YUSUFÇUK ALGORİTMASININ BROWNIAN HAREKET İLE İYİLEŞTİRİLMESİ

HAKAN GÜLCAN

MERSİN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

MERSİN
AĞUSTOS – 2018

YUSUFÇUK ALGORİTMASININ BROWNIAN HAREKET İLE İYİLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

HAKAN GÜLCAN

MERSİN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

Danışman
Dr. Öğr. Üyesi Çiğdem ACI

MERSİN
AĞUSTOS - 2018

ONAY

Hakan GÜLCAN tarafından Dr. Öğr. Üyesi Çiğdem ACI danışmanlığında hazırlanan "Yusufçuk Algoritmasının Brownian Hareket ile İyileştirilmesi" başlıklı çalışma aşağıda imzaları bulunan jüri üyeleri tarafından 15 Ağustos 2018 tarihinde yapılan Tez Savunma Sınavı sonucunda oy birliği ile Yüksek Lisans tezi olarak kabul edilmiştir.

Görevi	Ünvanı, Adı ve Soyadı	İmza
Başkan	Dr. Öğr. Üyesi Çiğdem ACI	
Üye	Doç. Dr. Alkan ALKAYA	
Üye	Dr. Öğr. Üyesi Esra SARAÇ EŞSİZ	

Yukarıdaki Jüri kararı Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 21.09.2018 tarih ve 2018.36/1310 sayılı kararıyla onaylanmıştır.

Prof. Dr. Cahit BİLİM
Fen Bilimleri Enstitü Müdürü

Bu tezde kullanılan özgün bilgiler, şekil, tablo ve fotoğraflardan kaynak göstermeden alıntı yapmak 5846 sayılı Fikir ve Sanat Eserleri Kanunu hükümlerine tabidir.

ETİK BEYAN

Mersin Üniversitesi Lisansüstü Eğitim-Öğretim Yönetmeliğinde belirtilen kurallara uygun olarak hazırladığım bu tez çalışmasında,

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlâk kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak kullandığımı,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Bu tezin herhangi bir bölümünü Mersin Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı,
- Tezin tüm telif haklarını Mersin Üniversitesi'ne devrettiğimi

beyan ederim.

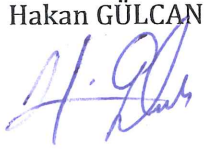
ETHICAL DECLARATION

This thesis is prepared in accordance with the rules specified in Mersin University Graduate Education Regulation and I declare to comply with the following conditions:

- I have obtained all the information and the documents of the thesis in accordance with the academic rules.
- I presented all the visual, auditory and written informations and results in accordance with scientific ethics.
- I refer in accordance with the norms of scientific works about the case of exploitation of others' works.
- I used all of the referred works as the references.
- I did not do any tampering in the used data.
- I did not present any part of this thesis as an another thesis at Mersin University or another university.
- I transfer all copyrights of this thesis to the Mersin University.

15 Ağustos 2018 / 15 August 2018

Hakan GÜLCAN



ÖZET

YUSUFÇUK ALGORİTMASININ BROWNIAN HAREKET İLE İYİLEŞTİRİLMESİ

Yusufçuk Algoritması son yıllarda geliştirilen önemli optimizasyon tekniklerinden biridir. Bu algoritma, doğadaki yusufçukların davranışlarını örnek almıştır. Yusufçukların doğada besin ararken veya düşmandan kaçarken yaptıkları rasgele uçuş hareketi, algoritmada rasgeleliğin temelini oluşturmaktadır. Yusufçuk Algoritmasında, optimum çözüme ulaşmak veya arama alanını genişletmek için rastgele bir uçuş yöntemi olan "Levy Uçuş Mekanizması" kullanılmıştır. Bu mekanizma, kıyaslama fonksiyonları üzerinde test edildiğinde kısmi bir başarı sağlamıştır. Buna karşın, yusufçukların rasgele hareketi bu mekanizma sebebiyle ve algoritmanın içinde uygulanan adım kontrolü hareketiyle zaman zaman kesintiye uğramaktadır. Bu kesinti, yusufçukların özgün davranmasını engelleyebilirken, uzun adımlar meydana geldiğinde de algoritmanın kendini tekrarlamasına yol açmaktadır. Bu çalışmada, kıyaslama fonksiyonları ile elde edilen sonuçların iyileştirilmesi için Brownian Hareket olarak bilinen ve randomizasyon alanında iyi bilinmesine rağmen, optimizasyon alanında örneği bulunmayan bir algoritma kullanılmıştır. Brownian Hareket ile modifiye edilen Yusufçuk Algoritması, 15 tek hedefli, 6 çok hedefli problem fonksiyonuna uygulanmış ve orjinal algoritma ile 200 iterasyon üzerinden karşılaştırılmıştır. Bu karşılaştırmalar sonucunda 15 tek hedefli problemden 12'si ve 6 çok hedefli problemden 6'sı orjinal algoritmaya göre daha başarılı sonuçlar vermiştir. Modifiye edilen algoritma, bir gerçek durum problemi olan Kaynaklı Kiriş Tasarımı Problemi'ne uygulanmış ve optimum maliyeti %20 daha düşük hesaplamayı başarabilmiştir. Sonuç olarak, Brownian Hareket, Yusufçuk Algoritması'nın keşif alanını genişletmek ve sömürü alanını taramak için iyi bir yeteneğe sahip olduğunu kanıtlamıştır.

Anahtar Kelimeler: dragonfly optimizasyonu, brownian hareket, rasgele uçuş, sömürü, tarama.

Danışman: Dr. Öğr. Üyesi Çiğdem ACI, Mersin Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Mersin.

ABSTRACT

IMPROVEMENT OF DRAGONFLY ALGORITHM BY BROWNIAN MOTION

Dragonfly Algorithm is one of the important optimization techniques developed in recent years. This algorithm takes the behavior of dragonflies in the nature as an example. The random flight motion of dragonflies in search of food in the nature or while running away from the enemy forms the basis of randomness in the algorithm. In the Dragonfly Algorithm, a "Levy Flight Mechanism", a random flight method, was used to reach optimal solution or to expand the search field. This mechanism provided a partial success when tested on benchmarking functions. However, the random movement of dragonflies is occasionally interrupted by this mechanism and by the step control action implemented within the algorithm. While this interruption can prevent dragonflies from behaving naturally, it also causes the algorithm to repeat itself when long steps are taken. In this study, to improve the results obtained by the comparison functions, an algorithm which is known as Brownian Motion and which is well known in the field of randomization but which lacks an example on the optimization field, is used. The Dragonfly Algorithm, modified by Brownian Motion, is applied to 15 single-objective, 6 multi-objective problem functions and compared over 200 iterations with the original algorithm. As a result of these comparisons, 12 of the 15 single objective problems and 6 of the 6 multi objective problems gave better results than the original algorithm. The modified algorithm was applied to the Welding Beam Design Problem, a real-state problem, and was able to calculate the optimal cost by 20%. As a result, the Brownian Motion proved that the Dragonfly Algorithm has a good ability to expand the field of exploration and to explore the area of exploitation.

Keywords: dragonfly optimization, brownian motion, random flight, exploitation, exploration.

Advisor: Ass. Prof. Dr. Çiğdem ACI, Computer Engineering, University of Mersin, Mersin.

TEŞEKKÜR

Yüksek lisans eğitimim boyunca, bilgi ve birikimlerini esirgemeden, sabırla bu çalışmanın ortaya çıkmasında katkıda bulunan ve daima yol gösteren, her zaman desteğini hissettiğim değerli danışman hocam Dr. Öğr. Üyesi Çiğdem ACI'ya sonsuz teşekkürlerimi sunarım. Tez savunmama katılımlarıyla ve değerli önerileriyle katkı veren Doç Dr. Alkan ALKAYA ve Dr. Öğr. Üyesi Esra SARAÇ EŞSİZ hocalarıma teşekkür ederim. Eğitim hayatım boyunca maddi, manevi hiçbir desteğini benden esirgemeyen annem Sibel GÜLCAN'a ve babam İlkay GÜLCAN'a, uzakta da olsa desteğini her zaman hissettiğim ağabeyim Hasan GÜLCAN'a teşekkür ederim.

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

KISALTMALAR ve SİMGELER

Kısaltma/Simge	Tanım
psi	Pounds per square inch
lb	Libre
DA	Yusufçuk Algorithm
MODA	Multi Objective Yusufçuk Algorithm
BH	Brownian Hareketi
PSO	Parçaçık Sürü Optimizasyonu

1. GİRİŞ

İnsanoğlu, varlığından bu yana doğa ile iç içe olmuş, çevresindeki olayları ve varlıkları anlamaya çalışmıştır. Bu ilişkileri anlamak ve herkesin anlayacağı şekilde anlatmak: madde ve madde bileşenlerini incelemede, bunların etkileşimini açıklamada fizik bilim dalının; maddenin iç yapısını ve bu yapıdaki değişiklikleri incelemede kimya bilim dalının; canlıların iç ve dış yapılarını ve hatta canlılık olaylarını incelemede biyoloji bilim dalının görevidir. Bu bilim dallarından elde edilen donanım ile insanlığa ve hatta tüm evrene hizmet edecek her türlü sistemi tasarlamak ise mühendislik alanının görevidir. Bu yüzden günümüze kadar süregelen süreçte elde edilen bilgi ve birikim; teknolojiyi, mühendisliği zorunlu kılmıştır. Fakat elde edilen bu bilgi ve birikimin tutarlı kullanılması pozitif sonuç için vazgeçilmez bir unsurdur.

Mühendislik çalışmalarının işleyişini anlamaya çalışırken iki önemli faktöre dikkat edilmesi gerekir: Analiz ve tasarım. Birbiriyle ayrı düşünülemeyecek olan bu iki kavram mühendisliğin temelini oluşturur. Tasarlanan bir sistemin ulaşacağı başarı analizine bağlıdır. Bu başarıya ulaşma, sadece tek bir yoldan olmayabilir. İhtiyaçları, arzu edilen şekilde sağlayan birden fazla çözüm bulunabilir. Bu çözümler arasından en iyisini (optimum), en kısa zamanda bulmaya çalışmak ise tam anlamıyla bir mühendislik çalışmasıdır [1]. Optimizasyon problemi olarak da adlandırılan bu çalışma, sistemden beklentiler arttıkça daha da karmaşıklaşmaktadır ve çözümü zor bir problem haline dönüşmektedir.

Optimizasyonun matematik ve/veya matematiksel programlamada ki manası şudur: Bir gerçel fonksiyonun minimum ya da maksimum noktasını bulmak amacı ile tanımlanan aralıkta bir gerçek değerini seçip fonksiyona yerine koyarak problemi incelemek veya çözmek işlemidir.

Mühendislikte ise optimizasyon, bir sistemde var olan kaynakların (işgücü, zaman, süreçler, hammaddeler, kapasite, ekipman gibi) en verimli şekilde kullanılarak, belirli hedeflere (maliyeti minimum düzeye indirgemek, kârı ve verimliliği olabilecek en üst düzeye çıkarmayı hedeflemek gibi) ulaşmayı sağlayan bir teknoloji olarak tanımlanmaktadır [2].

Mühendislik çalışmalarındaki optimizasyonda iki önemli bileşen vardır: modelleme ve çözümleme. Modelleme, günlük yaşantıdaki karşılaşılan problemin matematiksel formülasyonu; çözümleme ise bu modeli gerçekleyen optimum çözümün elde edilmesini kapsamaktadır.

1.1. Optimizasyon Çeşitleri

Varlıkların doğayla ilişkisi arttıkça, optimizasyona ihtiyaç duyulan problemlerin de artması, bu problemleri çözmek adına yapılan optimizasyon çalışmalarının da artmasına sebep olmuştur. Bu çalışmaları 6 temel alanda inceleyebiliriz [3]:

1.1.1. Deneme – Yanılma Optimizasyonu

Çıkışı etkileyen giriş parametrelerinin, işlemle ilgili fazla bilgili olmadan ayarlanma durumudur. Örneğin, televizyondaki görüntü ve sesin, antenin hangi açısında net ve iyi olacağı, ilgili kişiler tarafından sadece tahmin edilir. Çoğu kâşif ve deneysel çalışma yapanlar, bu tekniği kullanmıştır. Buna karşın matematiksel programlamada optimizasyon, matematiksel formülasyon ile açıklanır. Optimum çözüme ulaşmak için değişik metotlar uygulanabilir. Bu yaklaşım teorisyenler tarafından tercih edilir.

1.1.2. Tek ve Çok Parametreli Optimizasyon

Optimizasyonun boyutu parametre sayısına bağlıdır. Bir parametreli optimizasyon bir boyutlu iken, birden fazla parametreli fonksiyon için çok boyutlu optimizasyon gereklidir. Bu optimizasyon tipinde boyutla doğru orantılı olarak zorluk seviyesi de artar. Ayrıca çok boyutlu optimizasyon gerçeklenirken, bir boyutlu optimizasyon metodu yaklaşımı kullanılır.

1.1.3. Statik ve Dinamik Optimizasyon

Bu optimizasyon tipinde temel fark zamana bağımlılık durumudur. Statik optimizasyon zamandan bağımsız iken, dinamik optimizasyon zaman bağımlı sonuç üretir. Bu optimizasyon tipine örnek olarak bir çalışanın evinden şehrin merkezindeki işine gitmesi problemi düşünülebilir. Evinden işine gidilecek birden fazla yol olduğu kabul edilirse, problem en iyi yolu bulma problemidir. İlk olarak probleme mesafe bakımından bakılırsa problem statiktir. Çözüm ise, harita ve arabanın hız parametreleri kullanılarak bulunabilir. Bu tip problemlerde en kısa yol, en hızlı yol değildir. En hızlı

yolu bulmak dinamik bir problemdir ve zamana, havanın durumuna, kazalara vb. değişkenlere bağlıdır.

1.1.4. Sürekli ve Ayrık Parametreli Optimizasyon

Parametrelerin alacağı değere göre bu optimizasyon modeli ikiye ayrılır. Sonsuz değer alan parametreler sürekliyken, sınırlı değer alan parametreler ayrıktır. Ayrık parametreli optimizasyona örnek olarak yapılacak işlerin listesi verilebilir. İşlerin yapımı bağımsız olduğu için ayrık parametreli olarak düşünülebilir. Bu durum kombinasyonel optimizasyon olarak da isimlendirilebilir. Sürekli parametreli optimizasyona örnek olarak ise lineer ya da lineer olmayan bir fonksiyon grafiğinde f(x)'in minimizasyonu örnek verilebilir.

1.1.5. Sınırlı ve Sınırsız Optimizasyon

Sınırlı ve sınırsız optimizasyonda parametrelerin tanım aralığı önemlidir. Örneğin sınırlı optimizasyon, parametreleri bir tanım aralığında değerlendirir. Sınırsız optimizasyonda ise parametreler herhangi bir tanım aralığında olabilir. Bu optimizasyon tipinde değişkenlerin sınırları kaldırılarak, sınırlı parametreler sınırsız parametrelere çevrilirler. Çoğu nümerik optimizasyon fonksiyonları sınırsız parametrelerle çalışırlar. Örneğin bir f(x) fonksiyonunun sınırlarının $-1 < x < -1$ olduğunu varsayalım. Bu fonksiyon $x = \sin(u)$ tanımı kullanılarak sınırsız optimizasyona dönüştürülür. Burada u'nun değeri ne olursa olsun x'in değeri; (-1,1) aralığında değişecektir. Sınırlı optimizasyon, lineer denklemler ve lineer sınırlarla parametreleri optimize ettiği zaman, lineer program, sınırlar ve maliyet denklemleri lineer değil ise, program da lineer olmayan programlama problemi olarak adlandırılır.

1.1.6. Rasgele ve Minimum Araştırma Algoritmaları

Bazı algoritmalar, parametrelerin başlangıç değerlerini ayarlayarak en uygun değerleri azaltmaya çalışır. Bu araştırma tekniği hızlı olabilir fakat yerel minimumlara takılabilir. Bunlar sayısal yöntemlere dayalı klasik optimizasyon algoritmalarıdır. Bir parametreyle başlayarak, diğer parametrenin belirlenmesi bazı deterministik adımlarla gerçekleştirilir. Buna karşın rastgele yöntemler; parametrelerin en iyi çözümünü

bulmak için olasılık hesaplamaları kullanırlar. Bu yöntemler, küresel bir minimum bulmada daha yavaştır fakat daha başarılıdır.

Yukarıdaki gruplandırmanın sonucunda optimizasyon metotları, deterministik ve stokastik metotlar olmak üzere iki gruba ayırabilir. Deterministik optimizasyon metotları, lokal minimum veya maksimuma yakınsayan algoritmalardır. Türevsel hesaplamalar veya türevsel yaklaşımlar deterministik metotlara örnek verilebilir. Rasgele araştırma algoritmaları gibi stokastik metotlar ise global minimum veya maksimumu bulmada bazı stratejileri ve rasgele sayıları kullanırlar [3, 4].

Bu noktada, çalışmanın konusu dolayısıyla stokastik optimizasyon üzerinden devam etmek daha doğru olacaktır. Stokastik optimizasyon, rasgelelik mevcut olduğunda objektif fonksiyonu minimize veya maksimize etmek için bir yöntemler koleksiyonunu ifade eder. Rasgelelik, sorunda genellikle iki şeye işaret eder: maliyet fonksiyonu ve kısıtlamalar. Stokastik optimizasyonda da deterministik optimizasyon gibi, tüm problemler için iyi çalışan tek bir çözüm yöntemi yoktur. Sürekli en iyileme durumu söz konusudur.

Özellikle son 50 yılda, optimizasyon üzerinde çalışan bazı bilim insanları, yüksek performanslı, dinamik ve esnek yöntemler geliştirmek için dikkatlerini doğaya vermişlerdir. Bunun sebebi, doğada var olan ve kendiliğinden muazzam bir işleyişe sahip olan sistemlerin, optimizasyon problemlerini çok daha efektif bir şekilde çözebilecekleri düşüncesidir. Çünkü doğa, zaten var olan kompleks ve zor olan optimizasyon problemlerini, yine doğada var olan çoğu zaman gizemini sürdüren yöntemlerle çözmektedir. Doğada var olan sistemleri ve olayları temel alarak oluşturulan optimizasyon teknikleri sezgisel yöntemler olarak adlandırılır. Sezgisel yaklaşım en iyiyi bulma garantisine sahip değildir ve bu nedenle genel olarak optimumdan daha kötü çözümler getirir. Bununla birlikte, sezgisel algoritmalar genellikle 'makul' bir sürede iyi çözümler bulurlar.

Birçok sezgisel algoritma çok spesifik ve problem-bağımlıdır. Öte yandan, bir meta sezgisel algoritma, sezgisel optimizasyon algoritmalarını geliştirmek için bir dizi kılavuz veya strateji sağlayan yüksek seviyeli, problemden bağımsız algoritmik bir sistem çalışmasıdır. Ancak somut bir tanım zor olmuştur ve pratikte birçok araştırmacı bu terimleri değiştirebilmektedir. Bu nedenle, meta sezgisel terimi aynı zamanda, bir sezgisel optimizasyon algoritmasının probleme özel bir uygulamasına atıfta bulunmak için de kullanılır.

1.2 Meta Sezgisel Optimizasyon Algoritmaları

Meta sezgisel optimizasyonda kaynaklar ve zaman her zaman sınırlı olduğu için, bu mevcut kaynakların optimal faydası çok önemlidir. Çoğu gerçek dünya problemleri, çeşitli karmaşık kısıtlamalar altında yüksek oranda doğrusal olmayan ve çok biçimlidir. Farklı hedefler genellikle çelişkilidir. Tek bir hedef için bile, bazen optimal çözümler mevcut olmayabilir. Genel olarak optimal bir çözüm veya hatta optimale yakın çözümler bulmak kolay bir iş değildir [5].

Meta sezgisel optimizasyon algoritmaları genel olarak, biyoloji, fizik, sosyal, müzik, kimya ve sürü tabanlı olmak üzere altı farklı grupta değerlendirilmektedir. Sürü zekasına dayalı optimizasyon bu tezin ana çalışma alanıdır. Dolayısıyla sürü zekâsı ayrı bir başlıkta incelenmektedir.

1.2.1. Sürü Zekasına Dayalı Meta Sezgisel Optimizasyon

Bazen tek başına bir şey yapamayan bireyler, kolektif olarak hareket ettiklerinde çok akıllıca davranabilirler. Bir topluluğun bireyleri, en iyi bireylerin veya diğer bireylerin davranışlarını veya kendi deneyimlerini kullanır ve bu bilgileri gelecekte karşılaşacakları sorunları çözmek için bir araç olarak kullanırlar. Örneğin, sürüde yaşayan canlılardan biri, bir tehlikeyi algıladığında bu tehlikeye tepki gösterir ve reaksiyon sürü halinde hareket eder. Bu durum tüm bireylerin tehlike anında uyum içinde hareket etmesine izin verir. Doğadaki canlıların bu hareketleri gözlemlenerek sürü zekasına dayalı optimizasyon algoritmaları geliştirilmiştir [6].

Sürü zekâsı tabanlı optimizasyon algoritmaları kuş, balık, kedi ve arı gibi canlı sürülerinin hareketlerinin incelenmesiyle geliştirilmiştir. Bu tip algoritmalar arasında en bilineni Karınca Kolonisi Optimizasyon Algoritması'dır. Gerçek bir karınca kolonisi davranışlarının, matematiksel modellemeleri üzerine dayalı bir optimizasyon algoritmasıdır. İlk çalışma Dorigo ve arkadaşları tarafından 1991 yılında yapılmıştır [7]. Yapılan çalışma sonucunda kendi sistemlerine 'karınca sistemi', elde ettikleri algoritmaya ise "karınca algoritması" adını vermişlerdir. Algoritmaya göre karınca, çevresel şartları göz önünde bulundurarak besin kaynağı ile yuvası arasında gidebileceği yolları belirlemektedir. Belirlenen bu yolların birinden ilk geçen karınca, yola feromon adı verilen bir koku bırakmaktadır. Eğer yol kısa ise diğer karınca geçene

kadar koku tam olarak uçmadığından, bu koku daha yoğun olmaktadır ve diğer karıncalar da aynı şekilde yollarına devam etmektedirler. İki yolun kesiştiği noktada ise karınca, hangi yola gideceğini kendisi belirlemektedir. Hangi yolu seçeceğine ilk önce koku miktarının yoğunluğuna göre ikinci olarak ise rasgele bir ölçüte göre karar vermektedir. Bu rasgele seçimin nedeni ise bütün karıncaların aynı yolda gitmesini engelleyerek yeni ve daha kısa yolları keşfetmektir.

Bu alandaki diğer bir çalışma ise Parçacık Sürü Optimizasyonu (PSO)'dur. Bu optimizasyon temelde sürüye ait bireylerin birbirlerini geliştirmesine dayanan bir algoritmadır. Bu algoritma sürü halinde hareket eden balıklar ve böceklerden esinlenerek Dr. Kennedy ve Dr. Eberhart tarafından 1995 yılında geliştirilmiştir[8]. Sürüdeki her bireye parçacık denir ve bu parçacıkların oluşturduğu popülasyona da sürü denir. Bu optimizasyonun amacı sürüdeki en iyi konuma sahip parçacığının yerinin tespit edilip, diğer parçacıkların da o yöne hareketinin sağlanmasıdır. Parçacıklar, bir sonraki konumunu, geçmiş tecrübelerine ve sürüdeki en iyi pozisyona sahip bireye dayanarak iyileştirmeyi hedefler.

Meta sezgisel algoritmalara verilecek değerli örneklerden biri de Yapay Arı Koloni Algoritması'dır. Yapay Arı Kolonisi Algoritması, Derviş Karaboğa tarafından 2005'te arıların topluluk olarak yiyecek arama davranışlarını temel alarak geliştirilmiş bir optimizasyon algoritmasıdır[9]. Bu algoritma, bal arısı sürülerinin kendilerine özgü zeki davranışlarını örnek alarak, arıların besin ararken kullandıkları yöntemlerden esinlenerek oluşturulmuş bir optimizasyon algoritmasıdır. Sosyal bir düzen içinde yaşayan bal arıları, içgüdüsel olarak bu düzeni bilir ve hayatını bu kurallara uygun olarak devam ettirir. Kovandaki her arının görevi bellidir ve arılar asla bu görevlerinden sapmazlar. Belirtilen sosyal düzen, yiyeceklerin depolanması, balın getirilmesi, iletişim ve besin arama gibi işleri kapsamaktadır. Koloni halinde yaşayan bu sosyal yaşamda 3 arı çeşidi bulunmaktadır: Bunlar kraliçe arı, erkek arı ve dişi olan işçi arılardır. İşçi arılar, gözcü arılar ve kâşif arılar olmak üzere yapay arı kolonisi algoritmasında da üç grup arı bulunmaktadır. Bu modelde işçi arılar kaynaklara gönderilerek besin miktarlarını hesaplar. Aynı zamanda da gözcü arılar da başka kaynaklara gönderilerek besin miktarları hesaplanır. Daha sonra rasgele yeni kaynaklar bulmaları için kâşif arılar gönderilir. Durdurma kriteri sağlanana kadar bu döngü devam eder.

Geliştirilen bazı diğer meta sezgisel optimizasyon algoritmalarına örnek olarak, 2006 yılında geliştirilen Kedi Sürüsü Optimizasyonu [10], 2008 yılında Xin-She Yang tarafından geliştirilen Ateşböceği Algoritması [11], 2011 yılında Ramin Rajabion

tarafından geliştirilen Guguk Kuşu Optimizasyon Algoritması [12], 2012 yılında Kevin Passino tarafından geliştirilen Bakteriyel Besin Arama Optimizasyonu [13], 2015 yılında geliştirilen Fil Sürüsü Optimizasyonu [14] verilebilir.

Geçmişten günümüze, gerçek durum problemlerinin artmasıyla ve optimizasyon algoritmalarının bu problemler üzerindeki etkin çözümü ortaya çıktıkça, bilim insanlarını daha efektif çözümler üretmeye itmiştir. Bu efektif çözüm arayışı, optimizasyonda doğadaki canlıların olaylar karşısındaki muazzam çözümlerini inceleme anlayışını getirmiştir. Bilim insanları doğadaki varlıkların olaylar karşısındaki davranışlarını, tecrübelerini ve reaksiyonlarını izleyip çeşitli algoritmalar geliştirmiştir. Bu algoritmalar yapay zekâ alanında, doğadan esinli optimizasyon algoritmaları olarak bilinmektedir. Bugüne kadar bu doğa esinli optimizasyon algoritmaları, bir çok gerçek durum çalışmasını başarıyla çözmüştür.

Problemlerin bu denli efektif çözülmesindeki ana sebeplerden birisi rasgele davranma hareketidir. Rasgele hareketlilik, problemin çözümünde tek bir yoldan gidilmemesini sağlamaktadır. Optimizasyon sırasında bulunan çözüm, optimuma en yakın sonuç olsa bile ve hatta optimum olsa bile, rasgele davranış her zaman daha iyi bir çözüm olabileceği önerisini getirmektedir. Bu öneri çoğu zaman problemlerdeki yerel en iyi noktalara takılmaları önlemektedir.

Rasgele davranma hareketinin getirdiği önemli bir diğer fayda ise arama uzayında taranmayan alan bırakmama başarısıdır. Eğer rasgele davranma hareketi olmazsa, optimizasyon belirlenen arama uzayının sadece bir bölgesinde takılabilir ve diğer bölgelerdeki sonuçları hiç göremeyebilir. Rasgele hareket sayesinde arama uzayının her alanı didik didik aranabilmektedir.

Geliştirilen optimizasyon tekniklerinde genel olarak 2 tip randomizasyon kullanılmıştır. Bunlardan birisi klasik rasgele hareket, yani işlemcinin üreteceği rasgele sayıya dayalı randomizasyon, diğeri ise Levy Uçuş Mekanizması'dır. Bu mekanizmada da yine işlemcinin üreteceği rasgele sayı vardır fakat mekanizma bir istatistiksel matematik formülüne dayanmaktadır. Kullanılan her iki yöntem de problemlerin çözümünde önemli iyileştirmeler sağlamıştır.

Bu tez çalışmasında, sürü zekasına dayalı bir optimizasyon tekniği olan Yusufçuk Algoritması'nın, optimizasyon sırasında kullanılan Levy Uçuş Mekanizması yerine Brownian Hareket'i ile modifikasyonu gerçeklenmiştir. Modifiye edilen algoritma, tek hedefli optimizasyonda 15, iki hedefli optimizasyonda ise 6 farklı kıyaslama fonksiyonu üzerinde test edilmiştir. Tek hedefli optimizasyonda elde edilen sonuçlar hem bulunan minimum nokta üzerinden hem de problemin 200 ayrı

çözümünden elde edilen ortalama değerler üzerinden karşılaştırılmıştır. Bu karşılaştırma sonucunda minimum değerlerde, 15 fonksiyondan13'ünde; ortalama değerlerde ise 15 fonksiyondan 12'sinde daha optimal değerler elde edilmiştir. İki hedefli optimizasyonda ise elde edilen grafik sonuçlarında 6 fonksiyonun 6'sında da başarı sağlanırken, nümerik sonuçlarda toplam elde edilen 12 minimum değerin 6'sında kesin başarı sağlanmıştır. 1'inde ise orijinal algoritmanın sonucuna %1 uzaklıktadır. Modifiye edilen algoritma son olarak bir gerçek durum problemi olan Kaynaklı Kiriş Tasarı Problemi üzerinde uygulanmıştır. Bu problemden elde edilen sonuçlar da orijinal algoritmanın sonuçlarıyla karşılaştırıldığında %25 iyileşme sağlandığı gözlemlenmiştir.

2. KAYNAK ARAŞTIRMALARI

Bu bölümde metasezgisel yöntemlere dayalı geliştirilen optimizasyon algoritmalarına ait daha önce yapılmış çalışmalara yer verilmiştir.

2.1 Optimizasyon Algoritmalarının İyileştirilmesi Hakkında Yapılan Çalışmalar

Günümüze kadar gelinen süreçte, optimizasyon algoritmaları hem kıyaslama fonksiyonları üzerinde hem de gerçek hayat problemleri üzerinde önemli başarılar sağlamıştır. Buna karşın bilim dünyasının doğası gereği her zaman daha iyisi istenmiş ve amaçlanmıştır. Bu yüzden geliştirilen yeni algoritmaların dışında, var olan algoritmalar üzerinde de iyileştirilmeler yapılmıştır. Bu iyileştirmeler, kimi zaman diğer algoritmalarla hibritleme yoluna giderek olurken kimi zaman da çözülmesi amaçlanan gerçek dünya problemine uygun şekilde algoritmayı modifiye ederek olmuştur.

Bu bağlamda yapılan çalışmalara ilk örneklerden biri olarak Song ve arkadaşlarının 1999 yılındaki çalışması örnek verilebilir [15]. Bu çalışmada, arı kolonisi algoritması, çok hedefli bir problem yaklaşımındeki kısıtlamarı çözmek için dağıtık hesaplama ve sezgizel bir aç gözlü yaklaşımla iyileştirilmiştir. İlerleyen yıllarda gelindiğinde, 2005 yılında Yan ve arkadaşları, sistematik iş planlama problemi için karınca kolonisini, geçilen yollardaki feromon dengesiyle oynarak geliştirmişlerdir [16]. Araç yönlendirme problemi için ise yine karınca kolonisi algoritması birkaç kez modifiye edilmiştir. Bunlardan biri, Chen ve arkadaşlarının 2006 yılında yaptığı

çalışmadır [17]. Bu çalışmada yeni bir durum geçiş kontrolü, feromon dağılımı ve yerel arama algoritması geliştirilerek iyileştirme yapılmıştır. Aynı problemi Yu Bin ve arkadaşları 2009 yılında, feromon dağılımını değiştirmeye ek olarak mutasyon sistemiyle iyileştirdikleri karınca kolonisi algoritması ile çözmüşlerdir [18]. Bu problem ile ilgili bir diğer değerli çalışma ise Yu ve arkadaşlarının 2011 yılında yaptığı, çoklu depo sistemindeki araç yönlendirme sorununu, algoritmayı paralelleştirerek çözdükleri çalışmadır [19]. Karınca kolonisinin geliştirilerek kullanıldığı bir diğer alan ise görüntü işlemedir. Han ve Shi, 2007 yılındaki çalışmalarında, bulanık kümeleme için, karıncaları küme merkezlerine yerleştirmiş ve sezgisel yaklaşımı artırarak bu problemi çözmüşlerdir [20]. Karınca kolonisinin iyileştirmelerine son örnek olarak ise Kaveh ve Talahatari'nin 2010 yılındaki çalışmaları örnek verilebilir. Bu çalışmada, karınca kolonisi algoritması, alt optimizasyon mekanizması adı verilen ve bir arama alanı güncelleme tekniği olarak çalışan sonlu eleman yöntemi kullanılarak iyileştirilmiş, sınırlamalı mühendislik problemleri başarı ile çözülmüştür [21].

Sezgisel algoritmaların temel taşlarından birisi olan Parçacık Sürü Optimizasyonu ile ilgili de pek çok iyileştirme çalışması yapılmıştır. Bunlara ilk örnek olarak Peter J. Angeline'nin 1999 yılındaki çalışması gösterilebilir. Kendisi bu çalışmada, parçacık sürü optimizasyonunu, evrimsel algoritmadaki seçim metodu ile hibritleyerek iyileştirmiş ve başarılı sonuçları listelemiştir [22]. 2005 yılında ise Liu ve arkadaşları, parçacık sürü optimizasyonunu, kaos optimizasyon algoritması ile hibritleyerek iyileştirmişlerdir [23]. Bu çalışmada parçacıkların keşif ve sömürü yeteneklerini geliştirmek için bir adaptif eylemsizlik ağırlığı algoritmaya eklenmiştir. Eklenen bu ağırlık ile birlikte kaos optimizasyonu hibritlenerek kaotik arama davranışı algoritmaya dahil edilmiştir. 2007 yılında ise Jiang ve arkadaşları, algoritmadaki parçacıkları alt kümelere ayırarak bölgelere dağıtmışlardır ve bölgelerdeki en iyi sonuçlar paylaşılarak algoritma optimize edilerek iyileştirilmiştir [24]. Parçacık sürü optimizasyonundaki iyileştirmelere son örnek olarak ise Park ve arkadaşlarının 2010 yılındaki çalışması örnek verilebilir [25]. Bu çalışmada, ekonomik dağıtım problemini, doğrusal olarak azalan eylemsizlik ağırlıklarıyla birleştirilen kaotik dizileri kullanarak, PSO'nun hem keşif hem de sömürü kabiliyetini arttırarak çözmüştür.

Derviş Karaboğa tarafından geliştirilen bir metasezgisel optimizasyon algoritması olan Yapay Arı Kolonisi Algoritması ile ilgili de yapılan birkaç iyileştirme çalışması göze çarpmaktadır. Gao ve Liu 2011 yılındaki çalışmalarında, diferansiyel evrimden etkilenerek, yapay arı kolonisi algoritmasını global optimizasyon için küresel yakınsama bloğunda kaotik sistem ve zıtlık temelli öğrenme sistemiyle geliştirerek

iyileştirmişlerdir [26]. Günümüze yakın olarak 2016 yılında Forsati ve arkadaşları, veri kümeleme problemini, algoritmaya klonlama ve adalet kavramlarını ekleyerek çözmüşlerdir [27]. Bu çalışmada, klonlama özelliği, yeni çözümler üretilirken önceki deneyimlerden faydalanılmasını sağlarken, k-means algoritması ile yerel aramalarda daha iyi sonuçlar elde edilmesi iyileştirmede başrol oynamıştır. Son olarak ise 2012 yılında, Bahriye Akay ve Derviş Karaboğa'nın gerçek parametreli optimizasyon için modifiye ettiği yapay arı kolonisi algoritması incelemeye değerdir. Bu çalışmada pertürbasyon frekansı ve büyüklüğü kontrol edilerek algoritma modifiye edilmiştir [28].

Yusufçuk Algoritması'nın geliştirilmesinden bu yana ise, birkaç çalışmada kullanıldığı gözlemlenmiştir. Bu çalışmalardan biri Salam ve arkadaşlarının 2016 yılında gerçekleştirdiği tahmin problemi için algoritmayı, aşırı öğrenme makinesi ile hibritleme çalışmasıdır [29]. Bu çalışmada aşırı öğrenme makine modeli, veri regresyon ve sınıflandırma problemleri için ümit vaat eden bir yöntem olarak görülmektedir. Hızlı eğitim avantajı vardır, ancak her zaman gizli katmanda çok sayıda düğüm gerektirir. Gizli katmanda çok sayıda düğüm kullanılması test/değerlendirme süresini artırır. Ayrıca, gizli katmandaki ağırlık ve sapma ayarlarının optimallik garantisi yoktur. Yusufçuk algoritması, aşırı öğrenme makinesinin performansını arttırmak için gizli katmandaki daha az sayıda düğüm seçmek için kullanılmıştır. Ayrıca optimum gizli katman ağırlıklarını ve sapmalarını seçmek için kullanılır. 2016'daki diğer bir çalışma ise Daely ve arkadaşlarına aittir [30]. Yusufçuk Algoritması bu çalışmada, menzil temelli kablosuz düğüm yerleşimi için kullanılmıştır. Yer belirleme programı genellikle, yerel düğümlerin lokasyonlarına göre konumunu belirlemek için her kablosuz düğüme yerleştirilir. Sürü zekası, klasik optimizasyon algoritmasına kıyasla düşük hesaplama karmaşıklığı nedeniyle birçok kablosuz düğüm yerleşimi için kullanılmıştır. Çalışmanın sonuçları da belirlenmiş bir alanda rastgele dağıtılan düğüm noktalarının yerini tahmininde yusufçuk algoritmasının hata payının çok düşük olduğunu kanıtlamıştır. 2017 yılına gelindiğinde ise Ranjini ve arkadaşları hafıza tabanlı bir hibrit Yusufçuk Algoritması önermişlerdir [31]. Bu çalışmada nümerik optimizasyon problemleri için hibrit bir yaklaşım geliştirilmiştir. Yusufçuk Algoritması, yerel optimuma erken yakınsamasına yol açabilecek dahili hafızaya sahip değildir. Yapılan hibritlemede Parçacık Sürü Optimizasyonu'nun (PSO) pbest ve gbest konsepti, potansiyel aday çözümler için arama sürecini yönlendirmek amacıyla konvansiyonel yusufçuk algoritmasına eklenmiştir ve daha sonra PSO, arama alanını daha fazla kullanmak için Yusufçuk Algoriması'nın pbestiyle başlatılmıştır. Önerilen yöntem, küresel optimal

çözümler elde etmek için Yusufçuk Algoritması'nın arama yeteneğini ve PSO'nun sömürü kabiliyetini birleştirmektedir. Yusufçuk Algoritması'nın bir diğer uygulaması ise yine 2017 yılında Mafarja ve arkadaşları tarafından özellik seçimi üzerine yapılmıştır [32]. Sarmalayıcı özellik seçme yöntemleri, orijinal özellikten alınan özelliklerin sayısını azaltmayı ve sınıflandırma doğruluğunu eş zamanlı olarak geliştirmeyi amaçlamaktadır. Mafarja ve arkadaşlarının çalışmasında da, İkili Yusufçuk Algoritması'na dayanan bir sarmalayıcı özellik seçim algoritması önerilmiştir. Sonuçlar, önerilen algoritmanın özellik alanını arama ve sınıflandırma görevleri için en bilgilendirici özellikleri seçme yeteneğini göstermiştir. Son olarak Tharwat ve arkadaşlarının 2018 yılında yaptığı destek vektör makinesi ile parametre optimizasyonunda, Yusufçuk Algıritması kullanılmıştır [33].

Yapılan çalışmalarda da görüldüğü üzere, direkt olarak Yusufçuk Algoritması'nı iyileştirme amaçlı bir çalışma yapılmamıştır. Bu tezde Yusufçuk Algoritması'nın rasgele hareketini iyileştirme çalışması yapılmıştır.

2.2 Optimizasyonda Randomizasyon

Randomizasyon, sürü zekasına dayalı optimizasyon tekniklerinin vazgeçilmez ana unsurlarından biridir. Hem keşif hem de sömürü aşamalarında çok önemli bir rol oynamaktadır. Bu tür bir randomizasyonun özü rastgele yürüyüştür. Rastgele bir yürüyüş, bir dizi ardışık rastgele adımı içeren rastlantısal bir süreçtir [5].

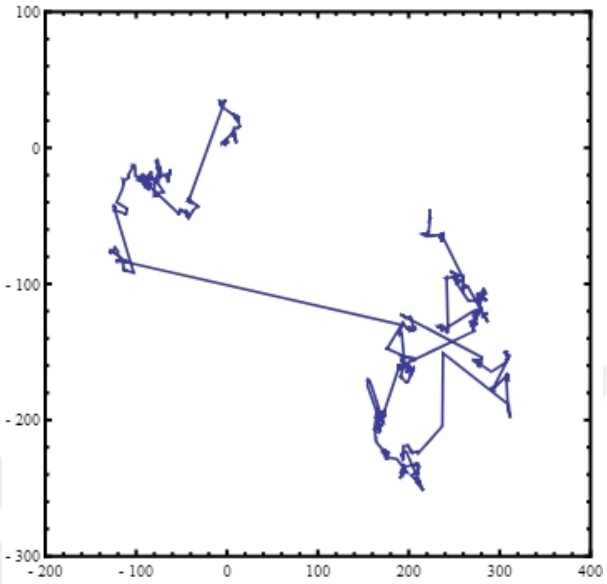
Randomizasyon, rastgele bir şey yapma sürecidir. Örneğin bir endeksin rastgele bir permütasyonunun oluşturulması, popülasyonun rastgele örneklemini seçmek, rasgele sayı üretimi veya veri akışını dönüştürme.

Randomizasyon isteğe bağlı değildir. Örneğin, bir popülasyondan bireylerin rastgele bir örneklemi, her bireyin bilinen bir örnekleme olasılığına sahip olduğu bir örnekte bulunur. Bu, keyfi bireylerin olasılık örneklemesi ile karşılaştırılabilir. Rastgele bir yürüyüş, rastgele hareket eden nesnelerin başladıkları yerden uzaklaştığı bir süreçtir.

2.2.1 Levy Uçuş Mekanizması

Levy Uçuş'u adını Fransız matematikçi Paul Levy'den almaktadır. Levy uçuşlarının karakteristiği, adım boyutlarının, uzun kuyrukların azalan olasılıklarının, artan uzunluklara baskın gelmesi için yeterince küçük olmadığı anlamına gelen, bir

baskın kuyruklu olasılık dağılımından seçilmesidir. Teknik olarak bu dağılım metodu sonsuz bir varyansa sahiptir (olası uzunlukta) [34]. Şekil 2.1'te Levy Uçuşu'nun ilk 1000 adımdaki örneği gösterilmektedir:



Şekil 2.1. Levy Uçuşu'nun ilk 1000 adımdaki simulasyonu

Geçmişte randomizasyon için bu mekanizmanın kullanıldığı birçok örnek bulunmaktadır. Bunlardan biri Haklı ve Uğuz'un 2014'te önerdikleri metotda Parçaçık Sürü Optimizasyonu'nu Levy Uçuş Mekanizması ile gerçeklemişlerdir [35]. Bu çalışmada, optimizasyon sırasında ajanların lokal minimaya takılması ve erken yakınsama problemi sebebiyle Levy Uçuş Mekanizması ile bir modifikasyon gerçeklenmiştir ve başarılı sonuçlar alınmıştır. Bu alandaki diğer bir çalışma ise, Heidari ve Pahlavani'nin 2016 yılındaki çalışmalarıdır [36]. Çalışmalarında, sezgisel bir optimizasyon algoritması olan Gri Kurt Optimizasyonu'na Levy Uçuş Mekanizması'nı adapte etmişlerdir ve PSO'daki soruna benzer şekilde, kurtların konum çeşitliliğinin fazla olmamasının lokal minimaya sebep olduğunu öngörmüşlerdir ve bu sorunu Levy Uçuş Mekanizması ile çözmüşlerdir. Ilya Pavlyukevich, 2007'de global optimizasyon için yeni bir stokastik algoritmayı teorik olarak doğrulamak ve teorik olarak gerekçelendirmek için yaptığı araştırmada Levy uçuşunu kullanmıştır [37]. 2008'de Pierre Barthelemy ve arkadaşları, ışığın iletimini ve yayılımını optimize etmek için Levy uçuş tekniğini kullanmışlardır [38]. Son olarak ise Seyedali Mirjalili tarafından 2016 yılında geliştirilen Yusufçuk Algoritması'nda Levy Uçuş Mekanizması kullanılmıştır [39]. Bir alt bölümde, bu algoritmada mekanizmanın nasıl kullanıldığı detaylıca

incelenmiştir. Levy Uçuş Mekanizması'nın optimizasyon problemlerinde kullanılmasıyla ilgili geniş kapsamlı bir çalışma olan, Kamaruzaman ve arkadaşlarının 2013 yılındaki çalışması daha detaylı bilgilendirme için incelenebilir [40].

2.2.1.1 Levy Uçuş Mekanizması ve Yusufçuk Algoritması

Rasgeleliği, olasılıksal davranışı ve yapay yusufçukların keşfini geliştirmek adına, komşuluk çözümü kalmadığında rasgele yürüyüş (Levy Uçuş Mekanizması) çözümüne gidilir. Buna göre yapay yusufçukların konumu şu şekilde güncellenir:

$$X_{t+1} = X_t + Levy(d) \times X_t \quad (2.1)$$

$$Levy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\beta}} \quad (2.2)$$

$$\sigma = \left(\frac{\tau(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\tau\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \quad (2.3)$$

$$\tau(x) = (x-1)! \quad (2.4)$$

Denklem 2.1'deki d pozisyon vektörünün boyutunu belirtirken, Denklem 2.2'deki r_1 ve r_2 [0, 1] aralığında rasgele sayılardır. β ise sabit bir değerdir.

Yusufçuk Algoritması'ndaki Levy Uçuş Mekanizması'nda, uçuş metodunun orijinal matematiksel formülünde yer almayan bir çarpım alınmıştır. Bu çarpım Denklem 2.2'de görüldüğü gibi Levy Uçuş büyüklüğünün %1'i alınarak sağlanmıştır. Buradaki amaç adım boyutunu kontrol edebilmektir. Bu çarpım, bir çözüm değerinin yani en iyi bireyin konumunun, Levy Uçuş'unu uyguladıktan sonra ne kadar saptığını tanımlar. %1'lik sapma değeri uygulamadaki değişkenlerin aralığına göre ayarlanabilir. Örneğin, uygulamadaki değişken aralığı [-10e6, 10e6] ise %1 lik çarpım değeri 1 olarak ayarlanabilir.

Levy Uçuş Mekanizması, Yusufçuk Algoritması'nı çözümünü belirli bir ölçüde yükseltmesine karşın olumsuz bir yönü göze çarpmaktadır. Mekanizmanın kendi karakteristik özelliğinden dolayı, bazen çok uzun adımlar meydana gelebilmektedir (Şekil 2.1). Bu büyük adımlar algoritmada 2 şekilde kontrol edilmeye çalışılmıştır. Bunlardan birincisi, eğer meydana gelen uzun adım neticesinde ajanlar arama uzayının

dışarısına çıkmak zorunda kalırsa yeni bir adım vektörü üretilmektedir. Fakat bu çözümün her zaman doğru sonuç vereceği bilinmemektedir. Yani üretilecek yeni adım, genel işleyişi geriye götürebilir. Algoritmanın 2. kontrol şekli ise, Denklem 2.2'da görüldüğü şekilde adım boyutunun %1'ini veya uygulamadaki değişken aralığına göre farklı bir yüzdesini almaktır. Bu çözüm metodu ise, 1. Çözüme oranla daha fazla doğru sonuç vermiştir. Fakat bu da Levy Uçuşu'unun doğasına aykırı bir şekilde adımı kontrol etmektir ve rasgele hareketi (uzun da olsa) bir ölçüde önlemektir.

2.2.2 Brownian Hareketi

Rastgele hareket mekanizmalarından bir diğeri Brownian Hareketi'dir. Bu metot serbest durumdaki sıvı/gaz moleküllerinin hareketinden ilham alır [41]. Literatüre girişi ise şöyle olmuştur: Jan Ingenhousz kömür ve toz parçacıklarının rastlantısal hareketini alkolün içinde yüzerken 1779'da gözlemlemişti [42]. Ancak Brownian hareketinin keşfi çoğunlukla 1828 yılında hareketi gözlemleyen botanik bilimci Robert Brown'a dayandırılır [43]. Brown, suda yüzen polen parçacıklarını mikroskop ile inceliyordu. Bu sırada polenin boşlukları içinde, rastlantısal olarak hareket eden ufak parçacıklar gözlemledi. Aynı deneyi toz ile tekrarlayarak hareketin polenin canlı olmasından kaynaklanmadığını doğruladıysa da hareketin kaynağını saptayamadı.

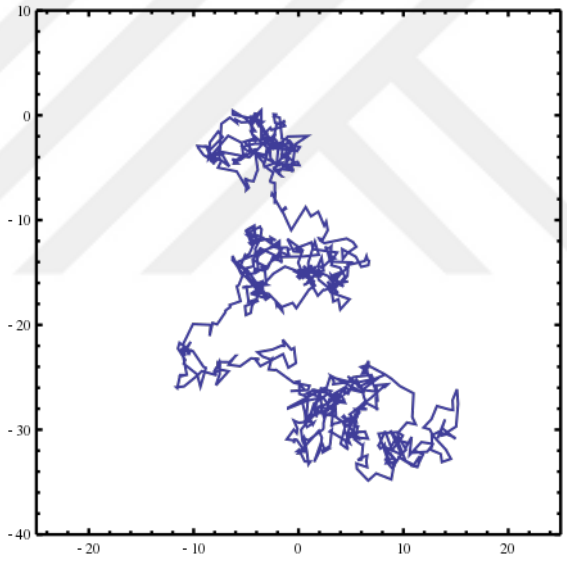
Brownian hareketi, bir miktarın sürekli olarak küçük, rastgele dalgalanmalardan geçtiği çeşitli fiziksel fenomenlerden herhangi biri olarak tanımlanır. Brownian hareketi, akışkandaki hızlı hareket eden moleküller ile çarpışmalarından kaynaklanan bir sıvı (bir sıvı veya bir gaz) içinde süspansiyon haline getirilmiş parçacıkların rastgele hareketidir.

Brownian hareketinin matematiksel tanımı nispeten basit bir olasılık hesaplamasıdır. Brown hareketinin matematiksel olarak ilk kez açıklanması, 1880 yılında en küçük kareler metodu üzerine yazdığı makalesiyle Thorvald N. Thiele tarafından olmuştur [44]. Ama ilk kez fizikçilerin konuya dikkatini çeken, Albert Einstein'ın bu konudaki bağımsız araştırması oldu. Maddenin atomik doğası o dönem hala tartışılırken, Einstein ve Marian Smoluchowski, eğer sıvıların kinetik teorisi doğru ise su moleküllerinin rastlantısal olarak hareket etmesi gerektiğini fark etti. Böylece küçük bir parçacık, rasgele yönlerden, rasgele şiddetlerde gelen birçok kuvvetin etkisi altında olmalıydı. Küçük parçacık, bu bombardıman altında aynen Robert Brown'un tarif ettiği gibi hareket etmeliydi. Bu araştırmalara ek olarak Theodor Svedberg, Brown

hareketini koloidlerde, Felix Ehrenhaft ise dünyanın atmosferinde asılı gümüş parçacıklarında gözlemledi.

Modern model, sürekli zamanlı bir stokastik sürecin işlevini tanımlayan Norbert Wiener onuruna verilen Wiener sürecidir. Brownian hareketi bir Gauss süreci ve sürekli süre boyunca meydana gelen sürekli yol ile bir Markov süreci olarak kabul edilir.

Brownian Hareketi'nin önemli bir özelliği izotropik olmasıdır. Yani modelin özelliği tamamen yönden bağımsızdır. Bu da keşif özelliğini yüksek oranda artırmaktadır. Öte yandan adımların boyutunun hem kontrol edilebilir büyüklükte hem de zamana dayalı rasgele bir şekilde oluşması, arama uzayının dışına çıkılmasını engelleyerek hareketin devamlılığını sağlamaktadır. Şekil 2.2, 1000 adımda Brownian Hareketi örneğini göstermektedir.



Şekil 2.2. Brownian Hareketi'nin ilk 1000 adımdaki simulasyonu

2.2.3 Brownian Hareketi ve Levy Uçuş Mekanizmasının Farkı

Matematiksel olarak bir rasgele yürüyüş şu şekilde tanımlanabilir:

$$X_{N+1} = X_N + W_N \tag{2.5}$$

Bu denklemde (2.5) X_N, N. adımda var olan çözüm; W_N ise bilinen bir olasılık dağılımından üretilmiş rasgele bir vektördür. Eğer burada W_N, Gauss dağılımından

üretilirse, rasgele yürüyüş izotropik olur. Bu durumda da hareket normal difüzyon şeklini alır ve Brownian Hareketi olarak adlandırılır. Beklenen adım boyutu ise karekök ölçekleme özelliğine sahip şu şekilde modellenebilir:

$$R(N) \propto \sqrt{N} \tag{2.6}$$

Eğer adımlar (W_N), Levy dağılımı veya Cauchy dağılımı gibi baskın kuyruklu bir olasılık dağılımından elde edilirse, difüzyon anormal hale gelir. Bu durumda beklenen adım boyutu şu hale dönüşür:

$$R(N) \propto N^q, q > 0 \tag{2.7}$$

Eğer q ≥ 1⁄2 ise, difüzyon süper difüzyon olarak adlandırılır. Adım boyutları için hem Levy dağılımı hem de Cauchy dağılımı, süper difüzyona yol açacak büyük adımların bir kısmına sahip olabilir. Bu, ortalama mesafenin normal difüzyon için olduğundan daha hızlı arttığı anlamına gelir.

2.3 Çalışmanın Motivasyonu

Bu hareketin şu ana kadar optimizasyon alanında kullanıldığı yalnızca 1 çalışma bulunmaktadır. Abdechiri ve arkadaşlarının 2012'de yaptığı çalışma bu manada bir temel taşı olmaktadır [41]. Bu algoritma ile direkt olarak doğadaki gaz moleküllerinin Brown Hareketi kullanılarak bir optimizasyon yöntemi geliştirilmiştir ve çok başarılı sonuçlar elde edilmiştir. Bu sonuçlar PSO ve Genetik Algoritma (GA) gibi iyi bilinen sezgisel algoritmalar ile karşılaştırılmıştır.

Bunun dışında, Yusufçuk Algoritması için, hibritleme veya bir gerçek dünya problemini çözmek için modifiye etmek dışında bir geliştirme veya iyileştirme yapılmamıştır. Biz bu çalışmada, direkt olarak Yusufçuk Algoritması'nı iyileştirmek adına Yusufçuk Algoritması'nı Brownian Hareket ile modifiye ettik ve elde ettiğimiz sonuçlar başarılı bir şekilde ortaya kondu.

3. MATERYAL ve YÖNTEM

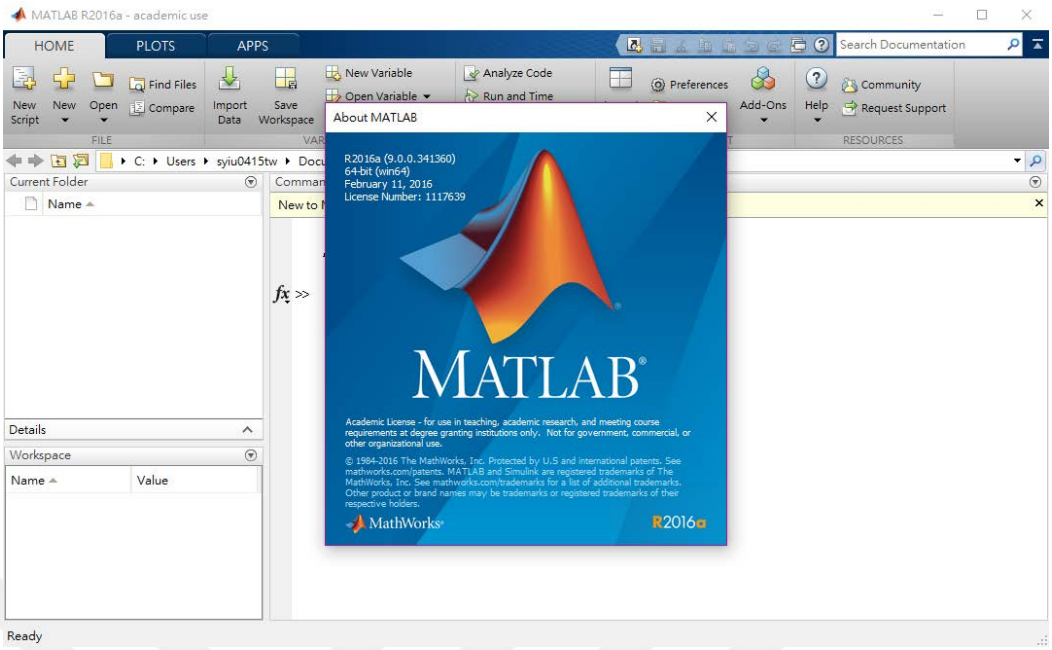
Bu bölüm, bu çalışmada kullanılan materyaller ve önerilen yöntemle ilgili ayrıntıları içermektedir. Tez çalışmalarında kullanılan Yusufçuk Algoritması, Tek ve Çok Hedefli Problemler İçin Yusufçuk Algoritması, Yusufçuk Algoritması'nın Brownian Hareketi ile modifikasyonu, bu bölümde sunulmuştur.

3.1. Materyaller

Çalışmada kullanılan materyallere bu bölümde yer verilmiştir.

3.1.1. Matlab

MATLAB, genellikle pozitif bilim ve mühendislik hesaplamaları için kullanılan bir bilgisayar programıdır. Amerika Birleşik Devletleri merkezli MathWorks firması tarafından geliştirilen MATLAB, aynı zamanda bir programlama dilidir. İngilizce "Matrix Laboratory" kelimelerinin birleştirilmesi ile oluşmuş olan MATLAB, isminden de anlaşılacağı gibi matris tabanlı bir çalışma sistemine sahiptir. Lineer cebir, istatistik, optimizasyon, nümerik analiz, optimizasyon, fourier analizi gibi pek çok matematiksel hesaplamanın etkili ve hızlı şekilde yapılmasına olanak sağlayan MATLAB programı aynı zamanda 2D ve 3D grafik çizimi için de kullanılır. MATLAB ile kullanıcılar kendi programlarını hazırlayabilirler. Matrisler ve onların etkileşim içinde olduğu fonksiyonlarla programlama yapılmasına izin veren MATLAB ile çok karmaşık matematik hesaplamaları bile birkaç saniye içinde tamamlanır. Temel programlama fonksiyonları ile benzer fonksiyonların kullanılabildiği MATLAB ile etkili ve pratik programlar hazırlanabilir. C, Java gibi programlama dillerindeki dizilerin kullanımı ile aynı mantıkla matrislerin kullanıldığı MATLAB programında bir, iki veya daha fazla boyutta matrisler ile çalışmak mümkündür. MATLAB ile temel matematik fonksiyonlarının iki ve üç boyutlu grafikleri çizilebilir. Polinomlar, parboller, sinüs dalgaları başta olmak üzere her tür iki ve üç boyutlu matematiksel grafik MATLAB ile elde edilebilir. Bu çalışmada da MATLAB programı, algoritmanın hem kıyaslama fonksiyonları üzerinde hem de Kaynaklı Kiriş Tasarımı Problemi'nin gerçeklenmesinde kullanılmıştır. Şekil 3.1'de Matlab programının ara yüzü görülmektedir [45].



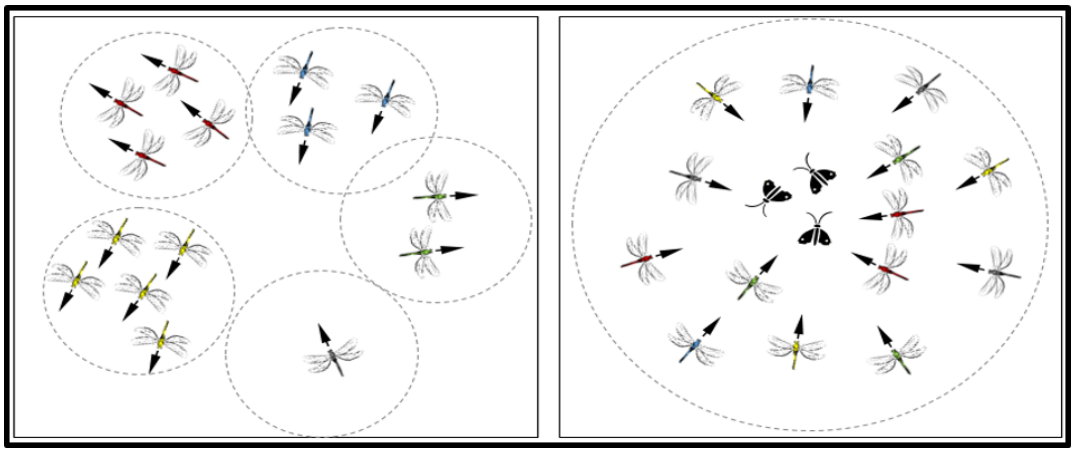
Şekil 3.1. Matlab Arayüzü

3.2. Yöntemler

Bu bölümde çalışmada kullanılan yöntemlere yer verilmiştir.

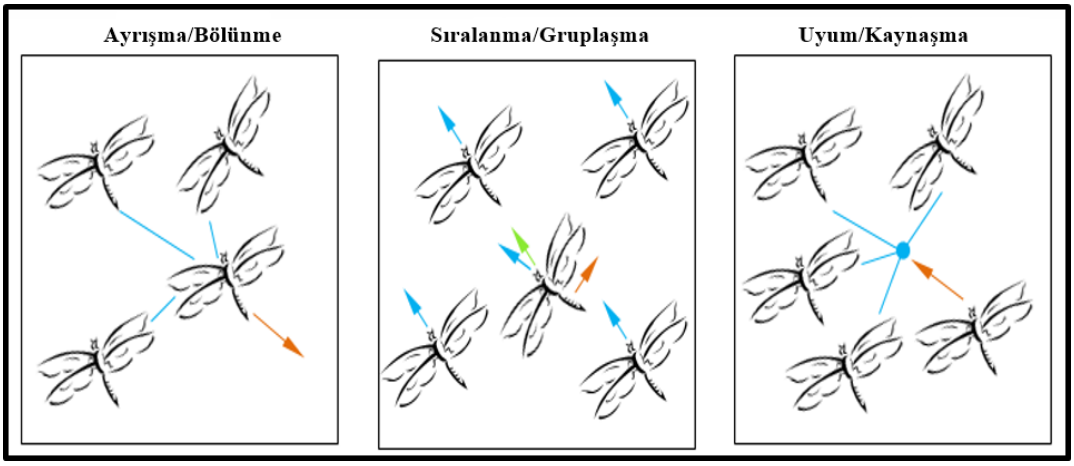
3.2.1. Yusufçuk Algoritması

Yusufçuk Algoritması, 2014 yılında Griffith Üniversitesi'nden Seyedali Mirjalili tarafından geliştirilmiştir [39]. Sürü zekasına dayalı meta sezgisel bir algoritma olan bu teknik, yusufçukların doğada sürü halindeki statik ve dinamik davranışlarından esinlenilmiştir. Optimizasyonun iki temel aşaması vardır: Keşif ve Sömürü. Bu iki aşama yusufçukların sürü halindeyken dinamik ya da statik olarak besin araması veya düşmandan kaçınması izlenerek modellenmiştir. Dinamik ve statik sürülerin kavramsal bir modeli Şekil 3.2.'de verilmiştir.



Şekil 3.2. Dinamik ve statik sürülerin kavramsal bir modeli

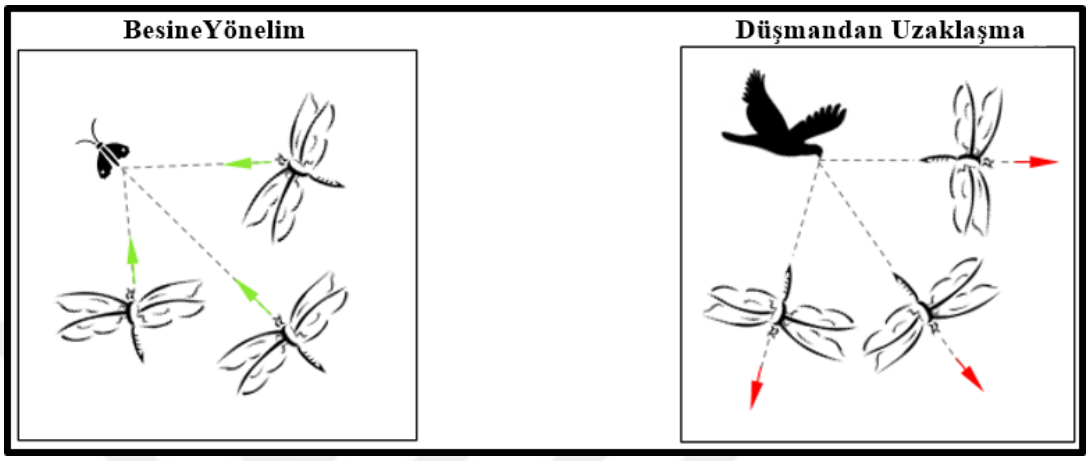
Sürü zekasının ise yusufçuklarda ortaya çıktığı 2 durum vardır: Avlanma ve Göç. Avlanma, optimizasyonda statik sürü olarak modellenirken; göç, dinamik sürü olarak modellenmiştir. Reynolds'a göre sürülerin 3 spesifik davranışı vardır: Ayrışma/Bölünme, Sıralanma/Gruplaşma ve Uyum/Kaynaşma [46]. Burada ayrışma/bölünme kavramı sürüdeki bir bireyin komşusu ile statik çarpışmasından kaçınması anlamına gelmektedir (Denklem 3.1). Sıralanma/gruplaşma ise sürüdeki bireylerin komşu bireylerle eşleşme hızını belirtir (Denklem 3.2). Son olarak uyum/kaynaşma kavramı ise bireylerin sürünün merkezine doğru olan eğilimini gösterir (Denklem 3.3). Bu 3 kavram Şekil 3.3.'de gösterilmiştir.



Şekil 3.3. Sürülerin 3 spesifik davranışı

Yusufçuk algoritmasında bu 3 temel davranışa, 2 ek davranış şekli eklenmiştir: Besine yönelim ve düşmandan uzaklaşma. Bu davranışların algoritmaya eklenme

sebebi ise yine sürü zekasından kaynaklanmaktadır. Yani, her sürünün temel amacı hayatta kalmaktır. Bu yüzden tüm bireyler besin kaynaklarına doğru giderken (Denklem 3.4), aynı zaman diliminde düşmandan kaçınmalıdırlar (Denklem 3.5). Bu davranış modeli Şekil 3.4.'te gösterilmektedir.



Şekil 3.4. Sürülerin hayatta kalmak için 2 çok önemli davranışı

Bu davranışların her biri aşağıda açıklanan şekilde matematiksel olarak modellenmiştir:

$$S_i = -\sum_{j=1}^{N} X - X_j \quad (3.1)$$

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \quad (3.2)$$

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \quad (3.3)$$

$$F_i = X^+ - X \quad (3.4)$$

$$E_i = X^- + X \quad (3.5)$$

Yukarıdaki denklemlerde X, bireyin anlık pozisyonu temsil ederken, X_j, j. bireyin anlık pozisyonunu temsil etmektedir. N, komşu birey sayısını temsil ederken, V_j ise j. komşu bireyin hızını temsil etmektedir. X^+ ve X^- sırasıyla besin ve düşman kaynaklarının konumlarını temsil etmektedir.

Arama uzayındaki yapay yusufçukların konumunu güncellemek ve hareketlerini simule etmek için iki vektör düşünülmüştür: Adım (ΔX) ve Pozisyon (X). Hız olarak da düşünülebilen adım vektörü yusufçukların hareket yönünü gösterir (Denklem 3.6). Adım vektörü hesaplandıktan sonra ise pozisyon vektörü güncellenir (Denklem 3.7).

$$\nabla X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\nabla X_t$$
(3.6)

$$X_{t+1} = X_t + \nabla X_{t+1}$$
(3.7)

Denklem 3.6'daki s, a, c değerleri sırasıyla ayrışma, sıralanma, uyum katsayılarını temsil ederken, f, e, w, t değerleri yine sırasıyla besin faktörü, düşman faktörü, eylemsizlik katsayısı ve iterasyon sayısını temsil etmektedir. Bu katsayı ve faktörler, optimizasyon sırasında keşifçi ve sömürücü davranışları gerçekleyebilmeyi sağlamaktadır. Dinamik sürüde yusufçuklar uçuşlarını hizalama eğilimindedirler. Statik sürü hareketinde ise hizalama çok düşükken, düşmana saldırı için uyum çok yüksektir. Bu yüzden keşif sürecinde sıralanma katsayısı yüksek, uyum katsayısı düşük verilirken; sömürü sürecinde sıralanma katsayısı düşük, uyum katsayısı yüksek verilir.

3.2.1.1. Tek Hedefli Problemler için Yusufçuk Algoritması

Tek hedefli optimizasyon, matematiksel olarak formüle edilmiş bir problemde, optimal değeri bulmak için sistemin hangi parametrelerle çalışacağının bulunması olarak açıklanabilir. Örneğin bir tankın 4 farklı maddeden yapıldığını varsayalım. Bu 4 farklı maddenin de 4 farklı fiyatı olduğundan, bu tankı en ucuz şekilde üretme problemi sadece maliyete odaklandığından tek amaçlı bir optimizasyon problemidir [47].

Birçok gerçek-dünya karar verme probleminin çeşitli hedeflere ulaşması gerekir: riskleri en aza indirme, güvenilirliği en üst düzeye çıkarma, sapmaları en aza indirme, maliyeti en aza indirme gibi. Tek hedefli optimizasyonun ana amacı, tüm farklı hedefleri bir araya getiren tek bir objektif fonksiyonun minimum veya maksimum değerine karşılık gelen "en iyi" çözümü bulmaktır. Bu tür bir optimizasyon, karar vericilere problemin doğası hakkında bilgi vermesi gereken bir araç olarak yararlıdır, ancak genellikle birbirlerine karşı farklı hedefler sunan bir dizi alternatif çözüm sunamaz [48].

Yusufçuk Algoritması, belirli optimizasyon problemleri için bir dizi rastgele çözüm oluşturarak optimizasyon işlemini başlatır. Aslında, yusufçukların pozisyonu ve adım vektörleri, değişkenlerin alt ve üst sınırları içinde tanımlanan rastgele değerler tarafından başlatılır. Her iterasyonda, her bir yusufçuğun pozisyonu ve adım vektörü, Denklem 3.6 ve Denklem 3.7 kullanılarak güncellenir. X ve ∇X vektörlerini güncellemek

için, her bir yusufçuğun komşusu, tüm yusufçuklar arasındaki öklid mesafesini hesaplayarak ve bunların N tanesi seçilerek belirlenir. Pozisyon güncelleme süreci, son kriter yerine getirilene kadar iteratif olarak sürdürülür [39].

3.2.1.2. Çok Hedefli Problemler için Yusufçuk Algoritması

Çok hedefli optimizasyon problemleri ise adından da anlaşılacağı üzere birden fazla hedef için çözüm üretilmesi beklenen problemlerdir. Tek hedefli optimizasyonda verdiğimiz tank hedefinden devam edecek olursak, ilgili 4 farklı maddenin farklı dayanıklılık oranları olduğunu varsayalım. Hem ucuz hem de dayanıklı bir tank yapımı hedeflendiğinde, daha dayanıklı malzemenin fiyatı daha yüksek olacağından, daha dayanıklı bir tank istedikçe fiyat daha fazla artacaktır. Hem dayanıklı hem de ucuz bir tank yapılmak istenirse, çok amaçlı bir optimizasyon problemi tanımlanmış olur ve burada tek bir çözüm olmaz. Çok hedefli optimizasyon algoritmaları bu çözümleri bulmayı amaçlamaktadır. Çakışan hedeflere sahip çok hedefli bir optimizasyonda, tek bir optimal çözüm yoktur. Farklı hedefler arasındaki etkileşim, büyük oranda trade-off, nondom, noninferior ya da Pareto-optimal çözümler olarak bilinen bir dizi çözümün ortaya çıkmasına neden olmaktadır [48] [49].

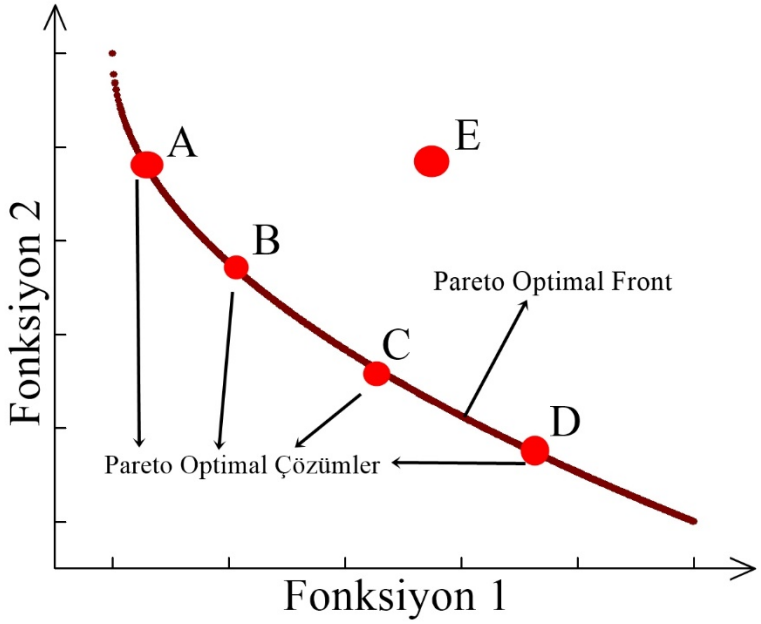
Pareto verimliliğini anlamak çok hedefli optimizasyon problemlerini çözmek için önemlidir. Pareto verimliliğinde bilinmesi gereken 3 önemli kavram vardır: Pareto baskınlığı, Pareto optimal front ve Pareto optimal çözümler kümesi. Tablo 2.1'de bir optimizasyon problemine ait parametreler ve minimize edilmesi gereken 2 fonksiyonun değerleri görülmektedir.

Tablo 3.1. Çok hedefli bir probleme ait rasgele çözümler kümesi

Çözümler	Parametre 1	Parametre 2	F1 (minimize)	F2 (minimize)
A	32	23	0.44	2.04
B	43	653	0.50	1.10
C	234	54	1.43	0.19
D	34	65	3.06	0.04
E	324	34	2.42	1.31

Yukarıdaki tabloya göre ilk olarak A ve D çözümlerini ele alalım. F1 fonksiyonu için A çözümü daha iyiyken, F2 fonksiyonu için D çözümü daha iyidir. Buna göre A ve D

çözümleri birbirlerine baskınlık kuramamışlardır. İkinci olarak B ve C çözümlerini ele alalım. F1 fonksiyonu için B çözümü daha iyiyken, F2 fonksiyonu için C çözümü daha iyidir. Buna göre de B ve C çözümleri birbirlerine baskınlık kuramamışlardır. Son olarak C ve E fonksiyonlarını ele alaım. Hem F1 hem de F2 fonksiyonları için C çözümü E çözümünden daha iyi sonuçlar üretmiştir. Bu noktada C çözümü E çözümünü domine etmiştir. Buna Pareto baskınlığı denir. Şekil 3.5'te görüleceği üzere; A, B, C ve D çözümleri birbirlerine baskınlık kuramamışlardır. Bu çözümler pareto optimal çözümler kümesi olarak adlandırılırken, çözümlerin oluşturduğu çizgi Pareto optimal front'u temsil eder.



Şekil 3.5. Pareto verimliliği için temsili grafik

Yusufçuk Algoritması'nı kullanarak çok hedefli problemleri çözmek için, ilk önce optimizasyon sırasında Pareto optimal çözümlerinin en iyi yaklaşımlarını saklamak ve almak için bir arşiv tasarlanmıştır. Yusufçukların güncellenme pozisyonu, tek hedefli optimizasyon ile aynıdır ancak gıda kaynakları arşivden seçilir. İyi yayılmış bir Pareto optimal front bulmak için elde edilen Pareto optimal front'un en az populasyona sahip bölgesinden bir besin kaynağı seçilmiştir. Pareto optimal front'un en az populasyonlu alanını bulmak için, arama alanı bölümlere ayrılmalıdır. Bu, elde edilen Pareto optimal çözümlerinin en iyi ve en kötü hedeflerini bularak, tüm çözümleri kapsayacak şekilde bir hiper-küre tanımlayarak ve hiper-küreleri her bir iterasyonda

eşit alt-hiper-kürelere bölerek yapılır. Segmentlerin oluşturulmasından sonra, seçim, Coello Coello tarafından önerilen her bir segment için bir rulet çarkı mekanizması ile yapılır [50].

Bu mekanizma, Yusufçuk Algoritması'nın daha az populasyonlu bölgelerden gıda kaynaklarını seçme olasılığını artırır. Bu nedenle, yapay yusufçuklar bu bölgelerde dolaşmaya ve tüm Pareto optimal front'unun yayılımını geliştirmeye teşvik edilecektir. Ancak, arşivden düşmanları seçmek için, yapay yusufçukların alakasız kalabalık alanlarda arama yapmasını engellemek amacıyla en kötü hiper-kürenin seçilmesi gerekir. Seçim her bölge için bir rulet mekanizmasıyla yapılır. Rulet çarkı mekanizması, düşman olarak seçilmek için en kalabalık hiper kürelere yüksek olasılıklar atar [39].

Arşiv, her yinelemede düzenli olarak güncellenmelidir çünkü optimizasyon sırasında dolu olabilir. Bu nedenle, arşivi yönetmek için bir mekanizma bulunmalıdır. Eğer bir çözüm, arşivdeki Pareto optimal çözümlerinden bazılarına hükmederse, bunların tümü arşivden kaldırılmalıdır. Eğer bir çözüm arşivdeki tüm çözümlerle birbirine baskın değilse, arşive eklenmelidir. Arşiv doluysa, arşivdeki yeni çözüm yada çözümlere yer açmak için en kalabalık bölümden bir veya birden fazla çözüm kaldırılabilir. Bu kurallar Coello Coello'nun çalışmalarından alınmıştır [50]. Çok hedefli optimizasyonda tüm parametreler, hiper-küreler ve arşiv boyutunun maksimum sayısını tanımlamak için iki yeni parametre dışında tek hedefli optimizasyondaki Yusufçuk Algoritması ile aynıdır.

3.2.2. Yusufçuk Algoritmasının Brownian Hareket ile İyileştirilmesi

Meta sezgisel bir sürü zekasına dayalı optimizasyon algoritması olan Yusufçuk Algoritması, problemi optimize ederken komşuluk çözümünün kalmaması durumunda tıkanmaya önlemek adına Levy Uçuş Mekanizması'nı kullanmaktadır. Bu tez çalışmasında bu mekanizma yerine, Brownian Hareketi kullanılmıştır.

Brownian Hareketi'nin algoritmaya modifikasyonunun matematiksel modellemesi aşağıdaki ifade edilmiştir:

$$X_{t+1} = X_t + h * rand() * \mathrm{Pg} \quad (3.8)$$

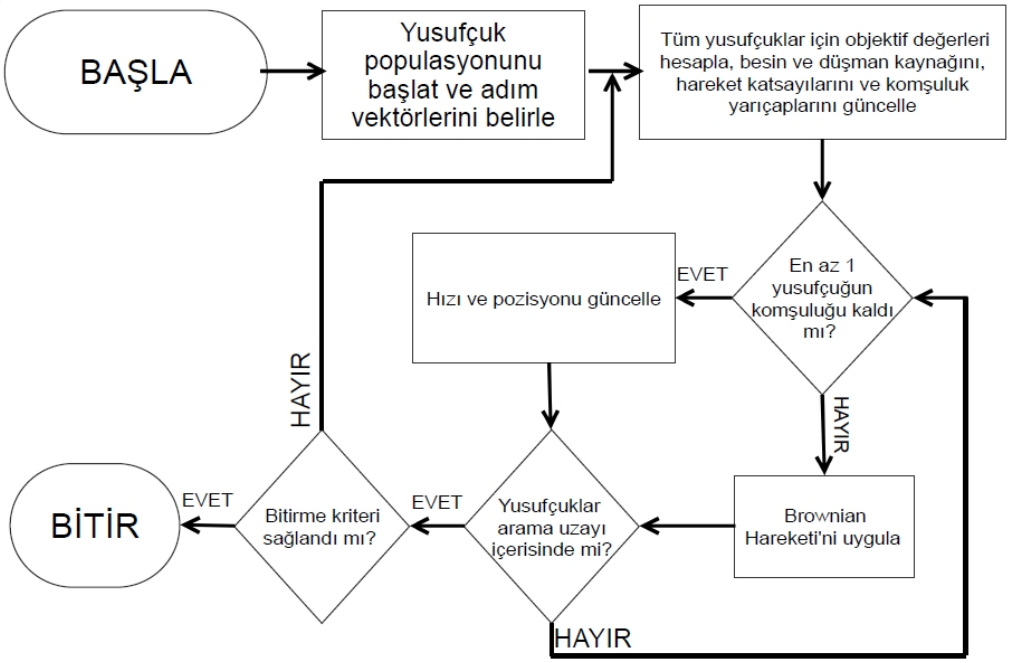
$$h = \sqrt{T/N} \quad (3.9)$$

$$N = 100 * T \tag{3.10}$$

$$Pg = \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{(dim-agents)^2}{2h^2}\right) \tag{3.11}$$

Denklem 3.9'da T terimi, 1 ajanın (yusufçuğun) hareket zaman periyodunu saniye cinsinden temsil etmektedir. Bu çalışmada T değeri 0.01 olarak alınmıştır. Denklem 3.10'daki N terimi ise aynı ajan için ani hareket sayısını (yoldaki değişimi) zamanla orantılı olarak ifade etmektedir.

Kullanılan yeni yöntem, hem tek hedefli problemler için hem de çok hedefli problemler için tasarlanan Yusufçuk Algoritması'na adapte edilmiştir. Bundan sonraki bölümlerde Yusufçuk Algoritması, modifiye edilen haliyle kullanılmıştır. Şekil 3.6, tek hedefli problemler için Yusufçuk Algoritması'nın akış şemasını göstermektedir:

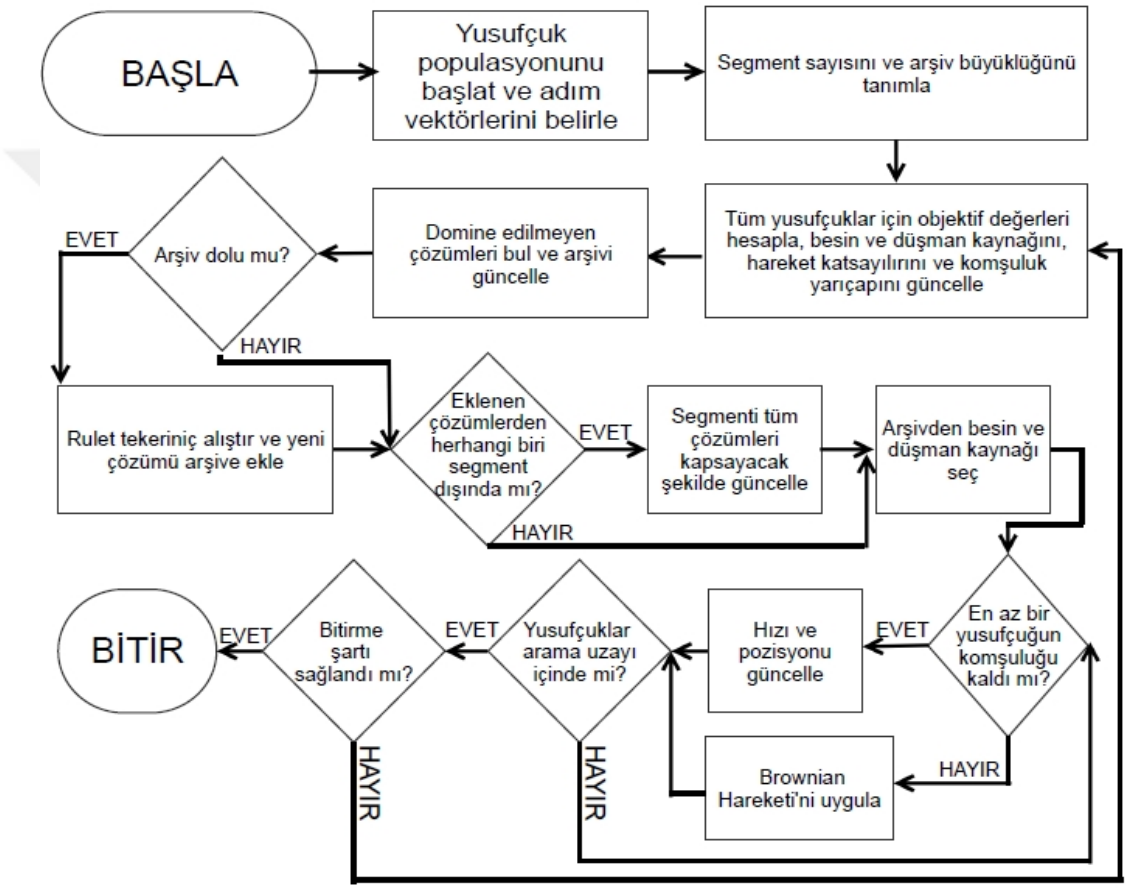


Şekil 3.6. Tek hedefli problemler için Yusufçuk Algoritmasına ait akış şeması

Optimizasyon yusufçukların arama uzayına rasgele olarak yerleştirilmesiyle ve adım vektörlerinin belirlenmesiyle başlıyor. Ardından o andaki konumları kıyaslama fonksiyonuna parametre olarak gönderiliyor ve en iyi-en kötü çözüm belirleniyor. Bu çözümlerin ardından, her yusufçuk için komşuluk sayısına bakılıyor. Eğer her yusufçuğunu en az bir komşusu varsa hız vektörü, algoritmanın başında belirlenen katsayılarla hesaplanıyor ve pozisyon vektörü güncelleniyor. Eğer herhangi bir yusufçuğun hiç komşuluğu kalmadıysa, Brownian Hareketi çözümüne gidiliyor ve

pozisyon vektörü buna göre güncelleniyor. Ardından yusufçukların arama uzayı içerisinde olup olmadığı kontrolü yapılıyor. Kontrol olumlu ise bitirme kriterinin sağlanıp sağlanmadığı kontrol ediliyor. Kontrol olumsuzsa tekrar komşuluk çözümüne gidiliyor. Ardından tekrar, en son bitirme kriterinin sağlanıp sağlanmadığı kontrol ediliyor ve optimizasyon sonlandırılıyor.

Çok hedefli optimizasyon için Yusufçuk Algoritması'nın akış şeması ise Şekil 3.7'de gösterildiği gibidir:



Şekil 3.7. Çok hedefli problemler için Yusufçuk Algoritmasına ait akış şeması

Çok hedefli optimizasyon ise, yine, yusufçukların arama uzayına rasgele olarak yerleştirilmesiyle ve adım vektörlerinin belirlenmesiyle başlıyor. Ardından maksimum arşiv büyüklüğü ve segment (hiper küre) sayısı tanımlanıyor. Yusufçukların anlık konumları kıyaslama fonksiyonuna parametre olarak gönderiliyor ve birbirini domine edemeyen çözümler bulunuyor. Eğer arşiv dolu ise rulet tekeri mekanizması ile bazı çözümler eleniyor ve yeni bulunan çözümşer arşive ekleniyor. Eğer eklenen çözümlerden herhangi biri hiper küre dışında kaldıysa, hiper küre tüm çözümleri

kapsayacak şekilde güncelleniyor. Bu işlemlerin ardında arşivden en iyi ve en kötü çözüm, sırasıyla besin ve düşman kaynağı olarak atanıyor. Ardından komşuluk kontrolü yapılıyor. Buradan itibaren algoritma, tek hedefli problem optimizasyonu ile aynı şekilde çalışıyor ve optimizasyon bitirme kriterinin ardından bitiriliyor.

4. BULGULAR ve TARTIŞMA

Bu bölümde modifiye edilmiş algoritmanın deneysel değerlendirmeleri sunulmuştur. Bu tez çalışmasında gerçekleştirilen tek hedefli problem optimizasyonu, çok hedefli problem optimizasyonu ve Kaynaklı Kiriş Tasarım Problemi çözümü için MATLAB programı kullanılmıştır. Bu tez çalışmasında tek hedefli problem optimizasyonu için 15, çok hedefli problem optimizasyonu için 6 kıyaslama fonksiyonu kullanılmıştır. Bulgular algoritmanın orijinal hali ve modifiye edilmiş hali ile karşılaştırılarak sunulmuş ve tartışılmıştır.

4.1. Optimizasyon İçin Kıyaslama Fonksiyonları

Test fonksiyonları, uygulamalı matematikte optimizasyon algoritmalarının şu özelliklerini değerlendirmek için yararlıdır: Yakınsama oranı, hassaslık, sağlamlık, genel performans. Burada bazı test fonksiyonları, bu tür problemlerle başa çıkarken optimizasyon algoritmalarının karşılaşacağı farklı durumlar hakkında bir fikir vermek amacıyla sunulmuştur. Sadece denklemin genel bir formu, objektif fonksiyonun bir çizimi, nesne değişkenlerinin sınırları ve global minimum koordinatları burada verilmiştir.

4.1.1. Tek Hedefli Problemler İçin Kıyaslama Fonksiyonları

Tablo 4.1. Tek Hedefli Problem Optimizasyonu için Kıyaslama Fonksiyonları

Fonksiyon	Boyut	Aralık
F1(x) = $\sum_{i=1}^{n} x_i^2$	10	[-100,100]
F2(x) = $\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[-10,10]
F3(x) = $\sum_{i=1}^{n} \left(\sum_{J-1}^{i} X_J\right)^2$	10	[-100,100]

F4(x) = $\max_i\{\|x_i\|, 1\le i\le n\}$	10	[-100,100]
F5(x) = $\sum_{i=1}^{n-1}\left[\mathbf{100}(x_{i+1}-x_i^2)^2+(x_i-\mathbf{1})^2\right]$	10	[-30,30]
F6(x) = $\sum_{i=1}^{n}([x_i+\mathbf{0.5}])^2$	10	[-100,100]
F7(x) = $\sum_{i=1}^{n} ix_i^4+random[\mathbf{0},\mathbf{1})$	10	[-1.28,1.28]
F8(x) = $\sum_{i=1}^{n}-x_i sin(\sqrt{\|x_i\|}$	10	[-500,500]
F9(x) = $\sum_{i=0}^{n}\left[x_i^2-\mathbf{10}cos(\mathbf{2}\pi x_i)+\mathbf{10}\right]$	10	[-5.12,5.12]
F10(x) = -20exp (-0.2$\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}$)-exp ($\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)$)+20+e	10	[-32,32]
F11(x) = $\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}+\mathbf{1}\right)$	10	[-600,600]
F12(x) = $\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	10	[-50,50]
F13(x) = $0.1\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	10	[-50,50]
F14: Özelleştirilmiş Küre Fonksiyonu	10	[-5,5]
F15: Özelleştirilmiş Griewank Fonksiyonu	10	[-5,5]

4.1.2. Çok Hedefli Problemler İçin Kıyaslama Fonksiyonları

Tablo 4.2. Çok Hedefli Problem Optimizasyonu için Kıyaslama Fonksiyonları

Problem	Tanım
Zitzler–Deb–Thiele 1	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1+\frac{9}{N-1}\sum_{i=2}^{N}x_i$ $h(f_1(x),g(x)) = 1-\sqrt{\frac{f_1(x)}{g(x)}}$ $0\le x_i\le 1, 1\le i\le 30$
Zitzler–Deb–Thiele 2	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1+\frac{9}{N-1}\sum_{i=2}^{N}x_i$ $h(f_1(x),g(x)) = 1-\left(\frac{f_1(x)}{g(x)}\right)^2$ $0\le x_i\le 1, 1\le i\le 30$
Zitzler–Deb–Thiele 1 (Lineer Pareto Front ile)	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1+\frac{9}{N-1}\sum_{i=2}^{N}x_i$ $h(f_1(x),g(x)) = 1-\frac{f_1(x)}{g(x)}$ $0\le x_i\le 1, 1\le i\le 30$

Zitzler–Deb–Thiele 3	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + \frac{9}{29}\sum_{i=2}^{N} x_i$ $h\left(f_1(x), g(x)\right) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right)\sin(10\pi f_1(x))$ $0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 4	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + 10(N-1) + \sum_{i=2}^{N}\left(x_i^2 - 10\sin(4\pi x_i)\right)$ $h\left(f_1(x), g(x)\right) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}\ 0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 6	Minimize: $f_1(x) = 1 - \exp(-4 * x_1) * \sin(6\pi x_1)^6$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + 9\frac{(\sum_{i=2}^{N} x_i)}{(N-1)^{0.25}}$ $h\left(f_1(x), g(x)\right) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2\ 0 \le x_i \le 1, 1 \le i \le 30$

4.2. Kaynaklı Kiriş Tasarımı Problemi

Mühendislik optimizasyonunda kullanılan fonksiyonlar, birçok değişken ve kısıtlamayla birlikte çok karmaşıktır. Konvansiyonel optimizasyon araçları bazen global optima noktasını bulamaz. Mühendislik problemleri; Genetik Algoritma, Isıl İşlem, Karınca Kolonisi gibi çok popüler algoritmalar ile global optimayı bulmak için çözülmüşlerdir. Modifiye edilmiş olan algoritma, bu mühendislik problemlerinden biri olan Kaynaklı Kiriş Tasarımı Problemi ile algoritmanın etkinliğini göstermek amacıyla gerçeklenmiştir.

Bu problem, mühendislik optimizasyon tekniklerinin başarısını kanıtlamak amacıyla geçmişte bir çok kez kullanılmıştır. Bunlardan biri 2009 yılında Kaveh ve Talatahari'nin Parçacık Sürü Optimizasyonu ve Karınca Kolonisi'ni hibritlediği çalışmadır [51]. Diğer bir çalışma 2010 yılında yine Kaveh ve arkadaşlarına ait olan iyileştirilmiş Karınca Kolonisi'nin sınırlı mühendislik problemleri için uygulaması çalışmasıdır [21]. Bu problem daha bir çok kez, Brajevic ve Tuba'ya ait geliştirilmiş Yapay Arı Kolonisi'nin sınırlı optimizasyon problemleri üzerindeki uygulaması [52], Liao ve arkadaşlarının 2014'teki Karınca Kolonisi'nin karışık değişkenli optimizasyon problemleri için uygulamasında [53] ve son olarak Ranjini ve Murugan'ın 2017 yılındaki Yusufçuk Algoritması'nın hafıza tabanlı modifiyesi ile uygulamasında [31] kullanılmıştır.

Kaynak, metalik parçaların, basınç uygulanarak veya uygulanmadan uygun bir sıcaklığa kadar ısıtılarak birleştirilmesi işlemi olarak tanımlanabilir. Kaynak, kalıcı metal parçaların elde edilmesini sağlayan ekonomik ve verimli bir yöntemdir.

Kaynak işlemi, aşağıdaki iki grup ile geniş bir şekilde sınıflandırılmıştır:

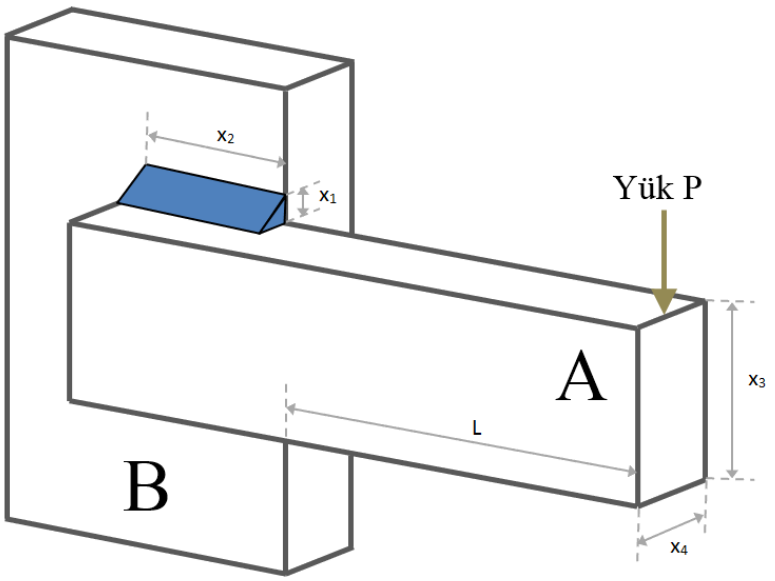
• İki parçayı birleştirmek için ısıyı tek başına kullanan kaynak işlemi

• İki parçayı birleştirmek için ısı ve basınç kombinasyonunu kullanan kaynak işlemi [54].

Yalnız ısı kullanan kaynak işlemine füzyon kaynak işlemi denir. Bu yöntemde birleştirilecek parçalar yerinde tutulur ve eriyik metale eklenir. İki parçanın birleşme yüzeyi, ısı etkisi altında plastik hale gelir veya erimez hale gelir. Eklem katılaştığı zaman, iki kısım tek bir birime geçer.

Bir kiriş, uzun boyutta enine uygulanan yüklere maruz kalan ve üyenin bükülmesine neden olan bir üyedir. Kirişler sıklıkla destek veya reaksiyonlar temelinde sınıflandırılır [55].

Kaynaklı kiriş tasarım problemi, kirişin dört yapısal parametrelerinin uygun bir setini bularak kaynaklı kirişin imalat maliyetini en aza indirmeyi amaçlamaktadır. Bu dört yapısal parametre ise kaynağın kalınlığı (x_1), kenetlenmiş çubuğun uzunluğu (x_2), çubuğun yüksekliği (x_3) ve çubuğun kalınlığıdır (x_4). İlgili kısıtlamalar ise makaslama gerilmesi (τ), kirişte eğilme gerilmesi (θ), burkulma yükü (P_c) ve kirişin son sapmasıdır (δ). Şekil 4.1'te problemin sistematik dizaynını görebilirsiniz:



Şekil 4.1. Kaynaklı Kiriş Tasarımı

Bu optimizasyon problemi 2000 yılında Coello [56] tarafından Genetik Algoritma ile, 2006'da Fukushima ve Hedar [57] tarafından Benzetimli Tavlama ile, 2008 yılında Efren Mezura ve Coello [58] tarafından "Evaluation Strategy" ile, 2009'da Rashedi [59] ve arkadaşları tarafından Yerçekimsel Arama Algoritması ile, 2010'da Kaveh ve Talatahari [21] tarafından Karınca Kolonisi Algoritması gibi birçok algoritma tarafından çözülmüştür.

Toplam maliyet, işgücü maliyetlerine (kaynak boyutlarının bir fonksiyonu) ve kaynak ile kiriş malzemesinin maliyetine eşittir.

Kiriş, kaynak ve eleman boyutlarını (x_1, x_2, x_3 ve x_4) değiştirerek minimum maliyet için optimize edilecektir. x_1, ve x_2 değişkenleri genellikle 0,0625 inç'lik tamsayı katlarıdır ancak bu uygulama için sürekli olarak kabul edilmektedir. Probleme ait parametreler ve değerleri şu şekildedir:

Young'ın modülü: (psi)

$$E = 30 \times 10^6 psi \tag{4.1}$$

Kiriş malzemesi için kesme modülü (psi)

$$G = 12 \times 10^6 psi \tag{4.2}$$

Üyenin çıkıntı uzunluğu (inch)

$$L = 14\ in \tag{4.3}$$

Kaynak tasarım stresi (psi)

$$\tau_{max} = 13600\ psi \tag{4.4}$$

Kiriş malzemesi için normal tasarım stresi (psi)

$$\sigma_{max} = 30000\ psi \tag{4.5}$$

Maksimum sapma (inch)

$$\delta_{max} = 0.25\ in \tag{4.6}$$

Yük (lb)

$$P = 6000\ lb \tag{4.7}$$

Problemin maliyet fonksiyonu ise şu şekildedir:

$$f(x) = C_0 + C_1 + C_2 \tag{4.8}$$

Denklem 4.8.'de C_0, başlangıç maliyetini temsil eder fakat kaynak sırasında çubuğun kurulumu ve tutulması için olan bağlantıların mevcut olduğu varsayılmaktadır. Bu nedenle, toplam maliyet modelinde C_0 maliyeti göz ardı edilebilir. C_1, kaynak maliyetini $\boldsymbol{C_2}$ ise malzeme maliyetini temsil eder.

4.3. Tek Hedefli Problem Optimizasyonu Sonuçları

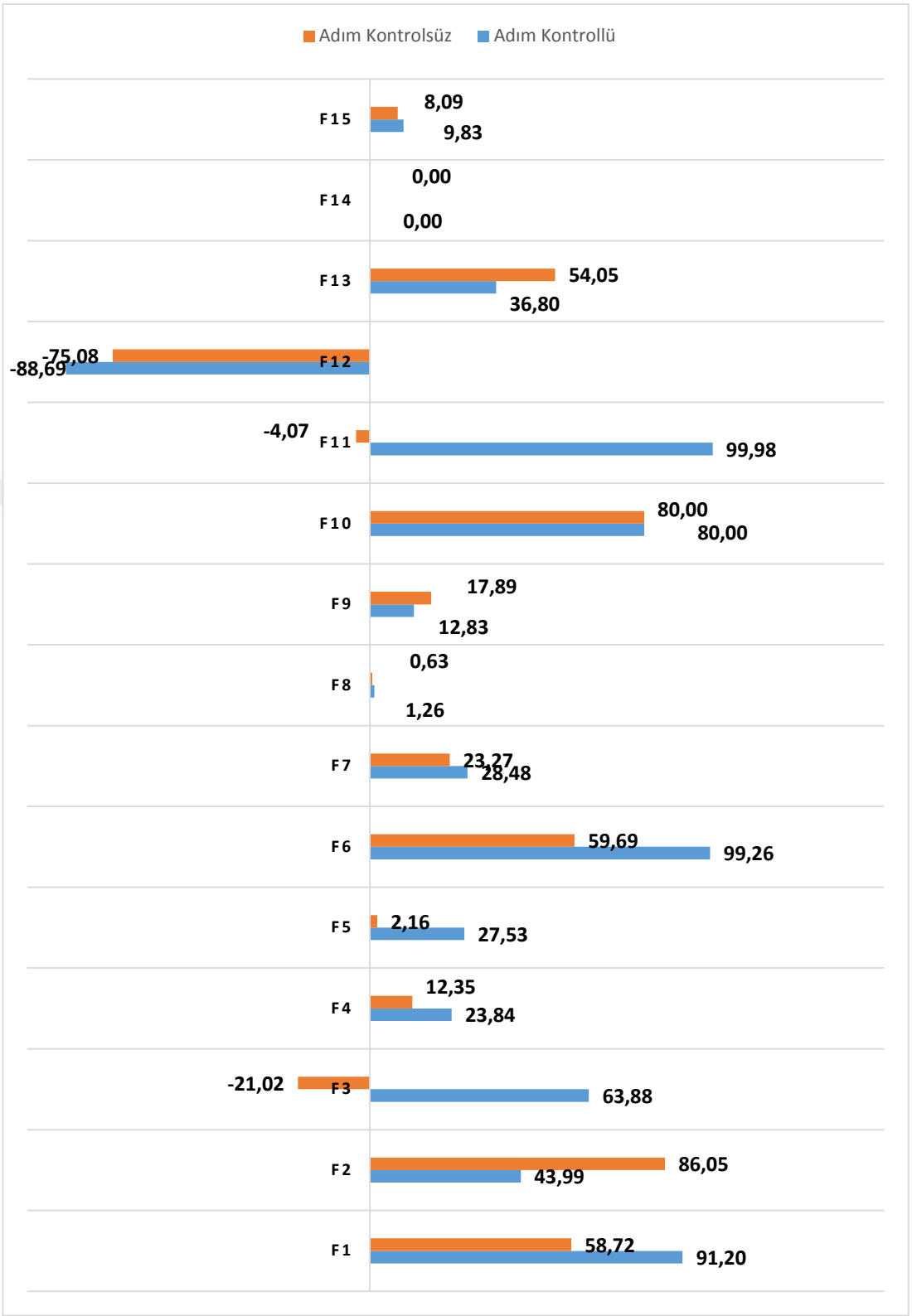
Modifiye edilen Yusufçuk Algoritması, öncelikle tek hedefli problem optimizasyonu için 15 adet kıyaslama fonksiyonu ile gerçeklendi. Bölüm 3.2.1.1'de detaylıca açıklanan tek hedefli problem optimizasyonuna ait akış şeması Şekil 3.6'da görülmektedir. Gerçeklemede kullanılan foksiyonlara ait tüm detaylar ise Bölüm 4.1.1.'de ve Tablo 4.1'de bulunmaktadır. Optimizasyon gerçeklenirken, sonuçlar 2 farklı kategoriye ayrıldı. İlk olarak 15 adet fonksiyonun her biri için bulunan minimum değer hesaplandı. Bu hesaplanan sonuçlar, modifiye edilmemiş algoritmanın sonuçları ile karşılaştırıldı ve yüzdesel olarak sağlanan başarı gösterildi. Modifiye edilmiş algoritmada Brownian Hareketi 2 farklı şekilde kullanıldı: İlk olarak Bölüm 3.2.1'deki orijinal hali, ikinci olarak ise Brownian Hareketi'nde hesaplanan adım boyutunun %1'i alınarak. Sonuçlardaki diğer kategori ise hesaplanan ortalama değer. Ortalama değer, hem modifiye edilen algoritmanın 2 farklı versiyonu (Orijinal Brownian – Adım Kontrollü Brownian) ile hem de modifiye edilmeden önceki hali (Levy Uçuş Mekanizması) ile 200 ayrı optimizasyon sonucunda elde edilen 200 ayrı sonucun ortalaması alınarak karşılaştırıldı. Sonuçlarda yine yüzdesel olarak sağlanan başarı gösterildi.

Gerçeklenen optimizasyona ait istatistiksel sonuçlar Tablo 4.3'de gösterilmektedir:

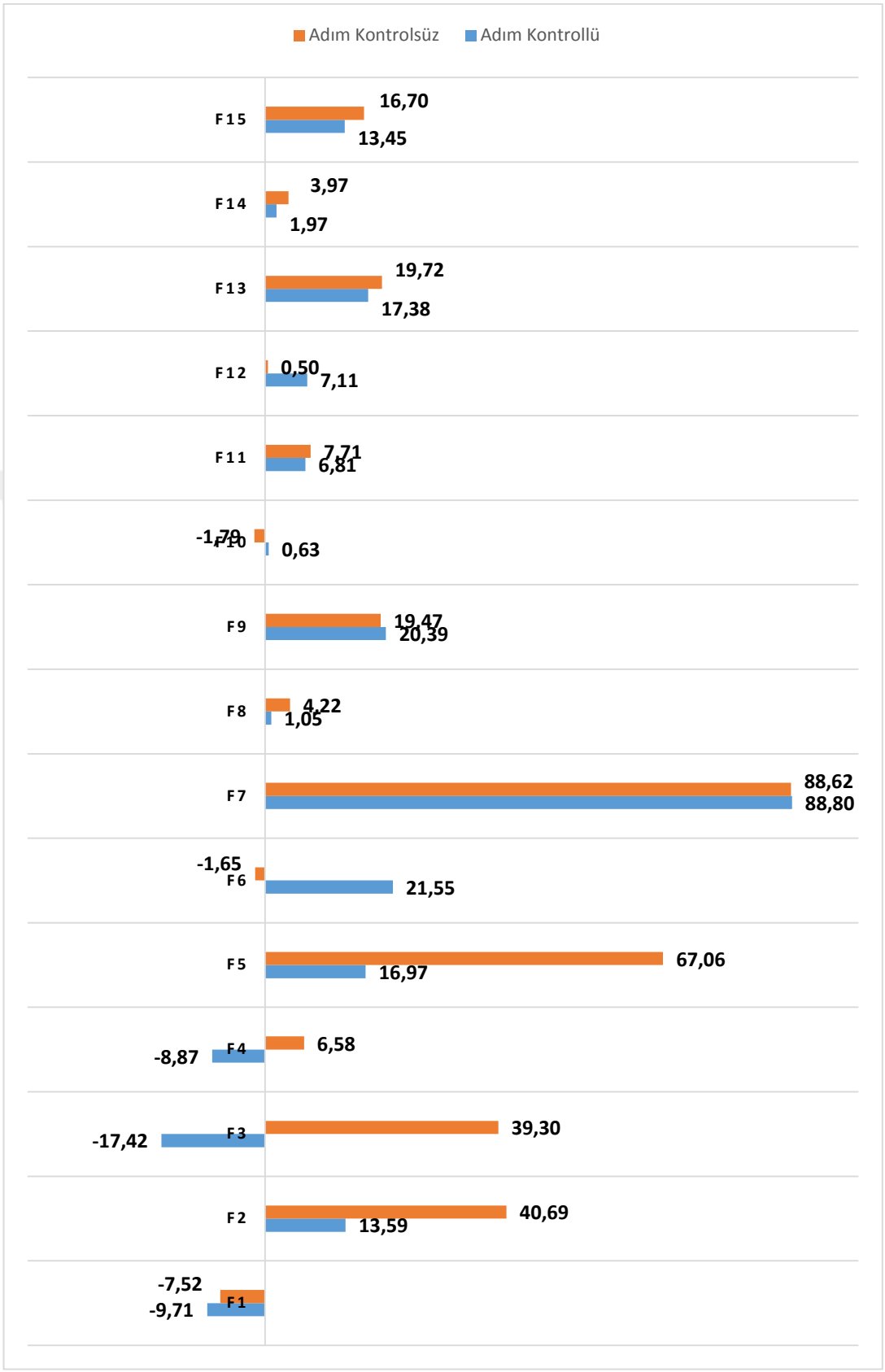
Tablo 4.3. Tek Hedefli Problem Optimizasyondan Elde Edilen Karşılaştırmalı İstatistiksel Değerler

Fonksiyonlar	Orijinal (Levy)	Brownian (Adım Kontrol ile)	Brownian	Adım Kontrollü Başarı Yüzdesi (%)	Normal Başarı Yüzdesi (%)
F1 (min)	5,56E-06	4,89E-07	2,29E-06	91,20	58,72
F1 (Ort)	4,59E+00	5,03E+00	4,93E+00	-9,71	-7,52
F2 (min)	1,29E-02	7,24E-03	1,80E-03	43,99	86,05
F2 (Ort)	1,46E+00	1,26E+00	8,68E-01	13,59	40,69
F3 (min)	8,39E-02	3,03E-02	1,02E-01	63,88	-21,02
F3 (Ort)	1,38E+02	1,63E+02	8,40E+01	-17,42	39,30
F4 (min)	3,45E-02	2,63E-02	3,02E-02	23,84	12,35
F4 (Ort)	1,91E+00	2,08E+00	1,78E+00	-8,87	6,58
F5 (min)	5,56E+00	4,03E+00	5,44E+00	27,53	2,16
F5 (Ort)	1,74E+03	1,45E+03	5,74E+02	16,97	67,06
F6 (min)	4,10E-07	3,04E-09	1,65E-07	99,26	59,69
F6 (Ort)	5,49E+00	4,30E+00	5,58E+00	21,55	-1,65
F7 (min)	1,41E-03	1,01E-03	1,09E-03	28,48	23,27
F7 (Ort)	1,95E-01	2,18E-02	2,22E-02	88,80	88,62
F8 (min)	-3,89E+03	-3,94E+03	-3,92E+03	1,26	0,63
F8 (Ort)	-2,82E+03	-2,84E+03	-2,93E+03	1,05	4,22
F9 (min)	2,99E+00	2,61E+00	2,46E+00	12,83	17,89
F9 (Ort)	3,06E+01	2,44E+01	2,47E+01	20,39	19,47
F10 (min)	4,44E-15	8,88E-16	8,88E-16	80,00	80,00
F10 (Ort)	2,28E+00	2,26E+00	2,32E+00	0,63	-1,79
F11 (min)	3,94E-03	8,59E-07	4,10E-03	99,98	-4,07
F11 (Ort)	4,70E-01	4,38E-01	4,34E-01	6,81	7,71
F12 (min)	1,63E-04	3,07E-04	2,85E-04	-88,69	-75,08
F12 (Ort)	1,29E+00	1,20E+00	1,29E+00	7,11	0,50
F13 (min)	6,70E-05	4,23E-05	3,08E-05	36,80	54,05
F13 (Ort)	8,35E-01	6,90E-01	6,71E-01	17,38	19,72
F14 (min)	9,98E-01	9,98E-01	9,98E-01	0,00	0,00
F14 (Ort)	1,25E+00	1,23E+00	1,20E+00	1,97	3,97
F15 (min)	3,41E-04	3,08E-04	3,13E-04	9,83	8,09
F15 (Ort)	2,45E-03	2,12E-03	2,04E-03	13,45	16,70

Elde edilen sonuçların grafiksel olarak gösterimi Şekil 4.2'de ve Şekil 4.3'de gösterilmektedir:



Şekil 4.2. Minimum Değerler için Modifiye Edilmiş Algoritmanın Başarı Yüzde Grafiği



Şekil 4.3. Ortalama Değerler için Modifiye Edilmiş Algoritmanın Başarı Yüzde Grafiği

Elde edilen sonuçlar istatistiksel olarak incelendiğinde karşımıza iki durum çıkmaktadır. İlk olarak minimum değerler sonuçlarına bakıldığında, Şekil 4.2'de görüldüğü gibi adım kontrollü Brownian Hareketi için kullanılan kıyaslama fonksiyonlarının 15'inin 13'ünde, algoritmanın ilk halinden elde edilen sonuçlarla karşılaştırıldığında başarı elde edilmiştir. Başarı sağlanan bu 13 fonksiyonun 10'unda ise en az %23,84 gelişim sağlanarak önemli bir başarı ortaya konmuştur. Minimum değerler karşılaştırmasında sadece 1 fonksiyonun sonucunda gerileme olurken, 1'inde ise gelişme kaydedilememiştir.

En büyük ilerlemenin sağlandığı 2 fonksiyon (F6 – F11) incelendiğinde ise karşımıza şu sonuç çıkmaktadır: Brownian Hareketi, küresel modellenen problemlerde lokal minimum tehlikesi olmadığından, algoritma kendi davranışını bozmadan sergiyelebildiği için Levy Uçuş Mekanizması'na göre %99'un üzerinde bir başarı sağlamıştır. Gerileme olan fonksiyon (F12) incelendiğinde ise Brownian Hareketi, Levy Uçuş Mekanizması'na göre daha erken yakınsamıştır.

Diğer yandan adım kontrolü yapılmadan gerçeklenen optimizasyonun sonuçları incelendiğinde ise 15 kıyaslama fonksiyonunun 11'de başarı sağlanmıştır. Bu 11 fonksiyonun 7'sında en az %17,89 kesin başarı gözlemlenmiştir. Toplamda 3 fonksiyonda başarısız olunurken, 1 fonksiyonda gelişme kaydedilememiştir. En büyük başarı sağlanan 2 fonksiyon (F2 – F10) gözlemlendiğinde, keskin sınırları olan ve tek bir minimum doğrultusunda çözümü beklenen problemlerde adım kontrolü %80'i aşacak kadar çok etkilidir. Gerileme olan 3 fonksiyon incelendiğinde ise küresel veya özellikle erken yakınsama beklenen problemlerde adım kontolü mantıklı bir çözüm yöntemi olmamaktadır.

Elde edilen sonuçların 2. kategorisinde ortalama değerler incelenmiştir (Şekil 4.3). Bu incelemelere göre adım kontrolü uygulanan Brownian Hareketi ile Levy Uçuş Mekanizması'nın kullanıldığı algoritma karşılaştırıldığında 15 fonksiyon üzerinden 12'sinde başarı kaydedilmiştir. Bu 12 fonksiyonun 11'inde elde edilen başarı yüzdesi maksimum %21,55 olduğundan, Brownian Hareketi'nin adım kontrolü uygulanan şeklinde, amaçlanan minimum nokta anlamında, ortalama değere göre çok daha kesin başarı elde ettiği söylenebilir. Burada en büyük başarı %88,8 ile fonksiyon 7'de elde edilmiştir. Bu fonksiyon incelendiğinde ise lokal minimumların oldukça fazla olduğu ve Brownian Hareketi'nin bu tip fonksiyonlarda Levy Uçuş Mekanizması'na göre daha başarılı olduğu kesin olarak söylenebilir. Gerileme olan 3 fonksiyon (F1 – F3 – F4) incelendiğinde ise küresel fonksiyonlarda minimum noktayı Levy'e göre daha net bulmasına karşın bu netliğin rasgele hareket kaynaklı olduğu anlaşılmaktadır. Çünkü

bu 3 fonksiyon şekilsel olarak benzerdir ve 200 ayrı değerin ortalamasına göre gerileme olmuştur.

Ortalama değerler adım kontrolü olmadan incelendiğinde ise 15 kıyaslama fonksiyonunun yine 12'sinde ilerleme görülmüştür. Elde edilen ilerlemeler ışığında şu sonuca varılmaktadır: Brownian Hareketi, ortalama değerler açısından adım kontrolü uygulanmadığında 12 fonksiyonun 7'sinde en az %16,70 gibi önemli bir başarı kaydetmiştir ve bu başarı Brownian Hareketi'nin doğasına uygun olarak hareket ettiğindeki başarısını kanıtlamaktadır. En büyük başarı adım kontrollü mekanizmada olduğu gibi fonksiyon 7'de elde edilmiştir. Bu durum da yine Brownian Hareketi'nin bir çok lokal minimum bulunan problemlerdeki başarısını kanıtlamaktadır.

Adım kontrolü uygulanmadığında gerileme olan 3 fonksiyona (F1 – F6 – F10) bakılırsa yine ortalama değerler açısından küresel fonksiyonlarda problem yaşandığı anlaşılmaktadır. Küresel fonksiyonlardan elde edilen minimum değerlerde Brownian Hareketi çok başarılıyken, ortalama sonuçlar alındığında istenilen başarı sağlanamamıştır.

4.4. Çok Hedefli Problem Optimizasyonu Sonuçları

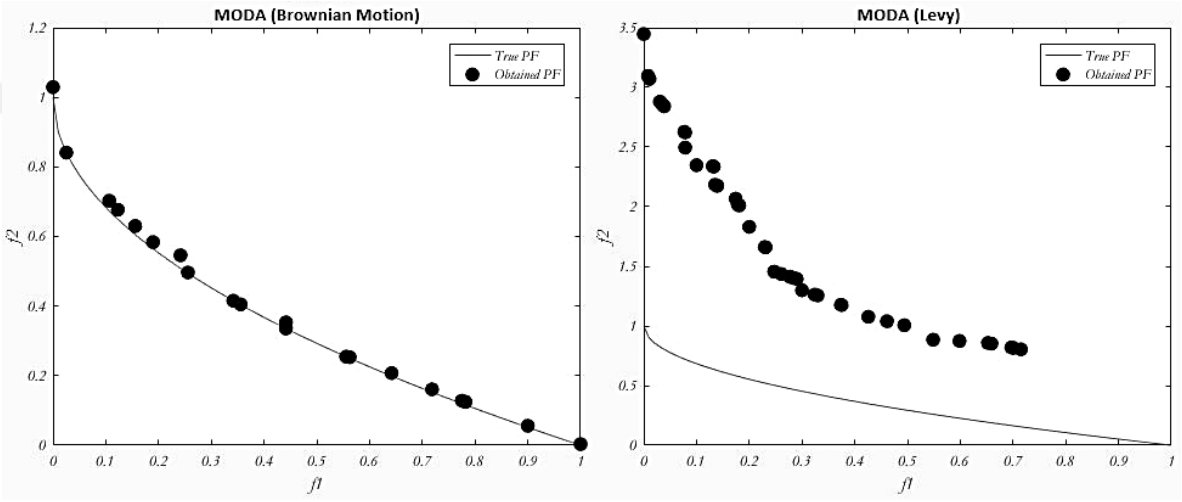
Modifiye edilen Yusufçuk Algoritması, ikinci olarak Bölüm 3.2.1.2'de detaylıca açıklanan çok hedefli problem optimizasyonu ile gerçeklenmiştir. Optimizasyona ait akış şeması Şekil 3.7'de görülmektedir. Gerçeklemede kullanılan foksiyonlara ait tüm detaylar ise Bölüm 4.1.2'de Tablo 4.2'de bulunmaktadır. Kıyaslamada kullanılan toplam 6 adet iyi bilenen algoritmadan 12 adet minimum değer elde edilmiştir. Elde edilen 12 minimum değer Yusufçuk Algoritması'nın orijinal halinin gerçeklenmesiyle elde edilen sonuçlarla karşılaştırılmıştır. Modifiye edilen algoritmada Brownian Hareketi adım kontrolsüz olarak uygulanmıştır ve sonuçlar 100 iterasyonun ardından birbirini domine etmeyen 100 çözüm üzerinden elde edilmiştir.

Gerçeklenen optimizasyona ait istatistiksel sonuçlar Tablo 4.4'de gösterilmektedir:

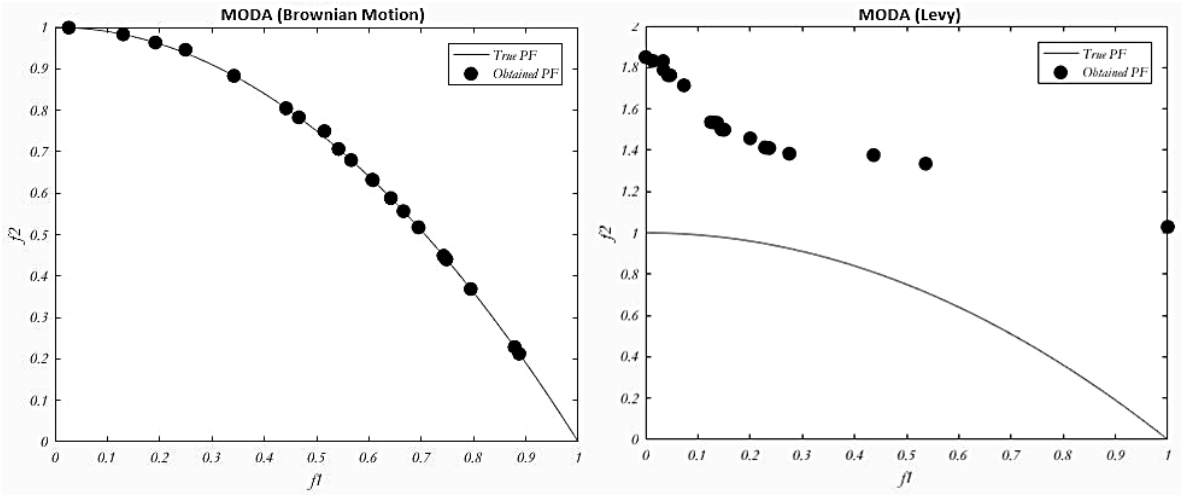
Tablo 4.4. Çok Hedefli Problem Optimizasyondan Elde Edilen Karşılaştırmalı İstatistiksel Değerler

	ZDT1		ZDT2		ZDT1 Lineer		ZDT3		ZDT4		ZDT6	
	f1	f2	f1	f2	f1	f2	f1	f2	f1	f2	f1	f2
MODA (Levy)	0,20	2,06	0,15	1,57	0,07	1,04	0,40	0,28	0,14	2,24	0,19	1,71
MODA (Brownian)	0,21	0,58	0,50	0,72	0,55	0,45	0,22	0,43	0,20	0,60	0,61	0,91

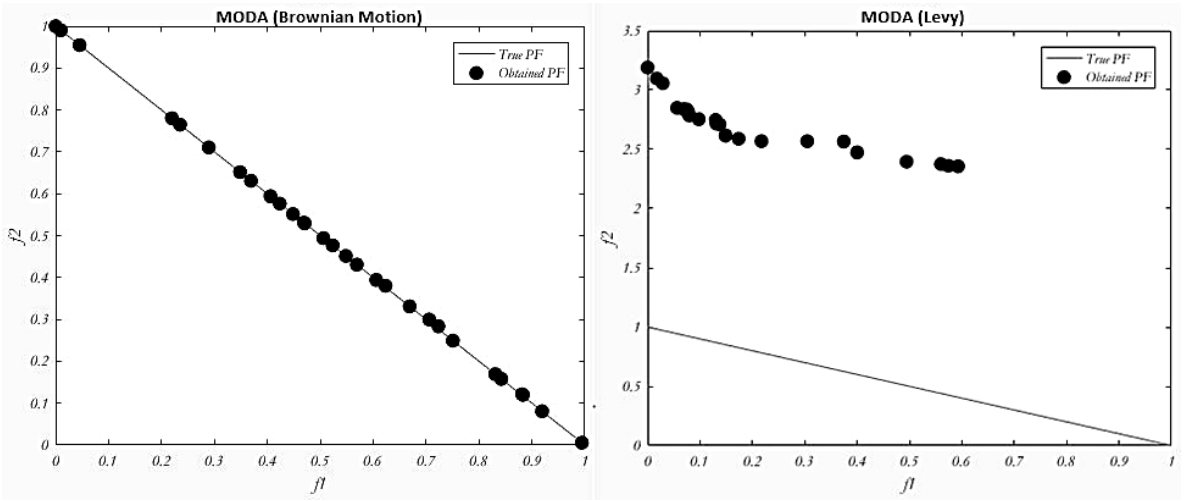
Elde edilen sonuçların grafiksel olarak gösterimi Şekil 4.4, Şekil 4.5, Şekil 4.6, Şekil 4.7, Şekil 4.8 ve Şekil 4.9'da gösterilmektedir:



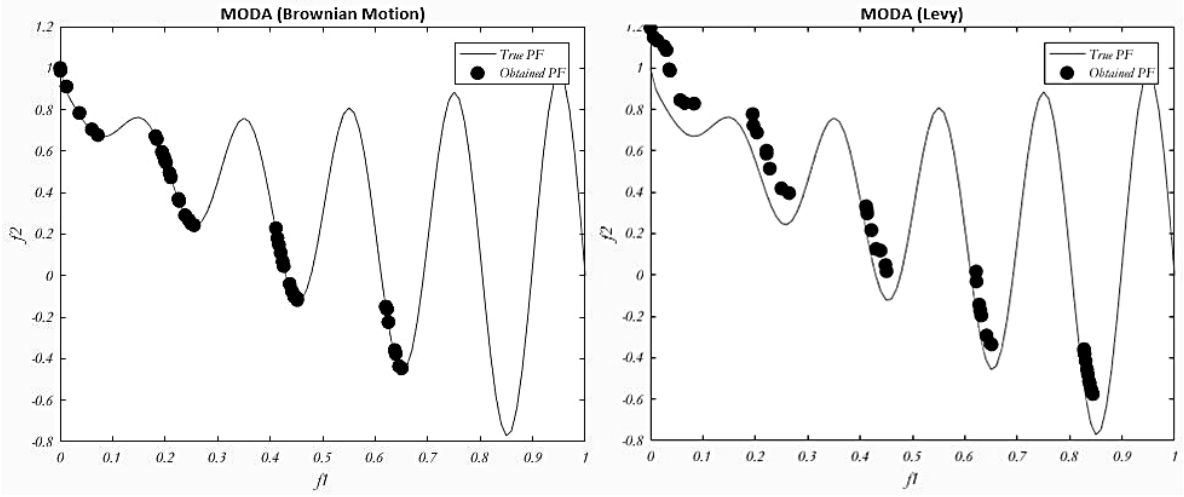
Şekil 4.4. ZDT1 optimizasyonun elde edilen Pareto Optimal Front Karşılaştırması



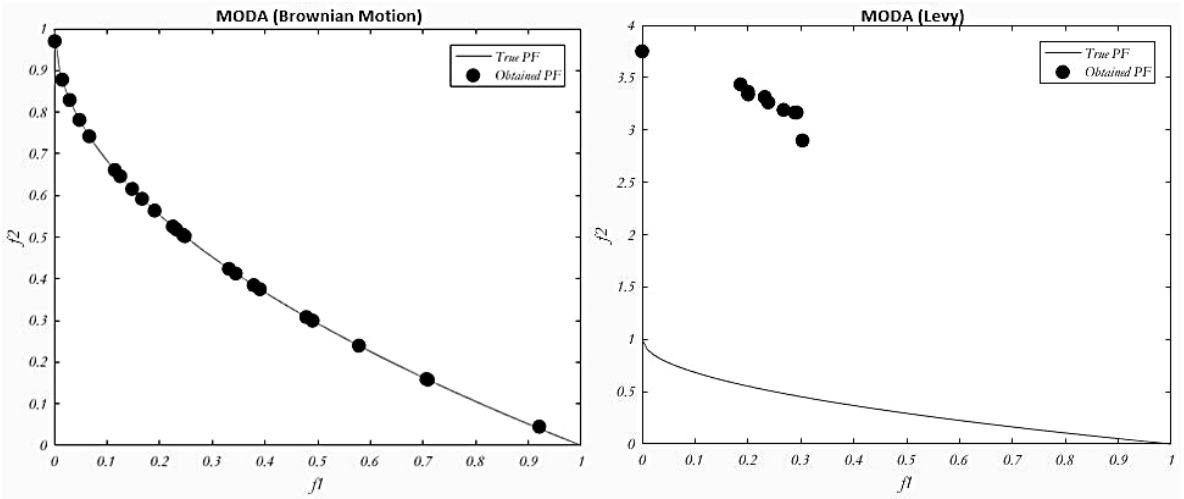
Şekil 4.5. ZDT2 optimizasyonun elde edilen Pareto Optimal Front Karşılaştırması



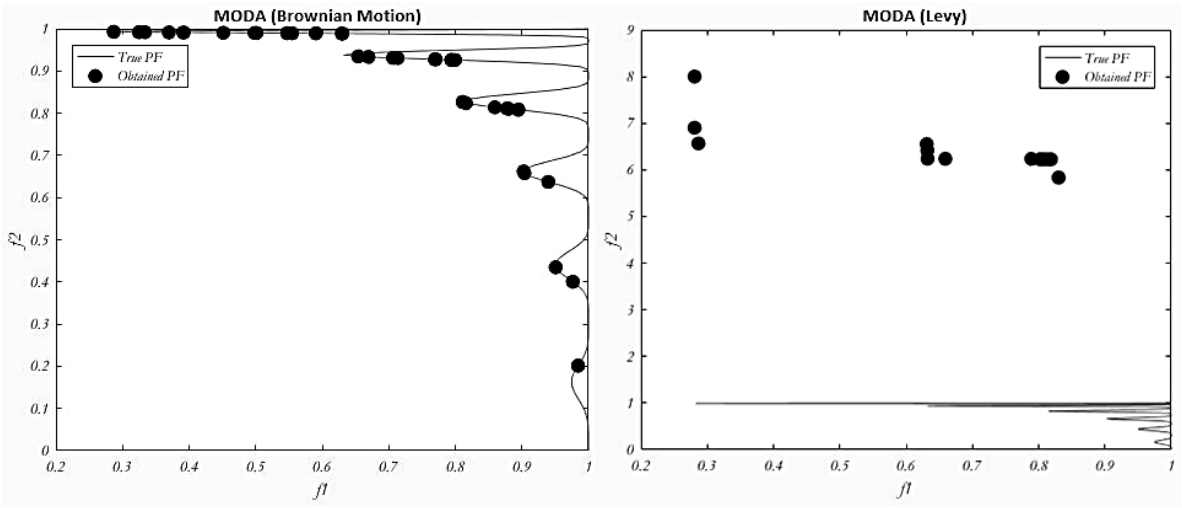
Şekil 4.6. ZDT1 lineer optimizasyonun elde edilen Pareto Optimal Front Karşılaştırması



Şekil 4.7. ZDT3 optimizasyonun elde edilen Pareto Optimal Front Karşılaştırması



Şekil 4.8. ZDT4 optimizasyonun elde edilen Pareto Optimal Front Karşılaştırması



Şekil 4.9. ZDT6 optimizasyonun elde edilen Pareto Optimal Front Karşılaştırması

Elde edilen sonuçlar istatistiksel olarak incelendiğinde (Tablo 4.4), şu sonuçlar göze çarpmaktadır: 6 farklı çok hedefli problemden beklenen toplam 12 farklı minimum değerden 6'sı orijinal algoritma sonuçları ile birbirine yakın elde edilmiştir. Bu 6 sonucun da f1 minimizasyonundan gelmesi beklenen bir sonuçtur. Çünkü f1 minimizasyonu 6 problemin 5'inde yapay yusufçukların x_1 konumuna denk gelmektedir. f2 minimizasyonun da ise beklenen fark ortaya çıkmıştır. Yani optimizasyonun gerçek rasgele hareketinin gerçeklendiği aşama f2 minimizasyonudur. Burada Brownian Hareketi, Levy Flight Mekanizması'na oranla 6 fonksiyonun 5'inde ortalama %50 gibi yüksek bir gelişme sağlamıştır. 1 fonksiyondaki (ZDT3) istatistiksel gerileme incelendiğinde ise trigonometrik köklü yaklaşımlarda rasgele hareketin adım kontrollü uygulanması gerektiği anlaşılmaktadır.

Sonuçlar grafiksel olarak incelendiğinde ise şu sonuç ortaya çıkmaktadır. Yakınsama ve kapsama anlamında Brownian Hareketi, Levy Uçuş Mekanizmasına oranla zorlayıcılıkla doğru orantılı şekilde başarı sağlamıştır. Yani, orijinal algoritma 5 boyutta gerçeklenirken Brownian Hareketi, Levy Uçuş Mekanizması ile aynı sonuçları elde etmiştir. Buna karşın modifiye edildikten sonra zorlayıcılığı artırmak adına arama uzayı 10 boyuta çıkarıldığında Brownian Hareketi, Levy Uçuş Mekanizması karşısında gerçek başarısını net şekilde ortaya çıkarmaktadır.

4.5. Kaynaklı Kiriş Tasarımı Problemi'nin Çözülmesi

Bir gerçek durum çalışması olan Kaynaklı Kiriş Tasarımı Problemi, 40 yusufçukla birlikte 100 iterasyon ile gerçeklenmiştir. Ortalama değerler optimizasyonun 200 kez gerçeklenmesinin ardından elde edilmiştir. Elde edilen sonuçlar Tablo 4.6'te gösterilmiştir.

Tablo 4.5. Kaynaklı Kiriş Tasarımı Optimizasyonunun Karşılaştırmalı Sonuçları

Optimum Maliyet		Adım Kontolsüz		%10 adım kontrolü		%1 Adım Kontrolü	
		Min	**Ort**	**Min**	**Ort**	**Min**	**Ort**
	DA (Levy)	1,302	4689,09	1,253	1,985	1,252	1,718
	DA (BH)	1,293	1,9598	1,252	2,079	1,204	1,930

Elde edilen sonuçlara göre, Brownian Hareketi adım kontrolünün uygulanmadığı durumda, yani rasgele hareketin özgün doğasına göre sağlandığı optimizasyonda bulunan minimum maliyet anlamında ufak bir başarı sağlasa da, Levy Uçuş Mekanizması ile hemen hemen aynı sonucu vermiştir. Brownian Hareketi'nin, net başarısı ise elde edilin ortalama değerde görülmektedir. Algoritmalara herhangi bir müdahalede bulunulmadığında, yani adım kontrolü olmadığında 200 iterasyon için Brownian Hareketi, Levy Uçuş Mekanizması'ndan çok net şekilde başarılıdır. Bu şu anlama gelmektedir: Levy Hareketi'nin zamansız uzun sıçramaları her zaman olumlu etki sağlamamaktadır. Brownian Hareketi'nin kontrollü hareketi, optimal sonuca ulaşma ihtimalini artırmaktadır. Diğer yandan %1 adım kontrollü uygulamada ise, minimum maliyet açından Brownian Hareketi, Levy Uçuş Mekanizması'ndan %20 daha başarılı sonuç elde etmiştir. Bu sonuçlar neticesinde modifiye edilen algoritmanın başarısı, diğer gerçek dünya sorunlarının çözümü için bir rehber niteliğinde olacaktır.

5. SONUÇLAR ve ÖNERİLER

2000'lerin başından bu yana, enerji tasarrufu, ders çizelgeleme ve araç yönlendirme gibi birçok sorunu çözmek için yapay zeka optimizasyonu algoritmaları etkili bir şekilde kullanılmıştır. Bu nedenle çok sayıda optimizasyon tekniği geliştirilmiş ve geliştirilmeye devam edilmiştir.

Bu tez çalışmasında optimizasyon alanında son yıllarda etkin bir şekilde kullanılan sürü zekasına dayalı meta sezgisel algoritmalardan olan Yusufçuk Algoritması, Brownian Hareketi ile modifiye edilmiştir.

Bir önceki bölümde modifiye edilen algoritmanın sonuçları detaylı olarak incelenmiştir. Elde edilen sonuçlar, tek hedefli problem optimizasyonu için 2 ayrı grafikte, 15 kıyaslama fonksiyonunda (farklı boyutlarda ve adım büyüklük değerleriyle) değerlendirilmiştir. Tek hedefli problem optimizasyonunun sonuçları, Yusufçuk Algoritması'nda Levy Uçuş Mekanizması yerine Brownian Hareketi mekanizmasını kullanırken, optimize edilen fonksiyonlarda minimum değerler bağlamında kesin bir başarının elde edildiğini açıkça göstermektedir.

Öte yandan, modifiye edilen algoritma, çok hedefli 6 problem üzerinde test edilmiştir. Modifiye edilmiş algoritmanın başarısını kanıtlamak için burada iki ek problem (ZDT4 - ZDT6) daha uygulanmıştır. İstatistiksel ve özellikle grafik sonuçları incelendiğinde, Brownian Hareketi'nin 10-boyutlu uzayda Levy Uçuş Mekanizması'na kıyasla belirgin bir başarısı vardır.

Olağan rasgele hareketle ilgili Levy Uçuş Mekanizması'nın başarısına ek olarak, Brownian Motion mekanizmasının bu başarısı, rastgele uçuş mekanizmalarının önemini de ortaya koymaktadır. Rastgele hareket algoritmalarının optimizasyon teknikleri üzerindeki etkisinin küçümsenemeyecek bir etkisi vardır. Bu çalışmanın ortaya çıkmasında bir rasgele hareket olan Levy Uçuş Mekanizması'nın Yusufçuk Algoritması içerisinde bir geliştirme sağlaması gerçeği vardı.

Bu tez çalışması kapsamında 2 amaç vardır. Bunlardan birincisi meta sezgisel algoritmalarda rasgele hareketin etkisinin arttırılması, ikincisi ise Brownian Hareketi gibi doğası gereği rasgele olan bu hareketin sürü zekasına vereceği katkıya dikkat çekilmek istenmiştir.

Son olarak, Brownian Motion mekanizmasının bu başarısı, diğer optimizasyon tekniklerinin sonuçlarını iyileştirmenin bir yolunu da göstermiştir. Gelecekteki çalışmalarda, bu mekanizma karmaşık problemleri çözmek için denenebilir ve belli başarılar elde edilebilir.

KAYNAKLAR

[1]. Karaboğa,D. (2017). *Yapay Zeka Optimizasyon Algoritmaları*. Ankara: Nobel Kitap.

[2]. Başdaş,Ö. (2016). Mühendislikte Optimizasyon. Tarihinde 12 Temmuz 2018, adresinden erişildi http://www.elektrikport.com/teknik-kutuphane/muhendislikte-optimizasyon/18764#ad-image-0

[3]. Meşeli,İ. (y.y.). Optimizasyon. Tarihinde 15 Temmuz 2018, adresinden erişildi http://www.ilkaymeseli.com/2011/12/optimizasyon-2/

[4]. Oyegoke,B. S. (2000). *Acta Polytechnica Scandinavica*. *Espoo 2000*.

[5]. Yang,X. (2010). *Nature-Inspired Metaheuristic Algorithms Second Edition*. doi:10.1016/B978-0-12-416743-8.00005-1

[6]. Akyol,S.,Alataş,B. (2012). Güncel Sürü Zekâsı Optimizasyon Algoritmaları. *Nevşehir Üniversitesi Fen Bilimleri Enstitü Dergisi*, *1*, 36–50. doi:10.17100/NBTD.29012

[7]. Dorigo,M.,Maniezzo,V.,Colorni,A. (1991). Ant System: An Autocatalytic Optimizing Process. *Tech. Rep. No. 91- 016*.

[8]. Kennedy,J.,Eberhart,R. (1995). Particle swarm optimization. *Swarm Intelligence*, *1*(1), 33–57. doi:10.1007/s11721-007-0002-0

[9]. Karaboga,D. (2005). An idea based on Honey Bee Swarm for Numerical Optimization. *Technical Report TR06, Erciyes University*, (TR06), 10. doi:citeulike-article-id:6592152

[10]. Chu,S.,Tsai,P.,Pan,J. (2006). Cat Swarm Optimization. *Trends in Artificial Intelligence*, 854–858. doi:10.1007/978-3-540-36668-3_94

[11]. Yang,X. (2010). *Firefly Algorithm*. *Nature-Inspired Metaheuristic Algorithms Second Edition*. doi:10.1016/B978-0-12-416743-8.00005-1

[12]. Rajabioun,R. (2011). Cuckoo optimization algorithm. *Applied Soft Computing Journal*, *11*(8), 5508–5518. doi:10.1016/j.asoc.2011.05.008

[13]. Passino,K. M. (2012). Bacterial Foraging Optimization. *Innovations and Developments of Swarm Intelligence Applications*, 219–234. doi:10.4018/978-1-4666-1592-2.ch013

[14]. Wang,G. G.,Deb,S.,Coelho,L. D. S. (2015). Elephant Herding Optimization. *Proceedings - 2015 3rd International Symposium on Computational and Business Intelligence, ISCBI 2015*, 1–5. doi:10.1109/ISCBI.2015.8

[15]. Song,Y. H.,Chou,C. S.,Stonham,T. J. (1999). Combined heat and power economic dispatch by improved ant colony search algorithm. *Electric Power Systems Research*, *52*(2), 115–121. doi:10.1016/S0378-7796(99)00011-5

[16]. Yan,H. U. I.,Shen,X. Q. X. X. Q.,Li,X.,Wu,M. H. M. (2005). An improved ant algorithm for job scheduling in grid computing. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, *5*(August), 2957–2961.

doi:10.1109/ICMLC.2005.1527448

[17]. Taylor,P.,Chen,C.,Ting,C.,Chen,C. (2006). An Improved Ant Colony System Algorithm For The Vehicle Routing Problem, (October 2014), 37–41. doi:10.1080/10170660609509001

[18]. Yu,B.,Yang,Z. Z.,Yao,B. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, *196*(1), 171–176. doi:10.1016/j.ejor.2008.02.028

[19]. Yu,B.,Yang,Z.-Z.,Xie,J.-X. (2011). A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, *62*(1), 183–188. doi:10.1057/jors.2009.161

[20]. Ã,Y. H.,Shi,P. (2007). An improved ant colony algorithm for fuzzy clustering in image segmentation, *70*, 665–671. doi:10.1016/j.neucom.2006.10.022

[21]. Kaveh,A.,Talatahari,S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations (Swansea, Wales)*, *27*(1), 155–182. doi:10.1108/02644401011008577

[22]. Angeline,P. J. (1999). Using selection to improve particle swarm optimization. *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 84–89. doi:10.1109/ICEC.1998.699327

[23]. Liu,B.,Wang,L.,Jin,Y. H.,Tang,F.,Huang,D. X. (2005). Improved particle swarm optimization combined with chaos. *Chaos, Solitons and Fractals*, *25*(5), 1261–1271. doi:10.1016/j.chaos.2004.11.095

[24]. Jiang,Y.,Hu,T.,Huang,C. C.,Wu,X. (2007). An improved particle swarm optimization algorithm. *Applied Mathematics and Computation*, *193*(1), 231–239. doi:10.1016/j.amc.2007.03.047

[25]. Roh,J. H.,Kim,M. J.,Song,H. Y.,Park,J. B.,Lee,S. U.,Son,S. Y. (2010). An improved particle swarm optimization for nonconvex economic dispatch problems. *Journal of Electrical Engineering and Technology*, *8*(1), 80–89. doi:10.5370/JEET.2013.8.1.080

[26]. Gao,W.,Liu,S. (2011). Improved artificial bee colony algorithm for global optimization. *Information Processing Letters*, *111*(17), 871–882. doi:10.1016/j.ipl.2011.06.002

[27]. Forsati,R.,Keikha,A.,Shamsfard,M. (2015). An improved bee colony optimization algorithm with an application to document clustering. *Neurocomputing*, *159*(1), 9–26. doi:10.1016/j.neucom.2015.02.048

[28]. Akay,B.,Karaboga,D. (2012). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, *192*, 120–142. doi:10.1016/j.ins.2010.07.015

[29]. Ghany,K. K. A.,Salam,M. A.,Zawbaa,H. M.,Emary,E.,Ghany,K. K. A.,Parv,B. (2016). A hybrid dragonfly algorithm with extreme learning machine for prediction A hybrid dragonfly algorithm with extreme learning machine for prediction, (August). doi:10.1109/INISTA.2016.7571839

[30]. Daely,P. T.,Shin,S. Y. (2016). Range Based Wireless Node Localization Using Dragonfly Algorithm Range Based Wireless Node Localization Using Dragonfly Algorithm. *Proceedings of Eighth International Conference onUbiquitous and Future Networks (ICUFN)*, (1), 1012–1015. doi:10.1109/ICUFN.2016.7536950

[31]. Sree Ranjini,S. R.,Murugan,S. (2017). Memory based Hybrid Dragonfly Algorithm for numerical optimization problems. *Expert Systems with Applications*, *83*, 63–78. doi:10.1016/j.eswa.2017.04.033

[32]. Mafarja,M. M.,Eleyan,D.,Jaber,I.,Hammouri,A.,Mirjalili,S. (2017). Binary Dragonfly Algorithm for Feature Selection. *Proceedings - 2017 International Conference on New Trends in Computing Sciences, ICTCS 2017*, *2018–Janua*, 12–17. doi:10.1109/ICTCS.2017.43

[33]. Tharwat,A.,Gabel,T.,Hassanien,A. E. (2018). Parameter Optimization of Support Vector Machine Using Dragonfly Algorithm, *639*. doi:10.1007/978-3-319-64861-3

[34]. Chris,W. (y.y.). Levy Flight. Tarihinde 18 Temmuz 2018, adresinden erişildi http://chriswarbo.net/projects/optimisation/levy.html

[35]. Hakli,H.,Uğuz,H. (2014). A novel particle swarm optimization algorithm with Levy flight. *Applied Soft Computing Journal*, *23*, 333–345. doi:10.1016/j.asoc.2014.06.034

[36]. Heidari,A. A.,Pahlavani,P. (2017). An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. *Applied Soft Computing Journal*, *60*, 115–134. doi:10.1016/j.asoc.2017.06.044

[37]. Pavlyukevich,I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, *226*(2), 1830–1844. doi:10.1016/j.jcp.2007.06.008

[38]. Barthelemy,P.,Bertolotti,J.,Wiersma,D. S. (2008). A Lévy flight for light. *Nature*, *453*(7194), 495–498. doi:10.1038/nature06948

[39]. Mirjalili,S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, *27*(4), 1053–1073. doi:10.1007/s00521-015-1920-1

[40]. Kamaruzaman,A. F.,Zain,A. M.,Yusuf,S. M.,Udin,A. (2013). Levy Flight Algorithm for Optimization Problems - A Literature Review. *Applied Mechanics and Materials*, *421*(September 2013), 496–501. doi:10.4028/www.scientific.net/AMM.421.496

[41]. Abdechiri,M.,Meybodi,M. R.,Bahrami,H. (2013). Gases brownian motion optimization: An algorithm for optimization (GBMO). *Applied Soft Computing Journal*, *13*(5), 2932–2946. doi:10.1016/j.asoc.2012.03.068

[42]. Jan,I. (1779). *Experiments on vegetables, discovering their great power of purifying the common air in sunshine, and of injuring it in the shade or at night*. Cambridge, Massachusetts: Harvard University Press.

[43]. Robert,B. (1828). Mr. R. Brown On the Existence of active Molecules. *The*

Philosophical Magazine and Annals of Philosophy, (21).

[44]. Thorvald Nicolai,T. (1880). Om Anvendelse af mindste Kvadraters Methode i nogle Tilfælde, hvor en Komplikation af visse Slags uensartede tilfældige Fejlkilder giver Fejlene en 'systematisk' Karakter. *Reitzel*, 381–408.

[45]. CadSay. (2018). Matlab Nedir Nerelerde Kullanılır? Tarihinde 25 Temmuz 2018, adresinden erişildi http://cadsay.com/matlab-nedir-nerelerde-kullanilir

[46]. Craig W.,R. (1987). Flocks , Herds , and Schools : A Distributed Behavioral Model. *Computer Graphics, Volume 21, Number 4*, (10305085), 1–2. doi:10.1145/280811.281008

[47]. Ahmet Cevahir,Ç. (2018). Tek amaçlı optimizasyon ile çok amaçlı optimizasyon arasındaki fark nedir? Tarihinde 21 Temmuz 2018, adresinden erişildi http://ahmetcevahircinar.com.tr/2018/06/01/tek-amacli-optimizasyon-ile-cok-amacli-optimizasyon-arasindaki-fark-nedir/

[48]. Savic,D. (2002). Single-objective vs Multiobjective Optimisation for Integrated Decision Support. *Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society*, 7–12.

[49]. Deb,K. (2005). Multi-Objective Optimization. *Search Methodologies*, 273–316. doi:10.1007/0-387-28356-0_10

[50]. Coello Coello,C. A. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279. doi:10.1109/TEVC.2004.826067

[51]. Kaveh,A.,Talatahari,S. (2009). Engineering Optimization With Hybrid Particle Swarm and Ant Colony Optimization, *10*(6), 611–628.

[52]. Brajevic,I.,Tuba,M. (2013). An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems. *Journal of Intelligent Manufacturing*, *24*(4), 729–740. doi:10.1007/s10845-011-0621-6

[53]. Liao,T.,Socha,K.,Oca,M. a M. De,St,T. (2014). Ant Colony Optimization for Mixed-Variable Optimization Problems. *Ieee Transactions on Evolutionary Computation*, *18*(4), 503–518. doi:10.1109/TEVC.2013.2281531

[54]. Bhandari,V. B. (2010). *Design of Machine Elements* (Third.). New Delhi: The Mc-Graw -Hill Company Limited.

[55]. Christu Nesam David,D.,Elizabeth Amudhini Stephen,S.,Ajay Joe,A. (2016). Cost Minimization of Welded Beam Design Problem Using PSO, SA, PS, GODLIKE, CUCKOO, FF, FP, ALO, GSA and MVO, *5*(1), 1–14.

[56]. Coello Coello,C. A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113–127. doi:10.1016/S0166-3615(99)00046-9

[57]. Hedar,A. R.,Fukushima,M. (2006). Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, *35*(4), 521–549. doi:10.1007/s10898-005-3693-z

[58]. Mezura-Montes,E.,Coello,C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems*, *37*(4), 443–473. doi:10.1080/03081070701303470

[59]. Rashedi,E.,Nezamabadi-pour,H.,Saryazdi,S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences*, *179*(13), 2232–2248. doi:10.1016/j.ins.2009.03.004

ÖZGEÇMİŞ

Adı ve Soyadı **:** Hakan Gülcan

Doğum Tarihi **:** 05/01/1993

E-mail **:** hgulcan333@gmail.com

Öğrenim Durumu **:**

Derece	**Bölüm/Program**	**Üniversite**	**Yıl**
Lisans	Bilgisayar Mühendisliği	Çukurova Üniversitesi	2011-2016
Yüksek Lisans	Elektrik Elektronik Müh.	Mersin Üniversitesi	2016-2018

ESERLER (Makaleler ve Bildiriler)

1. H. Gülcan, E. Avaroğlu, R. Sinekli, V. Yıldız, "Web Based AIS31 Test Software For Random Number Generators", International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2-4 Kasım 2017, Tokat, Türkiye
2. H. Gülcan, Ç. Acı, "Performance Comparison of Levy Flight Mechanism in Dragonfly Optimization and Gravitational Search Algorithm", International Conference on Engineering Technology and Innovation (ICETI), 07-11 Mart 2018, Budapeşte, Macaristan
3. H. Gülcan, Ç. Acı, "A Modification of Dragonfly Algorithm with Brownian Motion", Computational Intelligence and Neuroscience, Ağustos 2018 (Yayın İçin Başvuru Yapıldı)