyarlanmıştır [79]. Bu sebeple geliştirilen algoritmaya Ayrık Karga Arama Algoritması (District Crow Search Algorithm) ismi verilmiştir.

TSP probleminde her karganın konumu (çözümü) bir tur demektir. Mesela bir karganın bulduğu çözüm (şehir3 => şehir15 => şehir2 => şehir50 …) olarak verilmektedir. Bir karganın konumu ve diğer takip edecek karganın konumu arasındaki mesafe Hamming Mesafesi (Hamming Distance) ile tanımlanır. Yani iki karganın arasındaki mesafeyi hesaplamak için ikisinin türlerine bakılır ve şehirlerin sırasının farklıkları sayılır. Ne kadar büyük rakam çıkarsa o iki karga birbirine uzak demek ne kadar düşük rakam çıkarsa birbirine yakın demektir. Mesela şöyle iki çözüm olduğunu varsayalım:

x1 : [0,1,2,3,4,5,6,7]

x2 : [0,1,3,2,5,4,6,7]

x1 ve x2 arasındaki Hamming Mesafesi 4 olur.

Daha sonra karga algoritmasına göre her karga konumunu güncellemek gerekir. Bunun için 2-opt yerel arama işlemi kullanılmıştır. Karga Arama Algoritması'nda bir karga yiyeceklerini çalmak için diğer kargaları takip etmeye çalışır. Bir karganın diğer kargaları yanlış yönlendirmek için rastgele bir yere gitmesi gerekmektedir. Buna göre bir karganın yeni konumunu belirlemek için Hamming Mesafesi (Hamming Distance) ve 2-opt işlemini kullanarak o karganın yeni konumu hesaplanır. Hamming mesafesine göre 2-opt işleminin çalışma sayısı belirlenir. Mesela iki karganın türlerinin arasında 10 fark varsa 2-opt işlemi 10 sefer çalıştırılır.

Bir karganın rastgele bir yere uçması, mevcut turundan itibaren başka bir tur oluşturması demektir. Rastgele bir yere uçmak için 1 ve şehir sayısı arasından rastgele bir rakam üretilir ve o rakama göre yine 2-opt fonksiyonu çalıştırılır.

4.4. TSP Sonuçları

Bu karşılaştırmada 8 algoritma kullanılmıştır. Tepe Tırmanışı (Hill Climbing HC), Tavlama Algoritması (Simulated Annealing SA), Tabu Arama (Tabu Search TS), Kanguru Algoritması (Kangro Search KA), Yapay Arı Kolonisi (Artificial Bee Colony ABC), Genetik Algoritma (Genetic Algorithm GA) ve Karınca Kolonisi Opytimizasyonu Algoritması (Ant Colony Optimization ACO) [80]. Popülasyon sayısı 20'ye ve iterasyon sayısı 3000'e ayarlanmış olup her algoritma 20 kez çalıştırılarak sonuçların ortalaması alınmıştır.

Tablo 4.1’deki sonuçlara göre, Ayrık Karga Ama Algoritması-DSCA’nın diğer algoritmalara kıyasla daha iyi sonuçlar verdiği görülmektedir. **gr17, gr21, fr26** ve **bays29**' problemlerinde DSCA diğer algoritmalarla birlikte optimum sonuçlara ulaşmıştır. **gr48, hk48, brazil58, eil76** ve **pr76** problemlerinde hiç bir algoritma optimum sonuca ulaşamamış olup, ancak diğer DSCA optimuma en yakın sonuçlara ulaşmış, sadece **eil51** probleminde ise ABC(2018) algoritması optimum en yakın sonuca ulaşmıştır. **gr24, swiss42, eil51, berlin52** ve **st70** problemlerinde DSCA optimuma en yakın ikinci değere ulaşmıştır.

Tablo 4.1. Ayrık Karga Arama Algoritmasının diğer algoritmalar ile mukayesesi

TSP Problemi	Optimum	DCSA	HC	SA	TS	KA	ABC	ABC 2018	GA	ACO
		Average								
gr17	2085	**2085**	2354	2114,7	2120	2094,8	**2085**	**2085**	**2085**	**2085**
gr21	2707	**2707**	3767	3027,9	2831	2849,6	**2707**	-	2726,3	**2707**
gr24	1272	1275.2	1786	1335,5	1341	1342,7	1281	-	1291,2	**1272**
fri26	937	**937**	1395	969,7	988	1004,9	948	-	955,0	**937**
bays29	2020	**2020**	2187	2062,6	2100	2182,6	2034	**2020**	2034,5	**2020**
dantzig42	699	707.8	1459	796,0	748	811,2	744	**704**	727,6	**699**
swiss42	1273	**1280.2**	2444	1483,9	1378	1456,6	1370	1273	1342,2	1297,8
gr48	5046	**5115.35**	10388	5895,6	5364	5699,2	5595	-	5183,3	5172,8
hk48	11461	**11566.8**	23823	15399,9	12287	12621,5	12465	-	12011,5	11712
eil51	426	**437.66**	596	480,6	457	470,4	459	**427.01**	443,7	443,7
berlin52	7542	**7718.19**	10978	9100,6	8338	8841,5	8594	**7542**	8146,6	7819,1
brazil58	25395	**25518.2**	58144	32426,6	26666	29970,6	28951	-	26405,2	26263,7
st70	675	**691.65**	1880	863,2	849	829,4	830	**675**	731,3	715,7
eil76	538	**565.39**	1367	649,6	604	611,6	622	**538**	573,5	573,5
pr76	108159	**110507.12**	308382	165690,0	119510	130888,0	130729	-	116416,1	114392,2
Ortalama	11349	11542.17	28730	16153.09	12372.07	13444.97	13294.27	-	12071.53	11874.03
Başarı Sırası	**Optimum**	1	8	7	4	6	5	-	3	2

En iyi-Optimum sonuç | Optimuma en yakın 1.sonuç | Optimuma en yakın 2.sonuç

Tablo4.1'de Kıyaslanan algoritmalardan ACO algoritması 15 problemin ilk 6'sında optimum sonuçları alarak %40 oranında başarı sağlamıştır. Geliştirilen DCSA'nın genel olarak 15 problemin 4'unda optimum sonuca ulaşmış, 5'inde optimuma en yakın sonuca ulaşarak toplamda 9 birincilikle %60 oranında başarı sağlamış, 5'inde de diğer algoritmalara göre ikinci en iyi sonuca ulaşarak genelde %93,3 başarı sağlayarak yüksek performans ve diğer iyi bilinen algoritmalara göre rekabetçi sonuçlara sahip olduğu görülmüştür. Tablodaki başarı sırası incelendiğinde; tezde önerilen DCSA'nın optimum sonuçlara yada optimuma en yakın değerlere ulaştığı için sıralamada 1.inci olduğu, 2.nci sırada ACO, 3.üncü GA, son sıralamalarda ise SA ve HC olduğu açıkça görülmekte olup, DCSA'nın kıyaslanan algoritmalara göre en iyi performansı sağladığı görülmüştür.

5. BÖLÜM

HİBRİT KARGA-YARASA MODELİ GELİŞTİRME

Bu bölümde yeni bir hibrit algoritması tanıtılmaktadır. Bu hibrit algoritmada Karga Arama Algoritması ve Yarasa Algoritması kullanılmıştır. Her iki algoritmanın temel özellikleri bu yaklaşımda birlikte kullanılmış, ayrıca bu hibrit algoritmada kuantum davranışlı parçacık sürü optimizasyonu denklemleri de kullanmıştır.

5.1. Karga Arama Algoritması

Bölüm 4.1'de klasik Karga Arama Algoritması hakkında detaylı bilgiler verilmiştir.

5.2. Yarasa Algoritması (Bat Algorithm)

Yarasa algoritması, Xin-She Yang tarafından 2010 yılında geliştirilmiş bir meta-sezgisel algoritmadır [42][43]. Değişen nabız emisyon ve ses yüksekliği oranlarına sahip mini yarasaların ekolojik yerleştirme davranışından esinlenmiştir. Kanatlı tek memelilerdir ve ayrıca gelişmiş ekolokasyon yeteneğine sahiptirler. Tüm memeli türlerinin %20'sini oluşturan yaklaşık 996 farklı yarasa türün olduğu tahmin edilmektedir. Boyutları küçük yaban arısı yarasasından (yaklaşık 1.5 ila 2 g) kanat genişliği yaklaşık 2 metre ve ağırlığı yaklaşık 1 kg'a kadar değişir. Mini yarasaların tipik olarak yaklaşık 2.2 ila 11 cm önkol uzunluğuna sahiptir. Çoğu yarasa belirli bir dereceye kadar ekolokasyon kullanır; tüm türler arasında mini yarasalar tanınmış bir örnektir. Çünkü mini yarasalar ekolokasyonu yoğun bir şekilde kullanırken mega yarasalar kullanmaz. Mini yarasalar avı tespit etmek, engellerden kaçınmak ve karanlıkta tünek çatlaklarını bulmak için ekolokasyon adı verilen bir tür sonar kullanır. Bu yarasalar çok yüksek bir ses nabzı (ses çıkarma hızı) yayar ve çevresindeki nesnelerden geri dönen yankıyı dinler. Nabızları (ses çıkarma hızı) özelliklerine göre değişir ve türlere bağlı olarak avlanma stratejileri

ile ilişkilendirilebilir. Sinyal bant genişliği türlere göre değişir ve genellikle daha fazla harmonik kullanılarak arttırılır [42].

Yankıların Akustiği

Her bir nabzın süresi sadece saniyenin birkaç binde biri kadar sürse de (yaklaşık 8 ila 10 ms'ye kadar), genellikle 25kHz ila 150 kHz arasında olan sabit bir frekansa sahiptir. Av için avlanırken, nabız emisyonu (çıkarması), avlarının yakınında uçarken saniyede yaklaşık 200 nabza kadar hızlandırılabilir. Bu tür kısa ses patlamaları, yarasaların sinyal işleme gücünün fantastik yeteneğini ima eder. Havadaki ses hızı tipik olarak v = 340 m / s olduğundan, ultrasonik ses patlamasının dalga boyu λ, aşağıdaki denklemde verilmiştir [42]:

$$\lambda = \frac{v}{f} \tag{5.1}$$

25kHz ila 150 kHz arasındaki tipik frekans aralığı olurken dalga boyları av boyutlarıyla 2mm ila 14mm arasındadır. Çalışmalar, mini yarasaların çevrenin üç boyutlu senaryolarını oluşturmak için yankının emisyonu ve tespitinden kaynaklanan gecikmeyi, iki kulakları arasındaki zaman farkını ve ekoların ses yüksekliği varyasyonlarını kullandığını göstermektedir. Böylece hedefin mesafesini ve yönünü, avın türünü ve hatta küçük böcekler gibi avın hareket hızını tespit edebilirler. Çalışmalar, yarasaların, hedef böceklerin kanat çırpma hızlarının neden olduğu Doppler etkisinin varyasyonlarıyla hedefleri ayırt edebildiğini ortaya koymuştur. Mini yarasaların bu gibi yeniden yerleştirme davranışı, optimize edilecek objektif fonksiyon ile ilişkilendirilebilecek şekilde formüle edilebilir ve bu, yeni optimizasyon algoritmalarının formüle edilmesini mümkün kılar [42][43].

Basitlik için, aşağıda verilen yaklaşık veya idealize edilmiş kurallar kullanılır [42]:

- Tüm yarasalar mesafeyi algılamak için yankılanmayı kullanırlar ve ayrıca yiyecek / av ve arka plan engelleri arasındaki farkı büyülü bir şekilde 'bilirler'.
- Yarasalar, av aramak için sabit frekans fmin, değişken dalga boyu λ ve ses yüksekliği A0 (başlangıç ses yüksekliği) ile x (i) konumunda hız v i ile rastgele uçar. Yayılan nabızlarının dalga boyunu (veya frekansını) otomatik olarak

ayarlayabilir ve hedeflerinin yakınlığına bağlı olarak nabız emisyon oranını r ∈ [0,1] ayarlayabilirler.

- Ses yüksekliği birçok yönden değişebilse de ses şiddetinin büyük (pozitif) A0 ile minimum sabit Amin değeri arasında olduğu varsayılır.

Yarasa algoritmasının kaba kodu aşağıda verilmiştir [42]:

Hedef fonksiyonu f(x), x=(x1, ..., xd)T
Yarasaları başlatxiand vifor i=1...n
Frekans belirleQi∈[Qmin,Qmax]
Artış oranını belirle riand the loudness Ai
while(t<Tmax)//iterasyon sayısı
    Frekansı ayarlayarak yeni çözümler üret
    Çözümleri ve hızı güncelle[Eq.(2)to(4)]
    if(rand(0,1)>ri
        Yeni çözümler hesapla NewBat=BestBat+Velocity
        En iyi çözüm etrafından yeni bir çözüm üret
    end if
    Rastgele uçarak yeni çözüm üret
    if(rand(0,1)<Aiandf(xi)<f(x))
        Yeni çözümleri kaydet
        ri ve Ai değerini azalt
    end if
    Yarasaları sırala ve en iyi çözümü bul
end while

Sanal Yarasaların Hareketi

Simülasyonlarda sanal yarasalar kullanılır. Kurallar, d-boyutlu bir arama alanındaki x (i) ve hızları v (i) konumlarının nasıl güncellendiği konusunda tanımlanmalıdır. T adımında yeni çözümler x (t, i) ve v (t, i) hızları şu şekilde verilir [42]:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{5.2}$$

$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_i^t - \mathbf{x}_*)f_i, \tag{5.3}$$

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t, \tag{5.4}$$

Burada β ∈ [0,1], muntazam bir dağılımdan çizilmiş rastgele bir vektördür. Burada x tüm n yarasalar arasındaki tüm çözümleri karşılaştırdıktan sonra bulunan mevcut global optimum konumdur (çözüm). Çarpma λ (i)*f (i) hız artışı olduğundan, diğer faktörü λ (i) (veya f(i) sabitlerken hız değişikliğini ayarlamak için ilgili sorunun türüne bağlı olarak f (i) (veya λ (i)) kullanılabilir).

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \epsilon A^t$$

Yerel arama kısmı için, mevcut en iyi çözümler arasından bir çözüm seçildikten sonra, rasgele yürüyüş kullanılarak yerel olarak her yarasa için yeni bir çözüm üretilir.

$$\epsilon \in [-1, 1] \tag{5.5}$$

Burada A, t zamanında adımında tüm yarasaların ortalama ses şiddetidir.

Ses Yüksekliği ve Nabız Emisyonu

Ses yüksekliği A (i) ve nabız emisyonunun r(i) hızı, iterasyonlar ilerledikçe buna göre güncellenmelidir. Bir yarasanın avını bulduktan sonra ses yüksekliği genellikle azaldığından, nabız emisyon hızı artarken ses yüksekliği herhangi bir kolaylık değeri olarak seçilebilir. Basitlik için, A(0) = 1 ve A(min) = 0 da kullanılabilir, A(min) = 0 varsayıldığında, bir yarasanın avı yeni bulduğunu ve herhangi bir ses yaymayı geçici olarak durdurduğunu varsayar [42][43].

$$A_i^{t+1} = \alpha A_i^t, \qquad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{5.6}$$

Burada α ve γ sabittir. Aslında α, simüle edilen tavlamadaki bir soğutma programının soğutma faktörüne benzerdir.

0 <α <1 ve γ> 0 için.

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad \text{as} \quad t \to \infty \tag{5.7}$$

5.3. Yeni Geliştirilen Hibrit Karga-Yarasa Algoritması - HCSBA

Yeni algoritmada, sürüdeki her üye, sürüdeki diğer üyelerin yiyeceklerini nerede sakladıklarını görmek için gözlemlemek anlamında bir karga gibi davranır. Aynı zamanda yarasalarda olduğu gibi, her üye kendi çözümünü ararken yankı sistemini kullanacaktır. Yankı sistemi parçacık sürü optimizasyon denklemleri ile entegre edilmiştir. Her üyenin Karga Algoritmasında olduğu gibi bir farkındalık parametresi vardır. Farkındalık parametresine göre bir üye başka bir üyenin onu takip edip etmediğini bilebilir [79].

Sürü üyelerinin hareketinde iki ihtimal var. Birinci ihtimal, üye j, üye i'nin onu takip ettiğini bilmiyor ve sonuç olarak, üye i üye j'nin yiyeceğini sakladığı yere yaklaşacak. İkinci ihtimal ise, üye j, üye i'nin onu takip ettiğini biliyor ve sonuç olarak, önbelleğinin çalınmasını önlemek için, j üyesi, arama alanının başka bir konumuna giderek üye i'yi kandırıp götürür. Bir üye en iyi gıda (çözüm) yerini bulmak için kuantum davranışlı parçacık sürü optimizasyon denklemleri ile birlikte yankı sistemini kullanır. Aynı zamanda, izlediği ve peşinde düştüğü diğer üyelerin sakladıkları yiyeceklerin yerini (çözümlerini) değerlendirir. Bunun hepsine göre her üyenin arama alanında yeni yeri tanımlanmış olur.

Ayrıca, yerel arama uygulanır, böylece her üye yakındaki çözümlere bakarak kendi çözümünü geliştirmeye çalışır. Bu, algoritma keşif aşamasında yardımcı olur. Yukarıdaki verilenlere göre bu hibrit algoritmanın aşamaları ve denklemleri aşağıdaki gibi verilebilir:

Sürü bireylerinin konumları için rastgele değerler atanmıştır. Ayrıca ses yüksekliği (A) ve nabız emisyon (r) parametrelerine başlangıç değerler atanır.

Farkındalık parametresine (AP) 0 ve 1 arasında bir değer atmak gerekmektedir. Meta-sezgisel algoritmalar çeşitlendirme ve yoğunlaşma arasında iyi bir denge sağlamalıdır. Bu tez çalışmasında geliştirilen hibrit algoritmada yoğunlaşma ve çeşitlendirme esas olarak farkındalık olasılığı (AP) parametresi ile kontrol edilir.

Algoritmanın durma kriteri iterasyon sayısına bağlıdır ve son iterasyona kadar algoritma mevcut çözümden daha iyi bir çözüm bulmaya devam eder.

Bireylerin konumlarını güncellemek için farkındalık parametresine göre farklı denklemler kullanılır. İlk önce rastgele bir sayı üretiliyor ve rand <AP ise ses yüksekliği faktörünü kullanarak rastgele bir çözüm üretilir. Eğer rand> AP ise iki denklem kullanılır. Birinci denklem karga arama algoritmasındaki konum güncelleme denklemi gibi ve ikinci denklem ise kuantum davranışlı PSO denklemidir [81].

$$X_{i,j}(t+1) = \begin{cases} p_{i,j}(t) + \beta \cdot \left(M\text{best}_j(t) - X_{i,j}(t)\right) \cdot \ln\left(\frac{1}{u}\right), & \text{if } h > 0.5 \\ p_{i,j}(t) - \beta \cdot \left(M\text{best}_j(t) - X_{i,j}(t)\right) \cdot \ln\left(\frac{1}{u}\right), & \text{otherwise,} \end{cases} \quad (5.8)$$

h ve u iki rasgele sayıdır, β parametresine daralma-genleşme katsayısı denir, algoritmanın yakınsama hızını kontrol etmek için ayarlanabilir. Mbest, PSO Algoritmasındaki gibi popülasyonun ortalama en iyi pozisyonudur. p PSO Algoritmasındaki gibi yerel çekicidir ve değeri probleme göre ayarlanır.

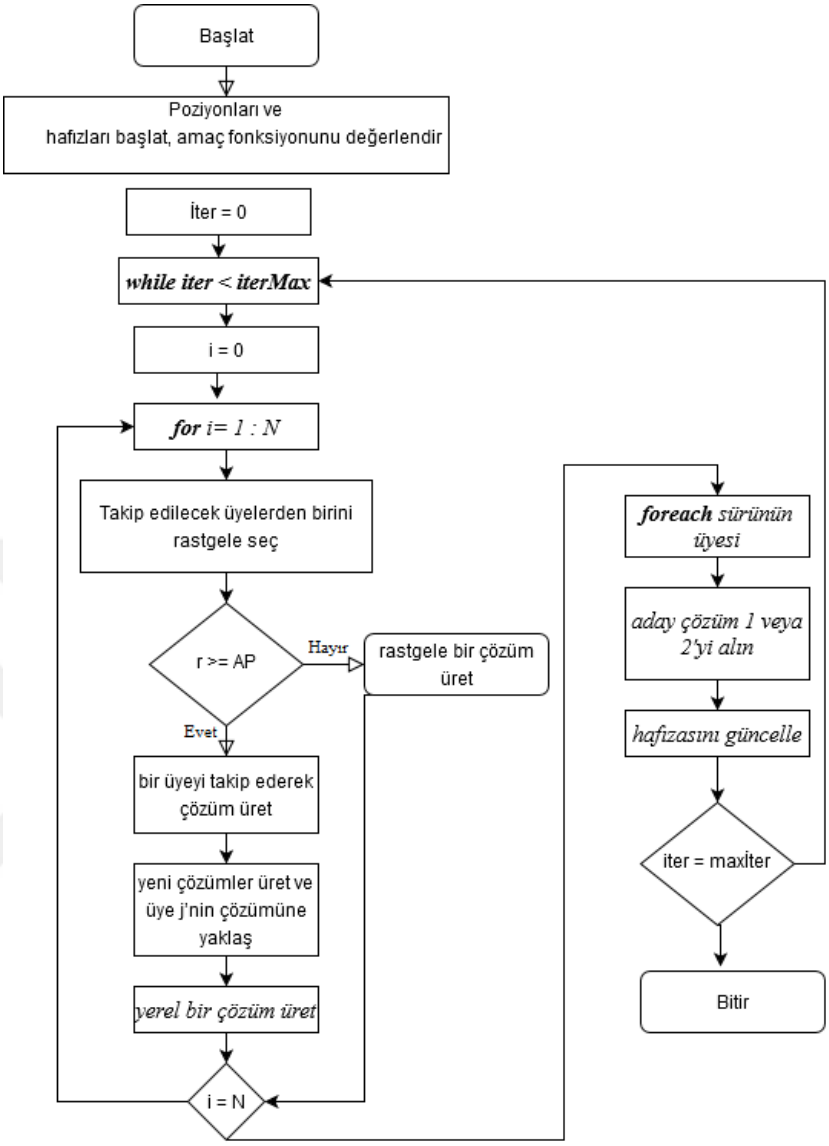
İki denklemden çıkan sonuçlar kaydedilir ve daha sonra algoritma ikisinden hangisi daha iyi sonuca yol açarsa onu seçer. Denklem 5.8 den çıkan konumlar için bir olasılık (rand > r) ile her çözümün etrafında yerel bir çözüm üretilir.

Buna göre, bazı üyelerin konumları karga arama algoritması gibi diğer üyeleri takip ederek, bazıları kuantum davranışlı PSO denklemi kullanarak ve bazıları ise ses yüksekliği parametresine göre bir konuma geçmiş olur.

Yeni oluşan çözümlerde eğer mevcut çözümlerden daha iyi bir çözüm varsa yarasa algoritması gibi bir olasılıkla (rand < A) alınır ve global en iyi çözüm güncellenir. Aşağıda Hibrit Karga Yarasa Algoritmasının kaba kodu ve Şekil 5.1'de algoritmanın akış diyagramını gösterilmiştir.

Hibrit Karga-Yarasa Kaba kodu

```
Arama alanındaki N üye sürüsünün konumunu rastgele üret
Üyelerin pozisyonlarını değerlendir
Her üyenin hafızasını tanımla
Bir farkındalık olasılığı tanımla (AP)
Her birey için amac fonksiyonunu değerlendir
while iter < iter max
        for i = 1 : N (N: sürünün sayısı)
          Takip edilecek üyelerden birini rastgele seç (örneğin j)
          If (r >= AP)
                Karga aramasında olduğu gibi başka bir üyeyi takip ederek çözüm üret (aday
                çözüm1)
                Kuantum davranışlı parçacık sürü optimizasyon denklemlerini kullanarak
                        yeni çözümler üret ve üye j'nin çözümüne yaklaş (aday çözüm2)
          else
                ses yüksekliği faktörünü kullanarak rastgele bir çözüm üret
          end
          Bir olasılık rand ile her çözüm (aday çözüm2 olanlar için) etrafında yerel bir çözüm
üret
        end for
        foreach sürünün üyesi
                Amaç fonksiyonun değerine göre aday çözüm 1 veya 2'yi alın
                if yeni çözüm, bireyin hafızasındaki çözümden iyiyse bireyin hafızasını güncelle
                end if
                if (rand < A i & f(x i ) < f(x *))
                        Yeni çözümü kabul et
                        Ai'yi azaltın
                end
        end foreach
        Tanımlanan adım sayısından sonra çözüm düzelmezse, ses yüksekliğini (A) ve nabız
        hızlarını (r) yeniden tanımla
end while
```



Şekil 5.1 Hibrit Karga-Yarasa Algoritmasının akış diyagramı

5.4. Benchmark Test Sonuçları

Bu araştırmada, önerilen HCSBA 'nın keşif ve sömürü yeteneklerinde iyi bilinen meta sezgisel algoritmaya karşı performansını değerlendirmek için birçok karşılaştırma fonksiyonu kullanılmıştır. Önerilen hibrid algoritmanın verimliliğini doğrulamak için HCSBA 'nın performansı CSA ve BA'nın yanı sıra iyi bilinen diğer algoritmalar ile karşılaştırılır. Sonuçları karşılaştırmak için ortalama değerler kullanılır.

Tablo l'de, tüm algoritmalar aynı parametre seti kullanılarak karşılaştırıldı. Her algoritmanın çalışma sayısı 2000 iterasyona, karar numarası değişkenleri (boyut) 10'a ve popülasyon boyutu 20'ye ayarlanmıştır.

Tablo 2'de yapılan karşılaştırmada her algoritmanın çalışma sayısı 2000 iterasyona, karar sayı değişkenleri (boyut) 10'a ve popülasyon büyüklüğü 50'ye ayarlanmıştır. Tablo1 ve Tablo 2, önerilen karşılaştırmanın sonuçlarını göstermektedir. Sırasıyla CSA, BA ve Geliştirilmiş Yarasa Algoritması (IBA) ile HCSBA [35, 43].

Tablo 5.1. HCSBA Karga Aramanın Algoritması ile mukayesesi

Benchmark Functions	**Fmin**	**HCSBA**	**CSA**
		AVERAGE	
F1: Sphere	0	**1.10E-117**	4.09E-11
F2: Rosenbrock	0	**3.2731**	10.86
F3: Griewank	0	**0.11141**	0.21
F4: Schwefel	0	**1.49E-74**	6.27E-03
F5: Ackley	0	**4.56E-15**	1.9

Tablo 5.2. HCSBA BA ve IBA ile mukayesesi

Benchmark Functions	**Fmin**	**HCSBA**	**BA**	**IBA**
		AVERAGE		
F1: Sphere	0	**1.41E-284**	7.90E-01	8.11E-06
F2: Zakharov	0	**1.20E-298**	3.38E+01	4.63E-03
F3: Sum of Different Power	0	**0**	2.72E-03	5.38E-06
F4: Dixon-Price	0	**0.66667**	7.90E+01	**0.66667**
F5: Step	0	**5.20E-19**	7.90E+01	6.67E-01
F6: Michalewicz	-9.66015	**-8.2989**	-5.16	-7.91
F7: Griewank	0	**0.12181**	1.14E+01	1.34
F8: Easom (d=2)	-1	**-1**	-3.25E-02	-9.99E-01
F9: Perm (d=4)	0	9.66E-02	3.54E-01	**7.16E-02**
F10: Six Hump Camel Back (d=2)	-1.0316	**-1.0316**	-1.03093	**-1.0316**

Tablo sonuçlarına göre, önerilen hibrid algoritmanın, BA ve CSA'nın iki farklı algoritmadan önemli ölçüde daha iyi performans gösterdiği açıktır. Bu başarı, önerilen algoritmanın yüksek keşif ve sömürü yeteneğinden kaynaklanmaktadır ki; burada geliştirilen BA ve CSA algoritmalarından türetilmiş olup, bunların en iyi özelliklerini kullanması nedeniyle normaldir.

Tablo 2 sonuçlarına göre literatürdeki geliştirilmiş Bat Algoritmasının (IBA) F9 Perm fonksiyonunda biraz daha iyi sonuç alabildiği görülmektedir. HCSBA, 10 test fonksiyonun 9'unda diğerlerine göre en iyi sonuca ulaşmış (f9 fonksiyonu hariç).

Tablo 3'da HCSBA literatürde [82] popüler diğer meta-sezgisel algoritmalar ile mukayese yapılmıştır. Bu algoritmalar Guguk Arama Algoritması (Cuckoo Search Algorithm), Diferansiyel Evrim Algoritması (Deferential Evolution Algorithm), Ateşböceği Algoritması(Firefly Algorithm), Genetik Algoritma(Genetic Algorithm), Parçacık Sürü Optimizasyon Algoritması (Particle Swarm Optimization).

Karşılaştırmada, algoritmanın hızlı yakınsama yeteneğini değerlendirebilecek, yerel optimumdan atlayabilecek, çok sayıda yerel optima elde edebilecek ve erken yakınsamayı önleyebilecek şekilde seçilmiştir. Yeni hibrit algoritması ve çeşitli test fonksiyonları üzerindeki diğer algoritmalar tarafından elde edilen ortalama değerler Tablo 3'te listelenmiştir [82]. Simülasyon sonuçları, HCSBA 'nın diğer algoritmalara kıyasla genellikle çok iyi performans verdiğini göstermektedir.

Tablo 5.3'te yapılan karşılaştırma için, her algoritma 10000 iterasyona ayarlanmış, karar numarası değişkenleri 30'a (veya 30 uygulanabilir değilse Karşılaştırma işlevi türüne bağlı olarak) ve popülasyon boyutu 50'ye ayarlanmıştır.

HCSBA sonuçları incelendiğinde, HCSBA'nın F2, F5, F7, F12, F14, F15, F18 ve F19 nolu fonksiyonlar olmak üzere 8'inde optimum değerlere ulaştığı, F1, F4, F6, F8, F13, F17 ve F20 olmak üzere 7 fonksiyonda ise optimuma en yakın sonuçlara ulaştığı, F3, F9, F10, F11, F16 ve F21 nolu fonksiyonla ise diğer algoritmaların başarılı olduğu görülmekte olup, neticede HCSBA'nın karşılaştırmada kullanılan diğer algoritmaların sonuçlarından daha iyi olduğu anlaşılmaktadır.

Tablo 5.3. Diğer algoritmalarla ile kıyaslama

Benchmark Functions	Fmin	HCSBA	CSA	BA	CS	DE	FA	GA	PSO
		AVERAGE							
F1: Sphere	0	**5.70E-285**	3.9294E-16	0.02	6.4894E−61	2.61E−165	7.86960E−04	2.2039E−02	1.76657E−51
F2: Beale	0	**0**	0	1.17E-20	9.7674E−61	**0**	1.3683E−04	9.6789E−03	1.12460E−50
F3: Step	0	3.28E-16	3.3564E-16	3.1304E-9	1.22	1.3000E−01	**0**	2.3026E−02	1.31167E−31
F4: Quartic	0	**1.03E-03**	4.5788E-3	4.006E-2	1.24E+01	2.61590E−03	5.95780E−02	2.7387E−03	3.740E+00
F5: Bohachevsky	0	**0**	0	1.03E-4	4.9218E−04	**0**	**0**	**0**	3.0393E−02
F6: Ackley	0	**5.86E-15**	2.9923	1.334	1.71E+01	1.72E+00	7.7334E−04	4.6683E−02	1.82E+01
F7: Griewank	0	**0**	9,3504E-3	4.5E-10	2.01E+01	9.220E−17	1.07E−07	5.5598E−02	4.05E+01
F8: Levy	0	**1.35E-31**	2.3498E-31	0.62891	1.62E+01	5.9824E−02	1.2E+00	1.3732E−04	4.93E+00
F9: Michalewiz	-9.66015	-7.8322	-8.7955	-3.5404	−8.06	−9.61550	−8.88	**−9.66**	−7.750
F10: Rastrigin	0	15.6037	21.1263	55.7366	1.08E+02	**1.25E+01**	30.7	1.65E+01	1.45E+02
F11: Alpine	0	3.8031	0.016018	46.133	1.06E+01	6.9E−16	1.1934E−02	**3.147E−17**	4.4402E−02
F12: Schaffer	0	**0**	0	0.00024	9.8155E−01	**0**	1.4304E−02	1.6896E−17	**0**
F13: Rosenbrock	0	**17.7875**	25.66	72.3622	3.950E+01	2.360E+01	1.840E+01	3.79E+01	2.57E+02
F14: Easom	-1	**-1**	**-1**	-0.049872	**-1**	**-1**	**-1**	**-1**	**-1**
F15: Shubert	−186.7309	**186.7309**	**-186.7309**	-147.2269	**− 186.7309**	**− 186.7309**	**− 186.7309**	**− 186.7309**	**− 186.7309**
F16: Schwefel 2.21	0	0.27355	2.0879E-3	6.4741	3.6728E−03	1.02E+00	**1.4317E−04**	4.6662E−02	1.0586E−01
F17:Schwefel 2.22	0	**4.10E-188**	0.65197	0.7487	1.12E+01	8.15E−91	1.5046E−03	5.4367E−02	7.28E+01
F18: Goldstein	3	**3**	**3**	3.793	**3**	**3**	**3**	**3**	**3**
F19: Matyas	0	**0**	1.702E-43	0.04037	1.3308E−06	**0**	**0**	8.3889E−104	**0**
F20: Powell	0	**2.64E-07**	0.0029875	4.7045	6.64E+00	1.00E+01	2.3957E−04	3.2713E−02	3.37E+03
F21: Power sum	0	1.66E-04	**1.8091E-12**	0.83	1.4015E−02	6.27E−04	1.2905E−04	1.2997E−04	1.98390E−04
Ortalama	-9.2567	-7.39	-2.36	2.00136	2.44	-6.92	-6.82	-6.65	1.77
Başarı Sırası	Optimum	1	5	7	8	2	3	4	6

	En iyi sonuç		İkinci en iyi sonuç

Tablo 5.3 sonuçlarına göre, kıyaslanan algoritmalardan DE algoritmasının 7'isinde optimum değere ulaştığı 1'inde ise optimuma en yakın değer ulaştığı, FA algoritmasının ise 6 algoritmada optimum değere ulaştığı, 1'inde ise optimuma en yakın değere ulaştığı görülmüş olup; DE algoritmasının 21 problemin 8'unda en iyi sonuçlara ulaşarak %38 oranında başarı sağladığı görülmüştür. Bu tez çalışmasında geliştirilen HCSBA'nın ise 21 problemin 15'inde en iyi sonuçlara ulaşıp %72,4 oranında başarı sağlayarak yüksek ve rekabetçi performans gösterdiği görülmüştür. Tablodaki başarı sırası incelendiğinde; tezde önerilen HCSBA'nın optimum sonuçlara ya da optimuma en yakın değerlere ulaştığı için sıralamada 1.inci olduğu, 2.nci sırada DE, 3.üncü FA, son sıralamalarda ise BA ve CS olduğu açıkça görülmekte olup, DCSA'nın kıyaslanan algoritmalara göre en iyi performansı sağladığı görülmüştür.

6. BÖLÜM

TARTIŞMA, SONUÇ VE ÖNERİLER

Matematikte eniyileme ya da optimizasyon terimi; bir fonksiyonu minimize ya da maksimize etmek amacı ile gerçek ya da tam sayı değerlerini tanımlı bir aralıkta seçip fonksiyona yerleştirerek sistematik olarak bir problemi incelemek ya da çözmek işlemlerini ifade eder. Optimizasyon, karar verme süreçlerini hızlandırmakta ve karar kalitesini arttırmakta kullanılarak gerçek hayatta karşılaşılan problemlerin etkin, doğru ve gerçek zamanlı çözümünde yararlanılmaktadır.

Bu çalışmada, bilinen Benchmark problemleri ve gezgin satıcı problemlerini çözmek için Hibrit Karga Arama Yarasa Algoritması (HCSBA) ve Ayrık Karga Arama Algoritması (DCSA) geliştirilmiştir.

İlk çalışmada, önerilen Hibrit Karga Yarasa Arama Algoritması (HCSBA), Yarasa Algoritması ve Karga Arama Algoritmalarına ait ses yüksekliği, yerel arama, farkındalık, çözüm hafızası ve takip etme özelliklerini kullanarak aday çözümler üretmektedir. Böylece aday çözümler arasından en iyi olanları seçerek mevcut çözümü iyileştirmektedir. Geliştirilen hibrit model ile hem yerel optimumdan kaçınma hem de global optimumu yakalamak için algoritmanın keşif ve sömürü yetenekleri arasında uygun bir denge kurulmuştur. Bu çalışmada HCSBA algoritması ele alınan diğer algoritmalardan Guguk Arama Algoritması, Diferansiyel Evrim Algoritması, Ateşböceği Algoritması, Genetik Algoritma, Parçacık Sürü Optimizasyon Algoritması ile karşılaştırılmış ve daha iyi sonuçlar elde edilmiştir. Tablo 5.3'deki Sonuçlara göre, HCSBA, 21 adet Benchmark probleminden 8'inde en optimum değere ulaşmış, 7'inde ise kıyaslanan algoritmalara göre HCSBA en optimum değere en yakın olan en iyi sonuçlara ulaşıp %68,2 oranında başarı sağlayarak yüksek ve rekabetçi performans gösterdiği görülmüştür.

İkinci çalışmada ise Gezgin Satıcı Problemlerinin çözümü için Ayrık Karga Arama Algoritması (DCSA) önerilmiştir. Klasik Karga Arama Algoritması süreklilik içeren problemlere uygun olup, süreklilik içermeyen Gezgin Satıcı problemlerine uygulanabilmesi için bu çalışmada Ayrık Karga Arama modeli geliştirilmiştir. Ayrık model geliştirilirken Hamming mesafesi ve 2-opt algoritması kullanılmıştır.
DCSA performansını test etmek için iyi tanınan algoritmalardan Tepe Tırmanışı (Hill Climbing HC), Tavlama Algoritması (Simulated Annealing SA), Tabu Arama (Tabu Search TS), Kanguru Algoritması (Kangro Search KA), Yapay Arı Kolonisi (Artificial Bee Colony ABC), Genetik Algoritma (Genetic Algorithm GA) ve Karınca Kolonisi Opytimizasyonu Algoritması (Ant Colony Optimization ACO) ile 15 adet farklı TSP problemleri üzerinde kıyaslanmış ve diğer algoritmalara göre daha başarılı ve rekabetçi sonuçlar alınmıştır. DCSA verilen 15 adet TSP problemlerinde 4 adet optimum çözüme ulaşılmış ve 5 adet problemde ise diğer algoritmaların çözümlerine göre optimuma en yakın çözüm sonuçları elde edilmiştir. Bu çalışmada geliştirilen DCSA'nın genel olarak 15 problemin 9'unda en iyi sonuçları alarak %60 oranında başarı sağlayarak yüksek performans ve diğer iyi bilinen algoritmalara göre rekabetçi sonuçlara sahip olduğu görülmüştür.

Bu tez çalışmasında geliştirilen HCSBA ve DCSA algoritmaları ilerde henüz uygulanmamış olan sürekli ve ayrık problemlerinden çeşitli kısıtlamalara sahip daha fazla Benchmark test problemlerinde ve kombinatorial test problemlerinin çözümlerinde kullanılabilir.
İleri çalışmalar olarak aşağıdaki öneriler sunulabilir:

- Geliştirilen HCSBA'ın diğer gerçek hayattaki problemlerin çözümlerinde kullanılması,
- Geliştirilen HCSBA'ın kombinatorial problemlerin çözümlerinde kullanılması,
- Geliştirilen Ayrık Karga Arama Algoritmasının daha büyük TSP problemlerin çözümlerinde kullanılması,
- Geliştirilen Ayrık Karga Arama Algoritmasının diğer kombinatorial problemlerinden kesme ve yerleşme probleminin çözümünde kullanılması,
- Bu tezde geliştirilen hibrit modeller gibi farklı algoritmalarında hibrit modelleri geliştirilerek problemlerin çözümlerinde kullanılabilir.

KAYNAKÇA

1. Coello, C.A.C. (2009) Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. **Frontiers of Computer Science in China, 3(1)**, 18-30.
2. Abraham, A., & Jain, L. (2005). Evolutionary multiobjective optimization. In Evolutionary Multiobjective Optimization (pp. 1-6). Springer, London.
3. Deb, K. (2003). Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions. In Advances in evolutionary computing (pp:263-292). Springer, Berlin, Heidelberg.
4. Mirjalili, S., & Lewis, A. (2016). **The whale optimization algorithm. Advances in engineering software, 95**, 51-67.
5. Beheshti, Z. & Shamsuddin, S.M.H. (2013). **A review of population-based meta-heuristic algorithms. Int. J. Adv. Soft Comput. Appl, 5(1)**, 1-35.
6. Giagkiozis, I., Purshouse, R.C. & Fleming, P.J. (2015). **An overview of population-based algorithms for multi-objective optimisation. Int. Journal of Systems Science, 46(9):** 1572-1599.
7. Trelea, I.C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. **Information processing letters, 85(6)**, 317-325.
8. Danacı, M., & Alızada, B., (2019). An Improvement Of Hybrid Whale Optimization Algorithm. **Euroasıa journal of mathematıcs-engıneerıng natural & medıcal scıences, vol.2**, 60-68.
9. Danacı, M., & Dıallo, M. A., (2019). A New Hybrid Fruit Fly Optimization Algorithm For Solving Benchmark Problems. **Euroasıa journal of mathematıcs-engıneerıng natural & medıcal scıences, vol.2**, 23-27.
10. Danaci, M., & Dogan, C., (2019). ***Geliştirilmiş Balina Optimizasyon Algoritması*. Uluslararası Erciyes Bilimsel Araştırmaları Kongresi** (pp.482-495). Kayseri, Turkey.

11. Karaboga, D., & Gorkemli, B. (2011, June). A combinatorial artificial bee colony algorithm for traveling salesman problem. **In 2011 International Symposium on Innovations in Intelligent Systems and Applications** (pp. 50-53). IEEE.

12. Khader, A. T., Al-betar, M. A., & Mohammed, A. A. (2013). Artificial bee colony algorithm, its variants and applications: a survey.

13. Khan, I., & Maiti, M. K. (2019). A swap sequence based artificial bee colony algorithm for traveling salesman problem. **Swarm and evolutionary computation, 44**, 428-438.

14. Karaboga, D., & Gorkemli, B. (2019). Solving traveling salesman problem by using combinatorial artificial bee colony algorithms. **International Journal on Artificial Intelligence Tools, 28(01)**, 1950004.

15. Danaci, M., & Üçgün, H., (2014). Ad Hoc Ağları İçin Kuyruk Ağ Analizi ve Yapay Arı Kolonisi Algoritmalarının Birleştirilerek Routing Probleminin Simülasyonu. **2nd Int. Symposium on Innovative Technologies in Engineering and Science, ISITES 2014** (pp.1123-1132). Zonguldak, Turkey.

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

17. Valian, E., Mohanna, S., & Tavakoli, S. (2011). Improved cuckoo search algorithm for global optimization. **International Journal of Communications and Information Technology, 1(1)**, 31-44.

18. Ouaarab, A., Ahiod, B., & Yang, X. S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. ***Neural Computing and Applications*, *24*(7-8)**, 1659-1669

19. Zheng, H. & Zhou, Y. (2012). A novel cuckoo search optimization algorithm based on Gauss distribution. **Journal of Computational Information Systems, 8(10)**, 4193-4200.

20. Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. **Engineering with computers, 29(1)**, 17-35.

21. Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization, 11(4)**, 341-359.

22. Zelinka, I., & Lampinen, J. (2000). On stagnation of the differential evolution algorithm. **In Proceedings of Mendel, 6th International Mendel Conference on Soft Computing**.

23. Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. ***Soft Computing*, *9*(6)**, 448-462.

24. Qin, A.K., Huang, V.L. & Suganthan, P.N. (2008). Differential evolution algorithm with strategy adaptation for global numerical optimization. **IEEE transactions on Evolutionary Computation, 13(2)**, 398-417.

25. Danaci, M., & Karataş, S. K., (2019). Diferansiyel Gelişim algoritmasında Yeni Yaklaşımlar Kullanılarak Kısıt Yönetim Metotlarının Performanslarının Karşılaştırılması. **Uluslararası erciyes bilimsel araştırmaları kongresi** (pp.806-818). Kayseri, Turkey.

26. Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. **In International symposium on stochastic algorithms** (pp. 169-178). Springer, Berlin, Heidelberg.

27. Yu, S., Yang, S., & Su, S. (2013). Self-adaptive step firefly algorithm. **Journal of Applied Mathematics**, *2013*.

28. Ariyaratne, M. K. A., Fernando, T. G. I., & Weerakoon, S. (2018). A self-tuning firefly algorithm to tune the parameters of ant colony system. **International Journal of Swarm Intelligence, 3(4)**, 309-331.

29. Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. **computer, 27(6)**, 17-26.

30. Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, *4*(2), 65-85.

31. Digalakis, J.G. & Margaritis, K.G. (2002). An experimental study of benchmarking functions for genetic algorithms**. International Journal of Computer Mathematics, 79(4)**, 403-416.

32. Zhong, W., Liu, J., Xue, M., & Jiao, L. (2004). A multiagent genetic algorithm for global numerical optimization. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34(2)**, 1128-1141.

33. Lim, S. P., & Haron, H. (2013, December). Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions. **In 2013 IEEE Conference on Open Systems (ICOS)** (pp. 41-46). IEEE.

34. Lo, K. M., Yi, W. Y., Wong, P. K., Leung, K. S., Leung, Y., & Mak, S. T. (2018). A genetic algorithm with new local operators for multiple traveling salesman problems. **International Journal of Computational Intelligence Systems, 11(1)**, 692-705.

35. Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. **Computers & Structures, 169**, 1-12.

36. Allaoui, M., Ahiod, B., & El Yafrani, M. (2018). A hybrid crow search algorithm for solving the DNA fragment assembly problem. **Expert Systems with Applications, 102**, 44-56.

37. Jain, M., Rani, A. & Singh, V. (2017). An improved Crow Search Algorithm for high-dimensional problems. **Journal of Intelligent & Fuzzy Systems, 33(6)**, 3597-3614.

38. Hassanien, A. E., Rizk-Allah, R. M., & Elhoseny, M. (2018). A hybrid crow search algorithm based on rough searching scheme for solving engineering optimization problems. **Journal of Ambient Intelligence and Humanized Computing**, 1-25.

39. Gupta, D., Sundaram, S., Khanna, A., Hassanien, A. E., & De Albuquerque, V. H. C. (2018). Improved diagnosis of Parkinson's disease using optimized crow search algorithm. **Computers & Electrical Engineering, 68**, 412-424.

40. Mohammadi, F., & Abdi, H. (2018). A modified crow search algorithm (MCSA) for solving economic load dispatch problem. **Applied Soft Computing, 71**, 51-65.

41. Huang, K. W., Girsang, A. S., Wu, Z. X., & Chuang, Y. W. (2019). A Hybrid Crow Search Algorithm for Solving Permutation Flow Shop Scheduling Problems. **Applied Sciences, 9(7)**, 1353.

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

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

44. Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., & Istanda, V. (2012). Bat algorithm inspired algorithm for solving numerical optimization problems. **In *Applied mechanics and materials* (Vol. 148, pp. 134-137)**. Trans Tech Publications Ltd.

45. Gandomi, A. H., Yang, X. S., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. **Neural Computing and Applications, 22(6)**, 1239-1255.

46. Nakamura, R. Y., Pereira, L. A., Costa, K. A., Rodrigues, D., Papa, J. P., & Yang, X. S. (2012, August). BBA: a binary bat algorithm for feature selection. **In 2012 25th SIBGRAPI conference on graphics, Patterns and Images** (pp. 291-297). IEEE.

47. Chakri, A., Khelif, R., Benouaret, M., & Yang, X. S. (2017). New directional bat algorithm for continuous optimization problems. **Expert Systems with Applications, 69**, 159-175.

48. Gupta, D., Arora, J., Agrawal, U., Khanna, A., & de Albuquerque, V. H. C. (2019). Optimized Binary Bat algorithm for classification of white blood cells. **Measurement, 143**, 180-190.

49. De Kort, J. B. (1993). A branch and bound algorithm for symmetric 2-peripatetic salesman problems. **European journal of operational research, 70(2)**, 229-243.

50. Jiang, Y., Weise, T., Lässig, J., Chiong, R., & Athauda, R. (2014, December). Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the tsp. **In 2014 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS)** (pp. 148-155). IEEE.

51. Volgenant, T., & Jonker, R. (1982). A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. **European Journal of Operational Research, 9(1)**, 83-89.

52. Grötschel, M., & Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. **Mathematical Programming, 45(1-3)**, 59-96.

53. Grötschel, M., Jünger, M., & Reinelt, G. (1984). A cutting plane algorithm for the linear ordering problem. **Operations research, 32(6)**, 1195-1220.

54. Konno, H. (1976). A cutting plane algorithm for solving bilinear programs. **Mathematical Programming, 11(1)**, 14-27.

55. Fischetti, M., Salazar González, J. J., & Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. **Operations Research, 45(3)**, 378-394.

56. Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. **Mathematical Programming, 100(2)**, 423-445.

57. Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. **SIAM review, 33(1)**, 60-100.

58. Chauhan, C., Gupta, R., & Pathak, K. (2012). Survey of methods of solving tsp along with its implementation using dynamic programming approach. **International journal of computer applications**, *52*(4).

59. Sahalot, A., & Shrimali, S. (2014). A comparative study of brute force method, nearest neighbour and greedy algorithms to solve the travelling salesman problem. **International Journal of Research in Engineering & Technology, 2(6)**, 59-72.

60. Fok, K. Y., Ganganath, N., Cheng, C. T., & Chi, K. T. (2016, May). A 3D printing path optimizer based on Christofides algorithm. **In 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)** (pp. 1-2). IEEE.

61. Genova, K., & Williamson, D. P. (2017). An experimental evaluation of the best-of-many Christofides' algorithm for the traveling salesman problem. **Algorithmica, 78(4)**, 1109-1130.

62. Caccetta, L., Alameen, M., & Abdul-Niby, M. (2013). An improved clarke and wright algorithm to solve the capacitated vehicle routing problem. **Engineering, Technology & Applied Science Research, 3(2)**, 413-415.

63. Battarra, M., Golden, B., & Vigo, D. (2008). Tuning a parametric Clarke–Wright heuristic via a genetic algorithm. **Journal of the Operational Research Society, 59(11)**, 1568-1572.

64. Karkory, F. A., & Abudalmola, A. A. (2013). Implementation of heuristics for solving travelling salesman problem using nearest neighbour and minimum spanning tree algorithms. **International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering, 7(10)**, 1524-1534.

65. Batista, G. E., & Monard, M. C. (2002). A Study of K-Nearest Neighbour as an Imputation Method. **His, 87(251-260),** 48.

66. Kataria, A., & Singh, M. D. (2013). A review of data classification using k-nearest neighbour algorithm. **International Journal of Emerging Technology and Advanced Engineering, 3(6)**, 354-360.

67. Özcan, S. C., & Kaya, H. (2018, October). An analysis of travelling salesman problem utilizing hill climbing algorithm for a smart city touristic search on OpenStreetMap (OSM). **In 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)** (pp. 1-5). IEEE.

68. Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: Theory and applications* (pp. 7-15). **Springer**, Dordrecht.

69. Wang, C., Lin, M., Zhong, Y., & Zhang, H. (2015). Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling. **International Journal of Computing Science and Mathematics, 6(4)**, 336-353.

70. Gendreau, M., Laporte, G., & Semet, F. (1998). A tabu search heuristic for the undirected selective travelling salesman problem. **European Journal of Operational Research, 106(2-3)**, 539-545.

71. Machado, J. M., Shiyou, Y., Ho, S. L., & Peihong, N. (2001). A common tabu search algorithm for the global optimization of engineering problems. *Computer methods in applied mechanics and engineering*, *190*(26-27), 3501-3510.

72. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. **IEEE computational intelligence magazine, 1(4)**, 28-39.

73. Brezina Jr, I., & Čičková, Z. (2011). Solving the travelling salesman problem using the ant colony optimization. **Management Information Systems, 6(4)**, 10-14.

74. Potvin, J. Y. (1996). Genetic algorithms for the traveling salesman problem. **Annals of Operations Research, 63(3)**, 337-370.

75. Hussain, K., Salleh, M. N. M., Cheng, S., & Naseem, R. (2017). Common benchmark functions for metaheuristic evaluation: A review. **JOIV: International Journal on Informatics Visualization, 1(4-2)**, 218-223.

76. Pasandideh, S.H.R. & Khalilpourazari, S. (2018). Sine cosine crow search algorithm: a powerful hybrid meta heuristic for global optimization. **arXiv preprint arXiv**:1801.08485.

77. Ilavarasi, K., & Joseph, K. S. (2014, February). Variants of travelling salesman problem: A survey. **In International Conference on Information Communication and Embedded Systems (ICICES2014)** (pp. 1-7). IEEE.

78. Rocki, K., & Suda, R. (2012, July). Accelerating 2-opt and 3-opt local search using GPU in the travelling salesman problem. **In 2012 International**

Conference on High Performance Computing & Simulation (HPCS) (pp. 489-495). IEEE.

79. Danacı, M., & AKHDIR, Z., (2019). A Novel Hybrid Crow Search BAT Algorithm For Solving Optimization Problems. **Euroasıa journal of mathematıcs-engıneerıng natural & medıcal scıences, vol.2**, 40-45.

80. Kuzu, S., Önay, O., Şen, U., Tunçer, M., Yıldırım, B., & Keskintürk, T. (2014). Gezgin satıcı problemlerinin metasezgiseller ile çözümü. **İstanbul Üniversitesi İşletme Fakültesi Dergisi, 43(1)**, 1-27.

81. Fu, X., Liu, W., Zhang, B. & Deng, H. (2013). Quantum behaved particle swarm optimization with neighborhood search for numerical optimization. **Mathematical Problems in Engineering**.

82. Arora, S. & Singh, S. (2019), Butterfly optimization algorithm: a novel approach for global optimization. **Soft Computing, 23(3)**, 715-734.

83. benchmarkFcns toolbox. http://benchmarkfcns.xyz,(Erişim tarihi: Temmuz 2020).

84. Kocer, H. E., & Akca, M. R. (2014). An improved artificial bee colony algorithm with local search for traveling salesman problem. **Cybernetics and Systems, 45(8)**, 635-649.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: ZAHER AKHDIR
Uyruğu: Suriye
Doğum Tarihi ve Yeri: 01.01.1990 - Halep
Medeni Durum: Bekar
e-mail: zaher_khdir@yahoo.com

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Lisans	Halep Üniversitesi, Bilgisayar Mühendisliği	2012
Lise	Üstün Öğrenciler Lisesi	2007

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2015-2016	WolvesInteractive	Bilgisayar Müh.

YABANCI DİL

Türkçe

İngilizce

Arapça (Ana dili)

YAYINLAR

1. M. Danacı And Z. Akhdır, "A Novel Hybrid Crow Search BAT Algorithm For Solving Optimization Problems," EUROASIA JOURNAL OF MATHEMATICS-ENGINEERING NATURAL & MEDICAL SCIENCES , vol.2, pp.40-45, 2019