

**YOL PLANLAMASINDA OPTİMİZASYON ALGORİTMALARININ KARŞILAŞTIRMALI ANALİZİ**

**FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ**

**LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI**

**İSTANBUL, 2024**

**YÜKSEK LİSANS FİNAL PROJE RAPORU**

**YOL PLANLAMASINDA OPTİMİZASYON ALGORİTMALARININ KARŞILAŞTIRMALI ANALİZİ**

ÖZET

Bu makalede, yol planlama (path planning) problemi için Gri Kurt Optimizasyonu (GWO), Karınca Kolonisi Optimizasyonu (ACO), Kum Kedisi Sürü Optimizasyonu (SCSO) ve Balina Optimizasyonu (WOA) olmak üzere dört farklı optimizasyon algoritmasının performansları karşılaştırılmıştır. Çalışmamızın ilk aşamasında, yol planlama probleminin tanımı ve bu bağlamda karşılaşılan zorlukları ele alınmıştır. Daha sonra, her bir algoritmanın temel prensipleri ve matematiksel modelleri ayrıntılı olarak incelenmiştir. Bu algoritmalar, yol planlama problemi üzerinde uygulanarak yakınsama eğrileri (convergence curves) ve kutup grafikleri (boxplots) kullanılarak sonuçlar analiz edilmiş ve performans değerlendirmesi, en iyi, en kötü, ortalama ve standart sapma değerlerinin karşılaştırılması ile gerçekleştirilmiştir. Ayrıca, algoritmaların istatistiksel olarak anlamlı olup olmadığını belirlemek için p-değerleri hesaplanmış ve karşılaştırılmıştır. Sonuçlar, belirtilen algoritmaların yol planlama problemi üzerindeki etkinliklerini göstermiş ve bu algoritmaların performansları detaylı bir şekilde kıyaslanmıştır. Bu çalışma, yol planlama problemlerinde kullanılan optimizasyon tekniklerinin etkinliğini artırmak amacıyla önemli sonuçlar sunmaktadır.

**Anahtar kelimeler:** Path Planning, GWO, ACO, SCSO ve WOA Algoritması

İÇİNDEKİLER

[ÖZET ii](#_Toc170586848)

[GİRİŞ 1](#_Toc170586852)

[BİRİNCİ BÖLÜM 2](#_Toc170586853)

[1. YOL PLANLAMA PROBLEMİ](#_Toc170586854)…………………...……………………………2

[1.2 YOL PLANLAMA PROBLEMİNİN TARİHİ………………………………...2  
1.2 YOL PLANLAMA PROBLEMİNİN TANIMI………………………………..2  
1.2 YOL PLANLAMA TÜRLERİ](#_Toc170586855)…………………………………………………3

[İKİNCİ BÖLÜM 5](#_Toc170586860)

[2. ALGORİTMALAR](#_Toc170586861)……………………………………………………………….5

[2.1. GRİ KURT OPTİMİZASYONU (GREY WOLF OPTİMİZATİON-GWO)](#_Toc170586862)…5

2.1.1.ALGORİTMANIN TEMEL BİLEŞENLERİ…………………………..5

[2.1.2.GWO ALGORİTMASININ MATEMATİKSEL MODELİ](#_Toc170586864)……..……6

[2.2. KARINCA KOLONİSİ OPTİMİZASYONU (ANT COLONY OPTİMİZATİON-ANT)](#_Toc170586862)…………………………………………………………...8

2.2.1. ALGORİTMANIN TEMEL BİLEŞENLERİ…………………………9

[**2.2.2. ANT ALGORİTMASININ MATEMATİKSEL MODELİ**](#_Toc170586864)**……....…10**  
[2.3. KUM KEDİSİ SÜRÜ OPTİMİZASYONU (SAND CAT SWARM OPTİMİZATİON-SCSO)](#_Toc170586862)…………………………………………………………12

2.3.1. ALGORİTMANIN TEMEL BİLEŞENLERİ………………………...12

[**2.3.2. SCSO ALGORİTMASININ MATEMATİKSEL MODELİ**](#_Toc170586864)**..………12**  
[2.4. BALİNA OPTİMİZASYONU ALGORİTMASI (WHALE OPTIMIZATION ALGORITHM -WOA)](#_Toc170586862)……………………………………………………………15

2.4.1. ALGORİTMANIN TEMEL BİLEŞENLERİ……………………..…15

[2.4.2. WOA ALGORİTMASINI MATEMATİKSEL MODELİ](#_Toc170586864)………......16

[ÜÇÜNCÜ BÖLÜM……………………………………………………………..….18](#_Toc170586866)

[3. DENEYSEL ÇALIŞMA 18](#_Toc170586867)

[3.1. GİRİŞ (INTRODUCTION) 18](#_Toc170586868)

[3.2. DENEYSEL KURULUM (EXPERIMENTAL SETUP) 18](#_Toc170586868)

[3.3. ALGORİTMALARIN UYGULANMASI (APPLICATION OF ALGORITHMS) 19](#_Toc170586868)

[3.3.1. GWO (GRİ KURTLAR OPTİMİZASYONU ALGORİTMASI) 19](#_Toc170586869)

[3.3.2 ACO (KARINCA KOLONİSİ OPTİMİZASYONU ALGORİTMASI ) 20](#_Toc170586870)

[3.3.3. SCSO (KUM KEDİSİ SÜRÜ OPTİMİZASYONU ALGORİTMASI) 21](#_Toc170586869)

[3.3.4. WOA (BALİNA OPTİMİZASYONU ALGORİTMASI) 22](#_Toc170586870)

[DÖRDÜNCÜ BÖLÜM……………………………………………………..….…..24](#_Toc170586866)

[4. SONUÇLAR VE DEĞERLENDİRME (RESULTS AND EVALUATION) 24](#_Toc170586867)

[4.1. ALGORİTMALARIN PERFORMANS SONUÇLARI 24](#_Toc170586868)

[4.2. PERFORMANS DEĞERLENDİRMESİ 24](#_Toc170586868)

[4.3. MANN-WHİTNEY U TESTİ SONUÇLARI 25](#_Toc170586868)

[4.4. GRAFİKSEL DEĞERLENDİRMELER 27](#_Toc170586868)

[4.1.1. CONVERGENCE CURVE (YAKINSAMA EĞRİSİ ) 27](#_Toc170586869)

[4.1.2. BOXPLOT GRAFİĞİ 28](#_Toc170586870)

[4.1.3. EXPLORATİON AND EXPLOİTATİON GRAFİĞİ (KEŞİF VE SÖMÜRÜ) 28](#_Toc170586869)

[SONUÇ VE ÖNERİLER 31](#_Toc170586873)

[KAYNAKÇA 33](#_Toc170586874)

GİRİŞ

Yol planlama (path planning), otonom robotlar, insansız hava araçları (İHA'lar) ve çeşitli akıllı sistemler için önemli bir araştırma alanıdır. Bu problem, belirli bir başlangıç noktasından hedef noktaya en kısa, en güvenli veya en az maliyetli yolu bulmayı amaçlar ve birçok uygulama alanında büyük bir öneme sahiptir. Yol planlama problemlerinin çözümünde optimizasyon algoritmaları yaygın olarak kullanılmaktadır.

Bu çalışmada, Gri Kurt Optimizasyonu (Grey Wolf Optimization - GWO), Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO), Kum Kedisi Sürü Optimizasyonu (Sand Cat Swarm Optimization - SCSO) ve Balina Optimizasyonu (Whale Optimization Algorithm - WOA) olmak üzere dört farklı doğadan esinlenmiş optimizasyon algoritmasının performanslarını yol planlama problemi üzerinde karşılaştırıyoruz. GWO, kurt sürülerinin avlanma davranışlarını taklit ederken, ACO, karıncaların yiyecek bulma ve feromon izleme mekanizmalarını kullanır. SCSO, kum kedilerinin hayatta kalma stratejilerinden esinlenirken, WOA ise balinaların avlanma tekniklerini simüle eder. Ayrıca bu çalışmada, bu algoritmaların temel prensiplerini ve matematiksel modellerini detaylandırarak, yol planlama problemi üzerindeki performanslarını değerlendireceğiz. Algoritmalar, yakınsama eğrileri ve kutup grafikleri ile analiz edilerek, en iyi, en kötü, ortalama ve standart sapma değerleri üzerinden karşılaştırılacaktır. Ayrıca, p-değerleri hesaplanarak algoritmaların istatistiksel anlamlılıkları incelenecektir. Bu araştırma, yol planlama problemlerinin çözümünde kullanılan optimizasyon tekniklerinin etkinliğini artırmayı amaçlamaktadır. Sonuçlar, bu alanda daha etkili ve verimli çözümler geliştirilmesine katkı sağlayacaktır.

BİRİNCİ BÖLÜM

1. YOL PLANLAMA PROBLEMİ
   1. **YOL PLANLAMA PROBLEMİNİN TARİHİ**

Yol planlama probleminin kökenleri, 18. yüzyılda Leonhard Euler'in grafik teorisi çalışmalarıyla başlamıştır. 19. yüzyılda en kısa yol ve gezgin satıcı problemi gibi klasik optimizasyon sorunları, bu alanın temelini oluşturmuş ve 20. yüzyılın ortalarında bilgisayarların gelişimiyle Dijkstra'nın Algoritması gibi önemli yöntemler ortaya çıkmıştır. 1960'lar ve 1970'lerde yapay zeka ve robotikteki ilerlemeler, daha karmaşık problemleri çözmeyi mümkün kılmıştır. Özellikle 60'lı yıllardan itibaren yol planlama konusunda çeşitli katkılar ortaya çıkmıştır. Bu araştırmalardan bazıları, bir robotun bir noktadan diğerine hareket etmesini sağlayan yörünge hesaplaması, insansı robotikte kinematik ve dinamik kısıtlamaları içeren yürüme hareketlerinin ve manipülasyonun hesaplanması, biyo-bilişimde moleküllerin hareketlerinin incelenmesi gibi farklı alanlarda çeşitli zorlukların çözülmesine izin vermiştir. 1980'lerden itibaren doğadan esinlenilmiş optimizasyon algoritmaları geliştirilmiş ve günümüzde bu algoritmalar, otonom araçlar ve akıllı şehirler gibi birçok uygulamada kullanılmaktadır. Yol planlama, robotik ve otomasyon alanlarında kritik bir rol oynamakta ve lojistikten sağlığa birçok sektörde büyük öneme sahiptir.

* 1. **YOL PLANLAMA PROBLEMİNİN TANIMI**

Yol planlama (path planning), bir başlangıç noktasından hedef noktaya en uygun yolu bulmayı amaçlayan bir problemdir. Bu en uygun yolu bulma sürecinde, belirli kısıtlamalar ve hedefler dikkate alınarak, en kısa, en güvenli veya en az maliyetli yolun belirlenmesi hedeflenir. Yol planlama, robotik, otonom araçlar, insansız hava araçları (İHA'lar) ve diğer akıllı sistemler için temel bir fonksiyondur. Yol planlama problemleri genellikle bir grafik veya grid üzerinde tanımlanır ve bu yapı, belirli düğümler (nodes) ve bu düğümleri birbirine bağlayan kenarlardan (edges) oluşur. Amaç, başlangıç düğümünden hedef düğümüne ulaşan en uygun yolu bulmaktır.  
Yol planlama problemi otonom araçlar, insansız hava araçları (İHA'lar), robotik, akıllı şehirler, askeri uygulamalar, oyun ve simülasyon gibi pek çok alanda kullanılır.

* 1. **YOL PLANLAMA TÜRLERİ**

Yol planlama türlerini aşağıdaki görselde de görüldüğü üzere dört farklı seviye de açıklıyoruz. Her seviye, farklı teknik gereksinimleri, matematiksel modelleri veya uygulama senaryolarını temsil eder.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Fig. 1.** Yol Planlaması Tür Diyagramı

Birinci seviye için path planning türleri :

* *Holonomic Path Planning:* Holonomic sistemler, bir noktadan başlayıp herhangi bir yöne serbestçe hareket edebilen sistemlerdir. Bu sistem için, dört tekerlekli bir robot düşünelim. Bu robotun her tekerleğinin bağımsız olarak kontrol edilebildiği bir sistemde, dört tekerlekli robot doğrudan her yöne hareket edebilir.
* *Nonolonomic Path Planning:* Nonholonomic sistemler ise belirli kısıtlar altında hareket eden sistemlerdir. Bu sınırlandırmalar, sistemin hareketini ve yönlendirmesini sağlar. Bu sistem için otomobil örneğini verecek olursak, otomobil sadece ileri ve geri hareket edebilir, yana hareket edemez. Yönünü değiştirmek için dönmek zorundadır.
* *Kinodynamic Path Planning:* Bu tür planlama ise hem holonomic hem de nonholonomictir. Yani sistemin, hareket kabiliyeti ve fiziksel sınırlamaları göz önünde bulundurulur. Bu sistemler için dronlar örnek olarak verilebilir.

İkinci seviye için path planning türleri:

* *Environment Modeling:* Yol planlamanın ilk adımı olan çevre modellemesi, bir robotun veya bir aracın etrafında bulunan boş alanları ve engelleri haritalamasını içerir.
* *Search Optimal/Feasible Path*: Burada ise belirli bir çevre modeline göre, verilen kısıtlar altında en uygun olan yolun aranması amaçlanmaktadır. Bir robotun başlangıç noktasından, varış noktasına en kısa yolu bulması bu kısım için örnek verilebilir.

Üçüncü seviye için path planning türleri:

* *Off-line Path Plannig*: Yol planlamanın bu türü, bir robotun veya sistemin, hareket etmeden önce bir yol planı oluşturduğu durumu ifade eder. Burada bir endüstriyel robotun, fabrikanın sabit bir düzeninde çalışırken önceden belirlenmiş yolları takip etmesi örneğini verebiliri
* *On-line Path Plannig:* Bu tür yol planlama ise, bir robotun veya sistemin, anlık çevre şartlarını ve değişkenleri dikkate alarak sürekli olarak yol planını güncellediği durumu ifade eder. Bir robotun, dinamik bir ortamda anlık olarak engelleri algılayıp yolunu güncellemesi örneği verilebilir.

Dördüncü seviye için path planning türleri:

* *Deterministic Path Plannig:* Bu tür yol planlama, belirli ve sabit bir çevrede, kesin ve tekrarlanabilir yolları kullanır. Bir robotun, belirli bir labirent de her zaman aynı yolu kullanarak hedefe ulaşması örneği verilebilir.
* *Probabilistic Path Planning*: Bu tür yol planlama, çevredeki belirsizlikleri ve rastgelelikleri dikkate alarak çeşitli olasılıklarla yolları arar. Bir otonom aracın, trafikte rastgele engellerle karşılaşabileceği bir durumda en uygun rotayı belirlemesi örneği verilebilir.

İKİNCİ BÖLÜM

1. ALGORİTMALAR

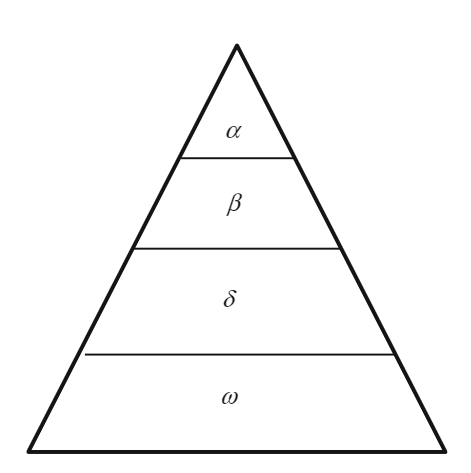
Bu bölümde yol planlama için kullanılan Gri Kurt Optimizasyonu (GWO), Karınca Kolonisi Optimizasyonu (ANT), Kum Kedisi Sürü Optimizasyonu (SCSO) ve Balina Optimizasyonu (WOA) gibi doğal metaheuristik algoritmalardan bahsedeceğiz. Bu algoritmalar, doğadan ilham alarak geliştirilmiş ve karmaşık optimizasyon problemlerinde etkili çözümler sunmaktadırlar. Her bir algoritma, farklı bir sürünün davranışlarını taklit ederek, potansiyel çözüm alanlarını keşfetmek ve iyileştirmek için kullanılmaktadırlar.

* 1. **GRİ KURT OPTİMİZASYONU (GREY WOLF OPTİMİZATİON-GWO)**

Gri Kurt Optimizasyonu (GWO), Mirjalili ve arkadaşları tarafından 2014 yılında geliştirilen, doğadan esinlenen bir metaheuristik optimizasyon algoritmasıdır. Algoritma, gri kurtların avlanma davranışlarını model alır ve özellikle küresel optimizasyon problemlerini çözmede etkili olduğu kanıtlanmıştır. GWO'nun temelinde, gri kurtların sosyal hiyerarşisi ve akıllı avlanma stratejileri yer alır.

* + 1. Algoritmanın Temel Bileşenleri

Gri kurtlar sosyal hiyerarşiye göre alfa (α), beta (β), delta (δ) ve omega (ω) olarak gruplara ayrılır. GWO algoritması, liderler (α, β, δ) ve takipçiler (ω) arasındaki işbirliğiyle çalışır. Gri kurtların sosyal hiyerarşisi aşağıdaki şekilde görüldüğü gibidir.



**Fig 2.** GWO Şematiksel Gösterim

Burada ;

* Alfa (α) : Sürünün ilk seviyesidir ve liderlerini temsil eder. Genellikle sürüleri için uyku yeri, avlanma ve uyanma zamanı gibi kararların verilmesinden sorumlu olan erkek ve dişi kurtlardır.
* Beta (β) : Gri kurtların sosyal hiyerarşisinde alfadan sonraki seviye betadır ve betanın rolü alfaya karar vermede yardımcı olmaktır. Beta, erkek veya dişi kurt olabilir ve beta, içlerinden biri yaşlandığında veya öldüğünde, alfanın yerini alacak en iyi aday olabilir.
* Delta (δ) : Delta seviyesindeki kurtlar alfa ve beta kurtlara itaat eder ve omega kurtlara hakim olurlar. Sürüde izci, nöbetçi, yaşlı, avcı ve bakıcı gibi davranırlar.
* Omega (ω) : Gri kurt sürüsünün en zayıf seviyesi omegadır. Omega seviyesindeki kurtlar diğer bireylerin emirlerine uymak zorundadırlar ve yemek yemelerine izin verilen son kurtlardır. Omega, sürüdeki en az önemli bireyler gibi görünüyor, ancak Omega olmadan iç kavgalar ve diğer sorunlar gözlemlenebilir.

Gri kurtların avlanma davranışları ise üç ana bölümden oluşur:

(1) Avı takip etmek, kovalamak ve yaklaşmak

(2) Avı hareket etmeyi bırakıncaya kadar takip etmek ve kuşatmak

(3) Avına saldırmak.

* + 1. GWO Algoritmasının Matematiksel Modellemesi

Gri kurt sürüsünün sosyal hiyerarşisi ve avlanma tekniği GWO algoritmasında modellenmiştir. GWO algoritması, avlanma davranışlarını üç ana aşamada gerçekleştirir. Bunlar :

1. **Avı Çevreleme: Gri** kurtlar avlarını bulmak ve çevrelemek için belirli bir strateji izlemektedirler. Bu süreci aşağıdaki denklemler ile ifade edebiliriz:

Buradaki matematik gösterimler

* : Av ile kurt arasındaki mesafeyi ifade eden vektör

* : alfa, beta ve delta gri kurtları için avın konum vektörü (en iyi çözüm) anlamına gelir
* : mevcut kurtun konum vektörü
* : bir sonraki iterasyondaki kurtun konum vektörü
* ve : katsayı vektörleri
* : mevcut iterasyon

temsil etmektedirler. Ayrıca ve katsayı vektörleri ise

* : Lineer olarak 2 ile 0 arasında azalan katsayı
* ve : [0,1] aralığında rastgele vektörler (kurdun herhangi bir konumu seçmesine izin verir.)

olmak üzere

şeklinde hesaplanır.

1. **Ava Saldırı :** Alfa, beta ve delta liderliği altındaki kurtlar avın yerini belirlemeye çalışırlar. En iyi üç çözüm yani alfa, beta ve delta belirlenir ve diğer kurtlar bu liderleri takip ederek çözüm arayışına devam eder. Üç ana liderin pozisyonları kullanılarak kurtların pozisyonları aşağıdaki matematiksel ifadeler ile güncellenir.

* : sırasıyla en iyi üç çözüm (alfa, beta ve delta kurtları için avın konum vektörü)
* : alfa, beta ve delta kurtları için katsayı vektörleri

1. **Avın Yakalanması :** katsayısı 0'a yaklaştıkça, kurtlar avlarına daha da yakınlaşır ve avı yakalarlar. Bu durum, küresel optimizasyon problemlerinde en iyi çözüme ulaşma sürecini temsil eder.  
     
   **GWO algoritmasının adımları:**
2. Başla
3. Başlangıç parametrelerini kullan (gri kurt sayısı, iterasyon sayısı vb.)
4. Farklı sosyal hiyerarşiye sahip α, β, δ ve ω gri kurtlarının ilk popülasyonunun belirlenmesi
5. α, β, δ gri kurtları ile avın konum bilgisini tahmin edin
6. Gri kurtların konumunu avın konumuna göre değerlendirin
7. Gri kurtları derecelendirin (en iyi çözüm alfa vb.olarak adlandırın.
8. Maksimum iterasyon sayısına ulaşıldı mı? Hayır, ise, 3. adıma geri dön.
9. Evet, ise en iyi çözümü döndür
10. Bitir.
    1. **KARINCA KOLONİSİ OPTİMİZASYONU (ANT COLONY OPTİMİZATİON – ANT)**

Karınca koloni optimizasyonu ilk olarak 1992 senesinde Marco Dorigo tarafından doktora tezi olarak tasarlanmış ve ilk olarak 1999 senesinde Dorigo ve Stützle tarafından yayınlanmıştır. Bu algoritma, bir karınca kolonisindeki bireysel karıncaların davranışlarını ve etkileşimlerini taklit ederek çalışır. Karıncalar, yiyecek kaynaklarına ulaşmak için çevrelerine izler bırakır. Bu izler, diğer karıncaların doğru rotayı bulmasına yardımcı olurken, aynı zamanda iz bırakan karıncalar da izlerin yoğunluğuna göre hareketlerini ayarlar. Karınca koloni algoritması da benzer bir şekilde, olası çözüm yollarını arayarak, daha iyi çözümlere ulaşmak için bir takım işaretler ve denemeler yoluyla yol haritası oluşturur.

çizgi, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Fig 3**. ACO Şematik Gösterim

Gerçek karıncaların en kısa yolu bulmasına ilişkin verilen yukarıdaki görseli incelediğimizde karıncaların, başlangıçta düz bir hattı takip ettiklerini ve bu esnada feromon olarak adlandırılan bir maddeyi yol güzergahına bırakarak kendilerinden sonra gelen karıncaların yollarını bulmalarını kolaylaştırmakta olduklarını görmekteyiz. Önlerine bir engel konulduğunda feromonları takip edemediklerinden, karıncalar gidebilecekleri iki yoldan birini öncelikle rastsal olarak seçmektedirler. Kısa olan yoldan birim zamandaki geçiş daha fazla olacağından bırakılan feromon miktarı da daha fazla olur. Buna bağlı olarak, zaman içerisinde kısa olan yolu tercih eden karıncaların sayısında artış olur. Belli bir süre sonra tüm karıncalar kısa yolu tercih ederler.

* + 1. Algoritmanın Temel Bileşenleri
* *Feromon Yoğunluğu (τ) :* Feromon, gerçek karıncaların bir haberleşme ve yön bulma aracı olarak kullandıkları, vücutlarından salgıladıkları bir kimyasaldır. Karıncalar, yollar üzerinde ilerlerken feromon bırakırlar ve diğer karıncalar bu feromon izlerini takip eder. Feromon izleri, karıncalar tarafından güncellenmekte ve bir bilgiyi temsil etmektedir. Bu feromon izleri zamanla buharlaşır, ancak sıkça kullanılan yollar üzerindeki feromon yoğunluğu artar. Bir yolda feromon izinin yoğun olması, yolun kalitesini gösterir ve tercih olasılığını arttırır.
* *Uygunluk (Sezgi) Değeri (η) :* Sezgi değeri, yolların uzunluğuna veya maliyetine dayalı olarak hesaplanır. Sezgi değeri karıncaların vardıkları bir düğümden varacakları başka bir düğüme kadar olan mesafenin uzaklığıyla ters orantılı bir başlangıç değeridir, yani daha kısa veya daha düşük maliyetli yollar daha yüksek sezgi değerine sahiptir.
* *Feromon Buharlaşması (ρ) :* Feromon buharlaşma oranı, feromonların zamanla azalmasını sağlar. Bu, algoritmanın arama alanında çeşitli yolları keşfetmesini teşvik eder ve eski yolların kalıcı olarak tercih edilmesini engeller.
* *Olasılık Hesabı () :* Karıncaların belirli bir yolu seçme olasılığı, o yoldaki feromon yoğunluğu ve sezgi değeri ile hesaplanır. Olasılık hesabı, karıncaların daha yoğun feromon izlerine sahip ve sezgi değeri yüksek yolları tercih etmesini sağlar.
  + 1. ANT Algoritmasının Matematiksel Modellemesi

##### ACO Algoritmasının Temel Adımları:

1. ***Başlatma****:* Rastgele çözümler (karıncalar) oluşturulur.
2. ***Feromon Güncelleme***: Her çözümün kalitesine göre feromon seviyeleri güncellenir.
3. ***Yol Seçimi***: Feromon yoğunluğuna ve heuristik bilgiye göre karıncalar yollarını seçer.

##### ***Feromon Güncelleme:***

##### Feromon seviyesi, her iterasyonda güncellenir. Feromonun buharlaşma oranı ve her bir karınca tarafından bırakılan feromon miktarı göz önünde bulundurulur. Feromon güncellemesi matematiksel olarak aşağıdaki gibi yapılır :

Burada;

* (t) : t-inci iterasyondaki i ve j noktaları arasındaki feromon miktarı
* : feromon buharlaşma miktarı oranı (0 < < 1)
* : k-nıncı karınca tarafından bırakılan feromon miktarı

***Feromon Güncelleme Miktarı :*** Bir karınca tarafından bırakılan feromon miktarı, karıncanın kat ettiği yolun kalitesiyle doğru orantılıdır.

Burada;

* : sabit katsayı
* ​: k-nıncı karınca tarafından kat edilen yolun toplam uzunluğu.

***Yol Seçimi (Olasılık Hesabı) :*** Karıncalar, feromon yoğunluğuna ve heuristik bilgiye göre yollarını seçer. Seçim olasılığı, feromon yoğunluğu ve heuristik bilgiye bağlıdır.

Burada;

* : k-nıncı karıncanın i noktasından j noktasına gitme olasılığı
* : t-inci iterasyondaki i ve j noktaları arasındaki feromon miktarı
* : i ve j noktaalrı arasındaki sezgisel değeri (genellikle ile ifade edilir, burada ​ iki düğüm arasındaki mesafedir).
* α : Feromon yoğunluğunun önemini belirleyen parametre (feromon katsayısı).
* β : Sezgi (uygunluk) değerinin önemini belirleyen parametre (sezgisel katsayı).
* : k-nıncı karıncanın i noktasında ziyaret etmediği komşu noktaların kümesi.
  1. **KEDİSİ SÜRÜ OPTİMİZASYONU (SAND CAT SWARM OPTİMİZATİON-SCSO)**

Kum Kedisi Sürü Optimizasyonu (Sand Cat Swarm Optimization **-** SCSO), doğadan esinlenen bir optimizasyon algoritmasıdır**.** Kum kedilerinin avlanma ve yaşam davranışlarını taklit ederek çalışır. Kum kedilerinin sergilediği davranış kalıplarına dayanarak, av arama süreci iki ayrı aşamaya dayanır: avını arama (keşif) ve avına saldırma (geliştirme). Bu algoritma, karmaşık optimizasyon problemlerini çözmek için kullanılır ve özellikle büyük veri kümelerinde ve karmaşık çözüm alanlarında etkili sonuçlar elde edebilir.

* + 1. Algoritmanın Temel Bileşenleri

1. ***Kum Kedisi Davranışı****:*

* ***Keşif ve Saldırı****:* Kum kedileri, avlarını bulmak için geniş alanları keşfeder ve ardından etkili saldırı stratejileri kullanarak avlarını yakalarlar. Algoritma, bu davranışı taklit ederek arama ve saldırı fazlarını içerir.
* ***Sosyal Davranış***: Kum kedileri, avlanma sırasında birbirleriyle işbirliği yaparlar ve avlarını paylaşırlar. Algoritma, bu sosyal davranışları modelleyerek, çözüm adaylarının birbirleriyle etkileşime girmesini sağlar.

1. ***Uygunluk (Fitness) Değeri***: Her çözüm adayı, belirli bir uygunluk değeri ile değerlendirilir. Bu değer, çözümün ne kadar iyi olduğunu gösterir ve algoritmanın amacına ulaşmasına yardımcı olur.
2. ***Pozisyon ve Hız Güncelleme****:* Kum kedilerinin pozisyonları ve hızları, avlarına ulaşmak için sürekli olarak güncellenir. Algoritma, benzer şekilde, çözüm adaylarının pozisyon ve hızlarını güncelleyerek, en iyi çözümü bulmaya çalışır
   * 1. SCSO Algoritmasının Matematiksel Modellemesi

SCSO algoritması, aşağıdaki adımlarla çalışır:

1. ***Başlangııç :*** Rastgele çözümler (kum kedileri) oluşturulur ve başlangıç pozisyonları belirlenir.

metin, ekran görüntüsü, çizgi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Fig 4**. SCSO'nun başlangıç ​​ve tanımlama aşamasındaki çalışma mekanizması

Şekil 4’ te görüldüğü üzere*d* boyutlu bir optimizasyon probleminde, kum kedisi  problemin çözümünü temsil eden  1 × *d'* lik birdizidir*.* Değişken değerlerinin her biri ( *x1 , x 2 , …, x d* ) bir düğümü temsil eder. Burada her *x,* alt ve üst sınırlar arasında yer almalıdır (∀ *x i*  ∈ [alt, üst]). SCSO algoritmasını başlatmak için öncelikle problemin büyüklüğüne ( *N*pop  ×  *N d* ), ( *pop*  = 1, …, *n* ) göre kum kedisi popülasyonundan bir aday matris oluşturulur.

1. ***Uygunluk Değerlendirmesi :*** Her çözüm adayının uygunluk değeri hesaplanır. Görselde kum kedisi popülasyonunu ve fitness (uygunluk) fonksiyonunu matematiksel ifadelerini görmekteyiz. Burada uygunluk fonksiyonunun değerlendirilmesi ile uygunluk maliyeti elde edilir.
2. ***Avı Arama ( Keşif) :*** Kum kedileri, keşif ve saldırı fazlarını kullanarak çözüm alanını tarar ve en iyi çözüme ulaşmaya çalışır.

**metin, yazı tipi, beyaz, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

****

****

Burada;

* + : kum kedilerinin algılama frekansı (2 ile 0 arasında doğrusal olarak azalır.
  +  : kum kedilerinin işitme özelliklerinden esinlenildiğinden 2 olarak varsayılır.
  +  : Keşif ve sömürü aşamaları arasındaki geçişi kontrol eden parametre
  +  : her kedinin sergilediği hassasiyet aralığı
  +  : geçerli yineleme
  +  : maksimum yineleme

Her bir kum kedisi, en ideal pozisyonuna göre konumunu ayarlar. Burada mevcut konum ve her bir kedinin hassasiyet aralığı olmak üzere



her bir kum kedisi bu denklem ile belirlenen alternatif av alanlarını ((t+1). iterasyondaki pozisyonunu) tanımlama yeteneğine sahiptirler.

1. **Avına Saldırma (Sömürü) :** SCSO’nun saldırı aşamasını matematiksel olarak modellemek için kum kedisinin en iyi konumu (en iyi çözüm) ve kum kedisinin mevcut konumu arasındaki mesafe hesaplanır.Kum kedisi hassasiyet aralığının bir daire olduğu varsayılır. Böylece kum kedisinin hareket yönü daire üzerindeki rastgele bir açı ile belirlenir. Bu açı Rulet Tekerleği yöntemi ile belirlenir.

****

****

Burada;

* + : rastgele pozisyonu gösterir ve bu pozisyona dahil olan kedilerin ava yakın olmasını sağlar.

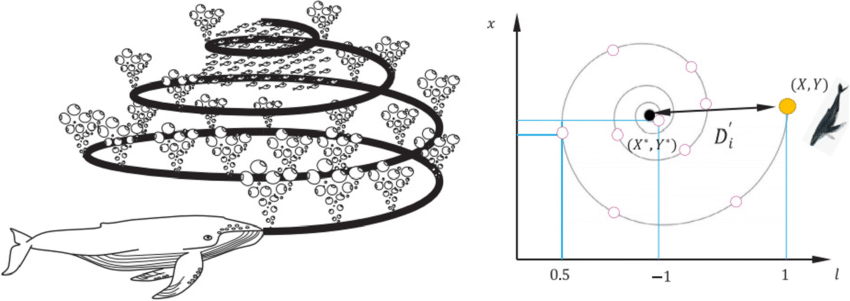
1. ***Keşif ve Geliştirme* :** Aşağıdaki denklem her bir kum kedisi için keşif ve sömürü aşamalarında konum güncellemesini gösterir. R , r G'nin doğrusal olarak yinelemeler boyunca 2'den 0'a düşürüldüğü [− 2 r G , 2 r G ] aralığındaki rastgele bir değerdir . R'nin rastgele değerleri [− 1, 1] aralığında olduğunda, bir kum kedisinin bir sonraki konumu, mevcut konumu ile av konumu arasındaki herhangi bir konumda olabilir. Burada |R| miktarı 1'den azsa, kum kedileri av saldırısına yönlendirilir. Tersine, |R| 1'den büyükse, kedilere tüm bölge üzerinde yeni bir potansiyel çözüm belirleme sorumluluğu atanır.

**metin, yazı tipi, beyaz, el yazısı içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* 1. **BALİNA OPTİMİZASYONU ALGORİTMASI (WHALE OPTİMİZATİON ALGORİTHM-WOA)**

Balina Optimizasyonu Algoritması (Whale Optimization Algorithm - WOA), 2016 yılında Seyedali Mirjalili ve Andrew Lewis tarafından geliştirilen, doğadan esinlenilmiş bir optimizasyon algoritmasıdır. Bu algoritma, kambur balinaların avlanma davranışlarını taklit eder. Kambur balinalar, balık sürülerini spiral hareketlerle sarmalayarak avlarını yakalarlar. WOA, bu davranışları modelleyerek, global optimizasyon problemlerini çözmek için kullanılır.



**Fig 5.** WOA Şematik Gösterim

* + 1. Algoritmanın Temel Bileşenleri

1. **Sarmalama Davranışı**:

* Kambur balinaların spiral hareketler yaparak avlarını sarmalaması, algoritmanın temel davranışlarından biridir. Bu davranış, balinaların avlarını çevreleyip yakalamalarını sağlar.

1. **Balina Davranışları**:

* **Avın Yerinin Güncellenmesi**: Balinalar, avlarının yerini tahmin ederek ve hareket ederek avlarını yakalarlar. Algoritma, çözüm adaylarının pozisyonlarını bu tahminlere göre günceller.
* **Kaçış Stratejileri**: Balinalar, avlarını yakalamak için çeşitli stratejiler kullanır. Algoritma, bu stratejileri taklit ederek, arama alanını etkili bir şekilde tarar.

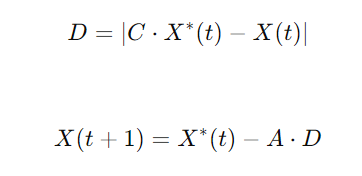
1. **Uygunluk (Fitness) Değeri**:

* Her çözüm adayı, belirli bir uygunluk değeri ile değerlendirilir. Bu değer, çözümün ne kadar iyi olduğunu gösterir ve algoritmanın amacına ulaşmasına yardımcı olur.
  + 1. WOA Algoritmasının Matematiksel Modellemesi

WOA algoritması, aşağıdaki adımlarla çalışır:

1. **Başlatma**: Rastgele çözümler (balinalar) oluşturulur ve başlangıç pozisyonları belirlenir.
2. **Uygunluk Değerlendirmesi**: Her çözüm adayının uygunluk değeri hesaplanır.
3. **Pozisyon Güncellemeleri**: Balinaların pozisyonları, sarmalama ve kaçış stratejilerine göre güncellenir.

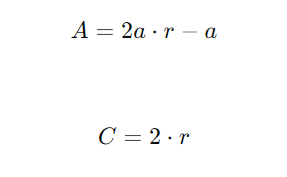
#### **Sarmalama Davranışı:** WOA, balinaların avlarını sarmalama davranışını aşağıdaki denklemlerle modeller:



Buradaki matematik gösterimler

* : Av ile balina arasındaki mesafeyi temsil eden vektör
* : Şu ana kadar elde edilen en iyi çözümün pozisyonu.
* : Balinanın t –inci iterasyonu
* ve : katsayı vektörleri
* : mevcut iterasyon

A ve C şu şekilde hesaplanır;



α : Lineer olarak 2’den 0 ‘ a doğru azalan bir skaler

r: [0,1] aralığında rastgele bir sayı.

#### **Spiral Güncelleme Pozisyonu:**

WOA, balinaların spiral hareketlerini aşağıdaki denklemlerle modeller:



Burada:

* D′: Balina ve av arasındaki mesafe.
* b: Spiral şeklinin sabit bir değeri.
* l: [-1, 1] aralığında rastgele bir sayı.

1. **Döngü**: Adımlar 2-3, belirli bir durma kriterine ulaşılana kadar tekrar edilir (örneğin, belirli bir iterasyon sayısına ulaşmak veya uygunluk değerinin belirli bir eşiğin altına düşmesi).

WOA algoritması, kambur balinaların sarmalama ve spiral hareketlerini modelleyerek, optimize edilecek problemi en iyi çözüme ulaştırmak için çalışır. Örneğin, üretim optimizasyonunda Balina Optimizasyonu algoritması, üretim süreçlerindeki parametreleri (örneğin, hammadde kullanımı, işçi sayısı, üretim hızı vb.) optimize etmek için çözümler üretebilir ve bu çözümleri daha sonra iyileştirmek üzere detaylı olarak işleyebilir. Bu şekilde, şirketin maliyetleri minimize edilirken verimliliği artırmasına yardımcı olabilir. Karmaşık ve çok boyutlu optimizasyon problemlerini çözmek için adaptif bir yaklaşım sunar.

ÜÇÜNCÜ BÖLÜM

**3. DENEYSEL ÇALIŞMA**

**3.1. Giriş (Introduction)**

**Deneyin Amacı:** Bu çalışmanın amacı, yol planlama problemlerini çözmek için Grey Wolf Optimization (GWO), Ant Colony Optimization (ACO), Sand Cat Swarm Optimization (SCSO) ve Whale Optimization Algorithm (WOA) gibi doğal metaheuristik algoritmalarını kullanmanın etkinliğini değerlendirmektir. Bu algoritmaların avantajlarını ve dezavantajlarını anlamak ve performanslarını karşılaştırmak hedeflenmektedir.

**3.2. Deneysel Kurulum (Experimental Setup)**

**Deney Ortamı:** Bu deneyler, Python programlama dili kullanılarak NumPy ve Matplotlib gibi kütüphaneler ile gerçekleştirilmiştir. Bilgisayar ortamı, Intel Core i7 işlemci ve 16 GB RAM ile donatılmış bir bilgisayarda yürütülmüştür.

**Problem Tanımı:** Yol planlama problemi, bir başlangıç noktasından bir hedef noktaya en uygun veya en kısa yolu bulmayı hedefler. Bu çalışmada, 2-boyutlu uzayda başlangıç ve hedef noktalar arasındaki en kısa yolun bulunması problemi ele alınmıştır.

**Senaryo Tanımı:**

* **Başlangıç ve Bitiş Noktaları:** Başlangıç noktası (0, 0) ve hedef noktası (10, 10) olarak belirlenmiştir.
* **Engeller ve Ortamın Boyutları:** 2-boyutlu uzay içerisinde engel bulunmamaktadır. Arazi sınırları (0, 0) ile (10, 10) arasında belirlenmiştir. Hiçbir algoritmanın yolunda engel bulunmamaktadır.

**Metrikler:** Performans değerlendirmesi için kullanılan metrikler şunlardır:

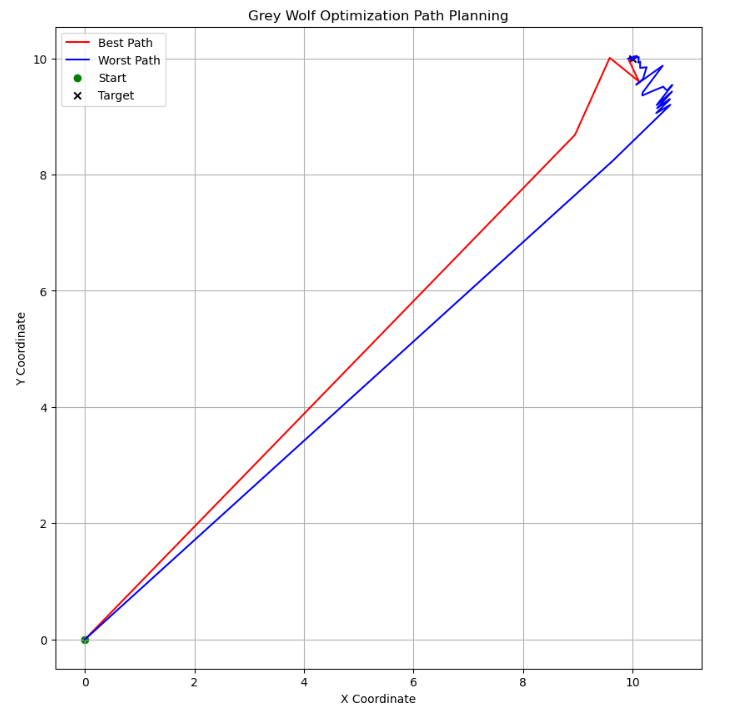
* Yolun Uzunluğu
* Hesaplama Süresi
* Konverjans Hızı

**3.3. Algoritmaların Uygulanması (Application of Algorithms)**

**3.3.1 GWO (Gri Kurtlar Optimizasyonu Algoritması)**

**GWO Algoritması için Parametreler**

1. **Number of Iterations (num\_iterations)**:
   * **Açıklama**: Algoritmanın çalışacağı iterasyon sayısını belirtir.
   * **Kullanılan Değer**: 100
   * **Etki**: Yüksek iterasyon sayısı, algoritmanın daha iyi bir çözüm bulma olasılığını artırır, ancak çalışma süresini uzatır.
2. **Number of Wolves (num\_wolves)**:
   * **Açıklama**: Sürüyü oluşturan kurt sayısını belirtir.
   * **Kullanılan Değer**: 20
   * **Etki**: Daha fazla kurt, daha geniş bir arama alanının keşfedilmesini sağlar, ancak hesaplama maliyetini de artırır.
3. **Dimension (dimension)**:
   * **Açıklama**: Çözüm uzayının boyutunu ifade eder. Bu çalışmada, 2D yol planlama için kullanılmıştır.
   * **Kullanılan Değer**: 2
   * **Etki**: Boyut sayısı arttıkça, çözüm uzayı daha karmaşık hale gelir.

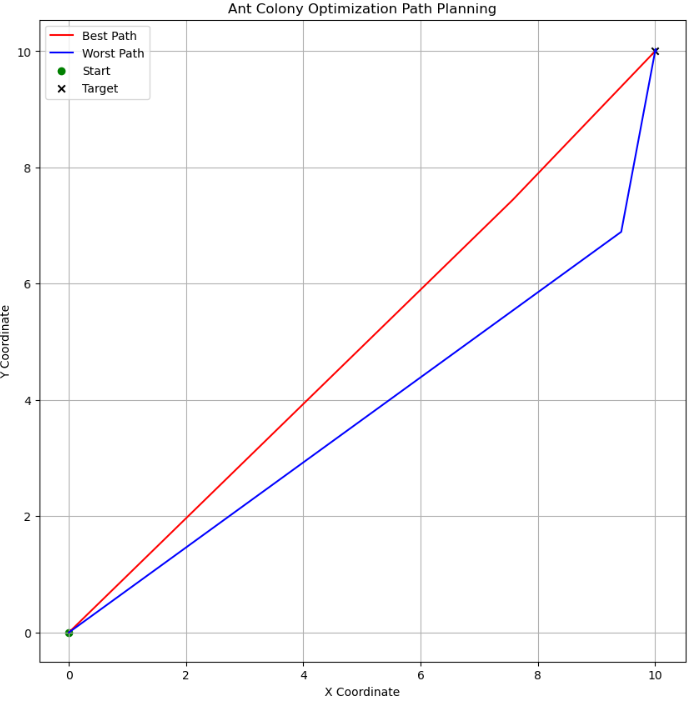


**Fig 5.** GWO worst (mavi) ve best(kırmzı) path plan resmi

**3.3.2. ACO (Karınca Kolonisi Optimizasyonu Algoritması )**

**ACO Algoritması için Parametreler**

1. **Number of Ants (num\_ants)**:
   * **Açıklama**: Kolonideki karınca sayısını belirtir.
   * **Kullanılan Değer**: 20
   * **Etki**: Daha fazla karınca, daha geniş bir arama alanının keşfedilmesini sağlar, ancak hesaplama maliyetini de artırır.
2. **Number of Iterations (num\_iterations)**:
   * **Açıklama**: Algoritmanın çalışacağı iterasyon sayısını belirler.
   * **Kullanılan Değer**: 100
   * **Etki**: Yüksek iterasyon sayısı, algoritmanın daha iyi bir çözüm bulma olasılığını artırır, ancak çalışma süresini uzatır.
3. **Dimension (dimension)**:
   * **Açıklama**: Çözüm uzayının boyutunu ifade eder. Bu çalışmada, 2D yol planlama için kullanılmıştır.
   * **Kullanılan Değer**: 2
   * **Etki**: Boyut sayısı arttıkça, çözüm uzayı daha karmaşık hale gelir.
4. **Alpha (alpha)**:
   * **Açıklama**: Feromon izlerinin yol seçimi üzerindeki etkisini belirler.
   * **Kullanılan Değer**: 1.0
   * **Etki**: Yüksek alpha değeri, karıncaların feromon izlerini daha fazla takip etmesine neden olur.
5. **Beta (beta)**:
   * **Açıklama**: Mesafe bilgisinin yol seçimi üzerindeki etkisini belirler.
   * **Kullanılan Değer**: 2.0
   * **Etki**: Yüksek beta değeri, karıncaların daha kısa yolları tercih etmesini sağlar.
6. **Evaporation Rate (evaporation\_rate)**:
   * **Açıklama**: Feromon izlerinin buharlaşma oranını belirler.
   * **Kullanılan Değer**: 0.1
   * **Etki**: Daha düşük bir buharlaşma oranı, eski yolların daha uzun süre hatırlanmasına neden olur.
7. **Constant for Pheromone Update (Q)**:
   * **Açıklama**: Feromon izlerinin güncellenmesinde kullanılan sabit değer.
   * **Kullanılan Değer**: 1.0
   * **Etki**: Yüksek Q değeri, karıncaların bıraktığı feromon miktarını artırır ve bu, gelecek iterasyonlarda o yolun tercih edilme olasılığını yükseltir.

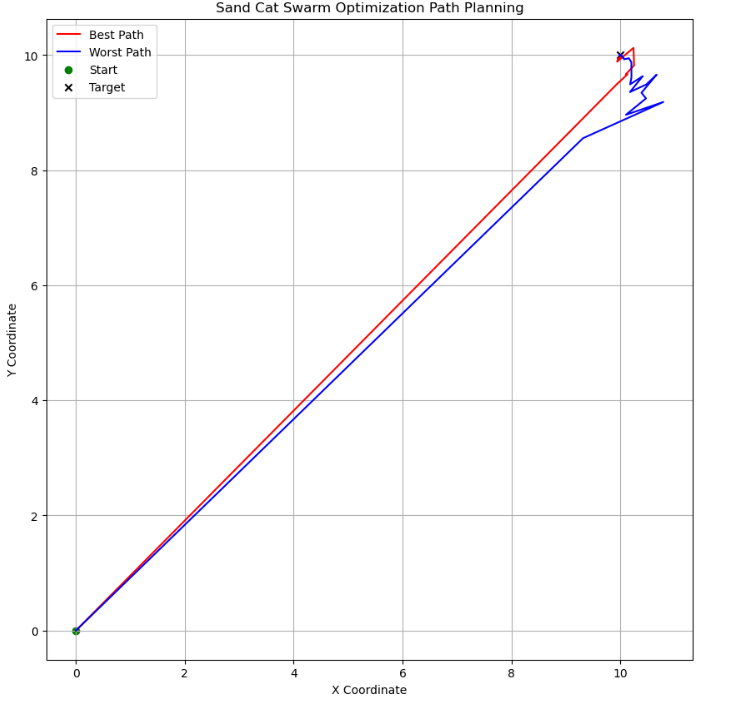


**Fig 6.** ACO worst (mavi) ve best(kırmzı) path plan resmi

**3.3.3. SCSO (Kum Kedisi Sürü Optimizasyonu Algoritması)**

**SCSO Algoritması için Parametreler**

1. **Number of Cats (num\_cats)**:
   * **Açıklama**: Algoritmadaki kedilerin (nüfusun) sayısını belirtir.
   * **Kullanılan Değer**: 20
   * **Etki**: Daha fazla kedi, arama alanının daha geniş bir şekilde keşfedilmesini sağlar, ancak hesaplama maliyetini de artırır.
2. **Number of Iterations (num\_iterations)**:
   * **Açıklama**: Algoritmanın çalışacağı iterasyon sayısını belirtir.
   * **Kullanılan Değer**: 100
   * **Etki**: Daha yüksek iterasyon sayısı, daha iyi bir çözüm bulma olasılığını artırır, ancak çalışma süresini uzatır.
3. **Dimension (dimension)**:
   * **Açıklama**: Çözüm uzayının boyutunu ifade eder. Bu çalışmada, 2D yol planlama için kullanılmıştır.
   * **Kullanılan Değer**: 2
   * **Etki**: Boyut sayısı arttıkça, çözüm uzayı daha karmaşık hale gelir.
4. **Exploration Factor (alpha)**:
   * **Açıklama**: Kedilerin avı keşfetme yeteneğini belirler.
   * **Kullanılan Değer**: 0.5
   * **Etki**: Daha yüksek bir alpha değeri, kedilerin avı daha geniş bir alanda aramasını sağlar.
5. **Prey Detection Influence (beta)**:
   * **Açıklama**: Kedilerin avı tespit etme yeteneğini etkiler.
   * **Kullanılan Değer**: 0.3
   * **Etki**: Daha yüksek bir beta değeri, kedilerin avı daha hassas bir şekilde tespit etmesine olanak tanır.

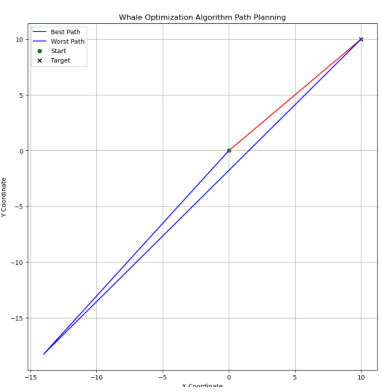
****

**Fig 7.** SCSO worst (mavi) ve best(kırmzı) path plan resmi

**3.3.4. WOA (Balina Optimizasyonu Algoritması)**

**WOA Algoritması için Parametreler**

1. **Number of Whales (num\_whales)**:
   * **Açıklama**: Algoritmadaki balinaların (nüfusun) sayısını belirtir.
   * **Kullanılan Değer**: 20
   * **Etki**: Daha fazla balina, arama alanının daha geniş bir şekilde keşfedilmesini sağlar, ancak hesaplama maliyetini de artırır.
2. **Number of Iterations (num\_iterations)**:
   * **Açıklama**: Algoritmanın çalışacağı iterasyon sayısını belirtir.
   * **Kullanılan Değer**: 100
   * **Etki**: Daha yüksek iterasyon sayısı, daha iyi bir çözüm bulma olasılığını artırır, ancak çalışma süresini uzatır.
3. **Dimension (dimension)**:
   * **Açıklama**: Çözüm uzayının boyutunu ifade eder. Bu çalışmada, 2D yol planlama için kullanılmıştır.
   * **Kullanılan Değer**: 2
   * **Etki**: Boyut sayısı arttıkça, çözüm uzayı daha karmaşık hale gelir.
4. **Coefficient aaa**:
   * **Açıklama**: Algoritmanın keşif ve sömürü dengesini ayarlayan katsayı.
   * **Kullanılan Değer**: 2.0
   * **Etki**: Keşif aşamasından sömürü aşamasına geçişi belirler. İterasyonlar ilerledikçe azalarak balinaların avlanma alanını daraltır.
5. **Spiral Coefficient bbb**:
   * **Açıklama**: Balinaların spiral hareketlerinin yoğunluğunu belirler.
   * **Kullanılan Değer**: 1.0
   * **Etki**: Spiral yolun kıvrılma derecesini kontrol eder.
6. **Lower Bound (l\_bound) ve Upper Bound (u\_bound)**:
   * **Açıklama**: Arama uzayının alt ve üst sınırlarını belirler.
   * **Kullanılan Değerler**: 0 ve 10
   * **Etki**: Balinaların arama alanı içinde kalmasını sağlar.



**Fig 8.** WOA worst (mavi) ve best(kırmzı) path plan resmi

### **DÖRDÜNCÜ BÖLÜM**

### **SONUÇLAR VE DEĞERLENDİRME (RESULTS AND EVALUATİON)**

Bu bölümde, dört farklı doğal metaheuristik algoritmanın (GWO, ACO, SCSO, WOA) yol planlama problemini çözmedeki performanslarını değerlendiriyoruz. Değerlendirme, algoritmaların en iyi, en kötü, ortalama ve standart sapma sonuçları üzerinden yapılmıştır. 30 tekrar çalıştırma sonucu Tablo 1 elde edilmiştir. Ayrıca, Mann-Whitney U testi kullanılarak algoritmalar arasındaki istatistiksel farklılıklar incelenmiştir.

#### **4.1. Algoritmaların Performans Sonuçları**

Aşağıdaki tabloda her algoritmanın performans sonuçları özetlenmiştir:

**Tablo 1.** Yapılan çalışmadaki algoritmaların best, worst, mean std sonuçları

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Algoritma** | **Best**  **Result** | **Worst**  **Result** | **Mean**  **Result** | **STD**  **Result** | | **GWO** | 0.001786 | 0.013557 | 0.007231 | 0.003621 | | **ACO** | 0.000197 | 0.011895 | 0.002481 | 0.002192 | | **SCSO** | 0.002878 | 0.019408 | 0.010164 | 0.004080 | | **WOA** | 0.013376 | 1.524067 | 0.415472 | 0.343074 | |  |  |  |  |

#### **4.2. Performans Değerlendirmesi**

Bu sonuçlar, farklı optimizasyon algoritmalarının path planning problemi için performansını göstermektedir. İlk olarak, en iyi performans gösteren algoritma ACO'dur. ACO, ortalama sonucu en düşük olan ve standart sapması da oldukça düşük olan bir algoritmadır. Bu, genelde başlangıç ve hedef arasındaki en kısa yolu bulma konusunda tutarlı ve etkili olduğunu gösterir.

Diğer yandan, en kötü performans gösteren algoritma WOA'dır. WOA, en yüksek standart sapmaya sahip olup, en kötü ve en iyi sonuç arasında büyük bir fark bulunmaktadır. Bu durum, WOA'nın bazı durumlarda iyi sonuçlar elde edebilse de genel olarak daha değişken ve tutarsız olduğunu işaret eder.

GWO ve SCSO ise orta düzeyde performans sergilemektedir. GWO, düşük standart sapmasıyla birlikte ortalama bir performans gösterirken, SCSO biraz daha yüksek standart sapmaya sahiptir ancak genel olarak kabul edilebilir bir ortalama sonuç verir.

Sonuç olarak, ACO genellikle en güvenilir ve tutarlı performansı sunan algoritma olarak öne çıkmaktadır, özellikle path planning gibi hassas ve doğruluğun önemli olduğu uygulamalarda tercih edilebilir.

#### **4.3. Mann-Whitney U Testi Sonuçları**

Mann-Whitney U testi, iki algoritma arasındaki sonuçların istatistiksel olarak anlamlı olup olmadığını belirlemek için kullanılmıştır. Aşağıdaki tabloda p ve h değerleri verilmiştir:

**Tablo 2.** Mann-Whitney ikili testi ile Algoritmaların p value sonuçları

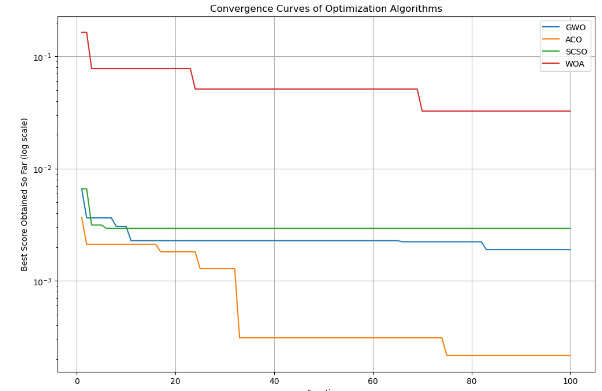
| **Algoritma 1** | **Algoritma 2** | **p Değeri** | **h Değeri** |
| --- | --- | --- | --- |
| GWO | ACO | 0.48571 | 0 |
| GWO | SCSO | 0.68571 | 0 |
| GWO | WOA | 0.05714 | 0 |
| ACO | SCSO | 0.20000 | 0 |
| ACO | WOA | 0.02857 | 1 |
| SCSO | WOA | 0.05714 | 0 |

**Mann-Whitney U Testi Kullanım Nedeni**: Mann-Whitney U testi, iki bağımsız örnek arasındaki medyanların karşılaştırılması için kullanılan non-parametrik bir testtir. Bu test, örneklerin normal dağılıma uymadığı durumlarda t-testine alternatif olarak kullanılır. Bu çalışmada, algoritmaların performans sonuçlarının normal dağılıma uygun olup olmadığı bilinmediği ve sonuçların dağılımı hakkında kesin bir varsayım yapılmadığı için Mann-Whitney U testi tercih edilmiştir. Testin uygulanmasının başlıca nedenleri şunlardır:

* **Normal Dağılım Gereksinimi**: Mann-Whitney U testi, t-testi gibi parametrik testlerin aksine, verilerin normal dağılıma sahip olmasını gerektirmez. Bu nedenle, verilerin dağılımı hakkında kesin bir bilgiye sahip olunmadığında daha güvenilir sonuçlar sağlar.
* **Örnek Büyüklüğü**: Örnek büyüklüklerinin küçük olduğu durumlarda Mann-Whitney U testi, parametrik testlere göre daha uygun sonuçlar verir.
* **Medyan Karşılaştırması**: Test, iki bağımsız grubun medyanlarının eşit olup olmadığını test eder. Bu çalışmada, algoritmaların performans medyanlarının karşılaştırılması amaçlandığından uygun bir testtir.
* **Ölçek Dışı Farklılıklar**: Performans sonuçlarının ölçekleri arasında farklılıklar olduğunda, Mann-Whitney U testi bu farklılıkları göz ardı ederek medyanlara odaklanır.

#### **4.4. Grafiksel Değerlendirmeler**

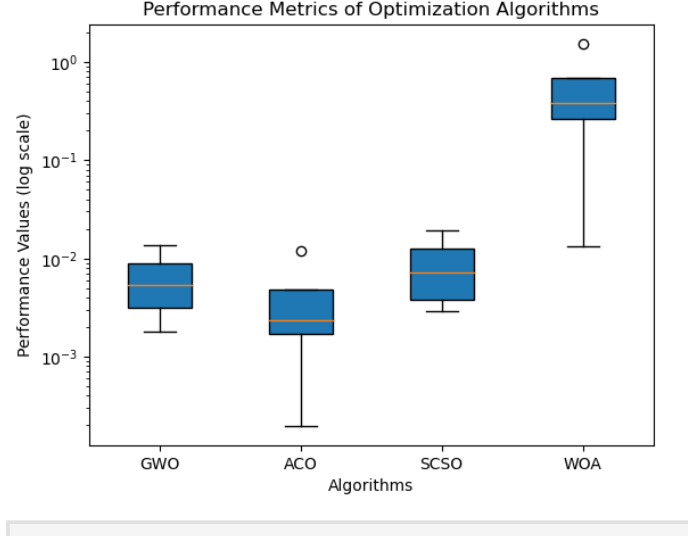
**4.4.1. Convergence Curve (Yakınsama Eğrisi )**

****

**Grafik 1.** Yakınsama Eğrisi

Bu grafik, GWO, ACO, SCSO ve WOA algoritmalarının bir yol planlama problemi üzerindeki performanslarını karşılaştırıyor. Tablo 1 de ayrıntılı olarak incelenmesinin haricinde grafik best resultlar için oluşturulmuştur değerler küçük olduğu için log olarak x ekseni gösterilmiştir. Aynı zamanda 100 iterasyon y eksenin de gözükmedir bu grafikten anlamalıyız ki en iyisi aco ve yaklaşık 70 . iterasyona kadar iyileştirme göstermiş diğerleri 80 iterasyona kadar iyileştirme göstermiş fakat aco kadar başarılı sonuç vermemiş. En kötü algoritmanında woa olduğunu fakat 1. İterasyondan 100. İterasyona kadar iyileştirme göstermeye çalıştığını gözlemliyorum.

**4.1.2. Boxplot Grafiği**

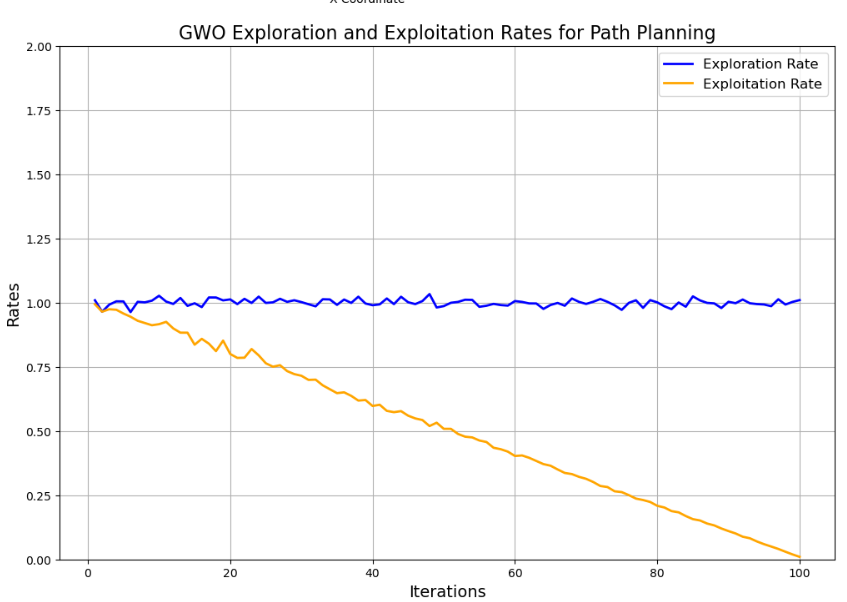
****

**Grafik 2.** Boxplot Grafiği

Grafik üzerinden yapılan analizde, GWO algoritmasının rota planlama problemleri için son derece etkili bir optimizasyon yöntemi olduğu gözlemlenmiştir. GWO, en kısa yol bulma konusunda diğer algoritmaları geride bırakarak en iyi çözüme en hızlı şekilde ulaşmıştır. ACO ve SCSO algoritmaları da sağlam performans sergilemiş, GWO'ya yakın sonuçlar elde etmiştir. Ancak WOA algoritması, diğerlerine kıyasla daha uzun süre en iyi çözüme ulaşamamıştır, bu da onun rota planlama problemleri için en uygun seçenek olmadığını göstermektedir. Her bir algoritmanın performansı, problem karmaşıklığına, algoritma parametrelerine ve veri setine bağlı olarak değişebilir, bu nedenle en uygun algoritmanın belirlenmesi için farklı seçeneklerin test edilmesi önemlidir.

**4.1.3. Exploration and Exploitation Grafiği (Keşif ve Sömürü)**

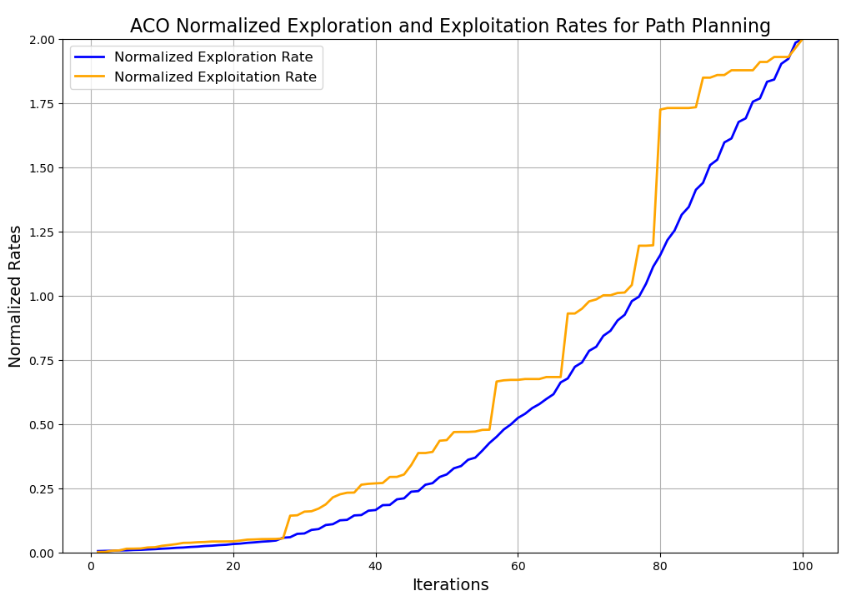
**1. GWO**

****

**Grafik 3.** GWO Keşif Sömürü Grafiği

Bu grafik, GWO (Grey Wolf Optimizer) algoritmasının pathplanning (yol planlama) problemleri için keşif (exploration) ve sömürü (exploitation) oranlarını göstermektedir. Yatay eksen iterasyon sayısını, dikey eksen ise oranları temsil etmektedir.Grafikten de anlaşılacağı üzere, mavi çizgiyle gösterilen keşif oranı (exploration rate) iterasyonlar boyunca oldukça sabit kalmakta ve yaklaşık olarak 1.0 civarında seyretmektedir. Bu, GWO algoritmasının keşif kapasitesini iterasyonlar boyunca koruduğunu ve yeni çözümler arayışını sürdürdüğünü göstermektedir.Turuncu çizgiyle gösterilen sömürü oranı (exploitation rate) ise iterasyonlar ilerledikçe düzenli olarak azalmaktadır. Başlangıçta 1.0 civarında olan sömürü oranı, iterasyonlar arttıkça azalmakta ve 100. iterasyonda 0'a yaklaşmaktadır. Bu, GWO algoritmasının başlangıçta mevcut en iyi çözümleri optimize etmeye odaklandığını, ancak zamanla daha fazla keşif yaparak yeni çözümler arayışına yöneldiğini göstermektedir.Bu davranış, GWO algoritmasının pathplanning problemlerinde dengeyi nasıl sağladığını anlamak için önemlidir. Algoritma, başlangıçta mevcut çözümleri iyileştirmeye odaklanarak hızlı bir performans artışı sağlarken, daha sonra daha iyi çözümler bulma potansiyelini artırmak için keşif oranını sabit tutarak yeni çözüm arayışını sürdürür. Bu strateji, GWO algoritmasının pathplanning problemlerinde etkili bir çözüm sunmasını sağlamaktadır.

2. ACO

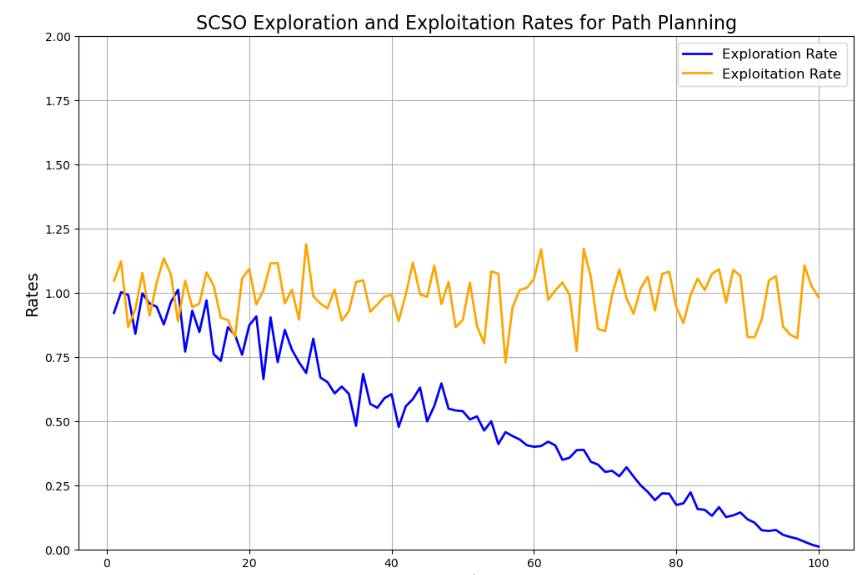


**Grafik 4.** ACO Keşif Sömürü Grafiği

Bu grafik, ACO (Ant Colony Optimization) algoritmasının pathplanning (yol planlama) problemleri için normalize edilmiş keşif (exploration) ve sömürü (exploitation) oranlarını göstermektedir. Yatay eksen iterasyon sayısını, dikey eksen ise normalize edilmiş oranları temsil etmektedir. Grafiğe göre, mavi çizgiyle gösterilen normalize edilmiş keşif oranı (normalized exploration rate) iterasyonlar boyunca düzenli olarak artmaktadır. Bu, ACO algoritmasının her iterasyonda daha fazla keşif yapma eğiliminde olduğunu ve yeni çözümler arayışını giderek artırdığını göstermektedir. Keşif oranının sürekli artması, algoritmanın çevresel bilgiyi sürekli olarak güncellediğini ve farklı çözüm yollarını araştırdığını işaret eder. Turuncu çizgiyle gösterilen normalize edilmiş sömürü oranı (normalized exploitation rate) ise dalgalı bir artış eğilimi göstermektedir. Başlangıçta düşük seviyelerde kalan sömürü oranı, belirli iterasyonlarda hızlı bir şekilde artış göstererek, mevcut en iyi çözümleri optimize etmeye yönelik ani atılımlar yapmaktadır. Bu dalgalanma, ACO algoritmasının belirli dönemlerde mevcut bilgiyi yoğun bir şekilde kullanarak mevcut çözümleri iyileştirmeye çalıştığını göstermektedir.

Bu davranış, ACO algoritmasının pathplanning problemlerinde nasıl çalıştığını anlamak için önemlidir. Algoritma, keşif oranını sürekli artırarak çevresel bilgiyi genişletirken, belirli aralıklarla sömürü oranını artırarak en iyi çözümleri optimize etmeye odaklanır. Bu çift yönlü strateji, ACO algoritmasının hem yeni çözümler arayışında hem de mevcut çözümleri iyileştirmede etkili olmasını sağlar. Özellikle karmaşık ve dinamik ortamlar için bu yaklaşım, daha iyi ve daha sağlam çözüm yolları bulunmasına yardımcı olabilir.

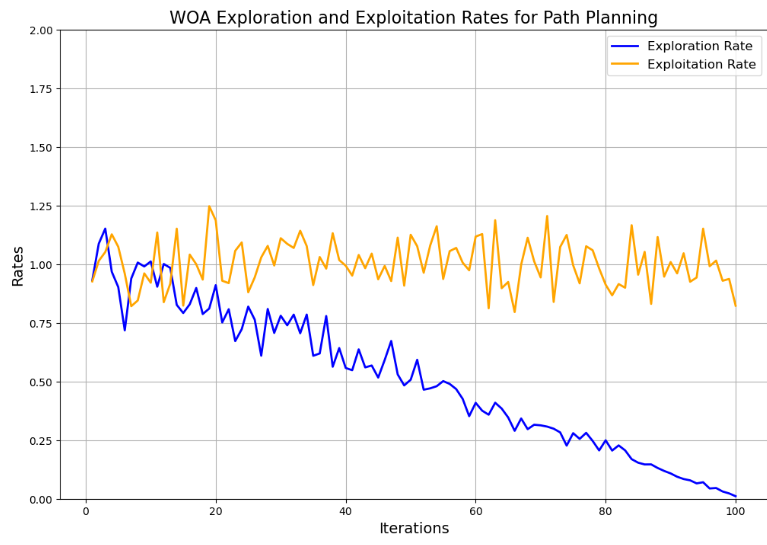
**3. SCSO**

****

**Grafik 5** SCSO Keşif Sömürü Grafiği

SCSO algoritması, pathplanning problemlerinde hem keşif hem de sömürü süreçlerini dengeli bir şekilde yönetmeye çalışmaktadır. Başlangıçta yüksek olan keşif oranı, algoritmanın farklı çözüm yollarını araştırmasına ve çevresel bilgiyi genişletmesine olanak tanırken, sabit kalan sömürü oranı mevcut en iyi çözümlerin optimize edilmesine odaklanmaktadır. Bu strateji, algoritmanın başlangıçta geniş bir çözüm aralığını araştırmasını ve daha sonra en iyi çözümleri optimize etmesini sağlamaktadır. Bu yaklaşım, pathplanning problemlerinde hem çeşitlilik hem de derinlemesine iyileştirme sağlar.

**4. WOA**

****

**Grafik 6 .**WOA Keşif ve Sömürü Grafiği

WOA algoritması, path planning problemlerinde keşif ve sömürü süreçlerinde keşif fazla olarak başlayıp zamanla azalmış aslında tam beklenildiği gibi sömürü ise baştan sona hep stabil kalmış scso da aynı grafik gözlenmekte fakat diğer değerlendirme sonuçlarına bakarsam woa keşif ve sömürüye rağmen başarılı bir sonuç elde edememiş sebebi spiral hareketten kaynaklı olabilir.

#### **SONUÇ VE YORUM**

Bu çalışmada, Path planning probleminde kullanılan dört farklı algoritmanın performans sonuçlarına dayanarak, çeşitli kriterler altında elde edilen sonuçları değerlendirebiliriz. Bu kriterler, en iyi sonuç, en kötü sonuç, ortalama sonuç ve standart sapma değerleridir. Algoritmalar arasında Gri Kurt Optimizasyonu (GWO), Karınca Kolonisi Optimizasyonu (ACO), Senkronize Sinüs Kozinüs Algoritması (SCSO) ve Balina Optimizasyonu Algoritması (WOA) bulunmaktadır.

**Gri Kurt Optimizasyonu (GWO)**: GWO algoritması, en iyi sonuç olarak 0.001786 değerini elde etmiş olup, oldukça düşük bir hata oranı göstermektedir. En kötü sonuç 0.013557 iken, ortalama sonuç 0.007231 olarak hesaplanmıştır. Standart sapma değeri ise 0.003621 olup, sonuçların tutarlılığını göstermektedir. GWO'nun genel performansı istikrarlı ve güvenilir olarak değerlendirilebilir.

**Karınca Kolonisi Optimizasyonu (ACO)**: ACO algoritması en iyi sonuç olarak 0.000197 değerine ulaşmış ve en düşük hata oranını göstermiştir. En kötü sonuç 0.011895 olup, ortalama sonuç 0.002481 olarak belirlenmiştir. Standart sapma değeri ise 0.002192 olup, sonuçların ACO algoritmasında da oldukça tutarlı olduğunu göstermektedir. ACO, en düşük hata oranına sahip olması nedeniyle oldukça etkili bir algoritma olarak öne çıkmaktadır.

**Senkronize Sinüs Kozinüs Algoritması (SCSO)**: SCSO algoritması en iyi sonuç olarak 0.002878 değerini elde etmiştir. En kötü sonuç 0.019408 iken, ortalama sonuç 0.010164 olarak hesaplanmıştır. Standart sapma değeri ise 0.004080 olup, SCSO'nun sonuçlarındaki dalgalanmayı göstermektedir. SCSO, diğer algoritmalara kıyasla daha yüksek bir hata oranına sahip olmasına rağmen, belirli senaryolarda etkili olabilir.

**Balina Optimizasyonu Algoritması (WOA)**: WOA algoritması, en iyi sonuç olarak 0.013376 değerine ulaşmış, ancak en kötü sonuç 1.524067 ile oldukça geniş bir hata aralığına sahip olmuştur. Ortalama sonuç 0.415472 olup, bu da diğer algoritmalara kıyasla daha yüksek bir hata oranı anlamına gelmektedir. Standart sapma değeri ise 0.343074 olup, sonuçların oldukça değişken olduğunu göstermektedir. WOA'nın performansı, diğer algoritmalara göre daha az tutarlı ve daha yüksek bir hata oranına sahip olmuştur.

Sonuç olarak, ACO algoritması en düşük hata oranını ve en tutarlı sonuçları elde etmiştir. GWO algoritması ise genel performansı ile istikrarlı bir seçenek olarak öne çıkmaktadır. SCSO ve WOA algoritmaları belirli senaryolarda etkili olabilirken, daha yüksek hata oranları ve tutarsızlıklar nedeniyle dikkatli değerlendirilmelidir. Path planning problemlerinde en iyi sonucu elde etmek için ACO ve GWO algoritmaları öncelikli olarak değerlendirilmelidir.

KAYNAKÇA

1. Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., & Feyzeau, P. (2013). “Path Planning: A 2013 Survey”. Presented at the 5th IESM Conference, October 2013, Rabat, Morocco. University of Valenciennes and Hainaut-Cambresis, LAMIH UMR 8201, 59313 France. EUROCOPTER, Aéroport International Marseille Provence, 13700 Marignane.
2. Seyyedabbasi, Amir, and Farzad Kiani. "Sand Cat Swarm Optimization: A Nature-Inspired Algorithm to Solve Global Optimization Problems." Springer-Verlag London Ltd., part of Springer Nature, 11 Apr. 2022.
3. Rezaei, H., Bozorg-Haddad, O., & Chu, X. “Grey Wolf Optimization (GWO) Algorithm.” In book : Advanced Optimization by Nature-Inspired Algorithms (Studies in Computational Intelligence, pp. 235-245). Springer. (2018) <https://doi.org/10.1007/978-981-10-5221-7_9>
4. İşcan, S., Kaplan, O., & Lokman, G. “Power loss and voltage stability optimization with meta-heuristic algorithms in power system.” Elektrik Elektronik Mühendisliği Bölümü, Teknoloji Fakültesi, Gazi Üniversitesi, Ankara, Türkiye. (2020). <https://doi.org/10.5505/pajes.2020.84152>
5. Şenaras, E., İnanç, Ş. "GSP Çözümü için Karınca Kolonisi Optimizasyonu." Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Ekonometri Bölümü ve Keles Meslek Yüksekokulu Bilgisayar Teknolojileri.
6. Söyler, H. ve Keskintürk, T. "Karınca kolonisi algoritması ile gezen satıcı probleminin çözümü." 8. Türkiye Ekonometri ve İstatistik Kongresiİnönü Üniversitesi, Malatya. 24-25 Mayıs 2007
7. Rezvanian, A., Vahidipour, S. M., & Sadollah, A. “An overview of ant colony optimization algorithms for dynamic optimization problems.” In book: Ant Colony Optimization - Recent Variants, Application and Perspectives [Working Title]. (2023) <https://doi.org/10.5772/intechopen.111839>
8. Li, Y., Yu, Q., & Du, Z. (2024). “Sand cat swarm optimization algorithm and its application integrating elite decentralization and crossbar strategy. Scientific Reports, 14, 8927. https://doi.org/10.1038/s41598-024-00000-x”