T.C.
BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ TASARIM PROBLEMLERİNİN METASEZGİSEL YÖNTEMLER KULLANILARAK ÇÖZÜMLENMESİ VE PERFORMANS SONUÇLARININ KARŞILAŞTIRILMASI

Oğuzhan DİLBER

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

OCAK 2024

T.C.
BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ TASARIM PROBLEMLERİNİN METASEZGİSEL YÖNTEMLER KULLANILARAK ÇÖZÜMLENMESİ VE PERFORMANS SONUÇLARININ KARŞILAŞTIRILMASI

Oğuzhan DİLBER

DANIŞMAN
Dr. Öğr. Üyesi Serhat KILIÇARSLAN

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

OCAK 2024

ONAY

Oğuzhan DİLBER tarafından hazırlanan "**Makine Mühendisliği Tasarım Problemlerinin Metasezgisel Yöntemler Kullanılarak Çözümlenmesi Ve Performans Sonuçlarının Karşılaştırılması**" adlı tez çalışması 29/01/2024 tarihinde yapılan sınavla aşağıdaki jüri tarafından oybirliği ile Bandırma Onyedi Eylül Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri		**İmza**
Dr. Öğr. Üyesi Serhat KILIÇARSLAN (Danışman)	(BANÜ)	_____
Doç. Dr. Üyesi Abdullah ELEN (Üye)	(BANÜ)	_____
Dr. Öğr. Üyesi Mehmet Akif BÜLBÜL (Üye)	(NEVÜ)	_____

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …/…/…gün ve …/… sayılı kararı ile onaylanmıştır.

Doç. Dr. Abdullah YEŞİL

Enstitü Müdürü

BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS ÇALIŞMASI
ETİK BEYANI

Bandırma Onyedi Eylül Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre hazırlamış olduğum "**Makine Mühendisliği Tasarım Problemlerinin Metasezgisel Yöntemler Kullanılarak Çözümlenmesi Ve Performans Sonuçlarının Karşılaştırılması** " adlı tezimin özgün bir çalışma olduğunu, tez hazırlanırken tüm aşamalarda bilimsel etik ilkelerine uygun davrandığımı, tez kapsamında sunulan tüm verileri bilimsel etik ilkelerine uygun elde ettiğimi, tezde faydalandığım tüm eserlere atıf yaptığımı ve kaynaklar kısmında bu eserleri gösterdiğimi beyan ederim. 29/01/2024

Oğuzhan DİLBER
İmza

ÖNSÖZ

Teknolojilerin sürekli gelişmesiyle birlikte mühendislik süreçleri, sürekli optimizasyon gerektiren daha karmaşık ve sofistike bir hale dönüşmektedir. Optimizasyonlar, tasarımdan üretim aşamasına kadar tüm mühendislik aşamalarında gereklidir. Geleneksel optimizasyon yöntemi, uzmanlar tarafından yapılan deneme yanılma testleri ile bilinmektedir. Ancak, bu gelişmelerle birlikte optimizasyon yöntemleri de ilerlemektedir. Buna ilaveten, bilgisayar teknolojilerinin gelişmesiyle birlikte optimizasyon işlemleri giderek daha fazla bilgisayarlar tarafından yapılmakta ve kararlar bilgisayar sistemlerinin katkısıyla verilmektedir.

Bu tez çalışmasında, mühendislik tasarım problemlerinin optimal çözümü için literatürde bilinen başarılı metasezgisel algoritmalar kullanıldı. Bu algoritmalar, farklı yineleme ve popülasyon sayılarına göre performanslarını karşılaştırmak amacıyla altı deney gerçekleştirildi. Ayrıca, algoritmların problem çözümündeki harcadıkları sürelerde ölçülerek, elde edilen tüm sonuçlar birlikte değerlendirildi.

Tez çalışması sürecinde, benden yardımlarını ve desteklerini esirgemeyen danışman hocam Sayın Dr. Öğr. Üyesi Serhat Kılıçarslan'a, teknik konularda deneyimlerinden faydalandığımız Doç. Dr. Abdullah Elen'e ve Dr. Öğr. Üyesi Cemil Közkurt'a teşekkürü borç bilirim.

İyi ve kötü günümde her zaman yanımda olan eşim Süheyla'ya, yüksek lisans öğrenim süresince beraber yol yürüdüğümüz arkadaşım Samet Panda'ya teşekkürler ederim. Son olarak, bugünlere gelmemde üzerimde büyük emeği olan rahmetli babaannem Nezihe Dilber'e Allah'tan rahmet dileyerek sonsuz şükranlarımı sunarım.

ÖZET

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ TASARIM PROBLEMLERİNİN METASEZGİSEL YÖNTEMLER KULLANILARAK ÇÖZÜMLENMESİ VE PERFORMANS SONUÇLARININ KARŞILAŞTIRILMASI

Oğuzhan DİLBER

Bandırma Onyedi Eylül Üniversitesi
Fen Bilimleri Enstitüsü
Mekatronik Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Serhat KILIÇARSLAN
Ocak 2024, 67 sayfa

Bir sistem veya süreç gibi bir şeyin mümkün olduğunca etkili ve verimli çalışmasını sağlama sürecine optimizasyon denir. Optimizasyon, maliyetleri azaltabilir, verimliliği artırabilir ve çıktıların kalitesini artırabilirken aynı zamanda bir sistemin, sürecin veya modelin performansını artırmaya da yardımcı olabilmesi bakımından önemlidir. Makine mühendisliğinin tasarım problemleri, geometrik, kinematik koşullar ve malzeme direnci üzerindeki eşitlik ya da eşitsizlikler ve genellikle doğrusal olmayan kısıtlamalar gibi farklı türdeki hedeflerden ve değişen zorluk düzeylerinden meydan gelmektedir. Bu tür problemleri çözmek için metasezgisel adı verilen özel bir algoritma türü kullanılmaktadır. Bu algoritmalar, özellikle bilgilerin yetersiz olduğu veya bilgisayar kapasitesinin kısıtlı olduğu durumlardaki optimizasyon problemleri için yeterince iyi bir çözüm bulmayı amaçlar.

Bu tez çalışmasında, makine mühendisliği tasarım problemlerinin çözülmesi amacıyla literatürde iyi bilinen Ateşböceği Algoritması, Gri Kurt Optimizasyon Algoritması, Yol Bulucu Algoritması ve Parçaçık Sürü Optimizasyonu algoritması kullanılmıştır. Bu algoritmalar, yineleme sayısı ve popülasyon sayısına bağlı olarak altı farklı senaryoda gerçek-dünya problemleri (Dört

Kademeli Dişli Kutusu Problemi, Gaz İletim Kompresörü Tasarımı, Dişli Takımı Tasarım Problemi, Himmelblau Fonksiyonu, Hidrostatik Baskı Yatağı Tasarımı Problemi, Çoklu Disk Kavramalı Fren Tasarım Problemi, Endüstriyel Soğutma Sisteminin Optimal Tasarımı ve Planet Dişli Zinciri Tasarım Problemi) ile test edilmiş ve elde edilen sonuçlara göre, metasezgisel algoritmaların performansları birbiriyle karşılaştırılmıştır.

Anahtar kelimeler: Optimizasyon, Metasezgiseller, Mühendislik Tasarım Problemleri.

ABSTRACT

M.Sc. THESIS

SOLVING MECHANICAL ENGINEERING DESIGN PROBLEMS USING METAHEURISTIC METHODS AND COMPARING PERFORMANCE RESULTS

Oğuzhan DİLBER

Bandırma Onyedi Eylül University
Graduate School of Natural and Applied Sciences
Department of Mechatronics Engineering

Supervisor: Assist. Prof. Dr. Serhat KILIÇARSLAN
January 2024, 67 pages

The process of ensuring that something, such as a system or process, works as effectively and efficiently as possible is called optimization. Optimization is important because it can reduce costs, increase efficiency, and improve the quality of outputs, while also helping to improve the performance of a system, process, or model. Design problems in mechanical engineering arise from different types of objectives and varying levels of difficulty, such as equations or inequalities and often non-linear constraints on geometric, kinematic conditions and material resistance. A special type of algorithm called metaheuristics is used to solve such problems. These algorithms aim to find a good enough solution for optimization problems, especially when information is insufficient or computer capacity is limited.

In this thesis study, Firefly Algorithm, Gray Wolf Optimization Algorithm, Pathfinder Algorithm and Particle Swarm Optimization algorithm, which are well known in the literature, were used to solve mechanical engineering design problems. These algorithms solve real-world problems in six different scenarios depending on the number of iterations and the number of population (Four-Stage Gearbox Problem, Gas Transmission Compressor Design, Gear Train Design Problem, Himmelblau Function, Hydrostatic Thrust Bearing Design Problem, Multiple Disc Clutch Brake Design Problem, Optimal Design of Industrial Cooling System and Planetary

Gear Chain Design Problem) and according to the results obtained, the performances of the metaheuristic algorithms were compared with each other.

Keywords: Optimization, Metaheuristics, Engineering Design Problems.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Şekil **Sayfa**

TABLO LİSTESİ

Tablo **Sayfa**

SİMGE VE KISALTMA LİSTESİ

Simgeler	**Açıklama**
σ	: Işık soğurma katsayısı
v	: Ateşböceğinin çekicilik değeri
x	: Gauss dağılımından alınan rastgele sayıların bir vektörü
ω	: Atalet momenti

Kısaltmalar	**Açıklama**
ABO	: Afrika Manda Optimizasyonu (African Buffalo Optimization)
ACO	: Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
AOA	: Aritmetik Optimizasyon Algoritması (Arithmetic Optimization Algorithm)
CS	: Gugukkuşu Arama Algoritması (Cuckoo Search)
ÇDKFTP	: Çoklu Disk Kavramalı Fren Tasarım Problemi
DKDKP	: Dört Kademeli Dişli Kutusu Problemi
DTTP	: Dişli Takımı Tasarım Problemi
ESSOT	: Endüstriyel Soğutma Sisteminin Optimal Tasarımı
FA	: Ateşböceği Algoritması (Firefly Algorithm)
FPA	: Çiçek Tozlaşma Algoritması (Flower Pollination Algorithm)
GA	: Genetik Algoritma (Genetic Algorithm)
GİKP	: Gaz İletim Kompresörü Tasarımı
GSA	: Yerçekimi Arama Algoritması (Gravitational Search Algorithm)
GWO	: Gri Kurt Optimizasyon Algoritması (Grey Wolf Optimizer)
HBYTP	: Hidrostatik Baskı Yatağı Tasarımı Problemi
HF	: Himmelblau'nun Fonksiyonu
HS	: Armoni Arama Algoritması (Harmony Search)
ICA	: Emperyalist Rekabetçi Algoritması (Imperialist Competitive Algorithm)
IWD	: Akıllı Su Damlaları (Intelligent Water Drops)
MOSS	: Çoklu Ortogonal Arama Stratejisi (multi-orthogonal search strategy)
PDZTP	: Planet Dişli Zinciri Tasarım Problemi
PFA	: Yol Bulucu Algoritması (Pathfinder Algorithm)
PSO	: Parçaçık Sürü Optimizasyonu (Particle Swarm Optimization)

1. GİRİŞ

Optimizasyon, belirli koşullar altında en iyi sonucu veya faydayı elde etme sürecidir. Tasarıma yönelik optimizasyona dayalı yaklaşımlar, operasyonel bir ihtiyacı önceden belirlenmiş bir performans düzeyi sağlayacak bir sisteme dönüştürmek için resmi bir yapı sağlama ve tasarlama açısından arzu edilir [1]. Aslında, mühendislik ürününün yaşam döngüsünün tüm aşamaları boyunca, mühendislik ve iş kararları, bir hedefi veya faydayı en üst düzeye çıkarma/en aza indirme, aynı zamanda sınırlı kaynakların kullanımını dengeleme veya kontrol etme ve sıfır veya daha fazla kısıtlamayı karşılama nihai hedefiyle alınır [2].

Optimizasyon, aday çözümler kümesi arasından "en iyi" çözümün seçimini içerir. Çözümün iyilik derecesi, minimuma indirilecek veya maksimuma çıkarılacak bir amaç fonksiyonu kullanılarak ölçülür [3]. Arama süreci, sistem modeline ve kısıtlamalar olarak adlandırılan kısıtlamalara tabi olarak gerçekleştirilir. Dolayısıyla optimizasyonun amacı, bir dizi kısıtlamaya tabi olarak bir amaç fonksiyonun değerini maksimuma çıkarmak veya en aza indirmektir [4]. Bu kısıtlar, eşitlik ve eşitsizlik ifadeleri şeklindedir. Eşitlik kısıtlamalarına örnek olarak malzeme ve enerji dengeleri, süreç modelleme denklemleri ve termodinamik gereksinimler verilebilir. Öte yandan, eşitsizlik kısıtlamalarına örnek olarak basınç, sıcaklık veya akış hızı belirli değerleri aşmamalıdır veya sistemin durumu termodinamiğin ikinci yasasını ihlal edemez şeklinde ifade edilebilir [5].

Genel olarak optimizasyon yöntemleri deterministik ve stokastik yaklaşımlara ayrılır. Deterministik tabanlı teknikler, hızlı yakınsama oranlarıyla yüksek doğrulukta çözümler bulabilen gradyan tabanlı yaklaşımlardır [6]. Bu yöntemler, yinelemeli sürecin sırasıyla arama yönünü ve adım boyutunu belirlemek için sürekli amaç fonksiyonlarına ve bunların gradyan bilgilerine ihtiyaç duyar. Ancak birçok karmaşık mühendislik problemi için sürekli amaç fonksiyonunu tanımlamak çok zor, hatta imkânsızdır. Ayrıca çoğu durumda amaç fonksiyonunun gradyan bilgisini elde etmek maliyetli bir süreçtir. Öte yandan algoritmanın başlangıç noktası bu algoritmalar için oldukça önemlidir. Bu algoritmalar arama alanındaki uygun konumlardan başlatılmazsa, kolaylıkla yerel minimumlara takılıp kalabilirler [2].

Mühendislik disiplinlerindeki optimizasyon problemlerinin birçoğu dışbükey olmayan bir arama alanına sahiptir. Bu tür problemlerde uygun bir çözüm bulmak için çeşitli kısıtlamalar ve sınır koşulları dikkate alınmalıdır [7]. Bu nedenle, gradyan tabanlı teknikler bu tür problemleri kolayca çözemez. Diğer yandan, stokastik yaklaşımlar bu türdeki problemlerin çözümü için Sürekli, türevlenebilir amaç fonksiyonu gerektirmeyen ve gradyan içermeyen yöntemler gibi bir

grup alternatif teknik sunmaktadır. Bu grubun en önemli yöntemlerinden biri metasezgisel algoritmalardır [2]. Metasezgiseller genellikle doğadan ilham alan çok çeşitli yöntemlerdir. Her metasezgiselin rastgele bir bileşeni vardır ve bu nedenle olasılıksal yöntemler olarak da bilinirler [8-11].

Bu tez çalışmasında, Makine Mühendisliği tasarım problemlerinin literatürdeki iyi bilinen bazı meta-sezgisel optimizasyon algoritmaları ile çözümlenmesi ve sonuçlarının karşılaştırılması amaçlanmıştır. Tez kapsamında ele alınan algoritmalar şunlardır: Ateşböceği Algoritması, Gri Kurt Optimizasyon Algoritması, Yol Bulucu Algoritması ve Parçaçık Sürü Optimizasyonu. Deneysel çalışmalarda bu algoritmalar, Dört Kademeli Dişli Kutusu Problemi, Gaz İletim Kompresörü Tasarımı, Dişli Takımı Tasarım Problemi, Himmelblau Fonksiyonu, Hidrostatik Baskı Yatağı Tasarımı Problemi, Çoklu Disk Kavramalı Fren Tasarım Problemi, Endüstriyel Soğutma Sisteminin Optimal Tasarımı ve Planet Dişli Zinciri Tasarım Problemi olmak üzere sekiz gerçek-dünya problemi üzerinde test edilmiştir.

1.1 Tezin Amacı ve Katkısı

Bu tez çalışmasının amacı, gerçek dünya problemlerinin optimal çözümü için literatürde iyi bilinen metasezgisel optimizasyon algoritmalarını kullanarak, ve bu algoritmaların yakınsama hızı ve çalışma süreleri bakımından karşılatırışmasını içermektedir. Deneysel çalışmalarla elde edilen sonuçların, bu tür problemler üzerine yapılan bilimsel çalışmalarda model seçimi için bir fikir vereceği düşünülmektedir.

1.2 Tezin Organizasyonu

Bu tez çalışmasının organizasyonu aşağıdaki gibi maddeler halinde belirtilmiştir.

- Giriş bölümünde, tez çalışmasının yapılmasınıdaki gerekçe, kullanılan yöntemlerden ve gerçek dünya problemlerinden bahsedilmiş ve bu tez çalışması kapsamında yapılan araştırmanın literatüre katkıları sunulmuş ve tezin takip eden bölümleri tanıtılmıştır.
- İkinci bölümde, metasezgisel optimizasyon teknikleri ve gerçek-dünya problemlerinin çözümünde kullanılan algoritmalar konusunda kapsamlı bir literatür bilgisi verilmiştir.
- Üçüncü bölümde, tez çalışmasında kullanılan optimizasyon algoritmaları, esin kaynakları ve çalışma prsensipleri ayrıntılı bir şekilde tanıtılmış ve bu metasezgisel yöntemlerin akış diyagramları verilmiştir.

- Dördüncü bölümde, algoritmaların performans değerlendirmesi amacıyla kullanılan sekiz gerçek dünya probleminin tanıtımına, uygunluk fonksiyonlarına ve bu problemlerin kısıt tanımlamalarına yer verilmiştir.
- Beşinci bölümde, metasezgisel algoritmaların gerçek dünya mühendislik problemlerinde, yineleme ve popülasyon değerlerinin farklı durumlarına göre altı deney gerçekleştirilmiş ve detaylı olarak sonuçlarından bahsedilmiştir.
- Altıncı ve son bölümde ise tez çalışması kapsamında gerçekleştirilen deneylerden elde edilen sonuçlar analiz edilmiş ve mühendislik problemlerinin çözümü için en uygun olan yöntemin hangisi olduğundan bahsedilmiştir.

2. LİTERATÜR ARAŞTIRMASI

Metasezgisel, bilgisayar bilimi ve uygulamalı matematik alanındaki algoritmalar ve hesaplama karmaşıklığı teorisi ile ilgili bir optimizasyon dalıdır. Karmaşık problemlere çözümler bulmak için kısa sürede uygun sonuçlar verir. Metasezgisel algoritmalar, karmaşık problemleri başarıyla çözebildiği için akademik araştırmalardan mühendislik uygulamalarına kadar birçok alanda ilgi odağı haline gelmiştir. Makul bir sürede çözülebilen optimizasyon problemlerinin boyutu ve karmaşıklığı, modern bilgi işlem teknolojilerinin ortaya çıkmasıyla birlikte ilerlemiştir. Bu teknolojiler, optimizasyon problemlerini çözmek için yeni yöntemlerin geliştirilmesini teşvik etmiştir.

Karınca kolonisi optimizasyonu (*Ant Colony Optimization,* ACO) [12] algoritması, karıncaların yiyecek arama davranışını modeller ve hedef olarak en kısa yolu bulmayı gerektiren problemler için kullanışlıdır. Gerçek dünyada karıncalar çevrelerini keşfettiklerinde birbirlerini kaynaklara yönlendirmek için feromonlar salgılarlar. ACO ayrıca bu yöntemi simüle eder ve her karınca konumunu benzer şekilde kaydeder, böylece daha sonraki yinelemelerde daha fazla karınca daha iyi çözümler bulur. Bu eğilim en iyi yol bulunana kadar devam eder. Geem ve ark. [13] tarafından geliştirilen Armoni Arama (*Harmony Search,* HS), müzikten ilham alan bir optimizasyon algoritmasıdır. Tasarım birleşimleri, çözüm olarak bir dizi armoni halinde kodlanır ve tasarım değişkenleri kümesi, çözüm aralığının farklı bir bölümünde olabilir veya optimum sonuçlar, mevcut çözümlere yakın olabilir. Tasarım çözümleri yakınlardaysa bu etkili olabilir. Shah-Hosseini [14] doğal nehirlerdeki su damlacıklarının davranışından esinlenerek Akıllı Su Damlaları (*Intelligent Water Drops*, IWD) algoritmasını önerdi. Nehirlerde su damlaları, kendi aralarında ve nehir yataklarında meydana gelen etki ve tepkimeler yoluyla hedeflerine ulaşmak için neredeyse en uygun yolu bulur. IWD algoritması bu eğilimi simüle eder ve bir problemde optimum çözümü bulmak için yapıcı yaklaşımı kullanır. Her yapay su damlası, problemin arama uzayında ilerleyerek ve çevresini değiştirerek bir çözüm (yol) oluşturur. Tüm bu yollar arasından optimum veya optimuma yakın olan yol algoritma tarafından seçilir. Atashpaz-Gargari ve Lucas [15] 2007 yılında Emperyalist Rekabetçi Algoritmasını (*Imperialist Competitive Algorithm,* ICA) önerdiler. ICA, çeşitli optimizasyon problemlerini çözmeye yönelik sosyo-politik açıdan küresel bir arama stratejisidir. Algoritma, imparatorluklar arasındaki rekabete dayanarak en iyi çözümü elde etmeyi amaçlamaktadır. En iyi çözüm, en iyi güce sahip emperyalisttir. Yang ve Deb [16] tarafından geliştirilen Gugukkuşu Arama (*Cuckoo Search,* CS) algoritması, popülasyona dayalı başka bir algoritmadır. Bazı guguk kuşu türlerinin yavru parazitliğinden esinlenilmiştir.

Yumurtalar ve yuvalar çözüm olarak kodlanarak en iyi yuvadaki kaliteli yumurtalar bir sonraki nesle aktarılacaktır. Arama alanını daha etkili bir şekilde keşfetmek için Lévy uçuşları kullanılarak yeni çözümler/yumurtalar üretilir. Rashedi ve diğ. [17] tarafından Yerçekimi Arama Algoritması (*Gravitational Search Algorithm,* GSA) adını verdikleri, Newton yerçekimi yasasına ve kütle etkileşimlerine dayanarak bir metasezgisel algoritma önerildi. Algoritmada, gerçek dünyadaki nesneler etmen olarak kabul edilir ve performansları, uygunluk fonksiyonu değerlerine bağlı olan kütleleri ile ölçülür. Her aracının arama uzayındaki konumu bir problemin çözümünü gösterir. En ağır kütle, arama uzayında optimum çözümü sunar. Zaman geçtikçe kütleler en ağır kütle tarafından çekilir ve en iyi çözüme yakınsar. Çiçek Tozlaşma Algoritması (*Flower Pollination Algorithm,* FPA), Yang [18] tarafından geliştirilen ve çiçekli bitkilerin tozlaşma sürecinin özelliklerinden esinlenen yeni bir meta-sezgisel algoritmadır. Tozlaşma iki şekilde yapılabilir. Polen böcekler, kuşlar, yarasalar, diğer hayvanlar veya rüzgâr gibi polen taşıyıcılar tarafından aktarılabilir. Bazı çiçek türleri kendi kendine tozlaşmayı kullanır. Tozlayıcılar tarafından yapılan tozlaşmaya çapraz tozlaşma denir ve küresel bir tozlaşma süreci olarak düşünülebilir. Yerel tozlaşma süreci kendi kendine tozlaşma ile üretilir. Çapraz tozlaşma, kendi kendine tozlaşma, çiçek sabitliği ve geçiş olasılığının birleşimi, çözümleri kodlamak ve böylece optimizasyon problemlerini etkili bir şekilde çözmek için kullanılabilir. Aslan algoritması (*Lion's Algorithm,* LA) [19], aslanların sosyal sistemi ve işbirliği özelliklerinden ilham alan popülasyon tabanlı bir algoritmadır. LO, aslanın iki benzersiz davranışı temel alınarak modellenmiştir: bölgesel savunma ve ele geçirme. Bu iki davranışa dayanarak, LO'nun çözümleri üç adımda üretilir: Birinci adımda, her bir cub çözümünün orijinal mi yoksa türetilmiş bir çözüm mü olduğunu ayırt etmektir. İkinci adımda, bölgesel savunma, mevcut ve yeni çözümleri değerlendirmeye ve karşılaştırmaya devam edecektir. Üçüncü ve son adımda, eğer mevcut çözüm yeni çözümden daha iyiyse, o zaman bölgesel devralma mevcut çözümü daha da geliştirmek için koruyacaktır. Afrika manda optimizasyonu (*African Buffalo Optimization,* ABO) algoritması [20], Afrika mandalarının geniş Afrika ormanlarında ve savanlarında mera bulma uygulamalarından ilham almıştır. ABO algoritmasını modellemek için bu hayvanların üç spesifik özelliği kullanılır. Öncelikle bu hayvanların hafıza kapasiteleri yüksektir. Bu beceri, Afrika kıtasında binlerce kilometreye kadar olan rotalarını takip edebilmelerini sağlıyor. İkinci olarak, hayatta kalma konusunda birbirlerini desteklemek için iki özel seslendirme (“maa” ve “waa”) kullanarak kendi aralarında iletişim kurarlar. Son olarak Afrika mandaları çoğunluğa göre karar alır. ChOA algoritmaları, Khishe ve Mosavi [21] tarafından 2020 yılında tanıtılan yeni meta-sezgisel algoritmalarından biridir. ChOA, şempanzelerin grup avlanma hareketlerinden ve cinsel motivasyonlarından ilham almaktadır.

ChOA'da optimizasyon probleminde optimal çözüme ulaşmak için av avcılığından yararlanılır. ChOA avlanmayı dört ana aşamaya ayırır: sürme, engelleme, kovalama ve saldırma. İlkinde ChOA, rastgele bir şempanze popülasyonunun oluşturulmasıyla başlatılıyor. Şempanzeler daha sonra rastgele dört gruba ayrılır: Saldırgan (attacker), kovalayan (chaser), bariyer (barrier) ve sürücü (driver). Abualigah ve ark. [22] dört matematiksel aritmetik operatörden ilham alan Aritmetik Optimizasyon Algoritması (*Arithmetic Optimization Algorithm*, AOA) adlı bir meta-sezgisel yaklaşımı tanıttılar. Algoritmanın yerel ve genel arama yeteneğini geliştirmek için dört aritmetik işlem (Çarpma, Bölme, Çıkarma ve Toplama) kullanılır. Çarpma ve bölme işlemleri, arama uzayında yeni noktalar keşfetmek için kullanılmış ve algoritmanın keşfetme yeteneği gelişmiştir. Arama uzayında yerel noktaları bulmak için çıkarma ve toplama işlemleri kullanılmış olup, algoritmanın kullanım yeteneği arttırılmıştır.

Gerçek dünyadaki mühendislik tasarım problemleri, hem iş dünyasında hem de akademik dünyada birçok araştırma alanında yaygın olarak ele alınmaktadır. Bu tür problemler çok sayıda optimizasyon algoritması tarafından irdelenmiştir. Ancak problemlerin karmaşıklığı arttıkça algoritmanın performansı önemli ölçüde düşmektedir. Literatürde bulunan mühendislik tasarım problemlerini etkili bir şekilde çözmek amacıyla farklı optimizasyon teknikleri ortaya konmuştur. Bunlardan bazıları şöyledir; Rizk [23] tarafından mühendislik tasarım problemlerini çözmek için Çoklu Ortogonal Sinüs Kosinüs Algoritması (*multi-orthogonal sine cosine algorithm,* MOSCA) olarak bilinen yeni bir yöntem sunulmuştur. SCA'nın dezavantajlarını azaltmak için önerilen MOSCA, Çoklu Ortogonal Arama Stratejisi (*multi-orthogonal search strategy,* MOSS) ve SCA'nın avantajlarını birleştirir. Önerilen MOSCA'nın iki aşaması vardır. İlk aşamada, araştırma becerilerini geliştirmek için keşif yaklaşımı kullanılır. İkinci aşamada ise daha önce sömürü eğilimlerini arttırdığı tespit edilen SCA, MOSS yöntemi için araştırmanın ilk hedefidir. MOSCA algoritmasını test etmek için dört mühendislik tasarım problemi ve çeşitli kıyaslama problemleri kullanılmıştır. Sonuç olarak, MOSCA'nın sıklıkla diğer yaklaşımlardan daha iyi performans gösteren kullanışlı bir strateji olduğunu göstermektedir. Pauline ve diğ. [24] tarafından uyarlanabilir bir guguk kuşu arama algoritması (*adaptive cuckoo search algorithm,* ACSA) adı verilen bir yöntem önerildi. ACSA, çeşitlendirme mekanizmasında optimize edilmiş bir faz boyutu seçim stratejisi kullanır. Yakınsama özelliğini koruyarak bu yaklaşım, CSA içerisinde yoğunlaştırma ve çeşitlendirme verimliliği arasında bir denge kurar. ACSA'nın yapısal optimizasyon problemlerini çözmedeki etkinliğini göstermek için üç yapısal mühendislik problemi kullanılmıştır. ACSA'nın performans değerlendirmesine göre çeşitli vaka çalışmalarında CSA standardını ve diğer literatür yaklaşımlarını uyguladığını ortaya koymaktadır. Francisco ve diğ.

[25], ateşböceği algoritmasını kullanarak kısıtlı küresel optimizasyon problemlerine dinamik bir ceza yaklaşımının nasıl uygulanacağını araştırdılar. Önerilen yöntemin uygulanabilirliği ve etkinliği, birkaç kıyaslamalı mühendislik tasarım optimizasyon problemi kullanılarak değerlendirilmiştir. Basak ve diğ. [26], Genetik Algoritma (GA) kullanarak, mühendislik tasarım problemlerini optimize etmek ve algoritmanın ideal çözüme doğru ve etkili bir şekilde nasıl yaklaştığını incelemek için bir çalışma gerçekleştirdi. Önerilen çalışmada, tasarım değişkenleri için bir matematiksel ifade seçilir ve optimize etmek için GA kullanılır. Küresel optimizasyon ve mühendislik tasarımı sorunlarının üstesinden gelmek amacıyla, Nadimi-Shahraki ve diğ. [27] tarafından Geliştirilmiş Gri Kurt Optimizasyonu (*Improved Gray Wolf Optimizer,* I-GWO) adlı bir yöntem önerilmiştir. I-GWO tekniği yeni bir hareket stratejisinden yararlanır. Komşular arasında bilgi paylaşımından başlayarak her kurdun komşusunu tanımak için değiştirilmiş bir yaklaşım kullanmaktadır. I-GWO keşif yaklaşımında kullanılan bu öğrenme stratejisi, çeşitliliği korurken yerel ve küresel aramayı dengeler. Önerilen yöntem CEC2018 kıyaslama veri setinde test edilmiş ve diğer tekniklerle karşılaştırılmıştır. Diferansiyel Evrim (DE) algoritmasının Ebeveyn Merkezli Çaprazlamalı DE (*Parent Centric Crossover*, DEPCX) ve olasılıksal Ebeveyn Merkezli Çaprazlamalı DE (*probabilistic Parent Centric Crossover*, Pro. DEPCX) olarak adlandırılan iki değiştirilmiş sürümü Ali ve diğ. [28] tarafından önerilmiştir. Mutasyon aşamasında ebeveyn merkezli mekanizmanın çaprazlama operatörü olarak uygulanması, bu iki değiştirilmiş sürüm arasındaki temel farktır. Test için çeşitli doğrusal olmayan (*non-linear*) mühendislik tasarım problemleri kullanıldığı ve deneylerde elde edilen sonuçların geleneksel DE'ye göre gelişme gösterdiğini bildirmişlerdir. Melo ve Carosio [29], temel ceza fonksiyonuna sahip değiştirilmiş bir DE kullanarak kısıtlı mühendislik problemlerini çözmek amacıyla beş farklı mutasyon stratejisi kullandılar. Bu teknikler, kısıtlamaları yönetmek için basit bir stratejiyle birlikte kullanılır. Deneysel çalışmada, basınçlı kap tasarımı ve çekme/bastırma yayı ağırlığının azaltılması problemleri test edildi. Ayrıca, birçok popüler algoritmayla karşılaştırıldığında, standart sapmalar açısından dikkate değer bir performansa ulaştıklarını bildirmişlerdir.

Literatür incelemesi sonucunda, gerçek dünya problemlerinin çözümü karmaşık hesaplar gerektirdiği, ancak metasezgisel optimizasyon algoritmaları aracılığıyla bu sorunların kolayca aşılabildiği görülmüştür. Bu yönüyle metasezgiseller, sosyal bilimlerden mühendislik bilimlerine kadar birçok alanda popüler hale gelmiştir.

3. METASEZGİSEL ALGORİTMALAR

Metasezgisel algoritmalar, özellikle optimizasyon problemlerinin karmaşık çözümü için kullanılan hesaplamalı tasarlanmış bir arama prosedürüdür. Sınırlı kaynakların olduğu gerçek dünyada kusurlu veya eksik bilgiye dayalı olarak ideal bir çözüm bulmak zorunludur. Bu tür optimizasyon problemlerini çözmek için metasezgisel yöntemlerin ortaya çıkışı, mühendislik alanındaki tasarım sorunlarının başarıyla çözülmesine olanak sağlamıştır.

Mevcut geleneksel yaklaşımlara göre daha iyi çözümler geliştirmeye dikkat edilmesi gereken zorluklar var. Doğrusal ve dışbükey olmayan optimizasyon problemlerini çözmek için çeşitli uygulamalara oldukça kapsamlı olan yazarlar tarafından farklı metasezgisel algoritmalar tanımlanmaktadır. Kombinatoryal optimizasyonda, NP-hard kategorisindeki bazı problemleri çözmek imkânsızdır. Bu nedenle, metasezgisel algoritmalar genellikle klasik optimizasyon algoritmalarına, yinelemeli yöntemlere ve basit açgözlü buluşsal yöntemlere göre daha az hesaplama çabasıyla iyi çözümler bulabilir [30-33].

Metasezgisel algoritmalar farklı alanlarda önemli bir rol oynayabilir. Esas itibarıyla pek çok optimizasyon problemi, doğrusal olmayan kısıtlamalara sahip çok amaçlı fonksiyonlardır. Örneğin, mühendislik optimizasyon problemlerinin çoğu oldukça doğrusal değildir ve çok amaçlı problemlere çözüm gerektirir. Öte yandan, yapay zekâ ve makine öğrenimi problemleri önemli ölçüde büyük veri kümelerine dayanır. Bu nedenle, optimizasyon problemini formüle etmek zordur. Metasezgisel yöntemler bu yönüyle, geleneksel optimizasyon yöntemleri kullanılarak çözülmesi zor olan pratik problemlerin çözümünde önemli bir rol oynamaktadır [34-36].

Metasezgisel yöntemler, doğadan ilham alan ve doğadan ilham almayan, popülasyon bazlı veya tek noktalı arama, dinamik ve statik amaç fonksiyonları, bir ve çeşitli komşuluk yapıları, bellek kullanımlı ve belleksiz yöntemler gibi arama alanı üzerinde nasıl çalıştıklarına göre sınıflandırılır [37, 38].

Bu çalışmada, mühendislik problemlerinin çözümünde test etmek amacıyla doğadan ilham alınan metasezgizel yöntemlerden Ateşböceği Algoritması, Gri Kurt Optimizasyon Algoritması, Yol Bulucu Algoritması ve Parçacık Sürü Optimizasyonu kullanıldı. Bu yöntemlere ait detaylı açıklamalar aşağıda verilmiştir.

3.1 ATEŞBÖCEĞİ ALGORİTMASI

Ateşböceği Algoritması (*Firefly Algorithm,* FA), 2008 yılında Yang [39] tarafından önerilmiştir. FA, ateşböceklerinin çiftleşmek amacıyla veya potansiyel yırtıcıları uyarmak için yanıp sönen ışığı kullanmalarını esas almaktadır. Açıkçası, yanıp sönen ışık ve yoğunluğu, fiziksel yasalar da dâhil olmak üzere bazı kurallara uyabilir. Yang, FA'nın arama modelini oluşturmak için aşağıda belirtilen kural önermiştir [40]:

- Tüm ateşböcekleri ünisekstir; bu, herhangi bir ateşböceğinin diğer tüm parlak ateşböcekleri tarafından çekilebileceği anlamına gelir.
- Ateşböceklerinin çekicilikleri, ışık yoğunluklarıyla orantılıdır ve uzaklık arttıkça her ikisi de azalır. Böylece, yanıp sönen herhangi iki ateşböceği için, parlaklık değeri az olan, daha parlak olan böceğe doğru hareket edecektir. Belirli bir ateşböceğinden daha parlak olanı bulunmazsa rastgele şekilde hareket edecektir.
- Bir ateşböceğinin ışık yoğunluğu, uygunluk fonksiyonunun peyzajına göre belirlenir.

Ateşböceği algoritmasında, ışık yoğunluğunun değişimi ve çekiciliğin formülasyonu olmak üzere iki önemli konu vardır. Basitlik açısından, bir ateş böceğinin çekiciliğinin, kodlanmış amaç fonksiyonuyla ilişkili olan parlaklığıyla belirlendiğini her zaman varsayabiliriz [41]. Kaynakla olan mesafe arttıkça ışık yoğunluğu ve dolayısıyla çekiciliği azaldığından, ışık şiddeti ve çekiciliğindeki değişimler monoton olarak azalan işlevler olmalıdır. Çoğu uygulamada, hem ters kare yasasının hem de soğurulmanın birleşik etkisi, Eşitlik 3.1'deki gibi Gauss formu kullanılarak yaklaşık olarak hesaplanabilir:

$$I(r) = I_0 e^{-\gamma r^2} \tag{3.1}$$

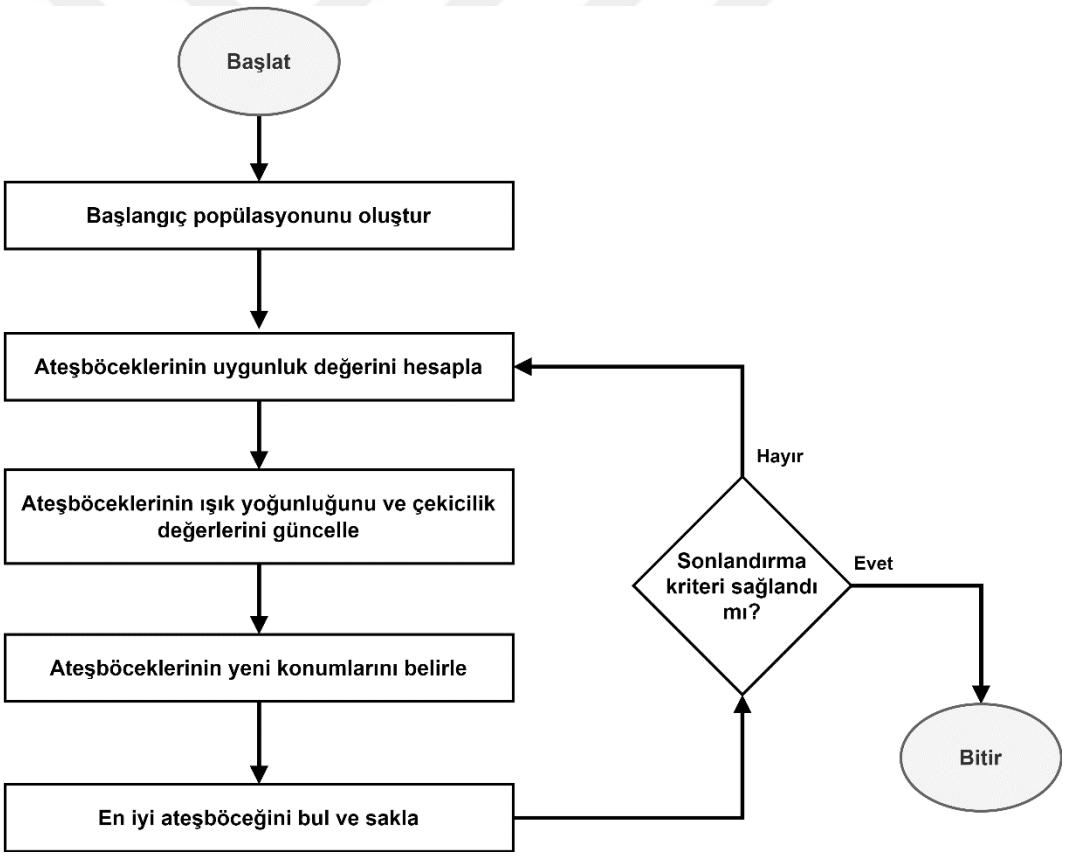
Burada I ışık yoğunluğunu, I_0 orijinal ışık yoğunluğunu, γ ise sabit olarak alınabilecek ışık soğurma katsayısını göstermektedir. Ateşböceğinin çekicilik değeri, etrafındaki ateşböceklerinin görerek farkettiği ışık yoğunluğuyla orantılı olduğundan, ateşböceğinin çekicilik değerini (β) Eşitlik 3.2'deki gibi tanımlayabiliriz:

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{3.2}$$

Burada β_0 bir sabit olup $r = 0$ mesafesindeki çekicilik değeridir. İki ateş böceği i ve j arasındaki mesafe, $r_{ij} = |x_i - x_j|$ şeklinde kartezyen mesafe olarak tanımlanabilir. Bir ateş böceği i'nin hareketi, daha çekici (daha parlak) başka bir ateş böceği j'ye doğru çekilmesi Eşitlik 3.3'teki gibi belirlenir.

$$x_i^{t+1} = x_i^t + \Delta x_i,$$
$$\Delta x_i = \beta_0 e_{ij}^{-\gamma r^2} \left(x_j^t - x_i^t\right) + \alpha \varepsilon_i \tag{3.3}$$

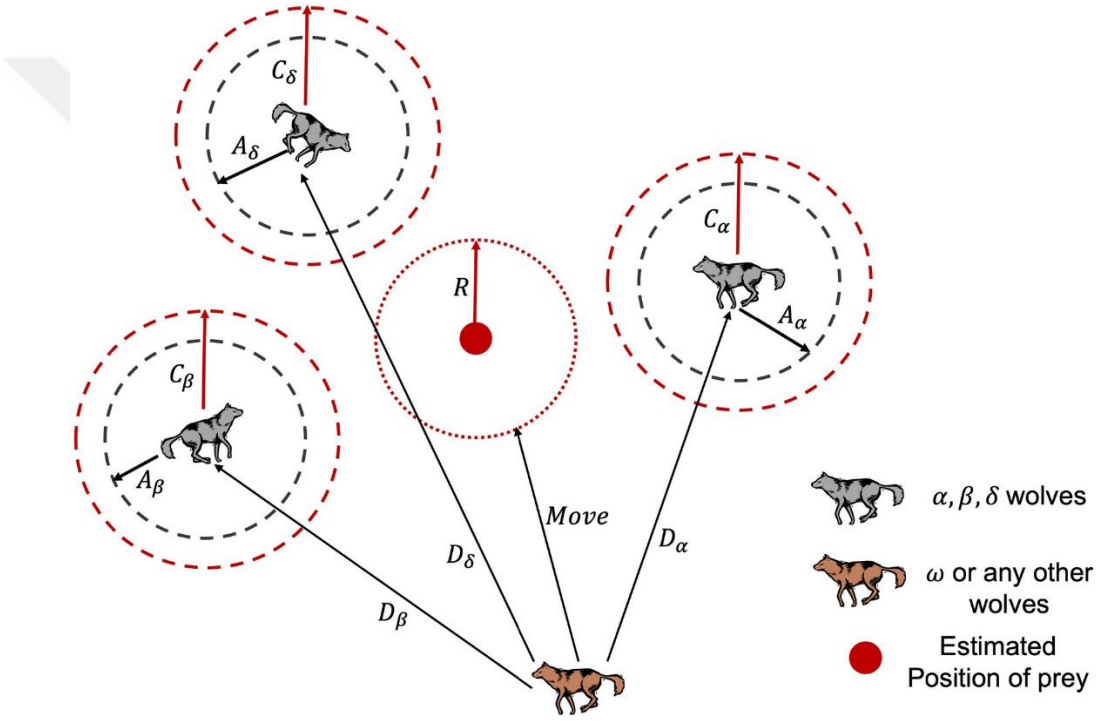
Burada, $\beta_0 e_{ij}^{-\gamma r^2}$ terimi çekicilikten kaynaklanmaktadır, ikinci terimdeki α, rastgelelik parametresidir ve ε_i ise Gauss dağılımından alınan rastgele sayıların bir vektörüdür. Şekil 3.1'de Ateşböceği algoritmasının akış şeması gösterilmektedir.



Şekil 3-1: Ateşböceği algoritmasının akış diyagramı.

3.2 GRİ KURT OPTİMİZASYON ALGORİTMASI

Gri Kurt Optimizasyon Algoritması (*Grey Wolf Optimizer,* GWO), Mirjalili ve ark. [42] tarafından geliştirilmiştir. Adından da anlaşılacağı gibi bu algoritma, gri kurt sürülerinin avlanma davranışlarından ilham almıştır. GWO algoritmasında, kurtların sosyal hiyerarşisini modellerken en uygun (optimal) çözümü alfa (α) olarak tayin edilir. Popülasyondaki 2'inci ve 3'üncü en iyi çözümler beta (β) ve delta (δ) olarak adlandırılmıştır. Bunların dışındaki aday çözümler ise omega (ω) olduğu varsayılmaktadır. GWO yönteminde arama eylemi Şekil 3.2'de gösterildiği gibi α, β ve δ kurtları tarafından yönlendirilir. Geriye kalan ω kurtları ise bu üç kurdu takip eder.



Şekil 3-2: Üç lider kurdun (α, β ve δ) ve avın tahmini konumları [43].

GWO'da avlanma süreci, avın aranması, avın kuşatılıp taciz edilmesi ve ava saldırma eylemi olmak üzere üç ana adımdan oluşur. Kuşatma süreci, her bir kurdun konumunun avın konumuna göre güncellenmesiyle matematiksel olarak modellenebilir. Eşitlik 3.4'te, bir sonraki yineleme için kurtların konumunu güncellemek için kullanılabilir.

$$
\begin{aligned}
&r_1, r_2 \in [0,1], \quad a = 2(1 - t/T), \\
&A = 2ar_1 - a, \quad C = 2r_2, \\
&D = \left| CX_p(t) - X(t) \right|, \\
&X(t+1) = X_p(t) - A \cdot D
\end{aligned}
\tag{3.4}
$$

Burada r_1 ve r_2, algoritmayı doğası gereği stokastik yapan tekdüze rastgele değişkenlerdir ve a keşif ve sömürü için yinelemeler boyunca ikiden sıfıra doğrusal olarak azalan bir uyarlama parametresidir. Şekil 3.3'te GWO algoritmasının sözde kodu verilmiştir.

Gri kurtların başlangıç popülasyonunu oluştur.
Gri kurtların uygunluk değerlerini hesapla.
En iyi gri kurdu α olarak sakla.
İkinci en iyi gri kurdu β olarak sakla.
Üçüncü en iyi gri kurdu δ olarak sakla.
while $(t < T)$
    $a \leftarrow 2(1 - t/T)$
    for *(Her bir gri kurt için)*
        Alfa, beta, delta katsayı vektörlerini oluştur.
        Mesafe vektörlerini hesapla.
        Gri kurdun konumunu güncelle.
        Gri kurdun uygunluk değerini (f_g) hesapla.
        if $(f_g < f_\alpha)$ ***then***
            $f_\alpha \leftarrow f_g$
        elseif $(f_g < f_\beta)$ ***then***
            $f_\beta \leftarrow f_g$
        elseif $(f_g < f_\delta)$ ***then***
            $f_\delta \leftarrow f_g$
        end if
    end for
end while

Şekil 3-3: GWO algoritmasının sözde kodu.

GWO algoritmasında, A ve C parametreleri ayarlanabilir yegâne iki parametredir. Önceki denklemlerde kullanılan varsayım, avın konumunun bilindiği yönündedir. Ancak bu, bir amaç fonksiyonunu optimize ettiğimiz soyut bir arama uzayında doğru değildir. GWO algoritması, sahadaki en iyi üç ajanın, yani α, β ve δ kurtlarının ava doğru ilerlediği ve en uygun konuma daha yakın olduğu fikrini varsayar. Bu varsayıma dayanarak, tüm kurtlar davranışlarını takip edecek ve

konumlarını, Eşitlik 3.5'te gösterildiği gibi lider kurtların (α, β ve δ) konumlarının ortalaması olarak tanımlanan tahmini av konumuyla güncelleyecektir.

$$
\begin{aligned}
D_\alpha &= \left|C_\alpha X_\alpha(t) - X(t)\right|, \\
D_\beta &= \left|C_\beta X_\beta(t) - X(t)\right|, \\
D_\delta &= \left|C_\delta X_\delta(t) - X(t)\right|, \\
X_1 &= X_\alpha(t) - A_\alpha D_\alpha, \\
X_2 &= X_\beta(t) - A_\beta D_\beta, \\
X_3 &= X_\delta(t) - A_\delta D_\delta, \\
X(t+1) &= \frac{X_1 + X_2 + X_3}{3}
\end{aligned}
\tag{3.5}
$$

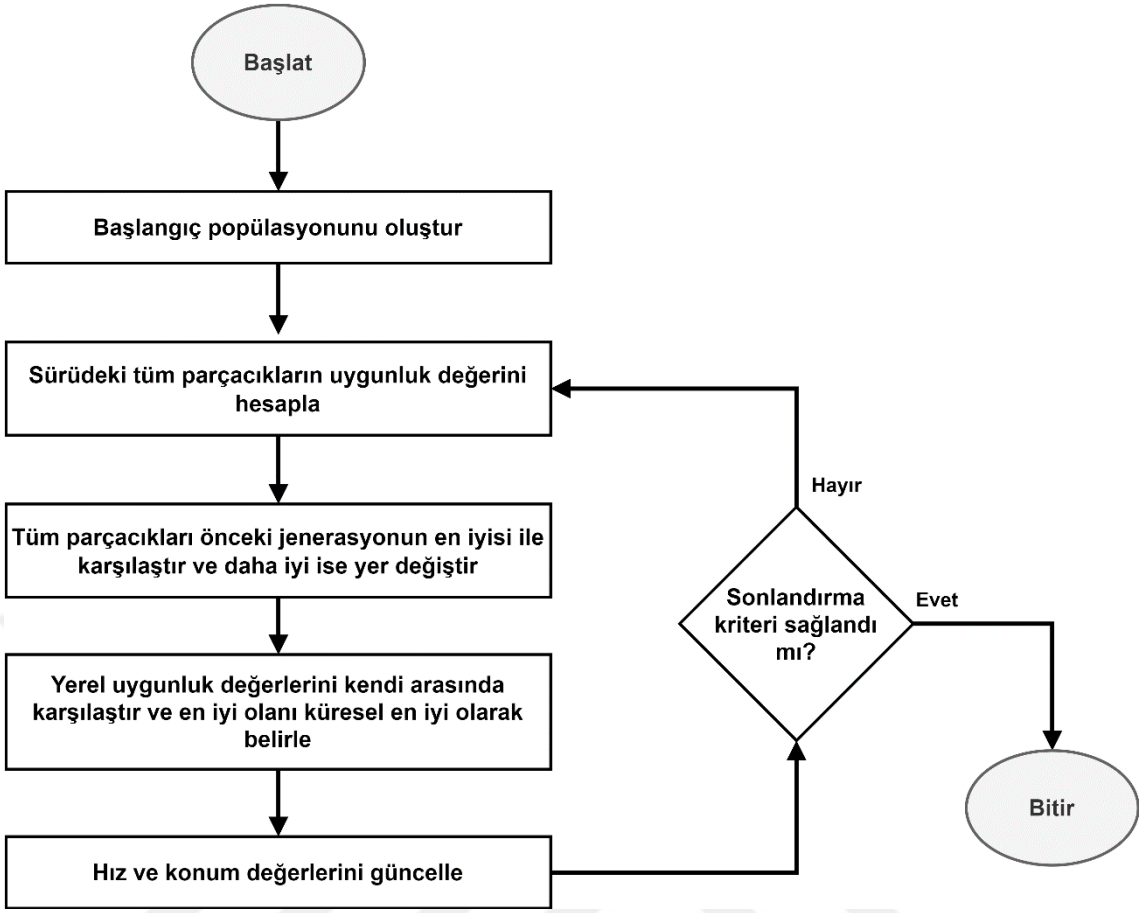
3.3 PARÇAÇIK SÜRÜ OPTİMİZASYONU

Parçaçık Sürü Optimizasyonu (*Particle Swarm Optimization*, PSO), Kennedy ve Eberhart [44] tarafından önerildi. PSO, kuş sürüleri veya balık sürüleri gibi bazı hayvanların akıllı kolektif davranışlarıyla motive edilen, popülasyona dayalı bir stokastik optimizasyon algoritmasıdır. PSO, 1995 yılında piyasaya sürülmesinden bu yana çok sayıda gelişme yaşandı. Araştırmacılar tekniği öğrendikçe, farklı talepleri hedefleyen yeni sürümlerini türettiler, birçok alanda yeni uygulamalar geliştirdiler, çeşitli parametrelerin etkilerine ilişkin teorik çalışmalar yayınladılar ve algoritmanın birçok varyantını önerdiler [45].

PSO, sürekli uzay koordinat sisteminde matematiksel olarak şu şekilde tanımlanabilir: Sürü boyutunun N olduğunu, her parçacığın (bireyin) D-boyutlu uzaydaki konum vektörünün $X_i = (x_{i1}, x_{i2}, \dots, x_{id}, \dots, x_{iD})$, hız vektörünün $V_i = (v_{i1}, v_{i2}, \dots, v_{id}, \dots, v_{iD})$, parçacığın en uygun konumunun $P_i = (p_{i1}, p_{i2}, \dots, p_{id}, \dots, p_{iD})$ ve sürünün optimal konumunun ise $P_g = (p_{g1}, p_{g2}, \dots, p_{gd}, \dots, p_{gD})$ temsil edildiğini varsayalım. Buna göre, PSO algoritmasında bireyin optimal pozisyonunun güncelleme formülü Eşitlik 3.6'daki gibidir.

$$
p_{i,t+1}^d = \begin{cases} x_{i,t+1}^d, & f\left(X_{i,t+1}\right) < p\left(P_{i,t}\right) \\ p_{i,t}^d, & \textit{değilse} \end{cases}
\tag{3.6}
$$

Şekil 3.4'te Parçaçık Sürü Optimizasyonu algoritmasının akış şeması verilmiştir.



Şekil 3-4: Parçaçık Sürü Optimizasyonu algoritmasının akış şeması.

Sürünün en uygun konumu, tüm bireylerin en uygun konumlarıdır. PSO'nun ilk versiyonu optimizasyon probleminde çok etkili olmadığından, ilk algoritma önerildikten hemen sonra değiştirilmiş bir PSO algoritması Shi ve Eberhart [46] tarafından önerildi. Bu sürümünde, hız güncelleme formülüne atalet ağırlığı eklendi. Hız ve konum güncelleme formülü sırasıyla Eşitlik 3.7 ve 3.8'deki gibi gösterilir.

$$v_{i,t+1}^{d} = \omega \cdot v_{i,t}^{d} + c_1 \cdot \text{rand} \cdot \left(p_{i,t}^{d} - x_{i,t}^{d} \right) + c_2 \cdot \text{rand} \cdot \left(p_{g,t}^{d} - x_{i,t}^{d} \right) \tag{3.7}$$

$$x_{i,t+1}^{d} = x_{i,t}^{d} + v_{i,t+1}^{d} \tag{3.8}$$

Burada ω, atalet momenti, c_1 ve c_2, 0 ile 1 arasındaki tekdüze dağılımdan alınan iki rasgele değişkendir.

3.4 YOL BULUCU ALGORİTMASI

Yol Bulucu Algoritması (*Pathfinder Algorithm,* PFA), gruplar halinde yaşayan hayvanların kolektif eyleminden ilham alır ve en iyi yiyecek alanlarını ya da avı bulmak için sürülerin liderlik hiyerarşisini taklit eder [47]. PFA'da, Eşitlik 3.9'da gösterilen matematiksel model, yol bulucuyu takip etmek ve etmenlerin arama uzayındaki yiyecek kaynaklarını bulmak için kullanılır.

$$x\left(t+\Delta t\right)=x^{0}\left(t\right)\bullet n+f_{i}+f_{p}+\varepsilon \tag{3.9}$$

Burada x, sürüdeki bir etkenin konum vektörünü, t zamanı, n açısal olmayan birim vektörünü, f_i, x_i ve $x_{i\pm1}$ konum vektörlerinin komşuluk etkileşimini, f_p etkileşim değerlerini x_i ile en iyi çözüm arasındaki küresel güç olarak adlandırılan ve son olarak ε bir titreşim vektörünü temsil etmektedir. Yol bulucu adı verilen sürü liderine ait konum vektörlerinin güncellenmesi işlemi Eşitlik 3.10'da verilmiştir.

$$x_{p}\left(t+\Delta t\right)=x_{p}\left(t\right)+\Delta x+A \tag{3.10}$$

x_p, sürü liderinin (yol bulucu) konum vektörleri olduğunda, Δx, konum değişikliği sırasında yol bulucunun kat ettiği mesafedir ve A dalgalanma oranı vektörüdür. Sürünün toplu olarak hareket edebilmesi ve en uygun çözümü bulabilmesi için Eşitlik 3.11'de gösterildiği gibi matematiksel modelin değiştirilmesi gerekmektedir.

$$x_{i}^{T+1}=x_{i}^{T}+R_{1}\left(x_{j}^{T}-x_{i}^{T}\right)+R_{2}\left(x_{p}^{T}-x_{i}^{T}\right)+\varepsilon \tag{3.11}$$

Burada x_i ve x_j sırasıyla popülasyondaki i-th ve j-th etmenlerinin konum vektörleridir. T, yinelemenin sayısıdır (veya geçerli indekstir), $R_1 = \alpha r_1$ ve $R_2 = \beta r_2$ denklemlerinde, r_1 ve r_2, sıfır (0) ile bir (1) aralığında rastgele bir sayı olarak belirlenir. Ayrıca α, etkileşimin büyüklüğünü tanımlayan etkileşim katsayısını ifade eder. Popülasyondaki bir etmenin komşusuyla hareketi ve son olarak β, sürüyü lidere daha yakın tutmak için rastgele mesafeyi belirleyen çekim katsayısıdır. Eşitlik 3.12'deki gibi yol bulucu için başka bir değişiklik yapılması gerekir.

$$x_{p}^{T+1}=x_{p}^{T}+2r_{3}\left(x_{p}^{T}-x_{p}^{T-1}\right)+A \tag{3.12}$$

Yukarıdaki denklemlerde kullanılan dalgalanma oranı (A) ve titreşim (ε) vektörlerinin açık ifadeleri Eşitlik 3.13'teki gibidir.

$$\varepsilon = \left(1-\left(T/T_{\max}\right)\right)\cdot u_1 \cdot D_{ij}, \quad D_{ij} = \left\| x_i - x_j \right\|, \\ A = u_2 \cdot \exp\left(-2T/T_{\max}\right) \tag{3.13}$$

Burada u_1 ve u_2, 0 ile 1 arasında rastgele oluşturulmuş bir sayıyı, D_{ij}, popülasyondaki i-th ve j-th öğeleri arasındaki mesafedir ve T_{max}, yinelemelerin maksimum sayısını (veya sınırını) temsil etmektedir. Yol Bulucu algoritmasının sözde kodu Şekil 3.5'te verilmiştir.

```
1: PFA parametrelerini belirle
2: Başlangıç popülasyonunu oluştur
3: Başlangıç popülasyonunun uygunluk değerini hesapla
4: Pathfinder ajanını tespit et ve sakla
5: while (T < T_max)
6:     α ve β katsayıları için [1, 2] arasında rastgele sayı üret
7:     Pathfinder'ın konumunu (x_p) güncelle
8:     if (f_p^new < f_p)
9:         Pathfinder'ı yenisi ile değiştir (x_p = x_p^new)
10:    end
11:    for i ← 2 to Popülasyon sayısı
12:        Ajanların konumlarını güncelle (x_i)
13:    end
14:    Ajanların uygunluk değerlerini hesapla
15:    En iyi uygunluk değerine sahip ajanı bul (x_best)
16:    if (f_best < f_p)
17:        x_p = x_best
18:        f_p = f_best
19:    end
20:    for i ← 2 to Popülasyon sayısı
21:        if (f_i^new < f_i)
22:            x_i = x_i^new
23:            f_i = f_i^new
24:        end
25:    end
26:    Yeni A ve ε değerlerini üret
27: end
```

Şekil 3-5: PFA'nın sözde kodu.

4. MÜHENDİSLİK TASARIM PROBLEMLERİ

Mühendislik tasarımı optimizasyon problemleri, genel olarak yeni geliştirilen metasezgisel algoritmaların etkinliğini göstermek için kullanılmaktadır ve bu durum literatürde oldukça yaygındır. Doğrusal olmayan (*non-linear*) bu mühendislik problemlerini çözmek için farklı yöntemler kullanan birçok araştırmacı tarafından araştırılmıştır [48].

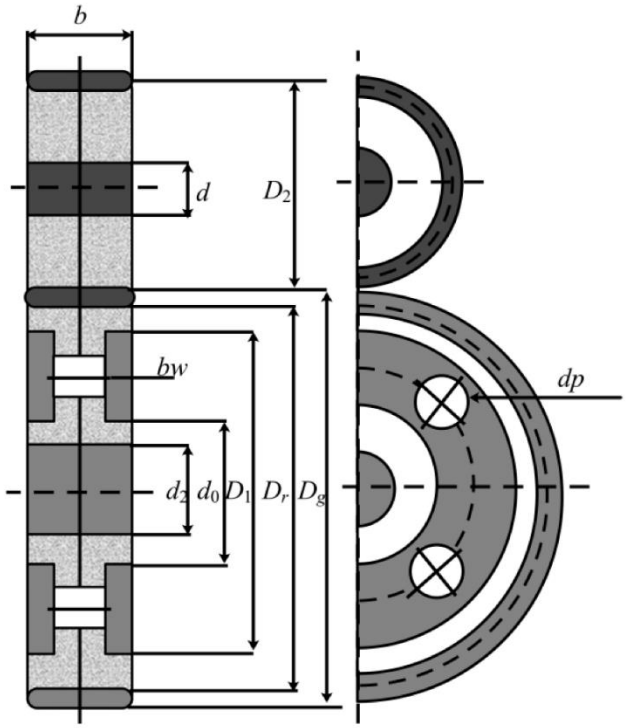
Bu çalışmada, Bölüm 3'te verilen metasezgisel algoritmaların performanslarını test etmek amacıyla literatürde iyi bilinen sekiz farklı mühendislik tasarım problemi kullanılmıştır. Bu problemler; Dört Kademeli Dişli Kutusu Problemi, Gaz İletim Kompresörü Tasarımı, Dişli Takımı Tasarım Problemi, Himmelblau Fonksiyonu, Hidrostatik Baskı Yatağı Tasarımı Problemi, Çoklu Disk Kavramalı Fren Tasarım Problemi, Endüstriyel Soğutma Sisteminin Optimal Tasarımı ve Planet Dişli Zinciri Tasarım Problemidir.

4.1 DÖRT KADEMELİ DİŞLİ KUTUSU PROBLEMİ

Dişli kutusu ağırlığının azaltılması amacıyla önerilen dört kademeli dişli kutusu tasarım probleminin şematiği Şekil 4.1'de gösterilmektedir. Dişlilerin konumları, pinyonların konumları, boşlukların kalınlığı ve diş sayısı da dâhil olmak üzere hesaplanması gereken yirmi iki özellik mevcuttur. Bu parametreler dört gruba ayrılabilir. Ayrıca karşılanması gereken 86 kısıtlama vardır. Dişlilerin boyutunu, kinematiğini, temas oranını, eğimini, dişlilerin gücünü, dişlilerin montajını ve eğimini gözden geçirirler. Birçok yerel çözüm ve küçük bir arama bölgesi olduğundan, dört aşamalı dişli kutusu tasarım problemi çoğu optimizasyon algoritması için zorluk teşkil etmektedir [49].

Dört Kademeli Dişli Kutusu probleminin uygunluk fonksiyonu Eşitlik 4.1'de verilmiştir.

$$f\left(\vec{x}\right)=\frac{\pi}{1000}\sum_{i=1}^{4}\frac{b_i c_i^2\left(N_{pi}^2+N_{gi}^2\right)}{\left(N_{pi}+N_{gi}\right)^2},\quad i=\{1,..,4\} \tag{4.1}$$



Şekil 4-1: Dört Kademeli Dişli Kutusu Problemi [49].

Eşitlik 4.2 ile 4.34 arasında Dört Kademeli Dişli Kutusu probleminin kısıtları verilmiştir.

$$g_1(\bar{x}) = \left(\frac{366000}{\Pi w_1} + \frac{2c_1 N_{p1}}{N_{pi} + N_{g1}}\right)\left(\frac{\left(N_{p1} + N_{g1}\right)^2}{4b_1 c_1^2 N_{p1}}\right) - \frac{\sigma_N J_R}{0.0167 W K_0 K_m} \le 0 \tag{4.2}$$

$$g_2(\bar{x}) = \left(\frac{366000 N_{g1}}{\Pi w_1 N_{p1}} + \frac{2c_2 N_{p2}}{N_{p2} + N_{g2}}\right)\left(\frac{\left(N_{p2} + N_{g2}\right)^2}{4b_2 c_2^2 N_{p2}}\right) - \frac{\sigma_N J_R}{0.0167 W K_0 K_m} \le 0 \tag{4.3}$$

$$g_3(\bar{x}) = \left(\frac{366000 N_{g1} N_{g2}}{\Pi w_1 N_{p1} N_{p2}} + \frac{2c_3 N_{p3}}{N_{p3} + N_{g3}}\right)\left(\frac{\left(N_{p3} + N_{g3}\right)^2}{4b_3 c_3^2 N_{p3}}\right) - \frac{\sigma_N J_R}{0.0167 W K_0 K_m} \le 0 \tag{4.4}$$

$$g_4(\bar{x}) = \left(\frac{366000 N_{g1} N_{g2} N_{g3}}{\Pi w_1 N_{p1} N_{p2} N_{p3}} + \frac{2c_4 N_{p4}}{N_{p4} + N_{g4}}\right)\left(\frac{\left(N_{p4} + N_{g4}\right)^2}{4b_4 c_4^2 N_{p4}}\right) - \frac{\sigma_N J_R}{0.0167 W K_0 K_m} \le 0 \tag{4.5}$$

$$g_5(\bar{x}) = \left(\frac{366000}{\Pi w_1} + \frac{2c_1 N_{p1}}{N_{p1} + N_{g1}}\right)\left(\frac{\left(N_{p1} + N_{g1}\right)^3}{4b_1 c_1^2 N_{g1} N_{p1}^2}\right) - \left(\frac{\sigma_H}{c_p}\right)^2 \left(\frac{\sin(\phi)\cos(\phi)}{0.0334 W K_0 K_m}\right) \le 0 \tag{4.6}$$

$$g_6(\bar{x}) = \left(\frac{366000N_{g1}}{\Pi w_1 N_{p1}} + \frac{2c_2 N_{p2}}{N_{p2}+N_{g2}}\right)\left(\frac{\left(N_{p2}+N_{g2}\right)^3}{4b_2c_2^2N_{g2}N_{p2}^2}\right) - \left(\frac{\sigma_H}{c_p}\right)^2\left(\frac{\sin(\phi)\cos(\phi)}{0.0334WK_0K_m}\right) \le 0 \quad (4.7)$$

$$g_7(\bar{x}) = \left(\frac{366000N_{g1}N_{g2}}{\Pi w_1 N_{p1}N_{p2}} + \frac{2c_3 N_{p3}}{N_{p3}+N_{g3}}\right)\left(\frac{\left(N_{p3}+N_{g3}\right)^3}{4b_3c_3^2N_{g3}N_{p3}^2}\right) - \left(\frac{\sigma_H}{c_p}\right)^2\left(\frac{\sin(\phi)\cos(\phi)}{0.0334WK_0K_m}\right) \le 0 \quad (4.8)$$

$$g_8(\bar{x}) = \left(\frac{366000N_{g1}N_{g2}N_{g3}}{\Pi w_1 N_{p1}N_{p2}N_{p3}} + \frac{2c_4 N_{p4}}{N_{p4}+N_{g4}}\right)\left(\frac{\left(N_{p4}+N_{g4}\right)^3}{4b_4c_4^2N_{g4}N_{p4}^2}\right) - \left(\frac{\sigma_H}{c_p}\right)^2\left(\frac{\sin(\phi)\cos(\phi)}{0.0334WK_0K_m}\right) \le 0 \quad (4.9)$$

$$g_{9-12}(\bar{x}) = -N_{pi}\sqrt{\frac{sin^2(\phi)}{4} - \frac{1}{N_{pi}} + \left(\frac{1}{N_{pi}}\right)^2} + N_{gi}\sqrt{\frac{sin^2(\phi)}{4} - \frac{1}{N_{gi}} + \left(\frac{1}{N_{gi}}\right)^2} + \frac{sin(\phi)(N_{pi}+N_{gi})}{2} + CR_{\min}\pi cos(\phi) \le 0 \quad (4.10)$$

$$g_{13-16}(\bar{x}) = d_{\min} - \frac{2c_i N_{pi}}{N_{pi}+N_{gi}} \le 0 \quad (4.11)$$

$$g_{17-20}(\bar{x}) = d_{\min} - \frac{2c_i N_{gi}}{N_{pi}+N_{gi}} \le 0 \quad (4.12)$$

$$g_{21}(\bar{x}) = x_{p1} + \left(\frac{\left(N_{p1}+2\right)c_1}{N_{p1}+N_{g1}}\right) - L_{\max} \le 0 \quad (4.13)$$

$$g_{22-24}(\bar{x}) = L_{\max} + \left(\frac{\left(N_{pi}+2\right)c_i}{N_{gi}+N_{pi}}\right)_{i=2,3,4} + x_{g(i-1)} \le 0 \quad (4.14)$$

$$g_{25}(\bar{x}) = -x_{p1} + \frac{\left(N_{p1}+2\right)c_1}{N_{p1}+N_{g1}} \le 0 \quad (4.15)$$

$$g_{26-28}(\bar{x}) = \left(\frac{\left(N_{pi}+2\right)c_i}{N_{pi}+N_{gi}} - x_{g(i-1)}\right)_{i=2,3,4} \le 0 \quad (4.16)$$

$$g_{29}(\bar{x}) = y_{p1} + \frac{\left(N_{p1}+2\right)c_1}{N_{p1}+N_{g1}} - L_{\max} \le 0 \quad (4.17)$$

$$g_{30-32}(\bar{x}) = -L_{\max} + \left(\frac{c_i\left(2+N_{pi}\right)}{N_{pi}+N_{gi}} + y_{g(i-1)}\right)_{i=2,3,4} \le 0 \quad (4.18)$$

$$g_{33}(\bar{x}) = \frac{\left(2+N_{p1}\right)c_1}{N_{p1}+N_{g1}} - y_{p1} \le 0 \quad (4.19)$$

$$g_{34-36}(\bar{x}) = \left(\frac{c_i\left(2+N_{pi}\right)}{N_{pi}+N_{gi}} - y_{g(i-1)} \right)_{i=2,3,4} \le 0 \tag{4.20}$$

$$g_{37-40}(\bar{x}) = -L_{\max} + \frac{c_i\left(2+N_{gi}\right)}{N_{pi}+N_{gi}} + x_{gi} \le 0 \tag{4.21}$$

$$g_{41-44}(\bar{x}) = -x_{gi} + \frac{\left(N_{gi}+2\right)c_i}{N_{pi}+N_{gi}} \le 0 \tag{4.22}$$

$$g_{45-48}(\bar{x}) = y_{gi} + \frac{\left(N_{gi}+2\right)c_i}{N_{pi}+N_{gi}} - L_{\max} \le 0 \tag{4.23}$$

$$g_{49-52}(\bar{x}) = -y_{gi} + \frac{\left(N_{gi}+2\right)c_i}{N_{pi}+N_{gi}} \le 0 \tag{4.24}$$

$$g_{53-56}(\bar{x}) = \left(b_i - 8.255\right)\left(b_i - 5.715\right)\left(b_i - 12.70\right)\left(-N_{pi} + 0.945c_i - N_{gi}\right)(-1) \le 0 \tag{4.25}$$

$$g_{57-60}(\bar{x}) = \left(b_i - 8.255\right)\left(b_i - 3.175\right)\left(b_i - 12.70\right)\left(-N_{pi} + 0.646c_i - N_{gi}\right) \le 0 \tag{4.26}$$

$$g_{61-64}(\bar{x}) = \left(b_i - 5.175\right)\left(b_i - 3.175\right)\left(b_i - 12.70\right)\left(-N_{pi} + 0.504c_i - N_{gi}\right) \le 0 \tag{4.27}$$

$$g_{65-68}(\bar{x}) = \left(b_i - 5.175\right)\left(b_i - 3.175\right)\left(b_i - 8.255\right)\left(0c_i - N_{gi} - N_{pi}\right) \le 0 \tag{4.28}$$

$$g_{69-72}(\bar{x}) = \left(b_i - 8.255\right)\left(b_i - 5.715\right)\left(b_i - 12.70\right)\left(N_{gi} + N_{pi} - 1.812c_i\right) \le 0 \tag{4.29}$$

$$g_{73-76}(\bar{x}) = \left(b_i - 8.255\right)\left(b_i - 3.175\right)\left(b_i - 12.70\right)\left(-0.945c_i + N_{pi} + N_{gi}\right) \le 0 \tag{4.30}$$

$$g_{77-80}(\bar{x}) = \left(b_i - 5.175\right)\left(b_i - 3.175\right)\left(b_i - 12.70\right)\left(-0.646c_i + N_{pi} + N_{gi}\right)(-1) \le 0 \tag{4.31}$$

$$g_{81-84}(\bar{x}) = \left(b_i - 5.175\right)\left(b_i - 3.175\right)\left(b_i - 8.255\right)\left(N_{pi} + N_{gi} - 0.504c_i\right) \le 0 \tag{4.32}$$

$$g_{85} = w_{\min} - \frac{w_1\left(N_{p1}N_{p2}N_{p3}N_{p4}\right)}{\left(N_{g1}N_{g2}N_{g3}N_{g4}\right)} \le 0 \tag{4.33}$$

$$g_{86} = \frac{w_1\left(N_{p1}N_{p2}N_{p3}N_{p4}\right)}{\left(N_{g1}N_{g2}N_{g3}N_{g4}\right)} - w_{\max} \le 0 \tag{4.34}$$

Eşitlik 4.35'te bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

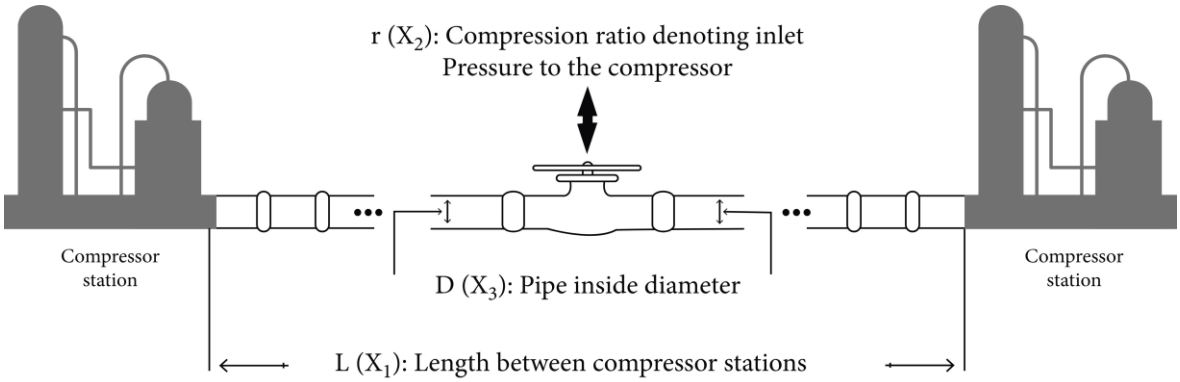
$$\vec{x} = \left\{N_{p1}N_{g1}N_{p2}N_{g2}...b_1b_2....x_{p1}x_{g1}x_{g2}...y_{p1}y_{g1}y_{g2}...y_{g4}\right\},$$
$$c_i = \sqrt{\left(y_{gi} - y_{pi}\right)^2 + \left(x_{gi} - x_{pi}\right)^2},\ K_0 = 1.5,\ d_{\min} = 25,$$
$$J_R = 0.2,\ \phi = 120°,\ W = 55.9,\ K_M = 1.6,\ CR_{\min} = 1.4, \quad (4.35)$$
$$L_{\max} = 127,\ c_p = 464,\ \sigma_H = 3290,\ w_{\max} = 255,\ w_1 = 5000,$$
$$\sigma_N = 2090,\ w_{\min} = 245.$$

Eşitlik 4.36'da bazı değişkenlerin alabileceği değerler ve eşitsizlik ifadeleri verilmiştir.

$$b_i \in \{3.175, 12.7, 8.255, 5.715\}$$
$$y_{p1}, x_{p1}, y_{gi}, x_{gi} \in \{12.7, 38.1, 25.4, 50.8, 76.2, 63.5, 88.9, 114.3, 101.6\} \quad (4.36)$$
$$7 \le N_{gi}, N_{pi} \le 76 \in \mathbb{Z}$$

4.2 GAZ İLETİM KOMPRESÖRÜ TASARIMI

Gaz iletim kompresörü tasarım probleminin temel amacı, kompresör istasyonları arasındaki uzunluk ($L = x_1$), kompresöre giriş basıncını gösteren sıkıştırma oranı ($r = x_2$) ve borunun iç çapı ($D = x_3$) olan dört tasarım değişkenini kullanarak amaç fonksiyonunu en aza indirmektir. Problemin şematiği Şekil 4.2'de gösterilmektedir [50].



Şekil 4-2: Gaz İletim Kompresörü Probleminin Şematiği [50].

Gaz İletim Kompresörü probleminin uygunluk fonksiyonu Eşitlik 4.37'de verilmiştir.

$$f\left(\vec{x}\right) = 8.61\times10^5 x_1^{1/2} x_2 x_3^{-2/3} x_4^{-1/2} + 3.69\times10^4 x_3 + 7.72\times10^8 x_1^{-1} x_2^{0.219} - 765.43\times10^6 x_1^{-1} \quad (4.37)$$

Gaz İletim Kompresörü probleminin kısıtları Eşitlik 4.38'de gösterilmektedir.

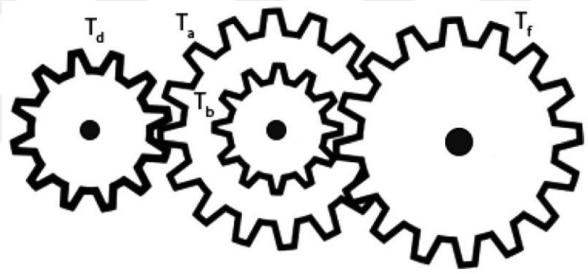
$$x_4 x_2^{-2} + x_2^{-2} - 1 \le 0, \tag{4.38}$$

Probleme ait durum değişkenlerinin alabileceği değer aralıkları Eşitlik 4.39'da verilmiştir.

$$20 \le x_1 \le 50,\ 1 \le x_2 \le 10,\ 20 \le x_3 \le 50,\ 0.1 \le x_4 \le 60. \tag{4.39}$$

4.3 DİŞLİ TAKIMI TASARIM PROBLEMİ

Dişli takımı tasarım problemi, tahrik eden ve tahrik edilen miller arasında belirli bir dişli oranını bulmak için tasarlanmıştır. Bu problemin amacı, Şekil 4.3'te gösterildiği gibi dişlilerdeki optimal diş sayısını ($T_d = x_1$, $T_b = x_2$, $T_a = x_3$ ve $T_f = x_4$) bulmaktır. Ayrıca diş sayısının tam sayı olması gerekmektedir [51]



Şekil 4-3: Dişli Takımı Tasarım Probleminin Şematik Görünümü [51].

Dişli takımı tasarım probleminin uygunluk fonksiyonu Eşitlik 4.40'ta verilmiştir.

$$f(\vec{x}) = \left(i_{trg} - i_{tot}\right)^2 = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right)^2 \tag{4.40}$$

Probleme ait kısıt fonksiyonları Eşitlik 4.41'de verilmiştir.

$$g_1(\vec{x}) = 12 \le x_i \le 60, \quad i \in \{1, ..., 4\} \tag{4.41}$$

4.4 HIMMELBLAU'NUN FONKSİYONU

Himmelblau [52], 1972'de doğrusal olmayan kısıtlı bir optimizasyon problemi önerdi ve bu bir makine mühendisliği problemi olarak kabul edildi. Bu problem altı doğrusal olmayan kısıtlama ve beş durum değişkeni içermektedir [53].

Himmelblau probleminin uygunluk fonksiyonu Eşitlik 4.42'de verilmiştir.

$$f(\vec{x}) = 5,3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{4.42}$$

Problemdeki kısıtlar Eşitlik 4.43'te verilmiştir.

$$\begin{aligned}
&g_1(\vec{x}) = -\left(85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5\right) \le 0,\\
&g_2(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0,\\
&g_3(\vec{x}) = 90 - \left(80.51249 + 0.0071317_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2\right) \le 0,\\
&g_4(\vec{x}) = 80.51249 + 0.0071317_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2 - 110 \le 0,\\
&g_5(\vec{x}) = 20 - \left(9.300961 + 0.0047026x_3x_5 + 0.0019085x_1x_3 - 0.0019085x_3x_4\right) \le 0,\\
&g_6(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0019085x_1x_3 - 0.0019085x_3x_4 - 25 \le 0,
\end{aligned} \tag{4.43}$$

Durum değişkenlerinin alabileceği değer aralıkları Eşitlik 4.44'te verilmiştir.

$$\begin{aligned}
&78 \le x_1 \le 102, \quad 33 \le x_2 \le 45,\\
&27 \le x_3 \le 45, \quad 27 \le x_4 \le 45,\\
&27 \le x_5 \le 45.
\end{aligned} \tag{4.44}$$

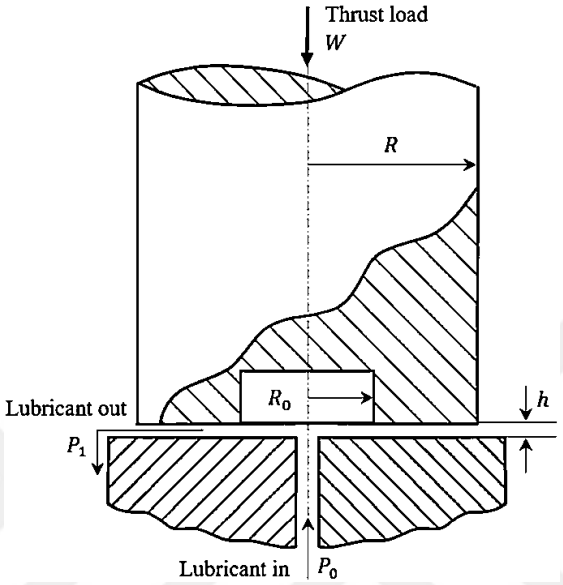
4.5 HİDROSTATİK BASKI YATAĞI TASARIMI PROBLEMİ

Hidrostatik baskı yatağı tasarım probleminin amacı, eksenel rulmanın çalışması sırasındaki güç kaybını azaltmaktır. Tasarım değişkenleri şunları içerir: yatak adımı yarıçapı (R), akış hızı (Q), yağ viskozitesi (μ) ve oluk yarıçapı (R_0). Hidrostatik baskı yatağı tasarım problemi Şekil 4.4'te şematik olarak gösterilmektedir [54].

Hidrostatik baskı yatağı tasarım problemine ait uygunluk fonksiyonu Eşitlik 4.45'te verilmiştir.

$$f\left(\vec{x}\right)=\left(QP_0/0.7\right)+E_f \tag{4.45}$$

Problemdeki kısıtlar Eşitlik 4.46'da verilmiştir.



Şekil 4-4: Hidrostatik Baskı Yatağı Tasarımının Şematiği [54].

$$\begin{aligned}
&g_1\left(\vec{x}\right)=1000-P_0\le 0,\\
&g_2\left(\vec{x}\right)=W-101000\le 0,\\
&g_3\left(\vec{x}\right)=5000-\frac{W}{\pi\left(R^2-R_0^2\right)}\le 0,\\
&g_4\left(\vec{x}\right)=50-P_0\le 0,\\
&g_5\left(\vec{x}\right)=0.001-\frac{0.0307}{386.4P_0}\left(\frac{Q}{2\pi Rh}\right)\le 0,\\
&g_6\left(\vec{x}\right)=R-R_0\le 0,\\
&g_7\left(\vec{x}\right)=h-0.001\le 0
\end{aligned} \tag{4.46}$$

Eşitlik 4.47'de bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

$$W = \frac{\pi P_0}{2} \frac{R^2 - R_0^2}{\ln\left(R/R_0\right)},$$

$$P_0 = \frac{6\mu Q}{\pi h^3} \ln\left(R/R_0\right),$$

$$E_f = 9336Q \times 0.0307 \times 0.5\Delta T,\ \Delta T = 2\left(10^P - 559.7\right), \tag{4.47}$$

$$P = \frac{\log 10 \log 10\left(8.122 \times 10^6 \mu + 0.8\right) + 3.55}{10.04},$$

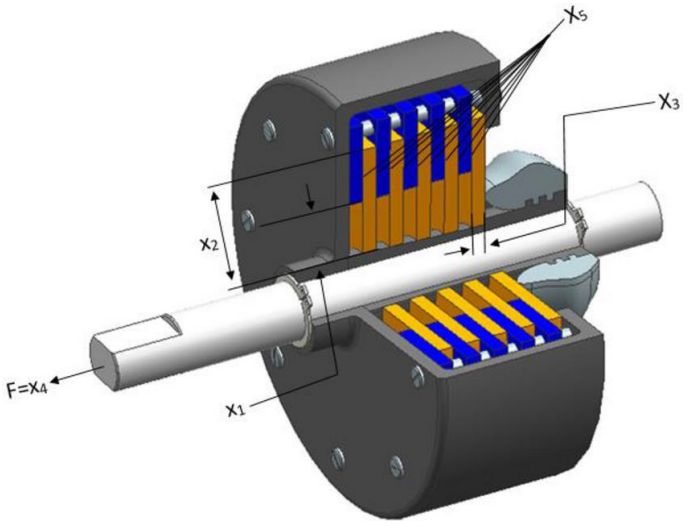
$$h = \left(\frac{2\pi \times 750}{60}\right)^2 \frac{2\pi\mu}{E_f}\left(\frac{R^4 - R_0^4}{4}\right)$$

Durum değişkenlerinin alabileceği değer aralıkları Eşitlik 4.48'de verilmiştir.

$$1 \le R \le 16,\ 1 \le R_0 \le 16,\ 1 \le Q \le 16,\ 1 \times 10^{-6} \le \mu \le 16 \times 10^{-6} \tag{4.48}$$

4.6 ÇOKLU DİSK KAVRAMALI FREN TASARIM PROBLEMİ

Çok disk kavramalı fren tasarım problemi, mühendislik kısıtlı optimizasyonunda iyi bilinen sorunlardan biridir. Bunun amacı çok diskli kavrama freninin kütlesini en aza indirmektir. Karar/durum değişkenleri x_1, x_2, x_3, x_4 ve x_5 sırasıyla iç yarıçap, dış yarıçap, diskin kalınlığı, harekete geçirme kuvveti ve sürtünme yüzeylerinin sayısıdır ve değişkenlerin tümü tam sayı aralığındadır. Çok diskli kavramalı fren tasarım probleminin şematiği Şekil 4.5'te gösterilmektedir [55].



Şekil 4-5: Çoklu Disk Kavramalı Fren Tasarımının Şematiği [55].

Çoklu disk kavramalı fren tasarımı probleminin uygunluk fonksiyonu Eşitlik 4.49'da verilmiştir.

$$f(\vec{x}) = \pi\left(x_2^2 - x_1^2\right) x_3 \left(x_5 + 1\right) \rho \tag{4.49}$$

Problemdeki kısıtlar Eşitlik 4.50'de verilmiştir.

$$\begin{aligned}
&g_1(\vec{x}) = -p_{\max} + p_{rz} \le 0, \\
&g_2(\vec{x}) = -p_{rz} V_{sr} - V_{sr,\max} p_{\max} \le 0, \\
&g_3(\vec{x}) = \Delta R + x_1 - x_2 \le 0, \\
&g_4(\vec{x}) = -L_{\max} + \left(x_5 + 1\right)\left(x_3 + \delta\right) \le 0, \\
&g_5(\vec{x}) = sM_s - M_h \le 0, \\
&g_6(\vec{x}) = T \ge 0, \\
&g_7(\vec{x}) = -V_{sr,\max} + V_{sr} \le 0, \\
&g_8(\vec{x}) = T - T_{\max} \le 0,
\end{aligned} \tag{4.50}$$

Eşitlik 4.51'de bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

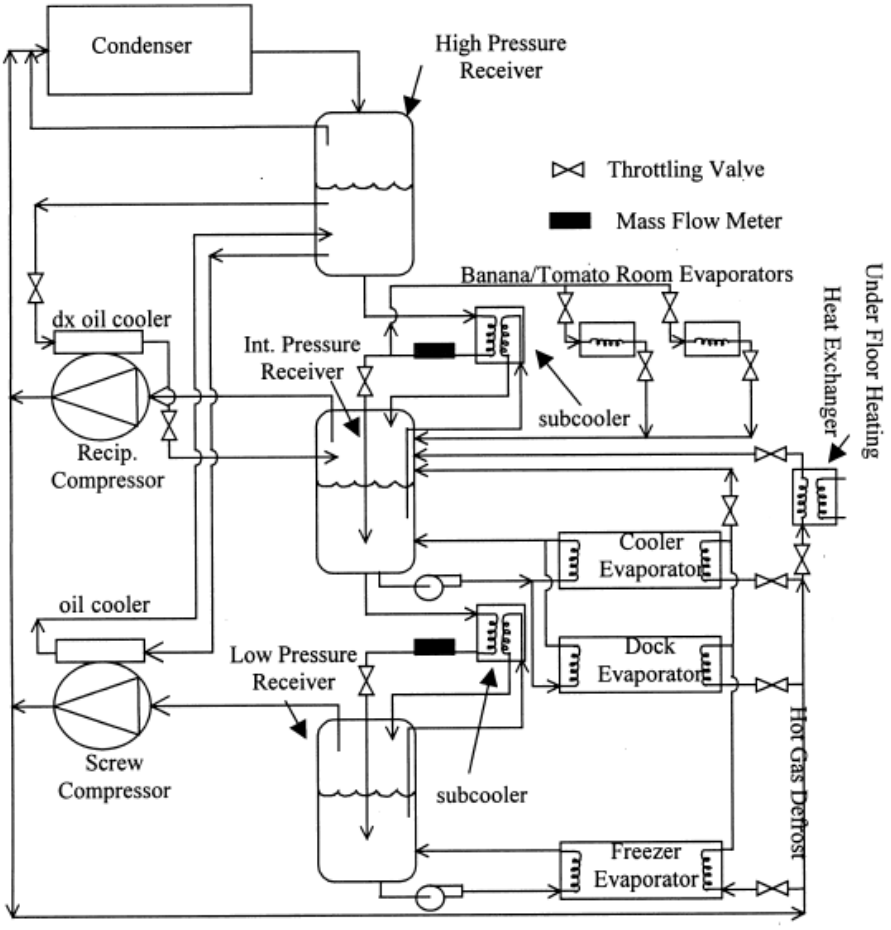
$$\begin{aligned}
&M_h = \frac{2}{3}\mu x_4 x_5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2} N.mm, \\
&\omega = \frac{\pi n}{30} rad/s, \quad A = \pi\left(x_2^2 - x_1^2\right) mm^2, \\
&p_{rz} = \frac{x^4}{A} N/mm^2, \quad V_{sr} = \frac{\pi R_{sr} n}{30} mm/s, \\
&R_{sr} = \frac{2}{3}\frac{x_2^3 - x_1^3}{x_2^2 x_1^2} mm, \quad T = \frac{I_z \omega}{M_h + M_f}, \\
&\Delta R = 20mm, \quad L_{\max} = 30mm, \quad \mu = 0.6, \\
&V_{sr,\max} = 10m/s, \quad \delta = 0.5mm, \quad s = 1.5, \\
&T_{\max} = 15s, n = 250rpm, I_z = 55Kg.m^2, \\
&M_s = 40Nm, M_f = 3Nm, and p_{\max} = 1.
\end{aligned} \tag{4.51}$$

Durum değişkenlerinin alabileceği değer aralıkları Eşitlik 4.52'de verilmiştir.

$$
\begin{aligned}
&60 \le x_1 \le 80,\\
&90 \le x_2 110,\\
&1 \le x_3 \le 3,\\
&0 \le x_4 \le 1000,\\
&2 \le x_5 \le 9.
\end{aligned}
\tag{4.52}
$$

4.7 ENDÜSTRİYEL SOĞUTMA SİSTEMİNİN OPTİMAL TASARIMI

Endüstriyel soğutma sisteminin problemi, maliyeti en aza indirmek ve müşteriler için en uygun soğutma sistemini üretmek için en iyi soğutucu akışkanları, ideal sıcaklık seviyelerini, uygun çevrim konfigürasyonunu ve en iyi sıkıştırma teknolojisini bulmayı amaçlamaktadır. Şekil 4.6'da temsili bir şematiği gösterilen problem, doğrusal olmayan eşitsizlik kısıtlamalı optimizasyon problemi olarak formüle edilebilir.



Şekil 4-6: Endüstriyel Soğutma Sisteminin Şematiği [56].

Endüstriyel soğutma sistemi probleminin uygunluk fonksiyonu Eşitlik 4.53'te verilmiştir.

$$\begin{aligned} f(\vec{x}) = {} & 63098.88x_2x_4x_{12} + 5441.5x_2^2x_{12} + 115055.5x_2^{1.664} + \\ & 6172.27x_2^2x_6 + 63098.88x_1x_3x_{11} + 5441.5x_1^2x_{11} + 115055.5x_1^{1.664}x_5 + \\ & 6172.27x_1^2x_5 + 140.53x_1x_{11} + 281.29x_3x_{11} + 70.26x_1^2 + 281.29x_1x_3 + \\ & 281.29x_3^2 + 11437x_8^{1.8812}x_{12}^{0.3424}x_{10}x_{14}^{-1}x_1^2x_7x_9^{-1} + 20470.2x_7^{2.893}x_{11}^{0.316}x_1^2 \end{aligned} \tag{4.53}$$

Problemdeki kısıtlar Eşitlik 4.54'te verilmiştir.

$$\begin{aligned} & g_1(\vec{x}) = 1.524x_7^{-1} \le 1, \\ & g_2(\vec{x}) = 1.524x_8^{-1} \le 1, \\ & g_3(\vec{x}) = 0.07789x_1 - 2x_7^{-1}x_9 - 1 \le 0, \\ & g_4(\vec{x}) = 7.05305x_9^{-1}x_1^2x_{10}x_8^{-1}x_2^{-1}x_{14}^{-1} - 1 \le 0, \\ & g_5(\vec{x}) = 0.0833x_{13}^{-1}x_{14} - 1 \le 0, \\ & g_6(\vec{x}) = 47.136x_2^{0.333}x_{10}^{-1}x_{12} - 1.333x_8x_{13}^{2.1195} + 62.08x_{13}^{2.1195}x_{12}^{-1}x_8^{0.2}x_{10}^{-1} - 1 \le 0, \\ & g_7(\vec{x}) = 0.04771x_{10}x_8^{1.8812}x_{12}^{0.3424} - 1 \le 0, \\ & g_8(\vec{x}) = 0.0488x_9x_7^{1.893}x_{11}^{0.316} - 1 \le 0, \\ & g_9(\vec{x}) = 0.0099x_1x_3^{-1} - 1 \le 0, \\ & g_{10}(\vec{x}) = 0.0193x_2x_4^{-1} - 1 \le 0, \\ & g_{11}(\vec{x}) = 0.0298x_1x_5^{-1} - 1 \le 0, \\ & g_{12}(\vec{x}) = 0.056x_2x_6^{-1} - 1 \le 0, \\ & g_{13}(\vec{x}) = 2x_9^{-1} - 1 \le 0, \\ & g_{14}(\vec{x}) = 2x_{10}^{-1} - 1 \le 0, \\ & g_{15}(\vec{x}) = x_{12}x_{11}^{-1} - 1 \le 0 \end{aligned} \tag{4.54}$$

Eşitlik 4.55'te bazı değişkenlerin sınır değerleri verilmiştir.

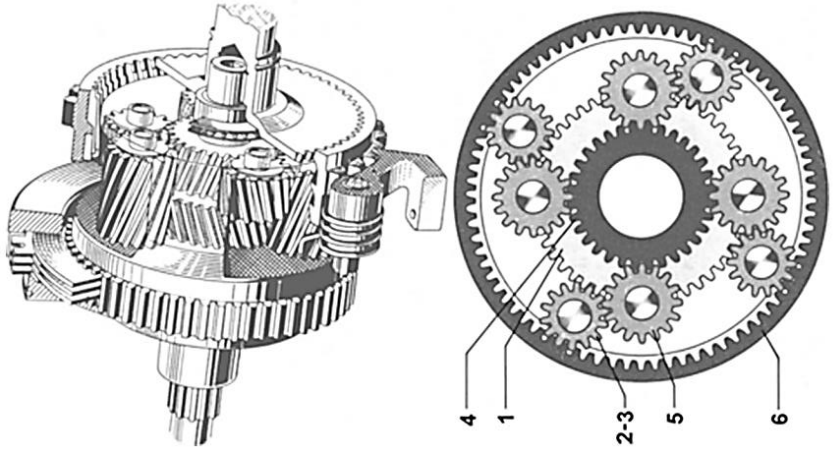
$$0.001 \le x_i \le 5, \; i = 1,...,14 \tag{4.55}$$

4.8 PLANET DİŞLİ ZİNCİRİ TASARIM PROBLEMİ

Bu problemin temel amacı otomobillerde kullanılan dişli oranlarındaki maksimum hataları en aza indirmektir. Maksimum hatayı minimuma indirmek amacıyla, planet dişli zincir sistemi

için toplam diş sayısı hesaplanır. Şekil 4.7'de gösterildiği gibi bu problem, altı tamsayı değişkeni olmak üzere farklı geometrik ve montaj kısıtlamalarını da içeren on bir kısıt içermektedir [57, 58]. Planet dişli zinciri tasarımı probleminin uygunluk fonksiyonu Eşitlik 4.56'da verilmiştir.

$$
\begin{aligned}
&f(\vec{x}) = \max\left|i_k - i_{0k}\right|, \quad k = \{1,2,...,R\}, \\
&i_1 = \frac{N_6}{N_4}, i_{01} = 3.11, \quad i_2 = \frac{N_6\left(N_1N_3 + N_2N_4\right)}{N_1N_3\left(N_6 + N_4\right)}, \quad i_{0R} = -3.11, \\
&I_R = \frac{N_2N_6}{N_1N_3}, \quad i_{02} = 1.84, \quad \vec{x} = \{p, N_6, N_5, N_4, N_3, N_2, N_1, m_2, m_1\}
\end{aligned}
\tag{4.56}
$$



Şekil 4-7: Planet Dişli Zinciri Tasarımının Şematiği [59].

Problemdeki kısıtlar Eşitlik 4.57'de verilmiştir.

$$
\begin{aligned}
&g_1(\vec{x}) = m_3\left(N_6 + 2.5\right) - D_{\max} \le 0, \\
&g_2(\vec{x}) = m_1\left(N_1 + N_2\right) + m_1\left(N_2 + 2\right) - D_{\max} \le 0, \\
&g_3(\vec{x}) = m_3\left(N_4 + N_5\right) + m_3\left(N_5 + 2\right) - D_{\max} \le 0, \\
&g_4(\vec{x}) = \left|m_1\left(N_1 + N_2\right) + m_3\left(N_6 + N_3\right)\right| - m_1 - m_3 \le 0, \\
&g_5(\vec{x}) = -\left(N_1 + N_2\right)\sin\left(\pi / p\right) + N_2 + 2 + \delta_{22} \le 0, \\
&g_6(\vec{x}) = -\left(N_6 + N_3\right)\sin\left(\pi / p\right) + N_3 + 2 + \delta_{33} \le 0, \\
&g_7(\vec{x}) = -\left(N_4 + N_5\right)\sin\left(\pi / p\right) + N_5 + 2 + \delta_{55} \le 0, \\
&g_8(\vec{x}) = \left(N_3 + N_5 + 2 + \delta_{35}\right)^2 - \left(N_6 + N_3\right)^2 - \left(N_4 + N_5\right)^2 + \\
&\qquad 2\left(N_6 - N_3\right)\left(N_4 + N_5\right)\cos\left(2\pi p^{-1} - \beta\right) \le 0, \\
&g_9(\vec{x}) = N_4 - N_6 + 2N_5 + 2\delta_{56} + 4 \le 0, \\
&g_{10}(\vec{x}) = 2N_3 - N_6 + N_4 + 2\delta_{34} + 4 \le 0, \\
&h_1(\vec{x}) = \frac{N_6 - N_4}{p},
\end{aligned}
\tag{4.57}
$$

Eşitlik 4.58'de bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

$$\begin{aligned}
&\delta_{22} = \delta_{33} = \delta_{55} = \delta_{35} = \delta_{56} = 0.5. \\
&\beta = \frac{\cos^{-1}\left(\left(N_4 + N_5\right)^2 + \left(N_6 - N_3\right)^2 - \left(N_3 + N_5\right)^2\right)}{2\left(N_6 - N_3\right)\left(N_4 + N_5\right)}, \\
&D_{\max} = 220, \quad p = (3,4,5), \\
&m_1 = (1.75, 2.0, 2.25, 2.5, 2.75, 3.0), \\
&m_3 = (1.75, 2.0, 2.25, 2.5, 2.75, 3.0),
\end{aligned} \tag{4.58}$$

Eşitlik 4.59'da durum değişkenlerinin sınır değerleri verilmiştir.

$$\begin{aligned}
&17 \le N_1 \le 96,\ 17 \le N_4 \le 46,\ 14 \le N_2 \le 54, 14 \le N_3 \le 51 \\
&14 \le N_5 \le 51, 48 \le N_6 \le 124,\ N_i \in \mathbb{Z}
\end{aligned} \tag{4.59}$$

5. DENEYSEL ÇALIŞMALAR

Bu bölümde, tez çalışmasında kullandığımız metasezgisel algoritmaların, Bölüm 4'te teknik ayrıntıları verilen mühendislik tasarım problemlerinin farklı yineleme ve popülasyon sayılarına göre yakınsama değerleri test edilmiştir. Deneysel çalışmalardaki yineleme sayısı sırasıyla 100, 500 ve 1000 olup, popülasyon sayısı ise 10 ve 50 olacak şekilde tertip edilmiştir. Ayrıca, her bir deneyde metasezgisel algoritmaların çalışma süreleri de ölçülerek sonuçları karşılaştırılmıştır.

Bazı metasezgisel yöntemler, daha iyi sonuçlar elde edebilmek için başlangıç ya da sabit parametrelerinin önceden ayarlanması gerekir. Deneysel çalışmada kullandığımız algoritmalarının parametreleri, geliştiriciler tarafından tercih edilen veya önerilen değerlere göre tayin edilmiştir.

FA algoritması için ışık emilim katsayısı (γ) 1.0, çekicilik katsayısının başlangıç değeri (β_0) 2.0, α parametresinin değeri 0.2 ve alfanın rastgelelik azaltma faktörü (ϑ) 0.97 olarak tayin edilmiştir. PSO'da atalet ağırlığı ve atalet sönümleme oranları sırasıyla 1.0 ve 0.99 olarak, c_1 ve c_2 değişkenleri ise sırasıyla 1.5 ve 2.0 olarak belirlenmiştir. GWO ve PFA algoritmaları için önceden ayarlanması gereken herhangi bir parametre bulunmamaktadır.

5.1 PERFORMANS ÖLÇÜM KRİTERLERİ

Deneysel çalışmalarda kullanılan metasezgisel algoritmaların, performans ölçüm kriterleri olarak (30 kez tekrarlı olarak çalıştırılmış) en iyi uygunluk değerlerinin ortalaması kullanılmıştır. Buna ek olarak, algoritmaların çalışma esnasında harcadıkları sürelerin ortalaması ise diğer bir ölçüm kriteri olarak belirlenmiştir. Çalışma zamanı analiz bilgisayar sisteminin özellikleri şu şekildedir: Windows 10 işletim sistemi, Intel i5 2.4 GHz işlemci ve 8 GB sistem belleği.

5.2 MÜHENDİSLİK PROBLEMLERİNİN TESTLERİ

Bu çalışmada, literatürde iyi bilinen dört algoritmayı (FA: Firefly Algorithm, GWO: Grey Wolf Optimizer, PSO: Particle Swarm Optimization ve PFA: Pathfinder Algorithm) test etmek amacıyla gerçek dünya mühendislik problemleri kullanılmıştır. Bu problemler; Dört Kademeli Dişli Kutusu Problemi (DKDKP), Gaz İletim Kompresörü Tasarımı (GİKP), Dişli Takımı Tasarım Problemi (DTTP), Himmelblau'nun Fonksiyonu (HF), Hidrostatik Baskı Yatağı Tasarımı Problemi (HBYTP), Çoklu Disk Kavramalı Fren Tasarım Problemi (ÇDKFTP), Endüstriyel

Soğutma Sisteminin Optimal Tasarımı (ESSOT) ve Planet Dişli Zinciri Tasarım Problemi (PDZTP) olmak üzere sekiz tanedir. Tablo 5.1'de deneysel çalışmalarda kullanılan mühendislik problemleri ve bunların özellikleri verilmiştir.

Tablo 5-1: Mühendislik Problemlerinin Özellikleri.

#	Problem	Boyut	Kısıtlar	Alt Sınır	Üst Sınır
1	DKDKP: Dört Kademeli Dişli Kutusu Problemi	22	86	6,51	76,49
2	GİKP: Gaz İletim Kompresörü Tasarımı	4	1	20	50
3	DTTP: Dişli Takımı Tasarım Problemi	4	1	12	60
4	HF: Himmelblau'nun Fonksiyonu	5	6	78	102
5	HBYTP: Hidrostatik Baskı Yatağı Tasarımı Problemi	4	7	1	16
6	ÇDKFTP: Çoklu Disk Kavramalı Fren Tasarım Problemi	5	8	60	80
7	ESSOT: Endüstriyel Soğutma Sisteminin Optimal Tasarımı	14	15	0,001	5
8	PDZTP: Planet Dişli Zinciri Tasarım Problemi	9	10	16,51	96,49

Tablo 5.1'de belirtilen özelliklerin açıklamaları şöyledir:

- ***Boyut***: Problemin tasarım/karar değişkenlerinin sayısı,
- ***Kısıtlar***: Problemin kısıt fonksiyonlarının sayısı,
- ***Alt Sınır***: Tasarım/karar değişkenlerinin alabileceği en küçük sayı değerleri,
- ***Üst Sınır***: Tasarım/karar değişkenlerinin alabileceği en büyük sayı değerlerini ifade etmektedir.

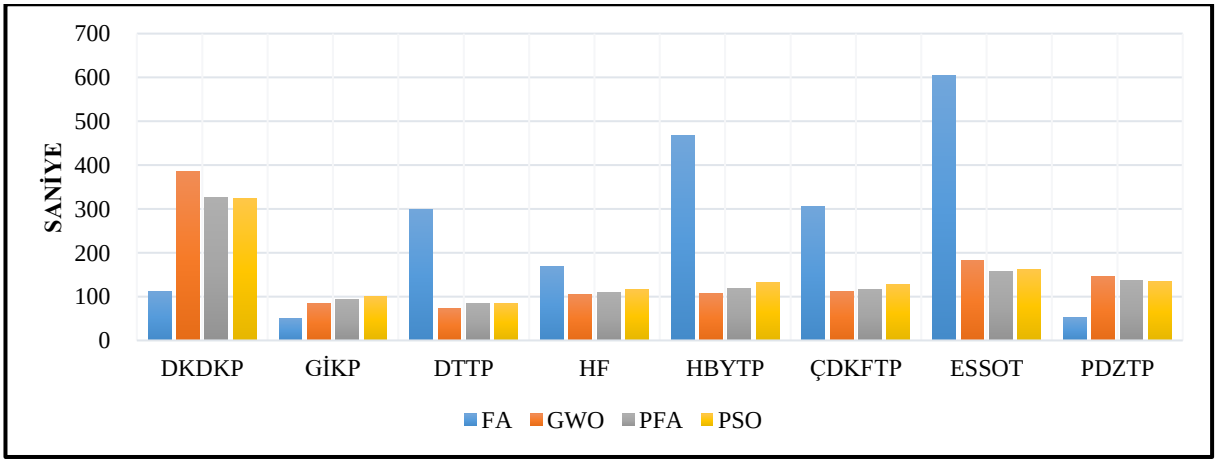
5.2.1 Deney I

Bu deneyde, yineleme sayısı 100, popülasyon sayısı 10 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen global en iyi (*global best*) skorların ortalaması alınmıştır. Tablo 5.2'de, Deney I'e göre mühendislik problemlerinin global en iyi skorları verilmiştir.

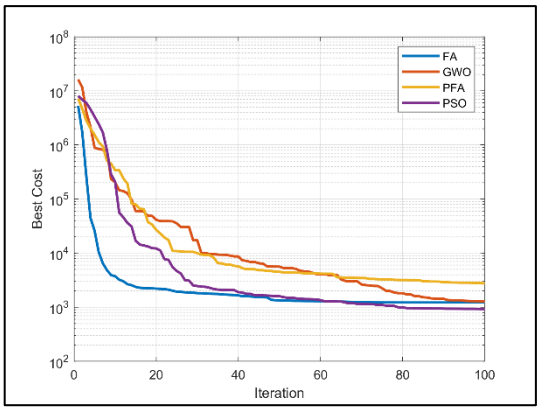
Tablo 5-2: Deney I'e Göre Algoritmaların Global En İyi Skorları.

#	Problem	En İyi Skor	FA	GWO	PFA	PSO
1	DKDKP	PSO	1223,3522	1287,3518	2748,6194	924,1798
2	GİKP	FA GWO, PFA	977874,1517	977874,1517	977874,1517	977876,6615
3	DTTP	FA	5,76105E-13	4,29007E-10	1,17273E-09	9,99127E-13
4	HF	GWO, PFA	-32214,6656	-32214,7639	-32214,7639	-32214,6287
5	HBYTP	FA	916,9199	3098,3970	2836,5964	1274,6639
6	ÇDKFTP	PFA	0,2390	0,2360	0,2353	0,2436
7	ESSOT	GWO	36,5383	0,1901	6,8543	100,2596
8	PDZTP	PFA	0,5505	0,5938	0,5351	0,5446

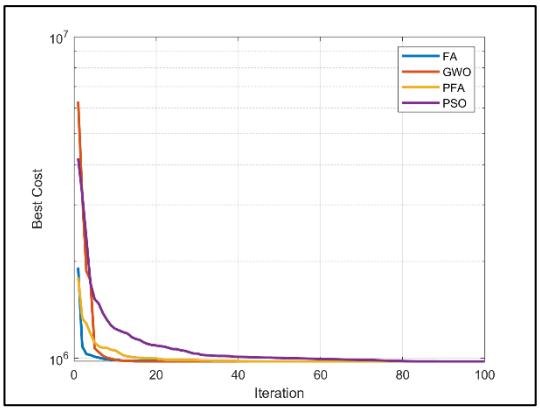
PFA algoritması 1, 3, 5 ve 7 nolu problem haricinde tüm deneylerde en iyi skoru elde etmiştir. FA ve GWO algoritmaları, üç deneyde en iyi skoru elde ederek ikinci sırada olup, PSO ise yalnızca DKDKP'da en iyi skoru elde ederek enson sırada yer almaktadır. Optimizasyon algoritmalarının çalışma süreleri, Şekil 5.1'de gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. GWO, 1, 7 ve 8 nolu problem dışında tüm deneylerde en hızlı çalışan algoritma olmuştur. Şekil 5.2'de, Deney I'e göre mühendislik problemlerinin yakınsama grafikleri verilmiştir.



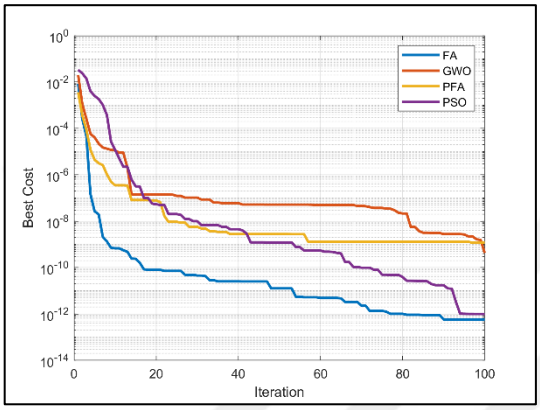
Şekil 5-1: Deney I'e göre algoritmaların çalışma süreleri.



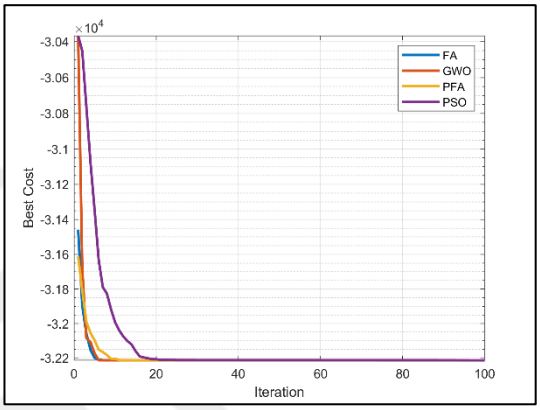
DKDKP



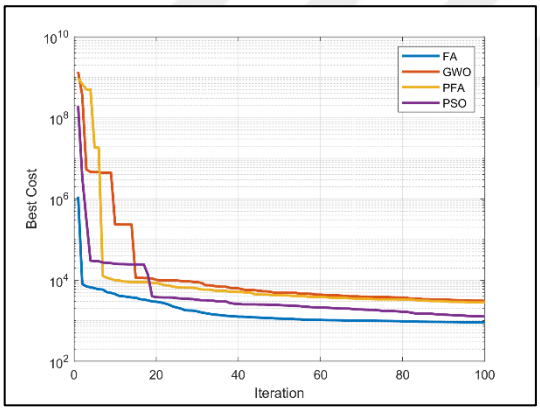
GİKP



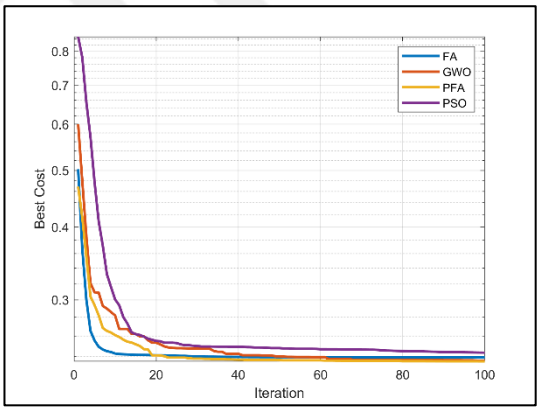
DTTP



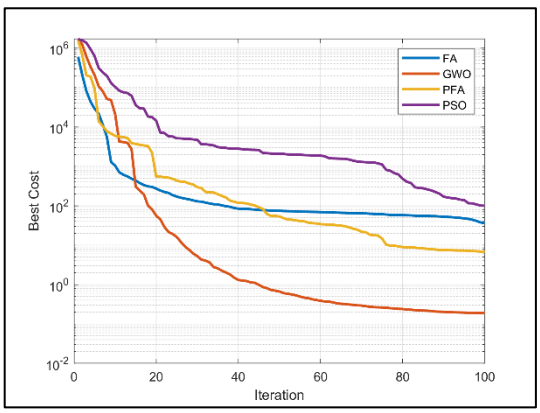
HF



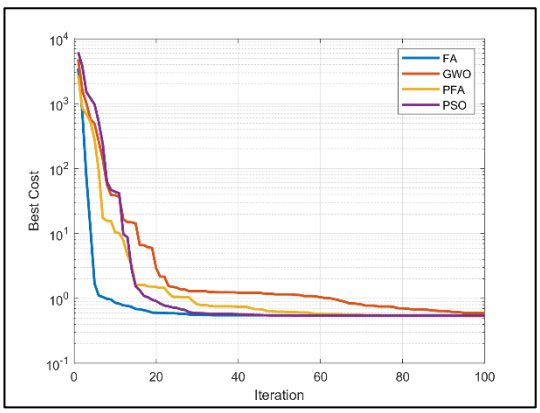
HBYTP



ÇDKFTP



ESSOT



PDZTP

Şekil 5-2: Deney I'e göre algoritmaların yakınsama grafikleri.

Deney I'e göre Şekil 5.2'de verilen algoritmaların yakınsama grafikleri incelendiğinde, HF problemi için tüm algoritmalar yaklaşık 20. iterasyonda ideal sonuca eriştiği görülmektedir. GİKP, ÇDKFTP ve ESSOT problemlerinde ise PSO algoritması en yavaş yakınsayan algoritma olmuştur. FA ve PSO algoritmaları, DTTP probleminde diğerlerine göre daha başarılı sonuçlar verdiği görülmektedir. PDZTP probleminde, tüm algoritmalar 100. yinelemede birbirlerine çok yaklaşmıştır. Ancak, DKDKP, DTTP, HBYTP ve ESSOT problemlerinin çözümü için iterasyon sayısının yetersiz olduğu da göze çarpmaktadır.

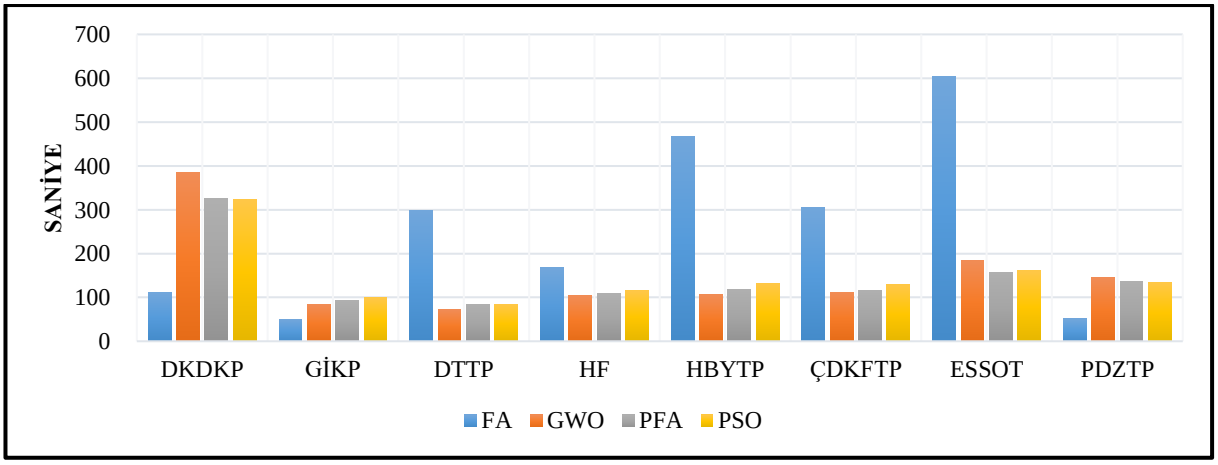
5.2.2 Deney II

Bu deneyde, yineleme sayısı 100, popülasyon sayısı 50 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen global en iyi (*global best*) skorların ortalaması alınmıştır. Tablo 5.3'te, Deney II'ye göre mühendislik problemlerinin global en iyi skorları verilmiştir.

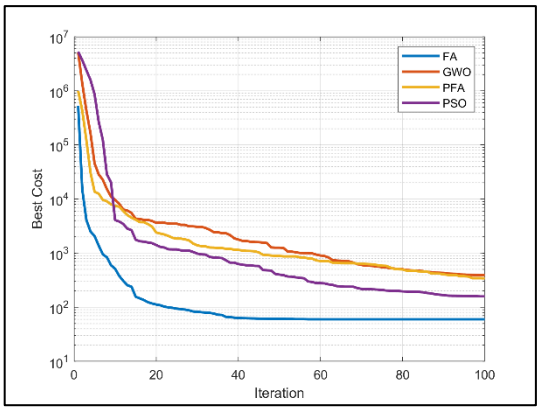
Tablo 5-3: Deney II'ye göre algoritmaların global en iyi (best) skorları.

#	Problem	En İyi Skor	FA	GWO	PFA	PSO
1	DKDKP	FA	59,8665	394,1841	342,3570	158,5389
2	GİKP	Hepsi	977874,1517	977874,1517	977874,1517	977874,1517
3	DTTP	FA	3,84232E-15	4,75318E-11	4,42424E-11	8,66905E-13
4	HF	Hepsi	-32214,7639	-32214,7639	-32214,7639	-32214,7639
5	HBYTP	FA	140,7186	1069,5782	1426,5683	849,1192
6	ÇDKFTP	FA, PFA, PSO	0,2352	0,2353	0,2352	0,2352
7	ESSOT	GWO	1,0806	0,8678	1,0940	2,6158
8	PDZTP	FA	0,5263	0,5293	0,5316	0,5302

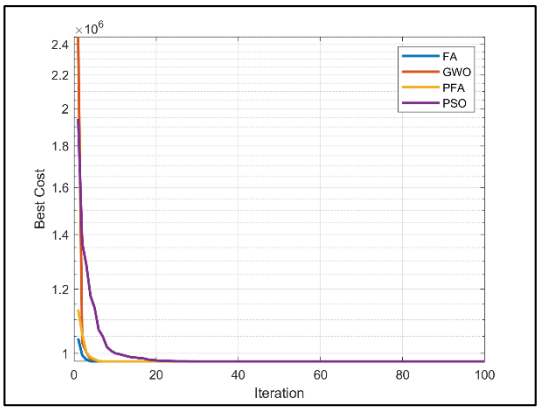
FA algoritması, ESSOT problemi dışındaki tüm deneylerde en iyi skoru elde etmiştir. Diğer optimizasyon algoritmaları ise üç deneyde en iyi skoru elde ederek ikinci sırada yer almaktadır. Algoritmalarının çalışma süreleri, Şekil 5.3'te gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. GWO, 1, 7 ve 8 nolu problem dışında tüm deneylerde en hızlı sonuç üreten algoritma olmuştur. Şekil 5.4'te, Deney II'ye göre mühendislik problemlerinin yakınsama grafikleri verilmiştir.



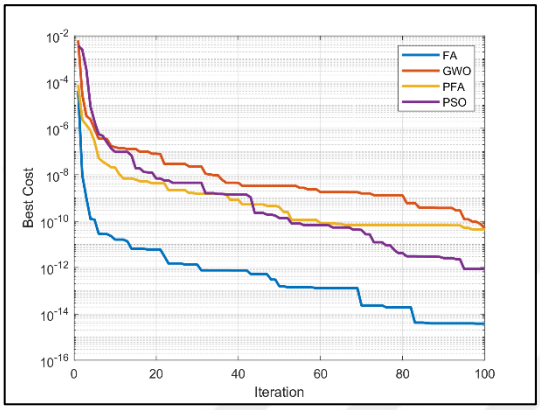
Şekil 5-3: Deney II'ye göre algoritmaların çalışma süreleri.



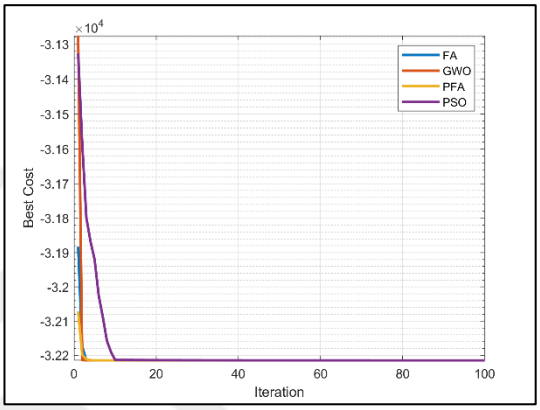
DKDKP



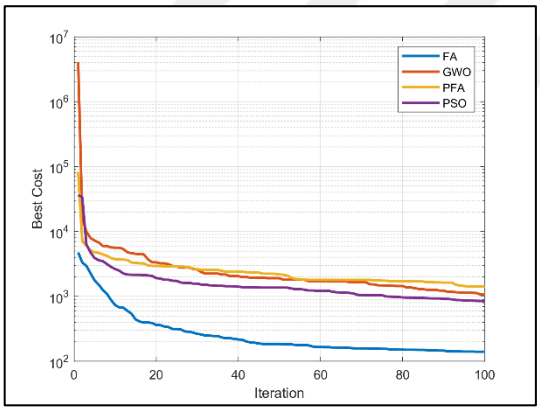
GİKP



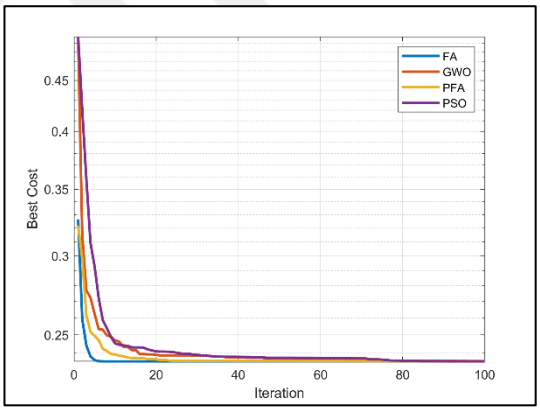
DTTP



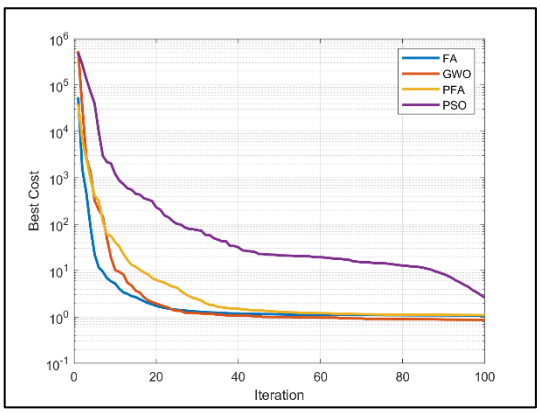
HF



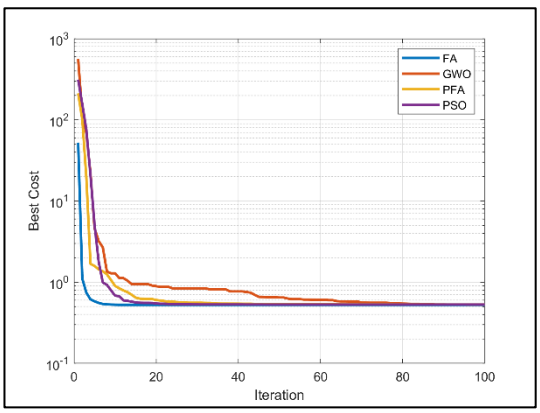
HBYTP



ÇDKFTP



ESSOT



PDZTP

Şekil 5-4: Deney II'ye göre algoritmaların yakınsama grafikleri.

Deney II'ye göre Şekil 5.4'te verilen algoritmaların yakınsama grafikleri incelendiğinde, ilk deneyde olduğu gibi DKDKP, DTTP, HBYTP ve ESSOT problemlerinin çözümü için iterasyon sayısının yetersiz olduğu ancak diğer problemler için makul bir değer olduğu görülmektedir. Poülasyon sayısının bir önceki deneye göre daha yüksek olması, algoritmaların daha hızlı yakınsamasına sebep olmuştur. FA algoritması, DKDKP, DTTP ve HBYTP problemlerinde diğerlerine göre çok daha iyi yakınsama sağlamıştır. PSO ise ESSOT probleminde en yavaş yakınsayan algoritma olduğu göze çarpmaktadır.

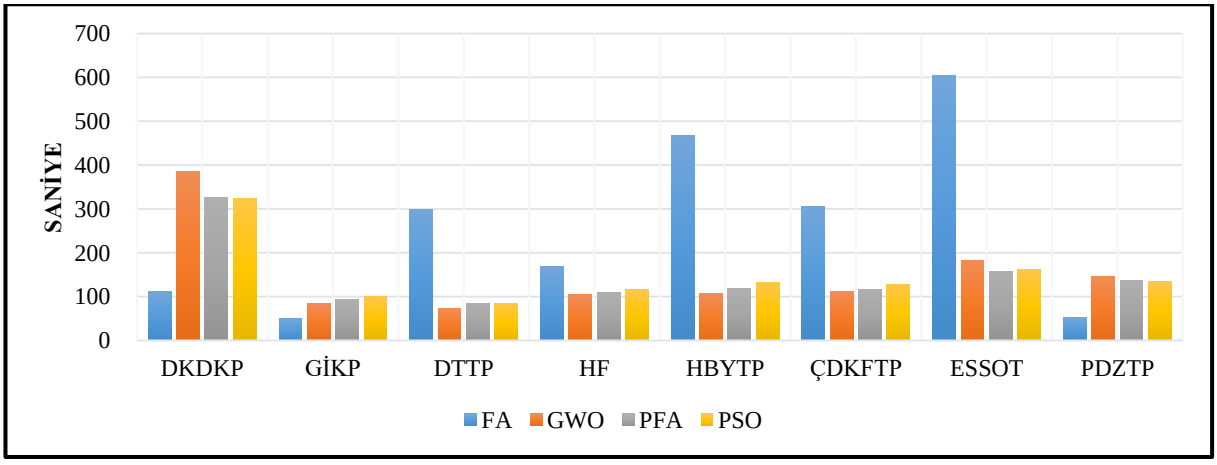
5.2.3 Deney III

Bu deneyde, yineleme sayısı 500, popülasyon sayısı 10 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen global en iyi (*global best*) skorların ortalaması alınmıştır. Tablo 5.4'te, Deney III'e göre mühendislik problemlerinin global en iyi skorları verilmiştir.

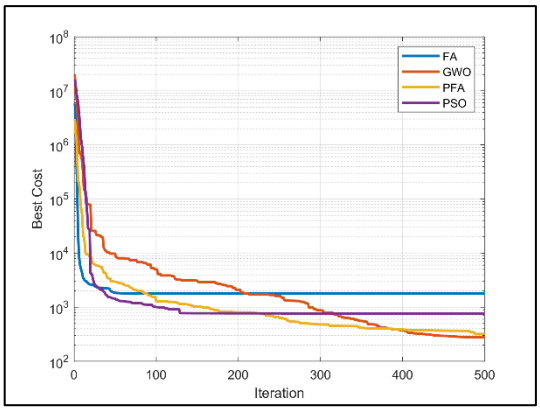
Tablo 5-4: Deney III'e göre algoritmaların global en iyi (best) skorları.

#	Problem	En İyi Skor	FA	GWO	PFA	PSO
1	DKDKP	GWO	1794,9195	277,1305	319,5092	763,6251
2	GİKP	Hepsi	977874,1517	977874,1517	977874,1517	977874,1517
3	DTTP	FA	1,57079E-23	3,89549E-11	2,47409E-11	1,26644E-18
4	HF	GWO, PFA, PSO	-32214,6909	-32214,7639	-32214,7639	-32214,7639
5	HBYTP	FA	335,8547	1184,9533	904,4497	917,2933
6	ÇDKFTP	GWO, PFA	0,2362	0,2352	0,2352	0,2375
7	ESSOT	GWO	3,8427	0,3343	0,6539	1,5893
8	PDZTP	PFA	0,5659	0,5322	0,5283	0,5688

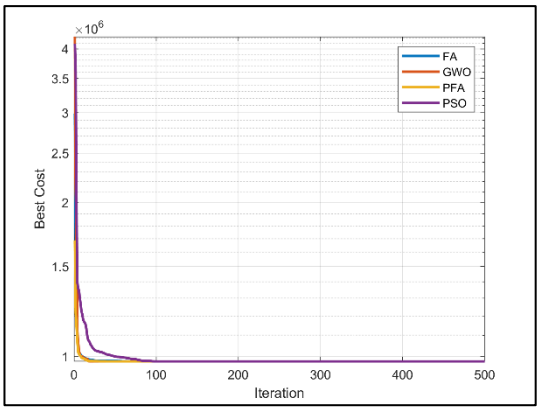
GWO algoritması 3, 5 ve 8 nolu problem haricinde tüm deneylerde en iyi skoru elde etmiştir. PFA, dört deneyde en iyi skoru elde ederek ikinci sırada, FA, üç deneyde en iyi skoru elde ederek üçüncü sırada ve PSO ise iki deneyde en iyi skoru elde ederek sonuncu sırada yer almaktadır. Algoritmalarının çalışma süreleri, Şekil 5.5'te gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. GWO, dört problem çözümünde, FA, üç problem çözümünde ve PFA ise yalnızca ESSOT probleminin çözümünde en hızlı çalışan algoritma olmuştur. Şekil 5.6'da, Deney III'e göre mühendislik problemlerinin yakınsama grafikleri verilmiştir.



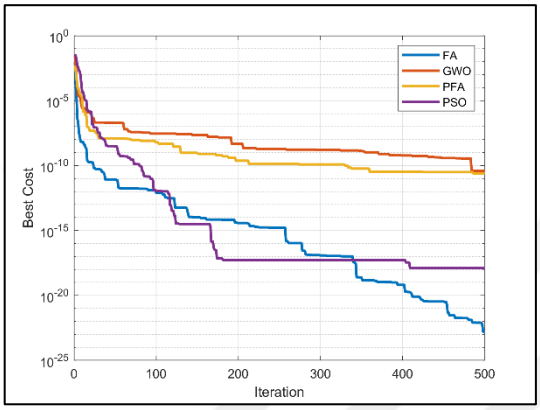
Şekil 5-5: Deney III'e göre algoritmaların çalışma süreleri.



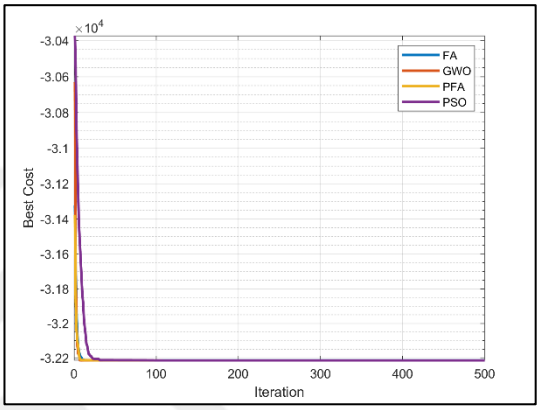
DKDKP



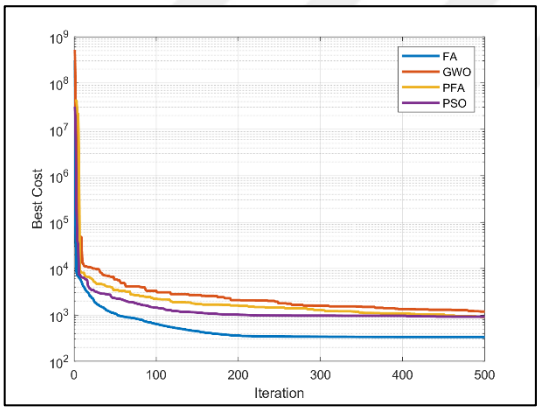
GİKP



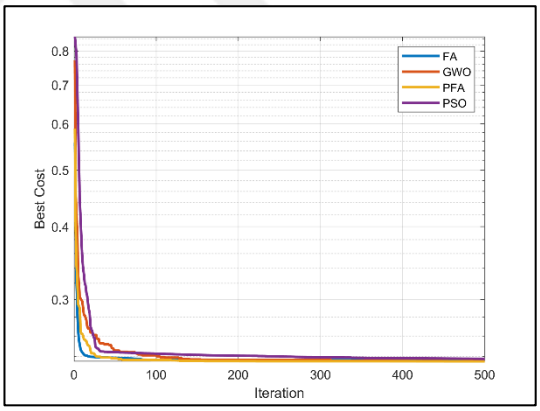
DTTP



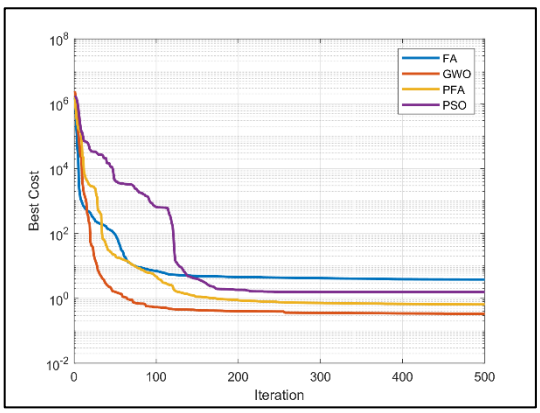
HF



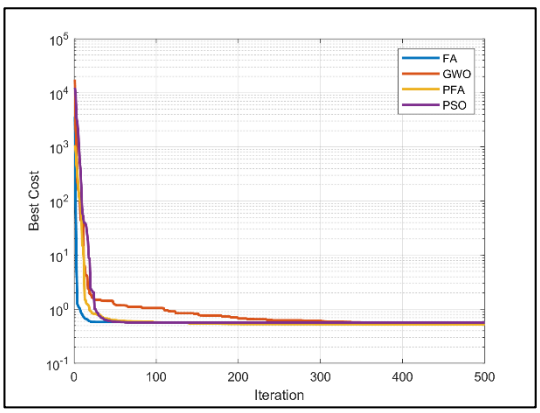
HBYTP



ÇDKFTP



ESSOT



PDZTP

Şekil 5-6: Deney III'e göre algoritmaların yakınsama grafikleri.

Deney III'e göre Şekil 5.6'da verilen algoritmaların yakınsama grafikleri incelendiğinde, metasezgisel yöntemlerin tümü GİKP, HF, ÇDKFTP ve PDZTP problemlerinde 500 yinelemeden daha az bir turda optimal sonuçları elde etmişlerdir. FA ve PSO algoritması, DKDKP probleminde yerel çözüme takıldığı için uzun bir yineleme boyunca değişim göstermemiştir. FA algoritması, DTTP ve HBYTP probleminde en iyi yakınsama gösteren yöntemdir.

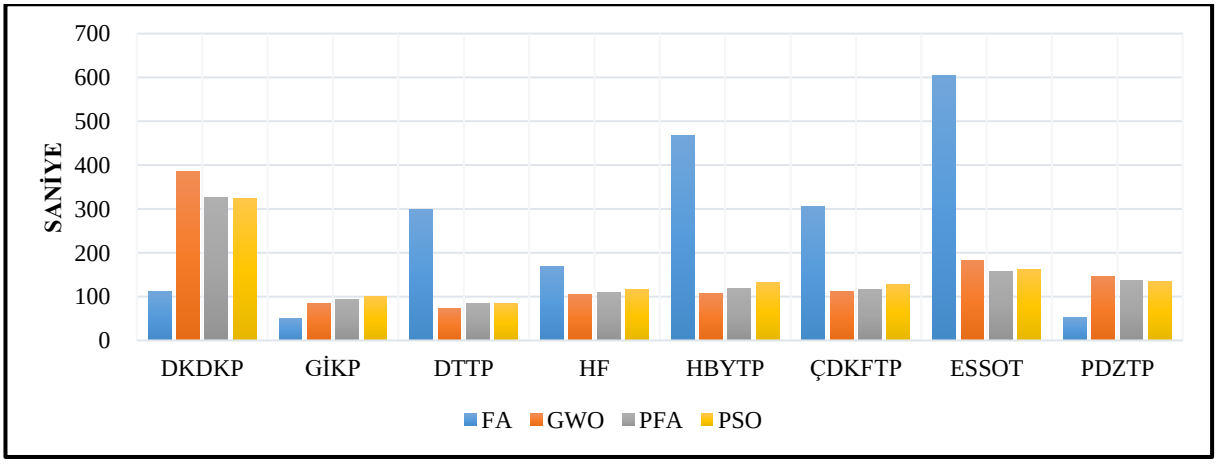
5.2.4 Deney IV

Bu deneyde, yineleme sayısı 500, popülasyon sayısı 50 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen global en iyi (*global best*) skorların ortalaması alınmıştır. Tablo 5.5'te, Deney IV'e göre mühendislik problemlerinin global en iyi skorları verilmiştir.

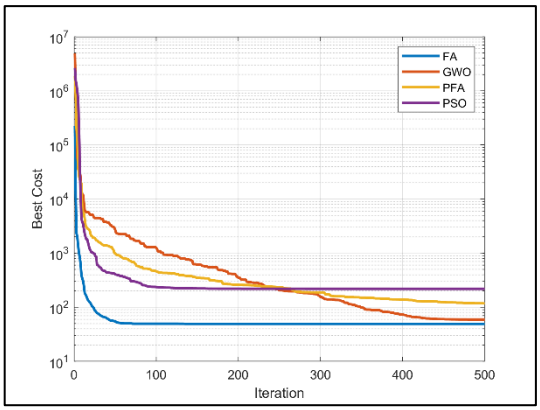
Tablo 5-5: Deney IV'e göre algoritmaların global en iyi (best) skorları.

#	Problem	En İyi Skor	FA	GWO	PFA	PSO
1	DKDKP	FA	49,2708	59,4591	117,6255	219,2433
2	GİKP	Hepsi	977874,1517	977874,1517	977874,1517	977874,1517
3	DTTP	PSO	3,07727E-21	1,71503E-12	1,24102E-12	8,97175E-31
4	HF	Hepsi	-32214,7639	-32214,7639	-32214,7639	-32214,7639
5	HBYTP	FA	126,1243	506,4470	544,8927	236,5831
6	ÇDKFTP	Hepsi	0,2352	0,2352	0,2352	0,2352
7	ESSOT	PSO	0,8034	0,0364	0,5473	0,0336
8	PDZTP	PFA	0,5267	0,5311	0,5261	0,5286

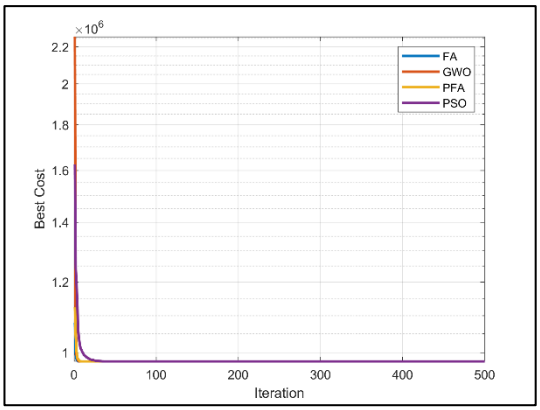
FA algoritması 3, 7 ve 8 nolu problem haricinde tüm deneylerde, PSO algoritması 1, 5 ve 8 nolu problem haricinde tüm deneylerde en iyi skoru elde ederek birinci sıradadır. PFA, dört deneyde en iyi skoru elde ederek ikinci sırada, GWO, üç deneyde en iyi skoru elde ederek sonuncu sırada yer almaktadır. Optimizasyon algoritmalarının çalışma süreleri, Şekil 5.7'de gösterilmiştir. GWO, dört problemin çözümünde, FA, üç problemin çözümünde ve PFA ise yalnızca bir problemin çözümünde en hızlı çalışan algoritma olmuştur. Şekil 5.8'de, Deney IV'e göre mühendislik problemlerinin yakınsama grafikleri verilmiştir.



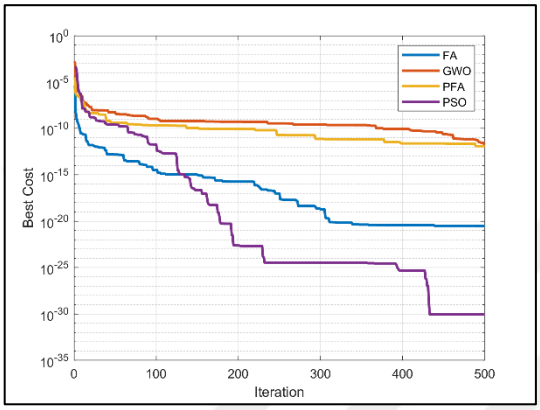
Şekil 5-7: Deney IV'e göre algoritmaların çalışma süreleri.



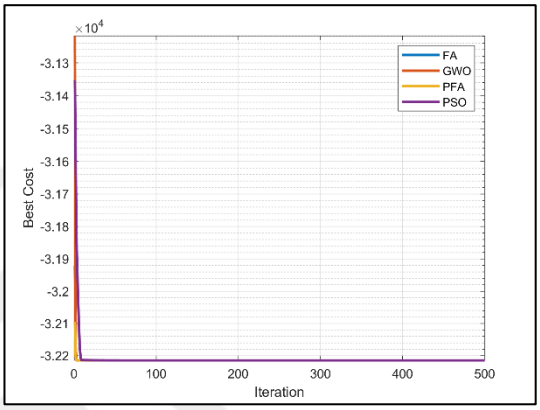
DKDKP



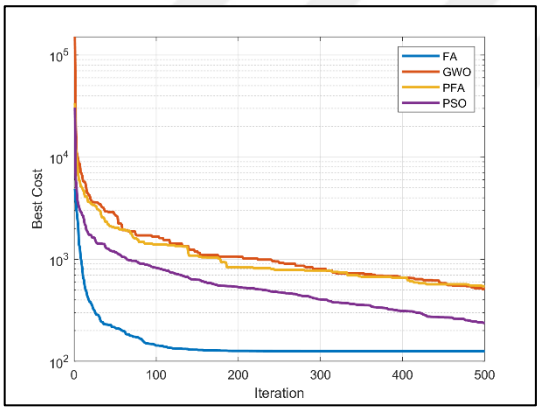
GİKP



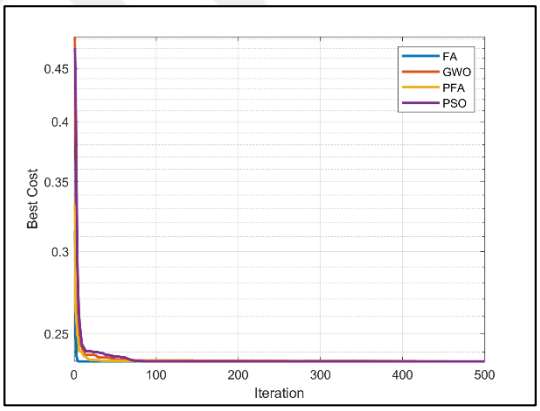
DTTP



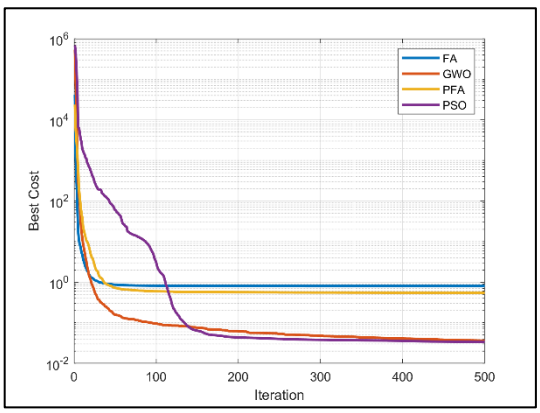
HF



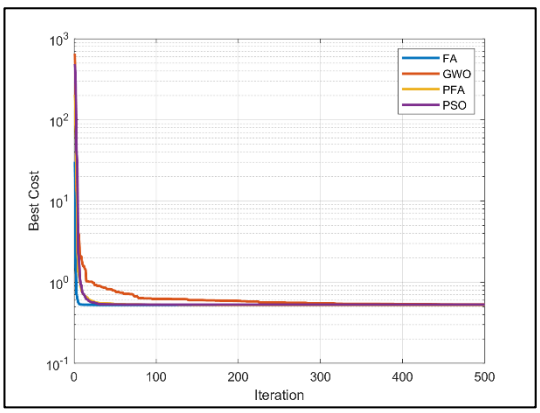
HBYTP



ÇDKFTP



ESSOT



PDZTP

Şekil 5-8: Deney IV'e göre algoritmaların yakınsama grafikleri.

Deney IV'e göre Şekil 5.8'de verilen algoritmaların yakınsama grafikleri incelendiğinde, GİKP, HF, PDZTP ve ÇDKFTP problemlerinde tüm algoritmaların 100 yinelemenin altında ideal sonuçları elde ettiği, ancak PDZTP probleminde GWO'nun 300 yineleme süresince iyileştirmeye devam ettiği görülmektedir. ESSOT probleminde en iyi yakınsayan algoritmalar GWO ve PSO olmuştur. PSO algoritması, DKDKP probleminde yerel çözüm engeline takılırken, DTTP'de ise en iyi yakınsama performasını sergilemiştir.

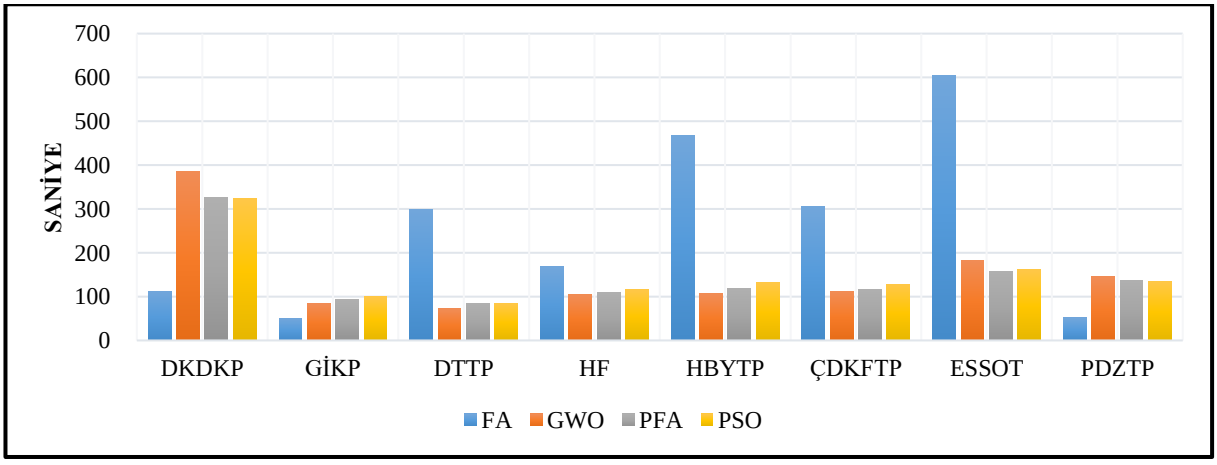
5.2.5 Deney V

Bu deneyde, yineleme sayısı 1000, popülasyon sayısı 10 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen global en iyi (*global best*) skorların ortalaması alınmıştır. Tablo 5.6'da, Deney V'e göre mühendislik problemlerinin global en iyi skorları verilmiştir.

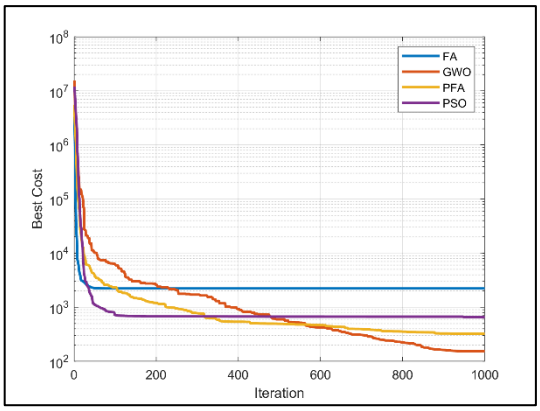
Tablo 5-6: Deney V'e göre algoritmaların global en iyi (best) skorları.

#	Problem	En İyi Skor	FA	GWO	PFA	PSO
1	DKDKP	GWO	2225,1136	153,3436	323,0229	658,7851
2	GİKP	Hepsi	977874,1517	977874,1517	977874,1517	977874,1517
3	DTTP	FA	2,79743E-27	8,96184E-12	6,86485E-12	1,64333E-20
4	HF	GWO, PFA, PSO	-32214,6044	-32214,7639	-32214,7639	-32214,7639
5	HBYTP	FA	214,1503	804,3361	653,0496	777,4236
6	ÇDKFTP	GWO, PFA	0,2363	0,2352	0,2352	0,2370
7	ESSOT	GWO	48,3294	0,3132	0,8641	1,0234
8	PDZTP	PFA	0,6053	0,5288	0,5284	0,5767

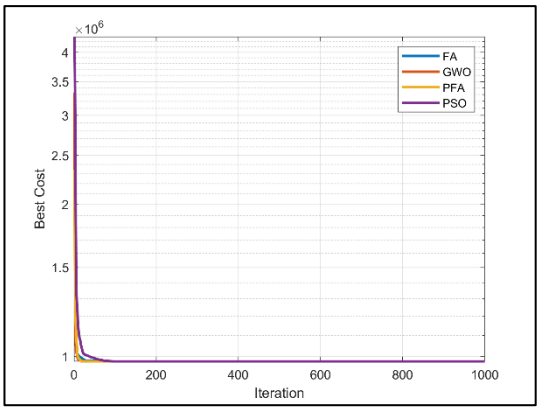
GWO algoritması 3, 5 ve 8 nolu problem haricinde tüm deneylerde en iyi skoru elde ederek birinci sıradadır. PFA, dört deneyde en iyi skoru elde ederek ikinci sırada, FA, üç deneyde en iyi skoru elde ederek sonuncu sırada yer almaktadır. Optimizasyon algoritmalarının çalışma süreleri, Şekil 5.9'da gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. GWO, dört problemin çözümünde, FA, üç problemin çözümünde ve PFA ise ESSOT probleminin çözümünde en hızlı çalışan algoritma olmuştur. Şekil 5.10'da, Deney V'e göre mühendislik problemlerinin yakınsama grafikleri verilmiştir.



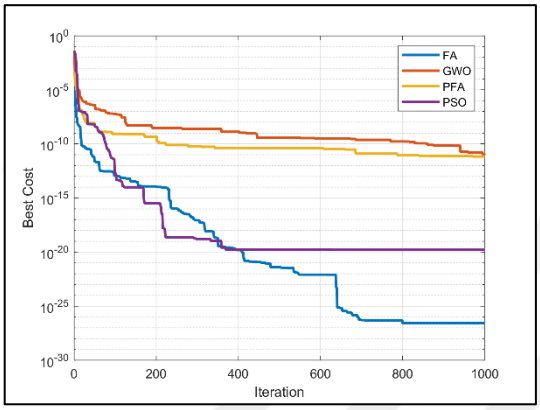
Şekil 5-9: Deney V'e göre algoritmaların çalışma süreleri.



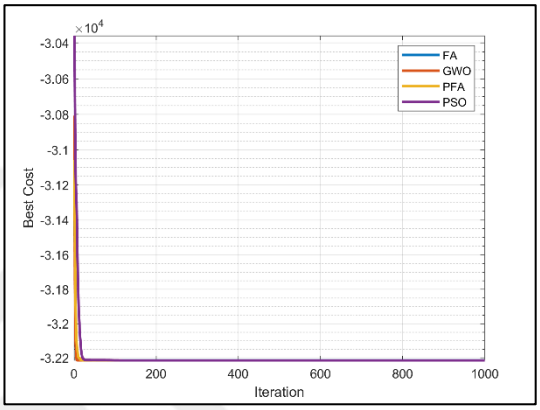
DKDKP



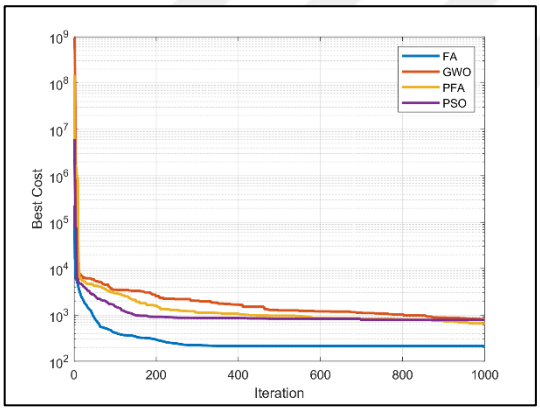
GİKP



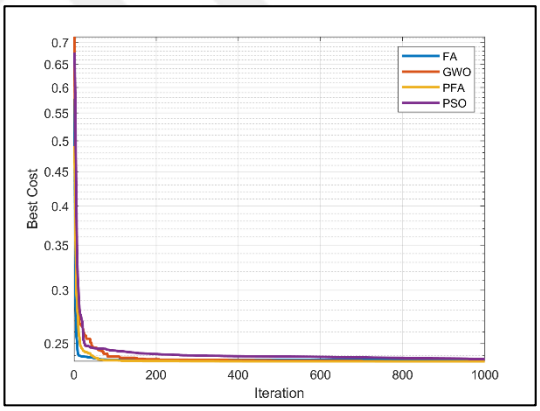
DTTP



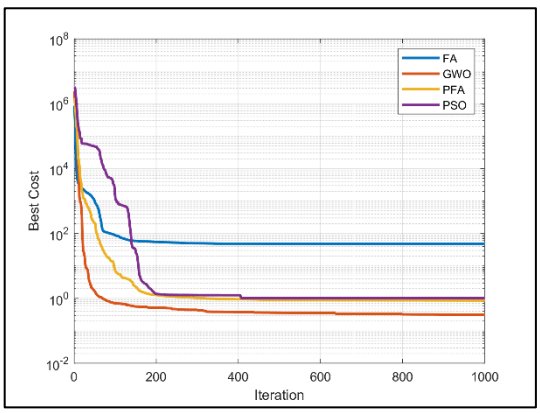
HF



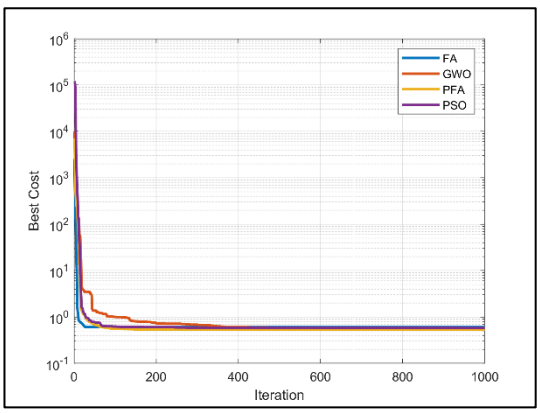
HBYTP



ÇDKFTP



ESSOT



PDZTP

Şekil 5-10: Deney V'e göre algoritmaların yakınsama grafikleri.

Deney V'e göre Şekil 5.10'da verilen algoritmaların yakınsama grafikleri incelendiğinde, önceki deneylerde de olduğu gibi metasezgisel yöntemlerin tümü GİKP, HF, PDZTP ve ÇDKFTP problemlerinde optimal sonuçları kısa sürede elde ettikleri görülmektedir. GWO algoritması, DTTP probleminde en kötü yakınsamaya sahip iken, DKDKP ve ESSOT'ta en başarılı yöntem olmuştur. FA ise HBYTP ve DTTP problemlerinde en iyi yakınsama değerine sahiptir. Ancak, ESSOT probleminde ise en kötü sonucu veren algoritma olmuştur.

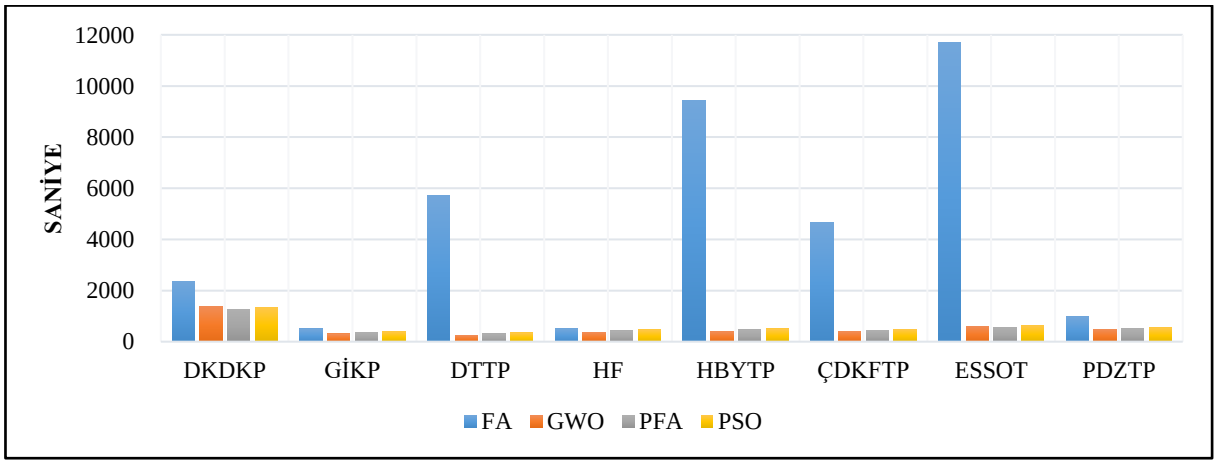
5.2.6 Deney VI

Bu deneyde, yineleme sayısı 1000, popülasyon sayısı 50 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen global en iyi (*global best*) skorların ortalaması alınmıştır. Tablo 5.7'de, Deney VI'ya göre mühendislik problemlerinin global en iyi skorları verilmiştir.

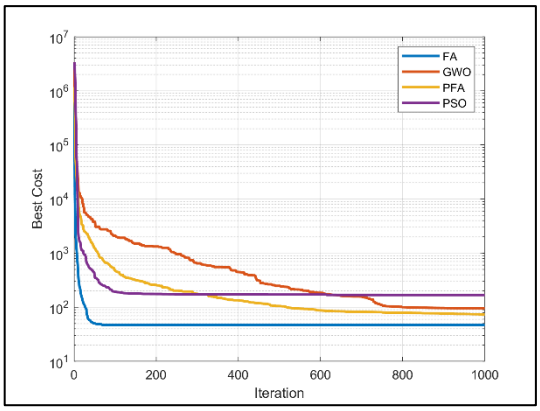
Tablo 5-7: Deney VI'ya göre algoritmaların global en iyi (best) skorları.

#	Problem	En İyi Skor	FA	GWO	PFA	PSO
1	DKDKP	GWO	2225,1136	153,3436	323,0229	658,7851
2	GİKP	Hepsi	977874,1517	977874,1517	977874,1517	977874,1517
3	DTTP	FA	2,79743E-27	8,96184E-12	6,86485E-12	1,64333E-20
4	HF	GWO, PFA, PSO	-32214,6044	-32214,7639	-32214,7639	-32214,7639
5	HBYTP	FA	214,1503	804,3361	653,0496	777,4236
6	ÇDKFTP	GWO, PFA	0,2363	0,2352	0,2352	0,2370
7	ESSOT	GWO	48,3294	0,3132	0,8641	1,0234
8	PDZTP	PFA	0,6053	0,5288	0,5284	0,5767

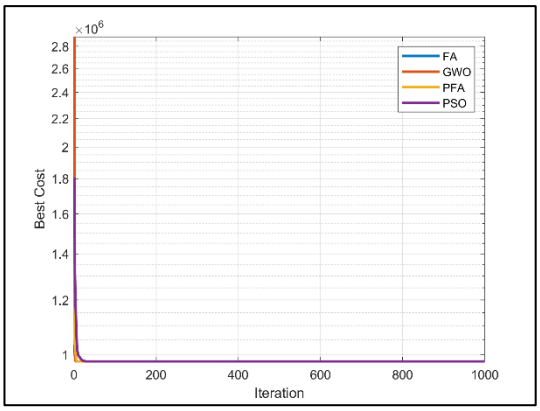
GWO algoritması 3, 5 ve 8 nolu problem haricinde tüm deneylerde en iyi skoru elde ederek birinci sıradadır. PFA, dört deneyde en iyi skoru elde ederek ikinci sırada, FA, üç deneyde en iyi skoru elde ederek sonuncu sırada yer almaktadır. Metasezgisel algoritmalarının çalışma süreleri, Şekil 5.11'de gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. GWO, altı problemin çözümünde, PFA, iki problemin çözümünde en hızlı çalışan algoritma iken FA ve PSO'nun hiçbir problemde derece alamamıştır. Ayrıca, FA bu deneyde en yavaş çalışan algoritma olduğu görülmektedir. Şekil 5.12'de, Deney VI'ya göre mühendislik problemlerinin yakınsama grafikleri verilmiştir.



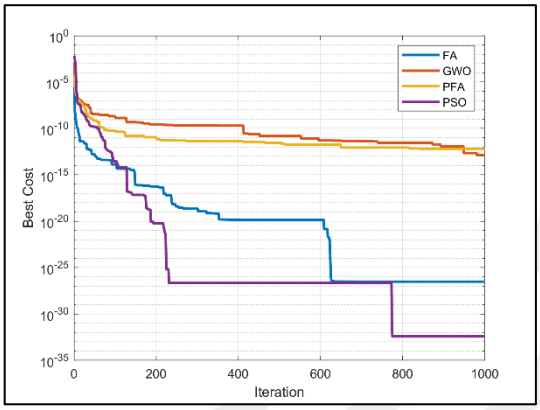
Şekil 5-11: Deney VI'ya göre algoritmaların çalışma süreleri.



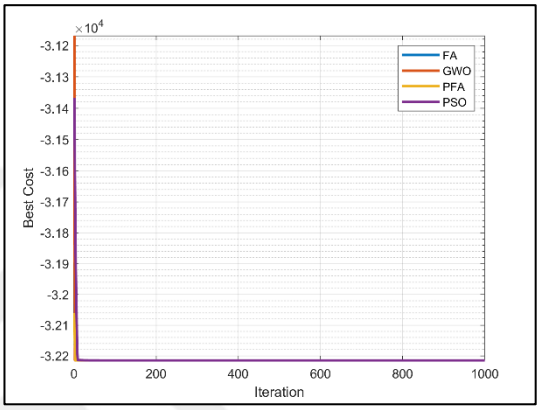
DKDKP



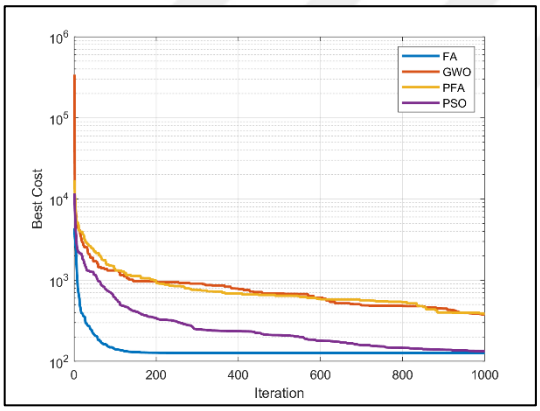
GİKP



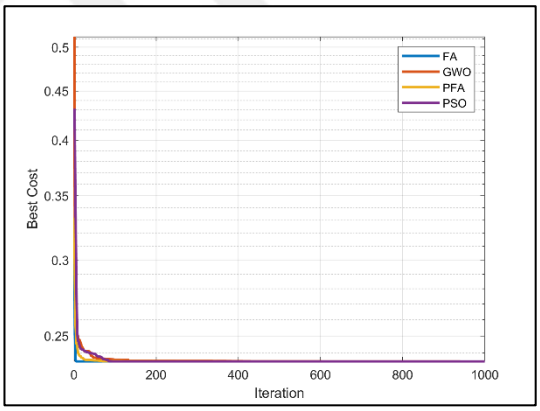
DTTP



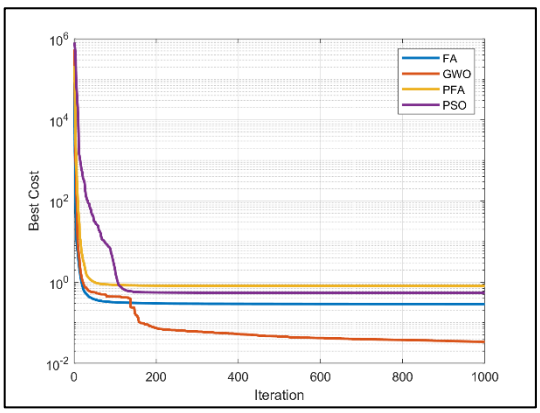
HF



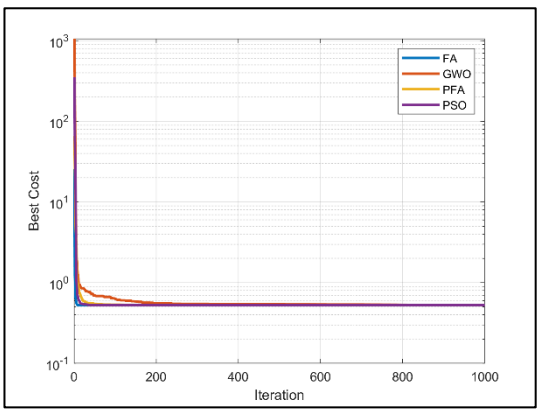
HBYTP



ÇDKFTP



ESSOT



PDZTP

Şekil 5-12: Deney VI'ya göre algoritmaların yakınsama grafikleri.

Deney VI'ya göre Şekil 5.12'de verilen algoritmaların yakınsama grafikleri incelendiğinde, HF ve GİKP problemlerinde tüm algoritmaların neredeyse 20-30 yinelemede en iyi sonuca eriştiği görülmektedir. Bu durum, ÇDKFTP'de 200 yinelemenin altında seyrederken, PDZTP probleminde GWO haricindeki algoritmalar da yaklaşık 50 yinelemede ideal sonuca ulaştığı görülmektedir. DDTP problemi sonuçlarına göre FA ve PSO algoritmaları hızlı yakınsama gösterirken, diğer algoritmalar iyi bir performans sergileyememiştir. ESSOT probleminde en iyi yakınsama değeri GWO'ya aitken, DKDKP'de ise FA dah başarılı olmuştur. HBYTP probleminde ise 1000 yinelemede PSO ve FA aynı değere ulaşırken, benzer biçimde PFA ve GWO algoritmları da birbirine yakın sonuçlar vermiştir.

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, makine mühendisliği tasarım problemlerinin çözülmesi amacıyla literatürde iyi bilinen metasezgisel algoritmar kullanılmıştır. Bu algoritmalar, yineleme sayısı ve popülasyon sayısında bağlı olarak altı farklı senaryo hazırlanarak gerçek-dünya mühendislik problemleri ile test edilmiş ve elde edilen deney sonuçlarına göre, metasezgisel algoritmaların performansları birbiriyle karşılaştırılmıştır.

Deneysel çalışmalarda kullanılan her bir mühendislik problemi için metasezgisel optimizasyon algoritmaları 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel (global) en iyi skorların ortalaması hesaplanmıştır. Aynı zamanda bu algoritmaların ortalama çalışma süreleri de ölçülerek, kıyaslamaları yapılmıştır. Deneysel çalışmalar, yineleme sayısı ve popülasyon sayısının değişen durumlarına bağlı olarak altı aşamada gerçekleştirilmiştir.

Deney I'de yineleme sayısı 100, popülasyon sayısı 10 olarak belirlenmiştir. Elde edilen sonuçlara göre PFA algoritması, sekiz mühendislik tasarım probleminin dördünde en iyi skoru elde ederken, FA ve GWO algoritmaları problemlerin üçünde ve PSO ise yalnızca bir problemde en iyi skoru etmiştir. Çalışma süresine göre değerlendirildiğinde ise GWO algoritması beş problemde en hızlı sonuç veren yöntem olmuştur.

Nümerik optimizasyon problemleri genellikle sürekli matematiksel fonksiyonların minimum veya maksimum değerlerini bulmayı amaçlayan problemlerdir ve genellikle daha hızlı çözülürler. Bu, sürekli fonksiyonların düzgün ve sürekli türevlere sahip olmalarından kaynaklanır. Nümerik olmayan optimizasyon problemleri ise ayrık, kategorik veya karma tip değişkenleri içerir. Örneğin, ürün seçimi gibi karar problemlerinde bu tür değişkenler kullanılır. Bu tür problemlerin çözümü genellikle daha zorlu olabilir ve çeşitli optimizasyon stratejilerini gerektirebilir, örneğin karma tamsayılı programlama veya genetik algoritmalar.Her iki tür optimizasyon probleminin de çalışma süreleri, problem büyüklüğüne, karmaşıklığına ve kullanılan algoritmaya bağlı olarak değişir. Problemin doğasını anlamak ve buna uygun bir çözüm stratejisi seçmek, verimli bir çözüm elde etmeye yardımcı olabilir.Deney II'de yineleme sayısı 100, popülasyon sayısı 50 olarak belirlenmiştir. Elde edilen sonuçlara göre FA algoritması, mühendislik tasarım problemlerinin yedisinde en iyi skoru elde ederken, diğer optimizasyon algoritmaları problemlerin üçünde en iyi skoru etmiştir. Çalışma süresine göre değerlendirildiğinde ise GWO algoritması, ilk deneyde olduğu gibi, beş problemde en hızlı sonuç veren yöntem olmuştur.

Deney III'te yineleme sayısı 500, popülasyon sayısı 10 olarak belirlenmiştir. Elde edilen sonuçlara göre GWO algoritması, sekiz mühendislik tasarım probleminin beşinde en iyi skoru elde

ederken, PFA problemlerin dördünde, FA problemlerin üçünde ve PSO algoritması ise problemlerin ikisinde en iyi skoru etmiştir. Çalışma süresine göre değerlendirildiğinde ise GWO ve FA algoritmaları en hızlı sonuç veren algoritmalar olmuştur.

Deney IV'te yineleme sayısı 500, popülasyon sayısı 50 olarak belirlenmiştir. Elde edilen sonuçlara göre FA ve PSO algoritmaları, mühendislik tasarım problemlerinin beşinde en iyi skoru elde ederken, problemlerin dördünde ve GWO ise problemlerin üçünde en iyi skoru etmiştir. GWO ve FA algoritmaları, çalışma süresine göre en hızlı sonuç veren yöntemler olmuştur.

Deney V ve VI'da yineleme sayısı 1000, popülasyon sayısı sırasıyla 10 ve 50 olarak belirlenmiştir. Elde edilen sonuçlara göre GWO algoritması, mühendislik tasarım probleminin beşinde en iyi skoru elde ederken, PFA ve FA algoritmaları sırasıyla problemlerin dördünde ve üçünde en iyi skoru etmiştir. Çalışma süresine göre değerlendirildiğinde ise GWO ve PFA algoritmaları en hızlı sonuç veren algoritmalar iken FA en yavaş çalışan algoritma olmuştur.

Genel olarak bir değerlendirme yapıldığında, çalışma süresine göre en iyi metasezgisel yöntemin GWO olduğu göze çarpmaktadır. Yineleme sayısı 1000 olduğu durumda GWO algoritması en iyi yakınsama değerine sahip olurken, yineleme sayısının daha düşük olduğu durumlarda FA, PFA ve PSO algoritmaları en iyi skorlara sahip olduğu görülmektedir.

Sonuç olarak, mühendislik tasarım problemlerinde daha yüksek yineleme sayısındaki deneylerde en hızlı ve en iyi sonuçları elde etmek için GWO algoritması tercih edilebileceğini söyleyebiliriz.

7. KAYNAKLAR

[1] Tang, J., Liu, G., & Pan, Q. (2021). A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends. IEEE/CAA Journal of Automatica Sinica, 8(10): 1627–1643. doi: 10.1109/jas.2021.1004129

[2] Gupta, S., Abderazek, H., Yıldız, B. S., Yildiz, A. R., Mirjalili, S. & Sait, S. M. (2021). "Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems". Expert Systems with Applications, 183:115351.

[3] Elshaboury, N., Attia, T., & Marzouk, M. (2020). Application of Evolutionary Optimization Algorithms for Rehabilitation of Water Distribution Networks. Journal of Construction Engineering and Management, 146(7): 04020069.

[4] Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., & Das, S. (2020). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm and Evolutionary Computation, 56: 100693.

[5] El-Halwagi, M. M. ed. (2006). Overview of optimization. Process Systems Engineering, Academic Press, 7:285–314. doi: 10.1016/s1874-5970(06)80012-3

[6] Zadeh, P. M., & Mohagheghi, M. (2022). An efficient Bi-level hybrid multi-objective reliability-based design optimization of composite structures. Composite Structures, 296: 115862.

[7] Elmahdi, L. A., Xu, Y., Khalil, E. M., & Mohamed, M. S. (2022). An application of adaptive normalization evolutionary optimization ANMOGA for missile fin design based on trajectory parameters. Alexandria Engineering Journal, 61(12): 12247-12257.

[8] Tsuzuki, M. S., Barari, A., Sato, A. K., Takimoto, R. Y., & Saka, T. (2022). Introductory Chapter: Optimization Problems in Engineering. In Engineering Problems-Uncertainties, Constraints and Optimization Techniques. IntechOpen.

[9] Elen, A. (2022). A Complete Solution to Exam Scheduling Problem: A Case Study. Journal of Scientific Reports-A, 2022(049): 12–34.

[10] Çayıroğlu, İ., & Elen, A. (2012). A heuristic optimization approach for a real-world university timetabling problem. Advances in Computer Science and Engineering, 9(2): 103–131.

[11] Elen, A., & Çayıroğlu, İ. (2010). Solving of scheduling problem with heuristic optimization approach. Teknoloji, 13(3): 159–172.

[12] Dorigo, M., Maniezzo, V., & Colorni, A., (1996). The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26(1): 29–41.

[13] Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. Simulation, 76(2): 60–68.

[14] Shah-Hosseini, H., (2007) Problem solving by intelligent water drops, Proceedings of IEEE Congress on Evolutionary Computation, 3226–3231.

[15] Atashpaz-Gargari, E., & Lucas, C., (2007). Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition, Proceedings of IEEE Congress on Evolutionary Computation, 4661–4667.

[16] Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. 2009 World Congress on Nature Biologically Inspired Computing, 210–214. Doi: 10.1109/NABIC.2009.5393690

[17] Rashedi, E., Nezamabadi, S., & Saryazdi, S., (2009). "GSA: a gravitational search algorithm", Information Sciences, 179(13): 2232– 2248.

[18] Yang, X. S. (2012). "Flower pollination algorithm for global optimization". Durand-Lose, J., Jonoska, N. (eds.) Unconventional Computation and Natural Computation, 240–249, Springer, Berlin.

[19] Rajakumar, B. R. (2012). "The lion's algorithm: A new nature-inspired search algorithm". Procedia Technology, 6: 126-135.

[20] Odili, J. B., Kahar, M. N. M. & Anwar, S. (2015). "African buffalo optimization: A swarm-intelligence technique". Procedia Computer Science, 76:443-448.

[21] Khishe, M. & Mosavi, M.R. (2020). "Chimp optimization algorithm". Expert systems with applications, 149: 113338. doi: 10.1016/j.eswa.2020.113338

[22] Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M. & Gandomi, A. H. (2021) "The arithmetic optimization algorithm". Comput Methods Appl Mech Eng, 376: 113609

[23] Rizk-Allah, R. M. (2018). Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. J Comput Des Eng, 5(2): 249–273

[24] Pauline, O., Sin, H. C., Sheng, D. D. C. V., Kiong, S. C., & Meng, O. K. (2017). Design optimization of structural engineering problems using adaptive cuckoo search algorithm. In: 2017 3rd international conference on control, automation and robotics (ICCAR), IEEE, 745–748

[25] Francisco RB, Costa MFP, Rocha AMA (2015) A firefly dynamic penalty approach for solving engineering design problems. In: AIP conference proceedings, 1648: 140010.

[26] Basak, R., Sanyal, A., Das, A., Ghosh, A., & Poddar, A. (2021). Performance analysis of genetic algorithm as a stochastic optimization tool in engineering design problems. Int. Journal of Recent Research in Electrical and Electronics Engineering, 2(4): 71-76.

[27] Nadimi-Shahraki, M. H., Taghian, S., & Mirjalili, S. (2021). An improved grey wolf optimizer for solving engineering problems. Expert Syst Appl 166: 113917.

[28] Ali, M., Pant, M., & Singh, V. (2010). Two modified differential evolution algorithms and their applications to engineering design problems. World J Model Simul, 6(1): 72–80.

[29] de Melo, V. V., & Carosio, G. L. C. (2012). Evaluating differential evolution with penalty function to solve constrained engineering problems. Expert Syst Appl, 39(9): 7860–7863.

[30] Rajwar, K., Deep, K., & Das, S. (2023). An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artificial intelligence review, 1–71. doi: doi.org/10.1007/s10462-023-10470-y

[31] Mandal, P. K. (2023). A review of classical methods and Nature-Inspired Algorithms (NIAs) for optimization problems. Results in Control and Optimization, 100315.

[32] Trojovský, P., Dehghani, M., & Hanuš, P. (2022). Siberian tiger optimization: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. Ieee Access, 10, 132396-132431. doi: 10.1109/ACCESS.2022.3229964

[33] Luo, Y. and Ma, X. (2020) A Complex Algorithm for Solving a Kind of Stochastic Programming. Journal of Applied Mathematics and Physics, 8, 1016-1030. doi: 10.4236/jamp.2020.86079.

[34] Xiong, N., Molina, D., Ortiz, M. L., & Herrera, F. (2015). A walk into metaheuristics for engineering optimization: principles, methods and recent trends. international journal of computational intelligence systems, 8(4): 606-636.

[35] Çimen, M. E., Garip, Z., & Boz, A. F. (2021). Comparison of metaheuristic optimization algorithms with a new modifieddeb feasibility constraint handling technique. Turkish Journal of Electrical Engineering and Computer Sciences, 29(7): 3270-3289.

[36] Muazu, A. A., Hashim, A. S., & Sarlan, A. (2022). Review of nature inspired metaheuristic algorithm selection for combinatorial t-way testing. IEEE Access, 10, 27404-27431.

[37] 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.

[38] Almufti, S. M., Shaban, A. A., Ali, Z. A., Ali, R. I., & Fuente, J. A. D. (2023). Overview of Metaheuristic Algorithms. Polaris Global Journal of Scholarly Research and Trends, 2(2): 10-32.

[39] Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.

[40] Yang, XS. (2009). Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (eds) Stochastic Algorithms: Foundations and Applications. SAGA 2009. Lecture Notes in Computer Science, vol. 5792. Springer, Berlin, Heidelberg.

[41] Fister, I., Fister Jr, I., Yang, X.-S., & Brest, J. (2013). A comprehensive review of firefly algorithms. Swarm and Evolutionary Computation, 13: 34–46.

[42] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software, 69, 46–61. doi: 10.1016/j.advengsoft.2013.12

[43] Meidani, K., Hemmasian, A., Mirjalili, S. et al. (2022). Adaptive grey wolf optimizer. Neural Comput & Applic 34, 7711–7731. doi: 10.1007/s00521-021-06885-9

[44] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks. doi: 10.1109/icnn.1995.488968

[45] Wang, D., Tan, D. & Liu, L. (2018). Particle swarm optimization algorithm: an overview. Soft Computing. 22, 387–408. doi: 10.1007/s00500-016-2474-6

[46] Shi, Y., & Eberhart, R.C. (1998). A modified particle swarm optimizer. In: Proceedings of the IEEE international conference on evolutionary computation, pp 69–73, Anchorage, Alaska, USA.

[47] Yapıcı, H. & Çetinkaya, N. (2019). A new meta-heuristic optimizer: Pathfinder algorithm. Applied Soft Computing. 78: 545–568. doi: 10.1016/j.asoc.2019.03.012

[48] Cagnina, L. C., Esquivel, S. C., & Coello, C.A.C. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica, 32(3): 319-326.

[49] Liang, Y., Hu, X., Hu, G., & Dou, W. (2022). An Enhanced Northern Goshawk Optimization Algorithm and Its Application in Practical Optimization Problems. Mathematics, 10(22), 4383. MDPI AG. doi: 10.3390/math10224383

[50] El-Shorbagy, M. A. (2022). Chaotic Fruit Fly Algorithm for Solving Engineering Design Problems. In A. Akgul (Ed.), Complexity vol. 2022, pp. 1–19. Hindawi Limited.

[51] Barshandeh, S., & Haghzadeh, M. (2020). A new hybrid chaotic atom search optimization based on tree-seed algorithm and Levy flight for solving optimization problems. Engineering with Computers. doi: 10.1007/s00366-020-00994-0

[52] D. M. Himmelblau, Applied nonlinear programming, McGraw-Hill Companies, 1972.

[53] Oztas, G. Z., & Erdem, S. (2023). A penalty-based algorithm proposal for engineering optimization problems. Neural Comput & Applic 35: 7635–7658.

[54] Yin, S., Luo, Q. & Zhou, Y. (2022) EOSMA: An Equilibrium Optimizer Slime Mould Algorithm for Engineering Design Problems. Arab J Sci Eng 47: 10115–10146.

[55] Yildiz, B. S., Pholdee, N., Bureerat, S., Yildiz, A. R., & Sait, S. M. (2022). Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems. Engineering with Computers. 38: 4207–4219.

[56] Andrei, N. (2013). Applications of Mechanical Engineering. In: Nonlinear Optimization Applications Using the GAMS Technology. Springer Optimization and Its Applications, vol 81. Springer, Boston, MA. doi: 10.1007/978-1-4614-6797-7_5

[57] Zhang, J., Xiao, M., Gao, L., & Pan, Q. (2018). Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. Applied Mathematical Modelling, 63, 464–490. doi: 10.1016/j.apm.2018.06.036

[58] Közkurt, C., & Elen, A. (2020) Planet dişli zinciri tasarım probleminin sezgisel optimizasyon yöntemleriyle çözümü, 2nd International Turkic World Congress on Science and Engineering, pp. 89-100, Astana, Kazakhistan.

[59] Singh, N., Kaur, J. (2021). Hybridizing sine–cosine algorithm with harmony search strategy for optimization design problems. Soft Comput 25:11053–11075.

8. EKLER

EK 1. Ateşböceği Algoritmasının Matlab ile Hazırlanmış Kaynak Kodu

```
% Ateşböceği Algoritması: MATLAB Kaynak Kodu
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% Geliştiriciler: S. Mostapha Kalami Heris, sm.kalami@gmail.com, info@yarpiz.com
% https://www.mathworks.com/matlabcentral/fileexchange/52900

clc; clear; % Komut penceresini ve değişkenleri temizle
%% Problem Tanımı
MaliyetFonk = @(x) sum(x.^2); % Maliyet Fonksiyonu
degiskenSayisi = 5; % Karar değişkenlerinin sayısı
boyut = [1, degiskenSayisi]; % Karar değişkenlerinin matris boyutu
altSinir = -10; % Karar değişkenlerinin alt sınırı
ustSinir = 10; % Karar değişkenlerinin üst sınırı
%% Ateşböceği algoritmasının parametreleri
maxIterasyon = 100;  % Maksimum iterasyon sayısı
popSayisi = 25;    % Ateşböceği sayısı (Sürü boyutu)
gamma = 1.0;    % Işık emilim katsayısı
beta0 = 2.0;    % Çekicilik temel kat sayısı
alpha = 0.2;  % Mutasyon katsayısı
alphaDamp = 0.98; % Mutasyon katsayısı güncelleme oranı
delta = 0.05 * (ustSinir - altSinir); % Tekdüze mutasyon aralığı
m = 2;
if isscalar(altSinir) && isscalar(ustSinir)
   dmax = (ustSinir-altSinir)*sqrt(degiskenSayisi);
else
   dmax = norm(ustSinir-altSinir);
end
%% Başlangıç
% Boş Atesboceği yapısı oluşturma
atesBocegi.Konum = [];
atesBocegi.Maliyet = [];
pop = repmat(atesBocegi, popSayisi, 1); % Baslangic populasyonu
BestCozum.Maliyet = inf; % En iyi cozum
% İlk popülasyonun oluşturulması
for i = 1 : popSayisi
   pop(i).Konum = unifrnd(altSinir, ustSinir, boyut);
   pop(i).Maliyet = MaliyetFonk(pop(i).Konum);
   if (pop(i).Maliyet <= BestCozum.Maliyet)
      BestCozum = pop(i);
   end
end
% En iyi maliyet değerlerinin saklanması
BestMaliyet = zeros(maxIterasyon, 1);
```

```
%% Firefly Algorithm Ana Döngü
for it=1:maxIterasyon
   yeniPop = repmat(atesBocegi, popSayisi, 1);
   for i=1:popSayisi
      yeniPop(i).Maliyet = inf;
      for j=1:popSayisi
         if (pop(j).Maliyet < pop(i).Maliyet)
            rij=norm(pop(i).Konum-pop(j).Konum)/dmax;
            beta=beta0*exp(-gamma*rij^m);
            e=delta*unifrnd(-1, 1, boyut);
            %e=delta*randn(VarSize);
            newsol.Konum = pop(i).Konum ...
               + beta*rand(boyut).*(pop(j).Konum-pop(i).Konum) ...
               + alpha*e;
            newsol.Konum=max(newsol.Konum,altSinir);
            newsol.Konum=min(newsol.Konum,ustSinir);
            newsol.Maliyet=MaliyetFonk(newsol.Konum);
            if newsol.Maliyet <= yeniPop(i).Maliyet
               yeniPop(i) = newsol;
               if yeniPop(i).Maliyet<=BestCozum.Maliyet
                  BestCozum=yeniPop(i);
               end
            end
         end
      end
   end
   % Birlestir
   pop=[pop; yeniPop];  %#ok
   % Sirala
   [~, SortOrder]=sort([pop.Maliyet]);
   pop=pop(SortOrder);
   % Populasyon sayisi kadar al
   pop=pop(1:popSayisi);
   % En iyi maliyete sahip cozumu sakla
   BestMaliyet(it)=BestCozum.Maliyet;
   % Iterasyon bilgisini goster
   disp(['Iterasyon ' num2str(it) ': En iyi maliyet = ' num2str(BestMaliyet(it))]);
   % Mutasyon katsayisini guncelle
   alpha = alpha * alphaDamp;
end
%% Sonuçlar
figure;
semilogy(BestMaliyet, 'LineWidth', 2);
xlabel('Iterasyon'); ylabel('En iyi maliyet');
grid on;
```

EK 2. Gri Kurt Algoritmasının Matlab Programlama Dili ile Hazırlanmış Kaynak Kodu

```
% Gri Kurt Algoritması: MATLAB R2011b(7.13) Kaynak Kodu
% Geliştiriciler: Seyedali Mirjalili
% E-posta: ali.mirjalili@gmail.com, seyedali.mirjalili@griffithuni.edu.au
% https://www.mathworks.com/matlabcentral/fileexchange/44974

clc; % Komut penceresini temizle
clear; % Değişkenleri temizle

%% Problem Tanımı
MaliyetFonk = @(x) sum(x.^2); % Maliyet fonksiyonu
boyut = 5; % Karar değişkenlerinin sayısı
altSinir=-10; % Karar değişkenlerinin alt sınırı
ustSinir= 10; % Karar değişkenlerinin üst sınırı
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % popülasyon sayısı

%% Gri Kurt algoritmasının parametreleri
% Alfa, beta ve delta kurtların başlangıç değerlerini oluştur.
AlphaKonum = zeros(1, boyut);
AlphaSkor = inf; % maksimize etme problemleri için bunu -inf olarak değiştirin
BetaKonum = zeros(1, boyut);
BetaSkor = inf; % maksimize etme problemleri için bunu -inf olarak değiştirin
DeltaKonum = zeros(1, boyut);
DeltaSkor = inf; % maksimize etme problemleri için bunu -inf olarak değiştirin

% Başlangıç konumlarını belirle
sinirBoyutu = size(ustSinir, 2);
% Tüm değişkenlerin sınır değerleri aynı ise
if sinirBoyutu == 1
   r = rand(popSayisi, boyut);
   Konumlar = r.*(ustSinir-altSinir)+altSinir;
end
% Her değişkenin farklı bir alt ve üst sınır değeri varsa
if sinirBoyutu > 1
   for i = 1 : boyut
      ubi = ustSinir(i);
      lbi = altSinir(i);
      r = rand(popSayisi, 1);
      Konumlar(:, i) = r.*(ubi-lbi)+lbi;
   end
end

YakinsamaEgrisi = zeros(1, maxIterasyon);
```

```
t = 0; % Döngü sayacı

% Ana döngü
while (t < maxIterasyon)
   for i = 1 : size(Konumlar, 1)
      % Arama alanının sınırlarını aşan kutları bul ve güncelle
      ust = Konumlar(i, :) > ustSinir;
      alt = Konumlar(i, :) < altSinir;
      Konumlar(i, :) = (Konumlar(i, :).*(~(ust+alt)))+ustSinir.*ust+altSinir.*alt;

      % Her bir ajanın maliyetini hesapla.
      uygunluk = MaliyetFonk(Konumlar(i, :));

      % Alpha'yı güncelle
      if (uygunluk < AlphaSkor)
         AlphaSkor = uygunluk;
         AlphaKonum = Konumlar(i, :);
      end

      % Beta'yı güncelle
      if (uygunluk > AlphaSkor && uygunluk < BetaSkor)
         BetaSkor = uygunluk;
         BetaKonum = Konumlar(i, :);
      end

      % Delta'yı güncelle
      if (uygunluk>AlphaSkor && uygunluk>BetaSkor && uygunluk<DeltaSkor)
         DeltaSkor = uygunluk;
         DeltaKonum = Konumlar(i, :);
      end
   end

   a = 2 - t*(2/maxIterasyon); % Doğrusal olarak 2'den 0'a azalt.

   % Update the Position of search agents including omegas
   for i = 1 : size(Konumlar, 1)
      for j = 1 : size(Konumlar, 2)
         r1 = rand(); % [0, 1] aralığında rastgele bir sayı.
         r2 = rand(); % [0, 1] aralığında rastgele bir sayı.

         A1 = 2.0 * a * r1 - a; % Eşitlik (3.3)
         C1 = 2.0 * r2; % Eşitlik (3.4)

         % Eşitlik (3.5)-bölüm 1
         dAlfa = abs(C1*AlphaKonum(j) - Konumlar(i, j));
         % Eşitlik (3.6)-bölüm 1
         X1 = AlphaKonum(j) - A1 * dAlfa;
```

```
            r1 = rand(); % [0, 1] aralığında rastgele bir sayı.
            r2 = rand(); % [0, 1] aralığında rastgele bir sayı.

            A2 = 2.0 * a * r1 - a; % Eşitlik (3.3)
            C2 = 2.0 * r2; % Eşitlik (3.4)

            % Eşitlik (3.5)-bölüm 2
            dBeta = abs(C2*BetaKonum(j) - Konumlar(i, j));
            % Eşitlik (3.6)-bölüm 2
            X2 = BetaKonum(j) - A2 * dBeta;

            r1 = rand(); % [0, 1] aralığında rastgele bir sayı.
            r2 = rand(); % [0, 1] aralığında rastgele bir sayı.

            A3 = 2.0 * a * r1 - a; % Eşitlik (3.3)
            C3 = 2.0 * r2; % Eşitlik (3.4)

            % Eşitlik (3.5)-bölüm 3
            dDelta = abs(C3*DeltaKonum(j) - Konumlar(i, j));
            % Eşitlik (3.5)-bölüm 3
            X3 = DeltaKonum(j) - A3 * dDelta;

            Konumlar(i, j) = (X1 + X2 + X3) / 3;% Eşitlik (3.7)
        end
    end
    t = t + 1;
    YakinsamaEgrisi(t) = AlphaSkor;
    % Iterasyon bilgisini goster
    disp(['Iterasyon ' num2str(t) ': En iyi maliyet = ' num2str(AlphaSkor)]);
end


%% Sonuçlar
figure;
semilogy(YakinsamaEgrisi, 'LineWidth', 2);
xlabel('Iterasyon');
ylabel('En iyi maliyet');
grid on;
```

EK 3. Parçacık Sürü Optimizasyonunun Matlab ile Hazırlanmış Kaynak Kodu

```
% Parçacık Sürü Optimizasyonu: MATLAB Kaynak Kodu
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% Geliştirici: S. Mostapha Kalami Heris
% https://www.mathworks.com/matlabcentral/fileexchange/52857

clc; clear; % Komut penceresini ve değişkenleri temizle
%% Problem Tanımları
MaliyetFonk = @(x) sum(x.^2); % Maliyet fonksiyonu
degiskenSayisi = 5; % Karar değişkenlerinin sayısı
VarSize = [1, degiskenSayisi]; % Karar değişkenleri
altSinir = -10; % Karar değişkenlerinin alt sınırı
ustSinir = 10;% Karar değişkenlerinin üst sınırı
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % Populasyon boyutu
%% PSO Parametreleri
w = 1; % Atalet ağırlığı
wdamp = 0.99; % Atalet ağırlığı azaltma oranı
c1 = 1.5; % Bireysel öğrenme katsayısı
c2 = 2.0; % Küresel öğrenme katsayısı
% Hız limitleri
maxHiz = 0.1 * (ustSinir - altSinir);
minHiz = -maxHiz;
%% Başlatma
bosParcacik.Konum = [];
bosParcacik.Maliyet = [];
bosParcacik.Hiz = [];
bosParcacik.Best.Konum = [];
bosParcacik.Best.Maliyet = [];
parcacik = repmat(bosParcacik, popSayisi, 1);
GlobalEnIyi.Maliyet = inf;
for i = 1 : popSayisi
   % Başlangıç Konumları
   parcacik(i).Konum = unifrnd(altSinir, ustSinir, VarSize);
   % Başlangıç Hızları
   parcacik(i).Hiz=zeros(VarSize);
   % Maliyet hesapla
   parcacik(i).Maliyet = MaliyetFonk(parcacik(i).Konum);
   % Bireysel en iyiyi güncelle
   parcacik(i).Best.Konum = parcacik(i).Konum;
   parcacik(i).Best.Maliyet = parcacik(i).Maliyet;
   % Update Global Best
   if (parcacik(i).Best.Maliyet < GlobalEnIyi.Maliyet)
      GlobalEnIyi = parcacik(i).Best;
```

```
  end
end

BestMaliyet = zeros(maxIterasyon, 1);

%% PSO Ana Döngüsü
for t = 1 : maxIterasyon
  for i = 1 : popSayisi
    % Hız Güncelleme
    parcacik(i).Hiz = w*parcacik(i).Hiz ...
      +c1*rand(VarSize).*(parcacik(i).Best.Konum-parcacik(i).Konum) ...
      +c2*rand(VarSize).*(GlobalEnIyi.Konum-parcacik(i).Konum);
    % Hız Limitlerini Güncelle
    parcacik(i).Hiz = max(parcacik(i).Hiz,minHiz);
    parcacik(i).Hiz = min(parcacik(i).Hiz,maxHiz);
    % Konumları Güncelle
    parcacik(i).Konum = parcacik(i).Konum + parcacik(i).Hiz;
    % Hız Ayna Etkisi
    IsOutside=(parcacik(i).Konum<altSinir | parcacik(i).Konum>ustSinir);
    parcacik(i).Hiz(IsOutside) = -parcacik(i).Hiz(IsOutside);
    % Konum Limitlerini Güncelle
    parcacik(i).Konum = max(parcacik(i).Konum, altSinir);
    parcacik(i).Konum = min(parcacik(i).Konum, ustSinir);
    % Maliyet hesapla
    parcacik(i).Maliyet = MaliyetFonk(parcacik(i).Konum);
    % Bireysel en iyiyi güncelle
    if (parcacik(i).Maliyet < parcacik(i).Best.Maliyet)
      parcacik(i).Best.Konum = parcacik(i).Konum;
      parcacik(i).Best.Maliyet = parcacik(i).Maliyet;
      % Global en iyiyi güncelle
      if (parcacik(i).Best.Maliyet < GlobalEnIyi.Maliyet)
        GlobalEnIyi = parcacik(i).Best;
      end
    end
  end
  w = w * wdamp; % Atalet ağırlığını güncelle
  BestMaliyet(t) = GlobalEnIyi.Maliyet;
  disp(['İterasyon ' num2str(t) ': En iyi maliyet = ' num2str(BestMaliyet(t))]);
end

%% Sonuçlar
figure;
semilogy(BestMaliyet, 'LineWidth', 2);
xlabel('İterasyon'); ylabel('En iyi maliyet');
grid on;
```

EK 4. Yol Bulucu Algoritmasının Matlab ile Hazırlanmış Kaynak Kodu

```
% Yol Bulucu Algoritması: MATLAB R2016b Kaynak Kodu
% Geliştirici: Hamza YAPICI, hyapici@erbakan.edu.tr
% This is a simple codes of PFA inspired by collective movements and leadership
behaviours of animals.
% https://www.mathworks.com/matlabcentral/fileexchange/82300

clc; clear; % Komut penceresini ve değişkenleri temizle
%% Problem Tanımları
MaliyetFonk = @(x) sum(x.^2); % Maliyet fonksiyonu
boyut = 5; % Karar değişkenlerinin sayısı
altSinir = -10*ones (1, boyut); % Karar değişkenlerinin alt sınırı
ustSinir = 10*ones (1, boyut);% Karar değişkenlerinin üst sınırı
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % Populasyon boyutu
% Başlatma
suru = zeros(popSayisi, boyut);
uygunluk = zeros(popSayisi, 1);
% Popülasyonun başlangıç değerlerini belirle.
for j = 1 : boyut
   r = rand(popSayisi, 1);
   suru(:, j) = altSinir(j) + (ustSinir(j)-altSinir(j)).*r;
end
% Popülasyonun uygunluk değerlerini hesapla.
for i = 1 : popSayisi, uygunluk(i) = MaliyetFonk(suru(i, :)); end
% Yol bulucuyu belirle
fitness = uygunluk;
[bestFitness, index] = min(uygunluk);
oldSuru = suru;
bestPos = suru(index, :);
pathPos = bestPos; pathOld = pathPos;
minMaliyet = inf(maxIterasyon, 1); minMaliyet(1) = bestFitness;
t = 0;
while (t < maxIterasyon) % Ana döngü
   t = t + 1;
   % PFA'nın parametlerini güncelle
   u1 = -1.0+(1.0-(-1.0)).*rand(popSayisi, 1);
   u2 = -1.0+(1.0-(-1.0)).*rand(1, boyut);
   r3 = rand(1, boyut);
   eps = (1 - t/(maxIterasyon)).*u1;
   A = u2.*exp(-2.0.*t/maxIterasyon);
   % Yol bulucuyu güncelle
   pathPos(1, :) = pathPos(1, :) + ...
      (2.0.*r3(1, :)).*(pathOld(1, :) - pathPos(1, :)) + 1.*A(1, :);
```

```
    ust = pathPos(1, :) > ustSinir(1, :); % alt sınır değelerini kontrol et.
    alt = pathPos(1, :) < altSinir(1, :); % üst sınır değelerini kontrol et.
    pathPos(1, :) = (pathPos(1, :).*(~(ust + alt))) + ...
        ustSinir.*ust + altSinir.*alt;
    pathOld = pathPos;
    path_fitness = MaliyetFonk(pathPos(1, :));
    [fitness, index] = sort(fitness);
suru = suru(index, :);
    % Eski uygunluk değeri ile yenisini karşılaştır.
    if (path_fitness < fitness(1))
        fitness(1) = path_fitness; suru(1, :) = pathPos(1, :);
    else
        fitness(1) = fitness(1); suru(1, :) = suru(1, :);
    end
    % Takipçi üyeleri güncelle.
    for ii = 2 : popSayisi
        d = sqrt(sum(((suru(ii,:)) - (suru(ii-1,:))).^2)); % calculate D(i,j)
        alpha = 1.0 + rand(1, boyut); % assign alpha
        beta = 1.0 + rand(1, boyut); % and beta
        % Takipçi üyelerin konumlarını güncelle
        suru(ii, :) = suru(ii, :) + ...
            ((alpha(1,:)).*rand).*(bestPos(1,:) - suru(ii,:)) + ...
            ((beta(1,:)).*rand).*(oldSuru(ii-1,:) - suru(ii,:)) + 1.*(eps(ii,1).*d);
        ust = suru(ii,:) > ustSinir(1,:);
        alt = suru(ii,:) < altSinir(1,:);
        suru(ii,:) = (suru(ii,:).*(~(ust+alt)))+ustSinir.*ust+altSinir.*alt;
    end
    for i = 1 : popSayisi
        % Takipçi üyelerin maliyet değerlerini güncelle
        fitness(i) = MaliyetFonk(suru(i, :));
        % update position via new fitness
        if (fitness(i) < uygunluk(i))
            uygunluk(i) = fitness(i);  oldSuru(i, :) = suru(i, :);
            bsf_index = i;
        end
    end
    % Find new pathfinder
    [bestFitness, index] = min(uygunluk);
    bestPos = oldSuru(index, :); pathPos = bestPos;
    minMaliyet(t+1) = bestFitness;
    disp(['İterasyon ' num2str(t) ': En iyi maliyet = ' num2str(bestFitness)]);
end
%% Sonuçlar
figure;
semilogy(minMaliyet, 'LineWidth', 2);
xlabel('İterasyon'); ylabel('En iyi maliyet');
grid on;
```