Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 32, S. 521-530, Aralık 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 32, pp. 521-530, December 2021

Research Article

Izgara Bazlı Yol Planlama için Matematik Tabanlı Metasezgisellerin Karşılaştırılması

Mustafa Yusuf Yıldırım[1*], Rüştü Akay[2]

[1*] Erciyes Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği Bölümü, Kayseri, Türkiye, (ORCID: 0000-0003-0302-8466), myyildirim@erciyes.edu.tr
[2] Erciyes Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği Bölümü, Kayseri, Türkiye (ORCID: 0000-0002-3585-3332), akay@erciyes.edu.tr

(International Conference on Design, Research and Development (RDCONF) 2021 – 15-18 December 2021)

(**DOI:** 10.31590/ejosat.1039899)

ATIF/REFERENCE: Yıldırım, M. Y., Akay, R. (2021). Izgara Bazlı Yol Planlama için Matematik Tabanlı Metasezgisellerin Karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*, (32), 521-530.

Öz

Robot navigasyonunun en önemli bileşenlerinden biri olan yol planlama son yıllarda da araştırmacılar tarafından kapsamlı bir şekilde incelenmekte ve bu problem için birçok farklı metasezgisel algoritma kullanılmaktadır. Bu çalışmada ızgara tipi bir ortamda bir mobil robotun küresel yol planlaması ele alınmış ve bu problem için farklı matematik tabanlı metasezgisel algoritmalarının etkileri incelenmiştir. Öncelikle ızgara tipinde ve farklı zorluk derecelerinde üç farklı ortam tasarlanmıştır. Ardından, son yıllarda geliştirilen farklı matematik tabanlı algoritmalar kullanılarak robotun ortamlardaki optimum yolları hesaplanmıştır. Çalışmada metasezgisel algoritma olarak stokastik fraktal arama (Stochastic Fractal Search, SFS), aritmetik optimizasyon algoritması (Arithmetic Optimization Algorithm, AOA) ve sinüs kosinüs algoritması (Sine Cosine Algorithm, SCA) kullanılmıştır. Bulgular değerlendirildiğinde SFS algoritmasının en kısa mesafe ve engelden kaçınma açısından diğer algoritmalara göre daha iyi sonuçlar verdiği gözlemlenmiştir.

Anahtar Kelimeler: Yol Planlama, Metasezgisel, Matematik Tabanlı Algoritmalar, Optimizasyon.

Comparison of Maths-Based Metaheuristics for Grid-Based Path Planning

Abstract

Path planning, one of the most important components of robot navigation, has been extensively studied by researchers in recent years and many different metaheuristic algorithms are used for this problem. In this study, the global path planning of a mobile robot in a grid-type environment is discussed and the effects of different maths-based metaheuristic algorithms for this problem are investigated. First, three different environments with grid type and different difficulty levels were designed. Then, the optimum paths of the robot in the environments were calculated by using different maths-based algorithms developed in recent years. Stochastic fractal search (SFS), arithmetic optimization algorithm (AOA) and sine cosine algorithm (SCA) were used as metaheuristic algorithms in the study. When the results were evaluated, it was observed that SFS algorithm has given better results than other algorithms in terms of shortest distance and obstacle avoidance.

Keywords: Path Planning, Metaheuristic, Maths-Based Algorithms, Optimization.

[*] Sorumlu Yazar: myyildirim@erciyes.edu.tr

1. Giriş

Robot navigasyonu, bir mobil robotun ortamı haritalaması, lokalizasyonu, hareket kontrolü ve yol planlaması bileşenlerinden meydana gelen önemli bir problemdir. Navigasyonda gerekli bileşenlerden biri olan yol planlama ise bir robotun bir başlangıç noktasından bir hedef noktaya optimum maliyetlerle ulaşması için gerekli hesaplamaların gerçekleştirilmesi sürecidir. Yol planlama, ortamların karakteristiklerine göre küresel yol planlama ve yerel yol planlama olarak iki sınıfta incelenmektedir (Muhammad ve ark., 2020). Küresel yol planlama problemi son yıllarda da araştırmacılar tarafından kapsamlı bir şekilde incelenmeye devam etmektedir (Zhong ve ark., 2020).

Bu çalışma için literatürde metasezgisel algoritmaların kullanıldığı güncel küresel yol planlama konulu çalışmalar incelenmiştir. Bu çalışmalar, metasezgisel algoritmalar arasından doğa tabanlı ve matematik tabanlı algoritmaları kullananlar olarak gruplandırılmıştır (Gabis ve ark., 2021). Doğa tabanlı algoritmaların kullanıldığı çalışmalar da ızgara tipi ve serbest uzay tipi ortamlar için sınıflandırılmıştır. Izgara tipi ortamların kullanıldığı doğa tabanlı algoritma konulu çalışmalar şu şekilde özetlenebilir: Liang ve ark. tavuk sürüsü optimizasyonu algoritmasında iyileştirmeler önermişlerdir. Parçacık sürü optimizasyonu ve geleneksel tavuk sürüsü optimizasyonu ile kıyasladıklarında, geliştirilen algoritmanın daha performanslı olduğu sonucuna ulaşmışlardır (Liang ve ark., 2020). Adamu ve ark. kartezyen uzayda nokta oluşturan özelleştirilmiş bir algoritma ve parçacık sürü optimizasyon algoritmasının hibrit olarak kullanıldığı bir yöntem önermişlerdir. Özelleştirilmiş algoritma iki ana nokta arasında bir takım noktalar oluşturmak için kullanılırken, parçacık sürü optimizasyon algoritması da bu noktalar arasındaki optimum yolu hesaplamaktadır. Yöntemin daha kısa sürede daha iyi sonuçların elde edildiği gözlemlemişlerdir (Adamu ve ark., 2019). Lamini ve ark. robotun yoldaki dönüş sayısını azaltan bir amaç fonksiyonu ve genetik algoritma için geliştirilmiş bir çaprazlama operatörü önermişlerdir. Geleneksel genetik algoritma ile kıyasladıklarında geliştirilen yöntemin daha performanslı olduğu sonucuna ulaşmışlardır (Lamini ve ark., 2018). Ajeil ve ark. karınca koloni algoritmasını iyileştirerek yaş tabanlı karınca koloni algoritmasını olarak ifade edilen bir algoritma önermişlerdir. Sonuç olarak geliştirilen algoritmanın parçacık sürü optimizasyonu, genetik algoritma gibi bazı metasezgisel algoritmalarla kıyasladıklarında daha iyi bir performans gözlemlemişlerdir (Ajeil ve ark., 2020). Dai ve ark. geleneksel karınca koloni algoritmasına A yıldız algoritmasını ve karınca koloni algoritmasının bir sürümü olan max-min sistemini entegre ederek iyileştirilmiş bir algoritma önermişlerdir. Yöntemin karınca koloni algoritmasının diğer sürümlerine kıyasla daha performanslı olduğunu gözlemlemişlerdir (Dai ve ark., 2019). Tuba ve ark. beyin fırtınası optimizasyon algoritmasına yerel arama sürecini entegre ederek yeni bir yöntem önermişlerdir. Parçacık sürü optimizasyonunun farklı sürümleri ile kıyasladıklarında geliştirilen yöntemin daha iyi sonuçlar ürettiği sonucuna ulaşmışlardır (Tuba ve ark., 2018). Akka ve Khaber karınca koloni algoritmasının yakınsamasını hızlandırmak için iyileştirmeler önermişler ve geleneksel karınca koloni algoritması ile kıyasladıklarında, geliştirilen algoritmanın daha performanslı olduğu sonucuna ulaşmışlardır (Akka & Khaber, 2018). Ali ve ark. karınca koloni algoritması ve A yıldız çok yönlü algoritma tabanlı bir yöntem önermişlerdir. Ayrıca yolun keskinliğini düzeltmek için Markov karar sürecini entegre etmişlerdir. Literatürdeki bazı çalışmalara kıyasla geliştirilen yöntemin daha iyi sonuçlar ürettiğini gözlemlemişlerdir (Ali ve ark., 2020). Luo ve ark. karınca koloni algoritmasını iyileştirerek yeni bir sürümünü önermişler. Simülasyonlar sonucunda literatürdeki karınca koloni algoritmasının sürümlerine kıyasla daha iyi sonuçlar ürettiği sonucuna ulaşmışlardır (Luo ve ark., 2020).

Serbest uzay tipi ortamların kullanıldığı doğa tabanlı algoritma konulu çalışmalar ise şu şekilde özetlenebilir: Wang ve ark. çok amaçlı parçacık sürü optimizasyon algoritmasını engebeli arazi ortamındaki küresel yol planlama probleminde uygulamışlardır. Yazarlar bu algoritmayı bastırılmamış sıralamalı genetik algoritma, Pareto gücü evrimsel algoritma gibi bazı çok amaçlı evrimsel algoritmalarla kıyaslamışlar ve kolay ortamlarda her zaman daha performanslı iken zor ortamlarda da diğer algoritmaların yerel minimuma takıldıklarında daha performanslı olduğu sonucuna ulaşmışlardır (Wang ve ark., 2018). Patle ve ark. genetik algoritmaya Sylvester eylemsizlik yasası ve matris simülasyonunu entegre ederek adaptif matris ikili kod tabanlı genetik algoritma olarak ifade edilen bir algoritma önermişlerdir. Sonuç olarak geliştirilen algoritmayı bulanık mantık, yapay sinir ağları ve karınca koloni algoritması ile kıyasladıklarında daha performanslı olduğunu gözlemlemişlerdir (Patle ve ark., 2018). Gul ve ark. gri kurt optimizasyon ve parçacık sürü optimizasyon algoritmalarının hibrit olarak kullanıldığı bir yöntem önermişler. Yazarlar, literatürdeki bazı hibrit ve iyileştirilmiş algoritmalara kıyasla daha iyi sonuçlar ürettiği sonucuna ulaşmışlardır (Gul ve ark., 2020). Hosseininejad ve Dadkhah guguk kuşu optimizasyon algoritmasının kullanıldığı ve özellik vektörünün optimize edildiği yeni bir yöntem önermişlerdir. Simülasyonlar sonucunda farklı ortam koşullarında geliştirilen yöntemin performanslı olduğunu gözlemlemişlerdir (Hosseininejad & Dadkhah, 2019). Elmi ve Efe hem mesafe ve planlanan yolun düzgünlüğü hem de algoritmanın çalışma süresini optimize edilmek için çekirge optimizasyon algoritması tabanlı çok amaçlı bir yöntem önermişlerdir. Parçacık sürü optimizasyonu ile kıyasladıklarında geliştirilen yöntemin daha performanslı olduğunu gözlemlemişlerdir (Elmi & Efe, 2018). Ajeil ve ark. parçacık sürü optimizasyonu ve yarasa algoritmalarının hibrit olarak kullanıldığı bir yöntem önermişlerdir. Yönteme bir yerel arama algoritmasını ve engel algılama özelliğini de entegre etmişler. Simülasyon sonuçları bakteri kolonisi, genetik algoritma gibi metasezgisel algoritmalara kıyasla daha iyi sonuçlar ürettiğini göstermektedir (Ajeil ve ark., 2020). Saraswathi ve ark. guguk kuşu arama ve yarasa algoritmalarının hibrit olarak kullanıldığı bir yöntem önermişlerdir. Yöntemin bu algoritmaların ayrı ayrı kullanıldığı simülasyonlara kıyasla daha az bir sürede sonuç alındığı sonucuna ulaşmışlardır (Saraswathi ve ark., 2018). Zhang ve ark. yalın kemik parçacık sürü optimizasyonu ve diferansiyel gelişim algoritmalarının hibrit olarak kullanıldığı çok amaçlı bir yöntem önermişlerdir. Bu yöntem mesafe, yolun düzgünlük derecesi ve güvenlik amaçları ile optimizasyon gerçekleştirmektedir. Simülasyonlar sonucunda geliştirilen yöntemin çok amaçlı parçacık sürü optimizasyonunun diğer sürümlerine kıyasla daha performanslı olduğu sonucuna ulaşmışlardır (Zhang ve ark., 2018). Dolicanin ve ark. beyin fırtınası algoritmasını, insansız hava araçlarının küresel yol planlama probleminde uygulamışlardır. Bu algoritmayı parçacık sürü optimizasyonu, genetik algoritma, diferansiyel gelişim gibi on bir farklı algoritma ile

kıyaslamışlardır. Sonuçlar beyin fırtınası algoritmasının hemen hemen tüm durumlarda daha iyi sonuçlar verdiğini göstermektedir (Dolicanin ve ark., 2018). Yıldırım ve Akay parçacık sürü optimizasyonu, yapay arı kolonisi ve genetik algoritmanın kullanıldığı kullanıcı tanımlı bir yazılım önermişler ve bu yazılımda farklı ortamlar için algoritmaların yol planlama problemindeki performanslarını karşılaştırmışlardır. Sonuç olarak yapay arı koloni algoritmasının daha iyi sonuçlar verdiğini gözlemlemişlerdir (Yıldırım & Akay, 2021). Yazarlar başka bir çalışmalarında ise yol planlama algoritmaların daha hızlı çalışması için metasezgisel ve kümeleme algoritmalarının hibrit olarak kullanıldığı bir yöntem önermişlerdir. Farklı metasezgisel ve kümeleme algoritmalarının test edildiği simülasyon sonuçları, en hızlı algoritmanın genetik algoritma, maliyet açısından en performanslı algoritmanın ise öğretme-öğrenme tabanlı optimizasyon ve yapay arı koloni algoritması olduğunu göstermektedir (Yıldırım & Akay, 2021).

Matematik tabanlı algoritma konulu çalışmalar şu şekilde özetlenebilir: Wang ve ark. aritmetik optimizasyon algoritmasını iyileştirerek adaptif paralel aritmetik optimizasyon algoritması olarak ifade edilen bir algoritma önermişlerdir. Bu algoritmayı önce benchmark fonksiyonları ve daha sonra da serbest uzay ortamındaki küresel yol planlama probleminde uygulamışlardır. Sonuç olarak geliştirilen algoritmayı parçacık sürü optimizasyonu, genetik algoritma gibi bazı metasezgisel algoritmalarla kıyasladıklarında kayda değer farklar gözlemlemişlerdir (Wang ve ark., 2021). Das, böbrek ilhamlı algoritma ve sinüs kosinüs algoritmasının hibrit olarak kullanıldığı bir yöntem önermiştir. Sinüs kosinüs algoritması, böbrek ilhamlı algoritmanın yakınsamasını hızlandırmak ve robotun dönüş sayısını azaltmak için entegre edilmiştir. Yazar geliştirilen yöntemi serbest uzay ortamındaki küresel yol planlama probleminde uygulamış ve geleneksel böbrek ilhamlı algoritma ile geleneksel sinüs kosinüs algoritmasına kıyasla daha iyi sonuçlar ürettiği sonucuna ulaşmıştır (Das, 2020). Li ve ark. stokastik fraktal arama algoritmasını, insansız hava araçlarının serbest uzay ortamındaki küresel yol planlama probleminde uygulamışlardır. Bunun sonucunda algoritmanın gerçek zamanlı performansının kayda değer olduğunu ve karmaşık ortamlarda da tatmin edici sonuçları üretebileceğini gözlemlemişlerdir (Li ve ark., 2018).

Matematik tabanlı algoritmaların kullanıldığı çalışmalarda ızgara tipi yerine serbest uzay tipi ortam tercih edilmiştir. Ayrıca literatürde küresel yol planlama problemi için matematik tabanlı algoritmaların kullanıldığı çalışmalar sayıca daha azdır ve bu algoritmaların karşılaştırmalı bir çalışmasına rastlanılamamaktadır. Literatürdeki bu boşluğu doldurmak için bu çalışmada ızgara tipi bir ortamda küresel yol planlama optimiasyonu için farklı matematik tabanlı algoritmaların performanslarının karşılaştırmalı incelenmesi amaçlanmıştır.

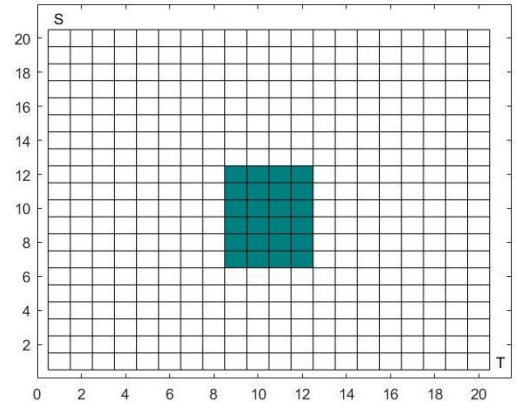
Çalışmanın bölümlerini şu şekilde sıralamak mümkündür: İkinci bölümde materyal ve yönteme, üçüncü bölümde bulgulara, dördüncü bölümde de sonuca yer verilmiştir.

2. Materyal ve Metot

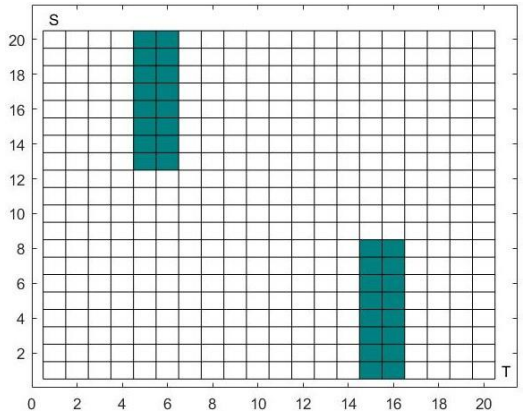
Bu bölümde öncelikle tasarlanan ortamlar ve amaç fonksiyonunun tanıtıldığı problem tanımı ifade edilmektedir. Daha sonra çalışmada kullanılan matematik tabanlı metasezgisel algoritmalardan kısaca bahsedilmektedir.

2.1. Problem Tanımı

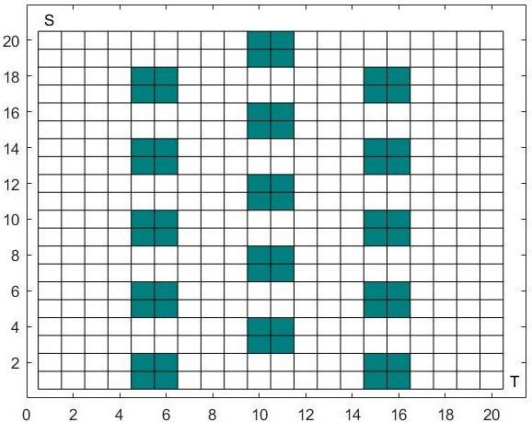
Çalışma için öncelikle farklı zorluk derecelerinde ızgara tipi üç ortam tasarlanmıştır. Tüm ortamlar 20 x 20 birim boyutunda olmakla birlikte, ortamların farklı engel karakteristikleri mevcuttur. Tasarlanan ortamlar Şekil 1'de gösterilmektedir. Burada engeller koyu yeşil renkte ızgara hücreleri ile temsil edilmekte, robotun başlangıç (sol üst hücre) ve hedef (sağ alt hücre) noktaları ise sırasıyla "S" ve "T" harfleri ile gösterilmektedir.



(a)



(b)



(c)

Şekil 1. Farklı zorluk derecelerinde ızgara tipi ortamlar: (a) Ortam 1, (b) Ortam 2, (c) Ortam 3

Bu ortamlardaki yollar başlangıç ve bitiş noktaları ile bir takım ara noktalar aracılığıyla oluşmaktadır. Bu ara noktalar da başlangıç ve bitiş noktaları gibi bir ızgara hücresi ile ilişkilendirilir. Yollar bu noktalar arasındaki yol parçalarından meydana gelir ve bu yol parçaları ızgara hücrelerinden bağımsızdır. Başlangıçta ara noktaların koordinatları belli bir değer aralığında rastgele oluşturulur ve her bir satırında tek bir ara noktanın koordinatlarının bulunduğu bir matris elde edilir. Bu matris Eşitlik 1'de gösterilen amaç fonksiyonunda değerlendirilir (Huang & Tsai, 2011).

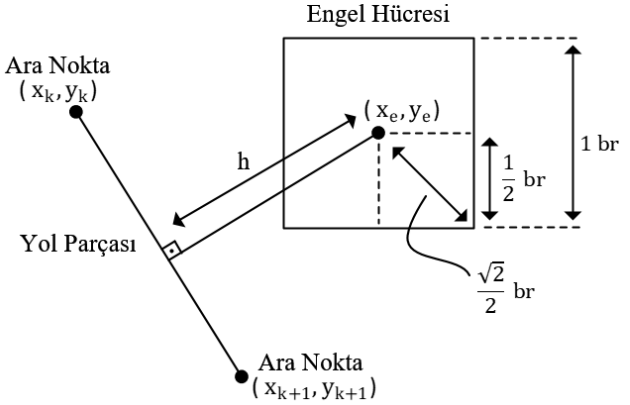
$$\min \ F = \sum_{i=1}^{P_{max}} (d_i + \alpha_i t) \tag{1}$$

Burada P_{max} yol parçalarının sayısı, d_i i. yol parçasında iki ardışık nokta arasındaki Öklit mesafesi fonksiyonu, t sabit bir sayı, α_i ise engel fonksiyonudur. d_i fonksiyonu Eşitlik 2'de ve α_i fonksiyonu Eşitlik 3'te gösterilmektedir.

$$d_i = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \tag{2}$$

$$\alpha_i = \begin{cases} 0 & ; \text{i. yol parçasıyla kesişen engeller yoksa} \\ \sum_{j=1}^{M} j & ; \text{i. yol parçasıyla kesişen engeller varsa} \end{cases} \tag{3}$$

Burada M i. yol parçasıyla kesişen engel sayısıdır. Bir yol parçasının engellerle kesişip kesişmediğini tespit etmek için özel bir kontrol stratejisi kullanılmaktadır. Bu strateji Şekil 2'de gösterilmektedir.



Şekil 2. Engel Kontrol Stratejisi

Bu stratejide i. yol parçasının her engel hücresine olan en kısa mesafeleri hesaplanır. Bu hesaplama için Eşitlik 4'te gösterilen bir doğrunun bir noktaya olan en kısa mesafe denklemi kullanılır.

$$h = \frac{|(x_{k+1} - x_k)(y_k - y_e) - (x_k - x_e)(y_{k+1} - y_k)|}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} \tag{4}$$

Bu h mesafesi kontrol değeri olan $\sqrt{2}/2$'den küçükse, ilgili yol parçası hesabı yapılan engeli kesmektedir. Kontrol değerinin $\sqrt{2}/2$ olarak belirlenmesinin sebebi, bir engel hücresi koordinatının engel çevresine olan en uzak mesafesinin esas alınması gerekliliğidir. Eşitlik 1'deki amaç fonksiyonu ile bir yol parçasının kesiştiği engel sayısı, toplamsal bir fonksiyonla mesafeye ek bir maliyet olarak yansımakta ve ayrıca t çarpanı ile bu maliyet daha da artırılmaktadır. Böylece çözümler farklı matematik tabanlı metasezgisel algoritmalar ile iyileştirilmektedir.

2.2. Metasezgisel Algoritmalar

Literatürde birçok optimizasyon problemi için farklı metasezgisel algoritmalar geliştirilmiştir. Bu algoritmalar Şekil 3'teki gibi sınıflandırılabilir.

Bu çalışmada matematik tabanlı metasezgisel algoritmalara odaklanılmış ve bu algoritmalardan stokastik fraktal arama algoritması (Salimi, 2021), aritmetik optimizasyon algoritması (Abualigah, 2021) ve sinüs kosinüs algoritması (Mirjalili, 2021) tercih edilmiştir. Bunlar matematikteki bir takım operatörlerle oluşturulan güncelleme denklemlerinin kullanıldığı popülasyon tabanlı algoritmalardır.

2.2.1. Stokastik Fraktal Arama Algoritması

Stokastik fraktal arama algoritması, Salimi tarafından 2015 yılında geliştirilen ve fraktal arama algoritmasındaki dezavantajları iyileştiren matematik tabanlı bir metasezgisel algoritmadır. Bu algoritma, her çözüm için alternatif çözümler üreten difüzyon süreci ve çözümlerin esas güncellendiği iki aşamalı güncelleme mekanizmasından oluşur. Difüzyon sürecinde fraktal aramada algoritmasındaki Levy yürüyüşü yerine tamamen Gauss yürüyüşünü kullanır. Algoritmada öncelikle başlangıç popülasyonunu rastgele oluşturulur. Popülasyondaki her bir çözüm difüzyon sürecinden geçer. Bu süreçte ilgili çözüm için iki farklı rastgele Gauss yürüyüşü ile yeni çözümler üretilir. Yürüyüş oranı parametresine göre bu iki yürüyüşten biri kullanılır. Bu yürüyüş denklemleri Eşitlik 5 ve 6'da gösterilmektedir.

$$GY_1 = \text{Gauss}(|BP|, \sigma) + (\varepsilon \times BP - \varepsilon' \times P_i) \tag{5}$$

$$GY_2 = \text{Gauss}(|P_i|, \sigma) \tag{6}$$

Burada ε ve ε' [0, 1] aralığında rassal sayılar, BP popülasyondaki en iyi çözüm, P_i i. çözüm ve rastgele yürüyüş için σ standart sapmadır. Standart sapma Eşitlik 7'deki gibi belirlenir.

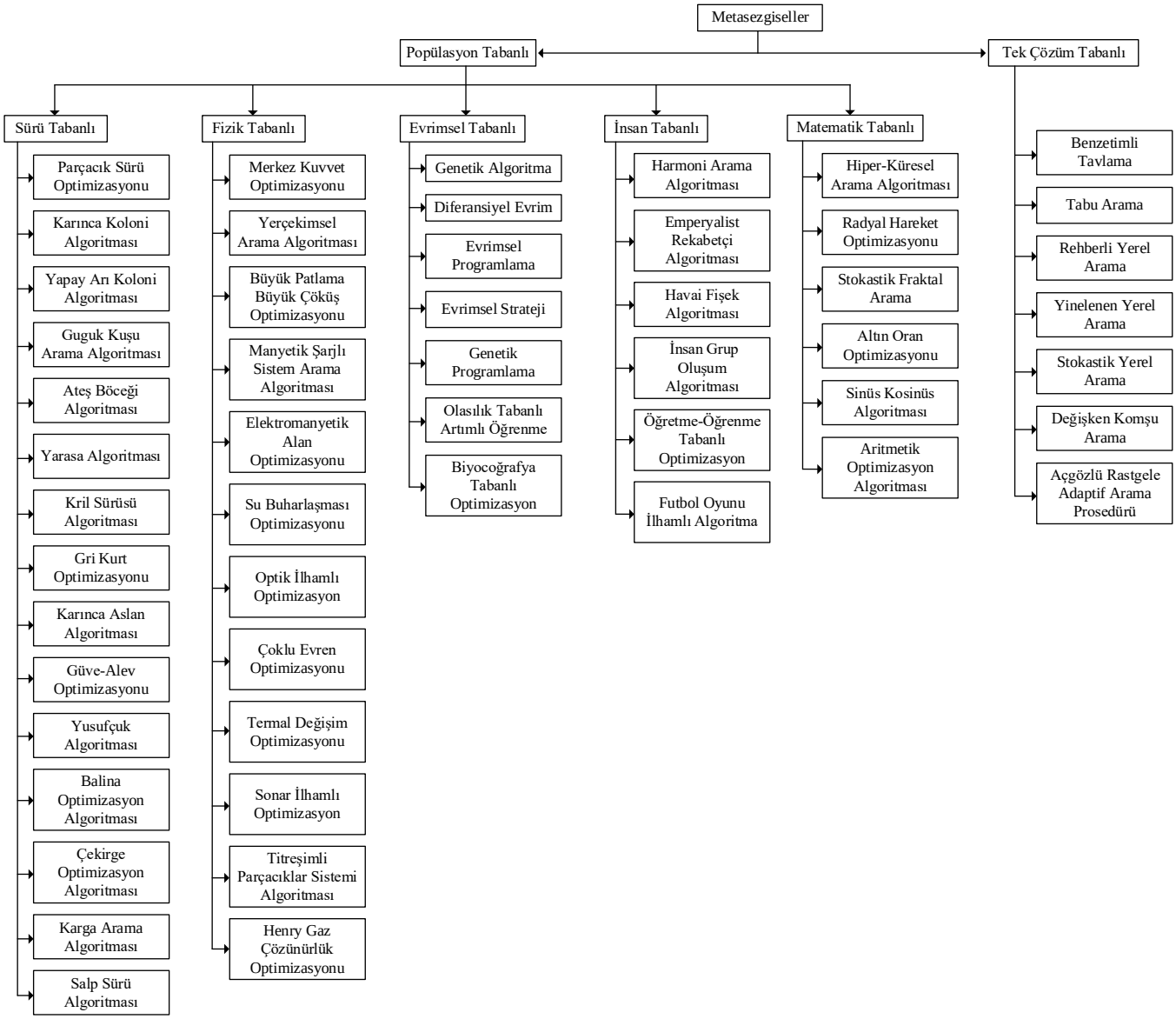
$$\sigma = \left| \frac{\log(g)}{g} \times (P_i - BP) \right| \tag{7}$$

Burada g değeri anlık iterasyon sayacıdır ve bu denklemdeki logaritmik ifadeli katsayı adım boyutunu azaltmak için kullanılır. Mevcut çözüm ile yeni çözümlerin uygunluk değerleri hesaplanır ve bunların arasındaki en iyi çözüm difüzyon süreci fonksiyonunun çıktısı olarak verilir. Ardından popülasyon iki güncelleme sürecinden geçer. Birinci güncelleme sürecinde ilk olarak çözümler Eşitlik 8'e göre sıralanır.

$$Pa_i = \frac{\text{rank}(P_i)}{N} \tag{8}$$

Burada Pa_i i. çözümün olasılıksal değeri, $\text{rank}(P_i)$ i. çözümün uygunluk değerine göre sıralamadaki yeri ve N ise popülasyon boyutudur. Her bir çözümün her bir boyutu için bir rassal sayı belirlenir. Bu sayı i. çözümün olasılıksal değerinden büyükse Eşitlik 9'daki denklem kullanılarak çözüm güncellenir.

$$P_i'(j) = P_r(j) - \varepsilon \times (P_t(j) - P_i(j)) \tag{9}$$



Şekil 3. Metasezgisel algoritmaların sınıflandırılması (Gabis ve ark., 2021)

Burada $P_i'(j)$ i. çözümün birinci güncel formu, P_r ve P_t popülasyonda rastgele seçilen çözümler ve ε [0, 1] aralığında rassal bir sayıdır. İkinci güncelleme sürecinde ise yine ilk aşamada birinci güncelleme aşamasında elde edilen tüm çözümler Eşitlik 8'e göre sıralanır. Her bir çözüm için tekrar bir rassal sayı belirlenir. Bu sayı i. çözümün olasılıksal değerinden büyükse Eşitlik 10'daki denklem kullanılarak çözüm güncellenir.

$$P_i'' = \begin{cases} P_i' - \hat{\varepsilon}\ x\ (P_t' - BP)\ ;\ \varepsilon' \leq 0.5 \\ P_i' + \hat{\varepsilon}\ x\ (P_t' - P_r')\ ;\ \varepsilon' > 0.5 \end{cases} \quad (10)$$

Burada P_i'' i. çözümün ikinci güncel formu, P_t've P_r' birinci güncelleme süreci sonucundaki popülasyonda rastgele seçilen çözümler ve $\hat{\varepsilon}$ ve ε' [0, 1] aralığında rassal sayılardır. Bu genel süreç durdurma kriteri sağlanana kadar devam eder. Algoritmanın sözde kodu Algoritma 1'de gösterilmektedir.

2.2.2. Aritmetik Optimizasyon Algoritması

Aritmetik optimizasyon algoritması, Abualigah tarafından 2021 yılında geliştirilen ve çözümlerin güncellenmesi için matematikteki ana aritmetik operatörleri kullanan matematik tabanlı bir metasezgisel algoritmadır. Bu algoritma toplama, çıkarma, çarpma ve bölme operatörleri tabanlı denklemler aracılığıyla optimizasyon gerçekleştirir. Algoritmada öncelikle başlangıç popülasyonunu rastgele oluşturulur ve çözümlerin uygunluk değerleri hesaplanır. Sonra anlık iterasyon sayacı tabanlı MOA (Math Optimizer Accelerated) adı verilen bir fonksiyonun değeri Eşitlik 11'deki gibi hesaplanır.

$$MOA(g) = Min + g\ x\ \left(\frac{Max-Min}{maxIter}\right) \quad (11)$$

Burada g anlık iterasyon sayacı, MOA(g) MOA fonksiyonunun g. iterasyondaki değeri, Min ve Max fonksiyonun sınır değerleri ve maxIter ise maksimum iterasyon sayısıdır. Daha sonra MOP (Math Optimizer Probability) adı verilen bir başka fonksiyonun değeri Eşitlik 12'deki gibi hesaplanır.

Algoritma 1. Stokastik fraktal arama algoritmasının sözde kodu (Salimi, 2015)

maxIt ← maksimum iterasyon sayısı
N ← popülasyon boyutu
D ← problem boyutu
q ← difüzyon sayısı
walk ← yürüyüş oranı
for i = 1 : N
  P_i ← i. çözümü rastgele üret
  if f(P_i) < f(BP)
    BP ← P_i
  end if
end for
for g = 1 : maxIt
  for i = 1 : N
    for j = 2 : (q+1)
      if rand < walk
        $P_i(q)$ ← Eşitlik 5 ile yeni bir çözüm üret
      else
        $P_i(q)$ ← Eşitlik 6 ile yeni bir çözüm üret
      end if
    end for
    P_i ← $P_i(q)$ ve P_i çözümlerinin en iyisini belirle
  end for
  P ← Eşitlik 8 ile popülasyonu sırala
  for i = 1 : N
    for j = 1 : D
      if rand ≥ Pa_i
        $P_i'(j)$ ← Eşitlik 9 ile çözümü güncelle
      else
        Herhangi bir güncelleme yapma
      end if
    end for
  end for
  for i = 1 : N
    if rand ≥ Pa_i'
      P_i'' ← Eşitlik 10 ile çözümü güncelle
    else
      Herhangi bir güncelleme yapma
    end if
  end for
end for
BP'yi göster

$$\mathrm{MOP}(g) = 1 - \frac{g^{\frac{1}{\alpha}}}{\mathrm{maxIter}^{\frac{1}{\alpha}}} \tag{12}$$

Burada MOP(g) MOP fonksiyonunun g. iterasyondaki değeri ve α ise hassasiyet parametresidir. Her bir çözümün her bir boyutu için, r_1, r_2 ve r_3 değişkenlerinde [0, 1] aralığında rassal değerler oluşturulur. r_1 değişkeni MOA değerinden büyükse keşif aşaması, küçükse kullanım aşaması gerçekleştirilir. Keşif aşamasında çözümün ilgili boyutu Eşitlik 13'te verilen denklem kullanılarak güncellenir.

$$x_{i,j}(g) = \begin{cases} \mathrm{best}(x_j) \,/\, (\mathrm{MOP} + \varepsilon) \times \left((UB_j - LB_j) \times \mu + LB_j \right) & ;\ r_2 < 0.5 \\ \mathrm{best}(x_j) \times \mathrm{MOP} \times \left((UB_j - LB_j) \times \mu + LB_j \right) & ;\ r_2 \geq 0.5 \end{cases} \tag{13}$$

Burada $x_{i,j}(g)$ g. iterasyonda i. çözümün j. boyutu, $\mathrm{best}(x_j)$ popülasyondaki en iyi çözümün j. boyutu, ε küçük sabit bir sayı, UB_j ve LB_j çözümlerin sınır değerleri ve μ ise arama sürecini düzenleyen bir kontrol parametresidir. Kullanım aşamasında ise çözümün ilgili boyutu Eşitlik 14'te verilen denklem kullanılarak güncellenir.

$$x_{i,j}(g) = \begin{cases} \mathrm{best}(x_j) - \mathrm{MOP} \times \left((UB_j - LB_j) \times \mu + LB_j \right) & ;\ r_3 < 0.5 \\ \mathrm{best}(x_j) + \mathrm{MOP} \times \left((UB_j - LB_j) \times \mu + LB_j \right) & ;\ r_3 \geq 0.5 \end{cases} \tag{14}$$

Bu genel süreç durdurma kriteri sağlanana kadar devam eder. Algoritmanın sözde kodu Algoritma 2'de gösterilmektedir.

Algoritma 2. Aritmetik optimizasyon algoritmasının sözde kodu (Abualigah ve ark., 2021)

maxIt ← maksimum iterasyon sayısı
N ← popülasyon boyutu
D ← problem boyutu
MOP_Max ← MOP fonksiyonunun maksimum değeri
MOP_Min ← MOP fonksiyonunun minimum değeri
α ← hassasiyet parametresi
μ ← arama sürecini düzenleyen kontrol parametresi
for i = 1 : N
  P_i ← i. çözümü rastgele üret
end for
for g = 1 : maxIt
  for i = 1 : N
    if f(P_i) < f(best)
      best ← P_i
    end if
  end for
  MOA ← Eşitlik 11 ile MOA değerini güncelle
  MOP ← Eşitlik 12 ile MOP değerini güncelle
  for i = 1 : N
    for j = 1 : D
      r_1, r_2, r_3 ← [0, 1] aralığında rassal değerler üret
      if r_1 > MOA
        $x_{i,j}$ ← Eşitlik 13 ile çözümü güncelle
      else
        $x_{i,j}$ ← Eşitlik 14 ile çözümü güncelle
      end if
    end for
  end for
end for
best'i göster

2.2.3. Sinüs Kosinüs Algoritması

Sinüs kosinüs algoritması, Mirjalili tarafından 2016 yılında geliştirilen matematik tabanlı bir metasezgisel algoritmadır. Bu algoritma çözümlerin güncellenmesi için matematikteki sinüs ve kosinüs fonksiyonlarına dayanan ve rastgele değişkenlerin de entegre edildiği matematiksel modelleri kullanır. Algoritmada öncelikle başlangıç popülasyonunu rastgele oluşturulur ve çözümlerin uygunluk değerleri hesaplanır. Sonra r_1, r_2, r_3 ve r_4 değişkenleri oluşturulur. r_1 için Eşitlik 15'teki denklem kullanılır.

$$r_1 = a - g \times \frac{a}{\mathrm{maxIter}} \tag{15}$$

Burada a sabit bir sayı, g anlık iterasyon sayacı ve maxIter maksimum iterasyon sayısıdır. r_2, r_3 ve r_4 değişkenleri rassal olarak oluşturulur. Ancak r_2 değişkeni [0, 2π] aralığında

üretilmektedir. Daha sonra çözümler Eşitlik 16'daki denklem kullanılarak güncellenir.

$$x_{i,j} = \begin{cases} x_{i,j} + r_1 \times \sin(r_2) \times \left| r_3 \times best_j - x_{i,j} \right| & ; \ r_4 < 0.5 \\ x_{i,j} + r_1 \times \cos(r_2) \times \left| r_3 \times best_j - x_{i,j} \right| & ; \ r_4 \geq 0.5 \end{cases} \quad (16)$$

Bu genel süreç durdurma kriteri sağlanana kadar devam eder. Algoritmanın sözde kodu Algoritma 3'te gösterilmektedir.

Algoritma 3. Sinüs kosinüs algoritmasının sözde kodu (Mirjalili, 2016)

```
maxIt ← maksimum iterasyon sayısı
N ← popülasyon boyutu
D ← problem boyutu
a ← r1 için kullanılan sabit sayı
for i = 1 : N
    xi ← i. çözümü rastgele üret
    if f(xi) < f(best)
        best ← xi
    end if
end for
for g = 1 : maxIt
    r1 ← Eşitlik 15 ile r1 değişkenini üret
    for i = 1 : N
        for j = 1 : D
            r2, r3, r4 ← Rassal sayılar üret
            xi,j ← Eşitlik 16 ile çözümü güncelle
        end for
        if f(xi) < f(best)
            best ← xi
        end if
    end for
end for
best'i göster
```

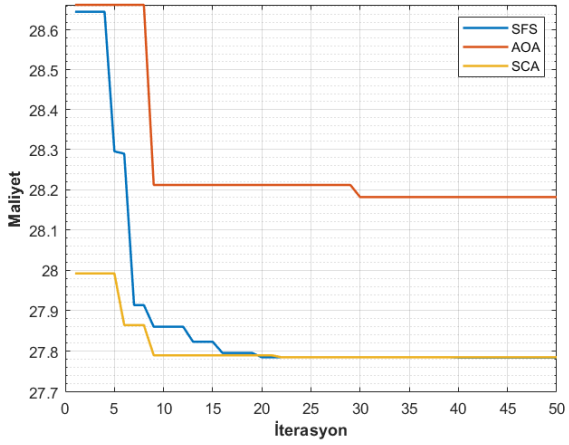
3. Bulgular ve Tartışma

Çalışma MATLAB 2020 programlama dilinde kodlanarak gerçekleştirilmiş ve bunun için Windows 10 işletim sistemi, INTEL CORE i7 işlemcisi, 16 GB RAM'e sahip bir bilgisayar kullanılmıştır. Çözümlere karşılık gelen ara noktaların sayısı 2 olarak belirlenmiştir. Bu ara noktalar [2 19] aralığında oluşturulmaktadır. Amaç fonksiyonundaki t çarpanı ise 10 alınmıştır. Tüm algoritmalarında maksimum iterasyon sayısı 50 ve popülasyon sayısı 30'dur. Stokastik fraktal arama algoritmasında difüzyon sayısı 2, yürüyüş oranı 1'dir (%100). Yani difüzyon aşamasında sadece birinci Gauss yürüyüşü kullanılmaktadır. Aritmetik optimizasyon algoritmasında α 5 ve μ 0.5 olarak belirlenmiştir. Ayrıca MOA fonksiyonunun maksimum ve minimum değerleri sırasıyla 1 ve 0.2'dir. Sinüs kosinüs algoritmasında r_1 denkleminde kullanılan a değeri 2 alınmıştır. Bu algoritmalar her bir ortamda 30 defa çalıştırılmıştır. Bu simülasyonlar sonucunda elde edilen ortalama maliyet, maliyetlerdeki standart sapmalar ve algoritmaların ortalama çalışma süreleri Tablo 1'de gösterilmektedir.

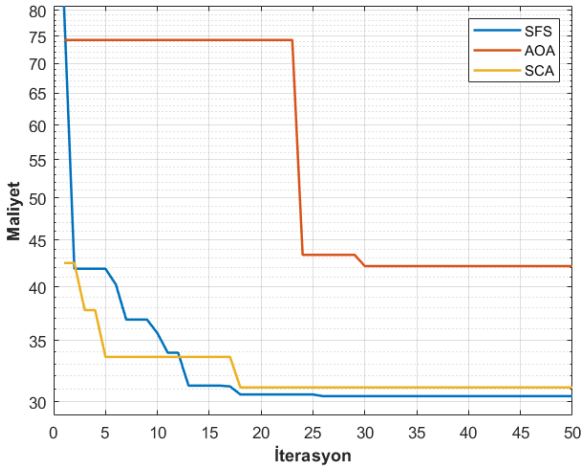
Tablo 1 ortalama maliyet açısından incelendiğinde, her ortam için en performanslı algoritmanın SFS olduğu görülmektedir. Ortam 1'den ortam 3'e doğru zorluğun arttığı göz önüne alınırsa, ortam 1 için ortalama maliyetler birbirine yakın olsa da en iyi performanslı algoritma seçilebilmektedir. Ortam 2 ve 3 için ortalama maliyetler arasındaki farkın yüksek olduğu ve SFS algoritmasının açık ara önde olduğu söylenebilir. En kötü performans gösteren algoritmanın ise AOA olduğu söylenebilir. Standart sapmalar incelendiğinde, her üç ortamda da en kararlı algoritmanın yine SFS olduğu görülmektedir. Bu sonuç da SFS algoritmasının performans kalitesini artırmaktadır. Kararlılıkta en düşük performansı gösteren algoritma ise yine AOA olarak görülmektedir. Algoritmaların çalışma süreleri incelendiğinde ise en hızlı algoritmanın AOA olduğu görülmektedir. SFS algoritması diğerlerine göre biraz daha yavaş sonuç üretmektedir, ancak burada kayda değer farkların olmadığı da söylenebilir. Örnek bir çalışma için algoritmaların yakınsamaları Şekil 4'te gösterilmektedir.

Tablo 1. Simülasyonlar Sonucunda Elde Edilen Ortalama Maliyet, Maliyetlerdeki Standart Sapmalar ve Algoritmaların Ortalama Çalışma Süreleri (Ort: Ortalama, Std: Standart Sapma)

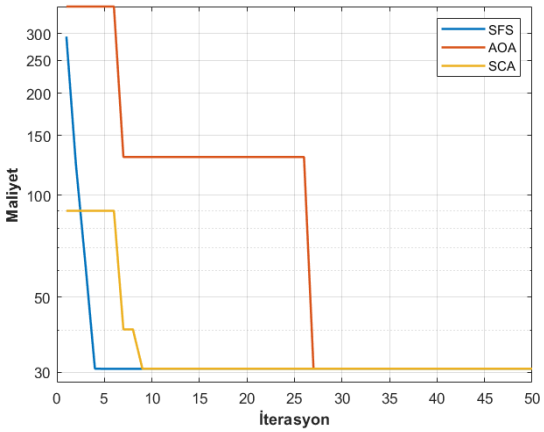
Algoritmalar	**Ortam 1**			**Ortam 2**			**Ortam 3**		
	Maliyet		**Algoritmanın Ortalama Çalışma Süresi (sn)**	**Maliyet**		**Algoritmanın Ortalama Çalışma Süresi (sn)**	**Maliyet**		**Algoritmanın Ortalama Çalışma Süresi (sn)**
	Ort.	**Std.**		**Ort.**	**Std.**		**Ort.**	**Std.**	
SFS	**27.7855**	**0.0014**	3.1590	**31.4098**	**5.3684**	3.4920	**30.7431**	**1.44e-14**	4.3800
AOA	28.8371	1.2027	**2.8414**	37.5400	14.5533	**3.1085**	112.8774	118.4936	**4.0377**
SCA	28.1195	0.5191	2.8452	33.1045	10.9593	3.1683	81.2305	102.1205	4.0402



(a)



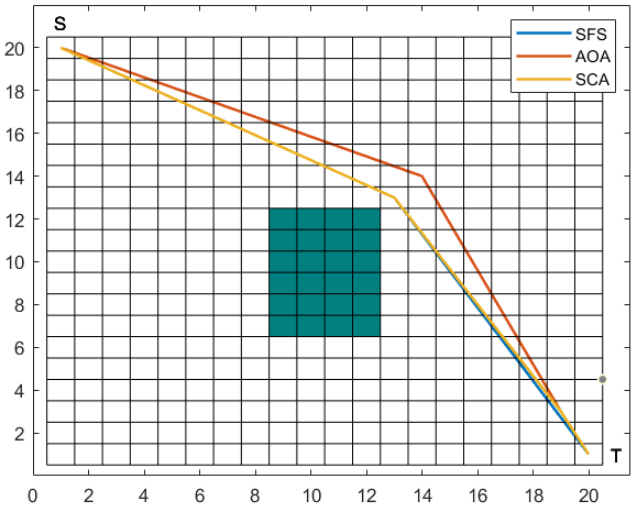
(b)



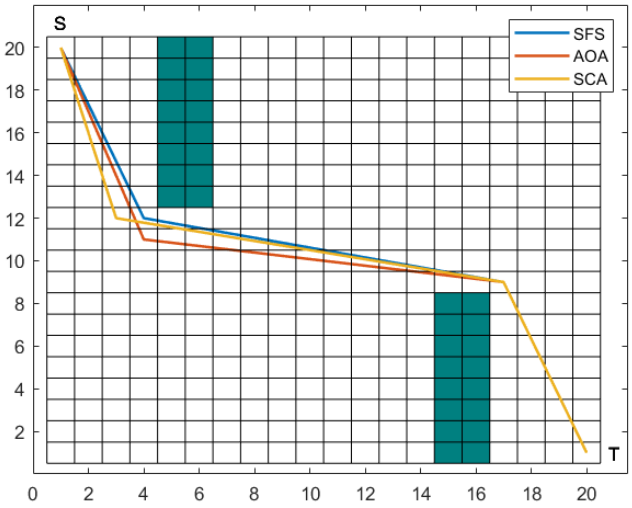
(c)

Şekil 4. Örnek bir çalışma için algoritmaların yakınsamaları: (a) Ortam 1, (b) Ortam 2, (c) Ortam 3

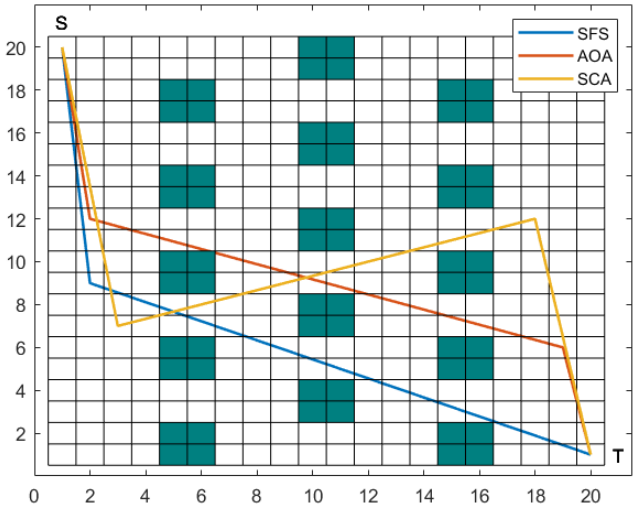
Şekil 4 incelendiğinde, SFS algoritmasının yakınsama performansının diğer algoritmalara göre daha iyi olduğu görülmektedir. Ortam 1'de algoritmaların yakınsamaları arasında küçük farklar görülürken, ortam 2 ve 3'te algoritmaların yakınsama hızları arasındaki fark açıkça görülebilmektedir. Örnek bir çalışma için algoritmaların planladıkları yollar Şekil 5'te gösterilmektedir.



(a)



(b)



(c)

Şekil 5. Örnek bir çalışma için algoritmaların planladıkları yollar: (a) Ortam 1, (b) Ortam 2, (c) Ortam 3

Şekil 5 incelendiğinde, ortam 1 ve 2'de algoritmaların planladığı yollar birbirine yakındır. Ancak bunların arasında en performanslı algoritmanın SFS olduğu söylenebilir. Ortam 3'te AOA ve SCA algoritmaları SFS algoritmasına göre yerel minimuma daha çok yakalandığı görülmektedir. AOA algoritması nispeten diğer algoritmalara göre daha maliyetli bir yol planlamıştır. Dolayısıyla ortam 3 için de planlanan yol açısından en performanslı algoritmanın SFS olduğu ifade edilebilir. Ancak yine de bu üç algoritma engelleri kesmeden yolları planlamışlardır.

4. Sonuç

Bu çalışmada ızgara tipi bir ortamda bir mobil robotun küresel yol planlamasında farklı matematik tabanlı metasezgisel algoritmaların etkileri incelenmiştir. Öncelikle ızgara tipinde ve farklı zorluk derecelerinde üç farklı ortam tasarlanmıştır. Ardından, son yıllarda geliştirilen farklı matematik tabanlı algoritmalar kullanılarak robotun ortamlardaki optimum yolları hesaplanmıştır. Çalışmada SFS, AOA ve SCA algoritmaları kullanılmıştır. Simülasyon sonuçları SFS algoritmasının en kısa mesafe ve engelden kaçınma açısından diğer algoritmalara göre daha iyi sonuçlar verdiğini göstermektedir. Kararlılık açısından en performanslı algoritmanın yine SFS olduğu görülmektedir. Yakınsama performansı ve planlanan yollar değerlendirildiğinde yine SFS algoritmasının diğer algoritmalara göre nispeten daha iyi sonuçlar verdiği görülmektedir. Ancak çalışma hızı açısından değerlendirildiğinde en hızlı algoritma AOA olmuştur. SFS algoritması diğerlerine göre biraz daha yavaş sonuç üretmektedir. Bu bulgular SFS algoritmasının mobil robotların küresel yol planlama probleminde kayda değer ölçüde sonuçlar verdiğini göstermektedir.

Kaynakça

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The Arithmetic Optimization Algorithm. *Computer Methods in Applied Mechanics and Engineering*, *376*, 113609. https://doi.org/10.1016/j.cma.2020.113609

Abualigah, L. (2021). *The Arithmetic Optimization Algorithm (AOA)*. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/84742-the-arithmetic-optimization-algorithm-aoa

Adamu, P. I., Okagbue, H. I., & Oguntunde, P. E. (2019). Fast and Optimal Path Planning Algorithm (FAOPPA) for a Mobile Robot. *Wireless Personal Communications*, *106*(2), 577–592. https://doi.org/10.1007/s11277-019-06180-w

Ajeil, F. H., Ibraheem, I. K., Sahib, M. A., & Humaidi, A. J. (2020). Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Applied Soft Computing Journal*, *89*, 106076. https://doi.org/10.1016/j.asoc.2020.106076

Ajeil, F. H., Ibraheem, I. K., Azar, A. T., & Humaidi, A. J. (2020). Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. *Sensors (Switzerland)*, *20*(7). https://doi.org/10.3390/s20071880

Akka, K., & Khaber, F. (2018). Mobile robot path planning using an improved ant colony optimization. *International Journal of Advanced Robotic Systems*, *15*(3), 1–7. https://doi.org/10.1177/1729881418774673

Ali, H., Gong, D., Wang, M., & Dai, X. (2020). Path Planning of Mobile Robot With Improved Ant Colony Algorithm and MDP to Produce Smooth Trajectory in Grid-Based Environment. *Frontiers in Neurorobotics*, *14*(July), 1–13. https://doi.org/10.3389/fnbot.2020.00044

Dai, X., Long, S., Zhang, Z., & Gong, D. (2019). Mobile robot path planning based on ant colony algorithm with a∗ heuristic method. *Frontiers in Neurorobotics*, *13*(April). https://doi.org/10.3389/fnbot.2019.00015

Das, P. K. (2020). Hybridization of Kidney-Inspired and Sine–Cosine Algorithm for Multi-robot Path Planning. *Arabian Journal for Science and Engineering*, *45*(4), 2883–2900. https://doi.org/10.1007/s13369-019-04193-y

Dolicanin, E., Fetahovic, I., Tuba, E., Capor-Hrosik, R., & Tuba, M. (2018). Unmanned combat aerial vehicle path planning by brain storm optimization algorithm. *Studies in Informatics and Control*, *27*(1), 15–24. https://doi.org/10.24846/v27i1y201802

Elmi, Z., & Efe, M. O. (2018). Multi-objective grasshopper optimization algorithm for robot path planning in static environments. *Proceedings of the IEEE International Conference on Industrial Technology*, *2018-Febru*, 244–249. https://doi.org/10.1109/ICIT.2018.8352184

Gabis, A. B., Meraihi, Y., Mirjalili, S., & Ramdane-Cherif, A. (2021). A comprehensive survey of sine cosine algorithm: variants and applications. In *Artificial Intelligence Review* (Vol. 54, Issue 7). Springer Netherlands. https://doi.org/10.1007/s10462-021-10026-y

Gul, F., Rahiman, W., Alhady, S. S. N., Ali, A., Mir, I., & Jalil, A. (2020). Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming. *Journal of Ambient Intelligence and Humanized Computing*, *12*(7), 7873–7890. https://doi.org/10.1007/s12652-020-02514-w

Hosseininejad, S., & Dadkhah, C. (2019). Mobile robot path planning in dynamic environment based on cuckoo optimization algorithm. *International Journal of Advanced Robotic Systems*, *16*(2), 1–13. https://doi.org/10.1177/1729881419839575

Huang, H. C., & Tsai, C. C. (2011). Global path planning for autonomous robot navigation using hybrid metaheuristic GA-PSO algorithm. *Proceedings of the SICE Annual Conference*, 1338–1343.

Lamini, C., Benhlima, S., & Elbekri, A. (2018). Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, *127*, 180–189. https://doi.org/10.1016/j.procs.2018.01.113

Li, W., Sun, S., Li, J., & Hu, Y. (2018). Stochastic Fractal Search Algorithm and its Application in Path Planning. *2018*

IEEE CSAA Guidance, Navigation and Control Conference, CGNCC 2018. https://doi.org/10.1109/GNCC42960.2018.9018694

Liang, X., Kou, D., & Wen, L. (2020). An Improved Chicken Swarm Optimization Algorithm and its Application in Robot Path Planning. *IEEE Access*, *8*, 49543–49550. https://doi.org/10.1109/ACCESS.2020.2974498

Luo, Q., Wang, H., Zheng, Y., & He, J. (2020). Research on path planning of mobile robot based on improved ant colony algorithm. *Neural Computing and Applications*, *32*(6), 1555–1566. https://doi.org/10.1007/s00521-019-04172-2

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

Mirjalili, S. (2021). *SCA: A Sine Cosine Algorithm.* MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/54948-sca-a-sine-cosine-algorithm

Muhammad, A., Ali, M., & Shanono, I. (2020). Path planning Methods for Mobile Robots: A systematic and Bibliometric Review. *Journal of Electrical Engineering*, *19*(3), 14–34.

Patle, B. K., Parhi, D. R. K., Jagadeesh, A., & Kashyap, S. K. (2018). Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot. *Computers and Electrical Engineering*, *67*, 708–728. https://doi.org/10.1016/j.compeleceng.2017.12.011

Salimi, H. (2015). Stochastic Fractal Search: A powerful metaheuristic algorithm. *Knowledge-Based Systems*, *75*, 1–18. https://doi.org/10.1016/j.knosys.2014.07.025

Salimi, H. (2021). *Stochastic Fractal Search (SFS).* MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/47565-stochastic-fractal-search-sfs

Saraswathi, M., Murali, G. B., & Deepak, B. B. V. L. (2018). Optimal Path Planning of Mobile Robot Using Hybrid Cuckoo Search-Bat Algorithm. *Procedia Computer Science*, *133*, 510–517. https://doi.org/10.1016/j.procs.2018.07.064

Tuba, E., Strumberger, I., Zivkovic, D., Bacanin, N., & Tuba, M. (2018). Mobile Robot Path Planning by Improved Brain Storm Optimization Algorithm. *2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings*. https://doi.org/10.1109/CEC.2018.8477928

Wang, B., Li, S., Guo, J., & Chen, Q. (2018). Car-like mobile robot path planning in rough terrain using multi-objective particle swarm optimization algorithm. *Neurocomputing*, *282*, 42–51. https://doi.org/10.1016/j.neucom.2017.12.015

Wang, R. B., Wang, W. F., Xu, L., Pan, J. S., & Chu, S. C. (2021). An Adaptive Parallel Arithmetic Optimization Algorithm for Robot Path Planning. *Journal of Advanced Transportation*, *2021*. https://doi.org/10.1155/2021/3606895

Yıldırım, M. Y., & Akay, R. (2021). A Comparative Study of Optimization Algorithms for Global Path Planning of Mobile Robots. *Sakarya University Journal of Science*, *25*, 417–428. https://doi.org/10.16984/saufenbilder.800067

Yıldırım, M. Y., & Akay, R. (2021). Fast path planning in multi-obstacle environments for mobile robots. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *36*(3), 1551–1564. https://doi.org/10.17341/gazimmfd.802646

Zhang, J. H., Zhang, Y., & Zhou, Y. (2018). Path planning of mobile robot based on hybrid multi-objective bare bones particle swarm optimization with differential evolution. *IEEE Access*, *6*, 44542–44555. https://doi.org/10.1109/ACCESS.2018.2864188

Zhong, X., Tian, J., Hu, H., & Peng, X. (2020). Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-Scale Dynamic Environment. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *99*(1), 65–77. https://doi.org/10.1007/s10846-019-01112-z