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

YAPAY ARI KOLONİ PROGRAMLAMA (ABCP) YÖNTEMİ İLE ROBOTİK YOL PLANLAMA

Hazırlayan
Fateh BOUDARDARA

Danışman
Dr. Öğr. Üyesi Beyza GÖRKEMLİ

Yüksek Lisans Tezi

Kayseri
Mayıs 2019

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

YAPAY ARI KOLONİ PROGRAMLAMA (ABCP) YÖNTEMİ İLE ROBOTİK YOL PLANLAMA

(Yüksek Lisans Tezi)

Hazırlayan
Fateh BOUDARDARA

Danışman
Dr. Öğr. Üyesi Beyza GÖRKEMLİ

Bu çalışma, Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FYL-2018-7937 kodlu proje ile desteklenmiştir.

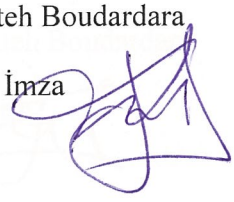
Mayıs 2019
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

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

Fateh Boudardara

İmza



"YAPAY ARI KOLONİ PROGRAMLAMA (ABCP) YÖNTEMİ İLE ROBOTİK YOL PLANLAMA" adlı Yüksek Lisans Tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi' ne uygun olarak hazırlanmıştır.

Hazırlayan

Fateh Boudardara

İmza

Danışman

Dr.Öğr.Üyesi Beyza Görkemli

İmza

Bilgisayar Mühendisliği ABD Başkanı

Prof. Dr. Veysel Aslantaş

İmza

KABUL VE ONAY

Dr. Öğr. Üyesi Beyza GÖRKEMLİ danışmanlığında **Fateh BOUDARDARA** tarafından hazırlanan **"YAPAY ARI KOLONİ PROGRAMLAMA (ABCP) YÖNTEMİ İLE ROBOTİK YOL PLANLAMA"** adlı bu çalışma jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği** Anabilim Dalında **Yüksek Lisans** tezi olarak kabul edilmiştir.

02/05/2019

JÜRİ:

Danışman : Dr. Öğr. Üyesi Beyza GÖRKEMLİ .....

Üye : Prof. Dr. Derviş KARABOĞA .....

Üye : Dr. Öğr. Üyesi Ebubekir KAYA .....

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun 14/05/2019 tarih ve 2019/30-07 sayılı kararı ile onaylanmıştır.

14/05/2019

Prof. Dr. Mehmet AKKURT

Enstitü Müdürü

TEŞEKKÜR

Bana çalışmalarım süresince her türlü yardımı ve fedakârlığı sağlayan, farklı bakış açıları ve bilimsel katkılarıyla beni aydınlatan, yakın ilgi ve yardımlarını esirgemeyen ve bu günlere gelmemde en büyük katkı sahibi sayın hocam Dr. Öğr. Üyesi Beyza Görkemli'ye teşekkürü bir borç bilirim.

Bu Yüksek Lisans eğitimim için burs imkânı sunan ve gereken desteği veren Yurtdışı Türkler ve Akraba Topluluklar Başkanlığı'na teşekkür ederim.

Bu Yüksek Lisans tez çalışma kapsamında FYL-2018-7937 kodlu proje ile destekleyen Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimine teşekkür ederim.

Ayrıca; bütün eğitim hayatım sürecinde sabır göstererek beni daima destekleyen ve bu günlere getiren kıymetli anne ve babama, ve kardeşlerime sonsuz teşekkürler.

Fateh BOUDARDARA

Mayıs 2019, KAYSERİ

YAPAY ARI KOLONİ PROGRAMLAMA (ABCP) YÖNTEMİ İLE ROBOTİK YOL PLANLAMA

Fateh BOUDARDARA

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Mayıs 2019
Danışman: Dr. Öğr. Üyesi Beyza GÖRKEMLİ

ÖZET

Robotik yol planlama özellikle insanın çalışamadığı veya riskin yüksek olduğu birçok alandaki öneminden dolayı son on yılda popüler bir araştırma alanı haline gelmiştir. Yapay karınca problemi (artificial ant problem-AAP) robotik yol planlamanın bir alt problemi olarak kabul edilmiştir. 1991 yılında Christopher Langton tarafından literatüre tanıtılmasından itibaren bu problem birçok araştırmacı tarafından incelenmiştir. Ayrıca, AAP birçok algoritmanın performansını değerlendirmek için bir test problemi olarak kullanılmıştır. Bu çalışmada, AAP iki yöntem kullanılarak çözülmüştür: yapay arı koloni programlama (artificial bee colony programming-ABCP) ve daralan ABCP (shrinking ABCP-shABCP) olarak adlandırılan ABCP'nin yeni bir versiyonu. ABCP yapay arı kolonisi algoritmasına dayalı yeni bir evrimsel hesaplama tabanlı otomatik programlama yöntemidir. shABCP bu çalışmada geliştirilmiştir ve temel fikri ABCP'deki gibi sabit bir miktar kullanmak yerine periyodik olarak yiyecek kaynağı miktarını azaltmaktır. İlk olarak, shABCP için bazı parametre ayarlama çalışmaları yapılmıştır. Daha sonra, ABCP, shABCP ve bazı diğer evrimsel hesaplama tabanlı otomatik programlama yöntemlerinin performansları kıyaslanmıştır. Simülasyon sonuçları ABCP'nin AAP'yi etkin bir şekilde çözmek için kullanılabileceğini göstermiştir. Ayrıca, ABCP'ye eklenen periyodik olarak azalan popülasyon büyüklüğü özelliği, algoritmanın performansını AAP'de iyiliştirmiştir.

Anahtar Kelimeler: robotik yol planlama, yapay karınca problemi, Los Altos Hills patikası, Santa Fe patikası, evrimsel hesaplamaya dayalı otomatik programlama, yapay arı kolonisi, yapay arı koloni programlama, daralan yapay arı koloni programlama, dinamik olarak azalan popülasyon büyüklüğü.

ROBOTIC PATH PLANNING WITH ARTIFICIAL BEE COLONY PROGRAMMING (ABCP) METHOD

Fateh BOUDARDARA

Erciyes University, Graduate School of Natural and Applied Sciences
Master Thesis, May 2019
Supervisor: Assist. Prof. Dr. Beyza GÖRKEMLİ

ABSTRACT

Robotic path planning has become a popular research field in the last decade due to its importance in many areas, especially where human being cannot work or the risk is high. Artificial ant problem (AAP) is considered as a sub-problem of robotic path planning. Since its introduction to the literature by Christopher Langton in 1991, this problem has been investigated by many researchers. In addition, AAP is used as a benchmark problem for many algorithms to evaluate their performances. In this study, the AAP is solved using two methods: artificial bee colony programming (ABCP) and a new version of ABCP called shrinking ABCP (shABCP). ABCP is a novel evolutionary computation based automatic programming method based on artificial bee colony algorithm. The shABCP is developed in this study and its basic idea is to reduce the number of food sources, periodically, instead of using a constant number as in ABCP. Firstly, some parameter tuning studies are carried out for shABCP. Then, performances of ABCP, shABCP and some other evolutionary computation based automatic programming methods are compared. The simulation results show that ABCP can be used to solve AAP effectively. Furthermore, the periodically decreasing population size property added to ABCP improves the performance of the algorithm on AAP.

Keywords: robotic path planning, artificial ant problem, Los Altos Hills trail, Santa Fe trail, evolutionary computation based automatic programming, artificial bee colony, artificial bee colony programming, shrinking artificial bee colony programming, dynamically decreasing population size.

İÇİNDEKİLER

YAPAY ARI KOLONİ PROGRAMLAMA (ABCP) YÖNTEMİ İLE ROBOTİK YOL PLANLAMA

BİLİMSEL ETİĞE UYGUNLUK i

BİLİMSEL ETİĞE UYGUNLUK ii

KABUL VE ONAY iii

TEŞEKKÜR iiv

ÖZET v

ABSTRACT vi

İÇİNDEKİLER vii

KISALTMALAR ix

TABLOLAR LİSTESİ x

ŞEKİLLER LİSTESİ xi

GİRİŞ 1

1. BÖLÜM

GENEL BİLGİLER ve LİTERATÜR ÇALIŞMASI

1.1. Robotik Yol Planlama 4

a. Klasik Yöntemler: 6

i. Grafik Araması 6

ii. APF 7

b. Esnek Hesaplama Yaklaşımları: 8

i. GA 8

ii. ACO 8

iii. ANN 9

1.2. AAP 9

a. SFT 10

a. LAHT 11

1.3. Literatür Taraması 13

2. BÖLÜM

YÖNTEM

2.1. ABC Algoritması 17

2.2. ABCP ve AAP ile ABCP 21

2.3. shABCP 25

3. BÖLÜM

DENEYSEL ÇALIŞMALAR VE TARTIŞMA

3.1 Patikalarla İlgili Parametrelerin Ayarları 27

3.2 shABCP'nin Başlangıç Koloni Büyüklüğü, Daralma Periyodu ve Daralma Oranı Parametreleri İle İlgili Deneyler 28

3.3. Kıyaslama Çalışması 42

4. BÖLÜM

SONUÇ VE ÖNERİLER

Sonuç ve Öneriler 50

KAYNAKÇA 52

ÖZGEÇMİŞ 58

KISALTMALAR

ECBAP	Evrimsel Hesaplama Tabanlı Otomatik Programlama (Evolutionary Computation Based Automatic Programming)
GP	Genetik Programlama (Genetic Programming)
GA	Genetik Algoritma (Genetic Algorithm)
LGP	Lineer Genetik Programlama (Linear Genetic Programming)
λ-LGP	λ-Lineer Genetik Programlama (λ-Linear Genetic Programming)
ABC	Yapay Arı Kolonisi (Artificial Bee Colony)
ABCP	Yapay Arı Koloni Programlama (Artificial Bee Colony Programming)
shABCP	Daralan Yapay Arı Koloni Programlama (Shrinking Artificial Bee Colony Programming)
shP	Daralma periyodu (Shrinking period)
shR	Daralma oranı (Shrinking rate)
MaxEval	Maksimum uygunluk değerlendirme sayısı (Maximum number of fitness evaluations)
minMaxCyc	Maksimum çevrim sayısının elde edilebilir minimum değeri (Obtainable minimum value of maximum number of cycles)
CS	Koloni büyüklüğü (colony size)
initCS	Koloni büyüklüğünün başlangıç değeri (initial colony size)
FS	Yiyecek kaynağı sayısı (Number of food sources)
Std. Dev.	Standart sapma (Standard deviation)
SR	Başarı oranı (Success rate)
minE	Minimum hesaplama eforu (Minimum computational effort)
AAP	Yapay Karınca Problemi (Artificial Ant Problem)
SFT	Santa Fe trail (Santa Fe patikası)
LAHT	Los Altos Hills trail (Los Altos Hills patikası)

TABLOLAR LİSTESİ

Tablo 1.1: Patikaların şekillerinde kullanılan sembollerin anlamları. 12
Tablo 3. 1: Patikalarla ilgili paramatrelerin ayarları. 28
Tablo 3. 2: Test durumları için initCS ve minMaxCyc değerleri.. 29
Tablo 3. 3: initCS, shR ve shP deneylerinde kullanılan shABCP'nin parametre ayarları 30
Tablo 3. 4: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 50) 32
Tablo 3. 5: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 70) 33
Tablo 3. 6: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 100) 34
Tablo 3. 7: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 140) 35
Tablo 3. 8: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 200) 36
Tablo 3. 9: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 280) 37
Tablo 3. 10: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 400) 38
Tablo 3. 11: Algoritmaların ortak parametrelerinin ayarları 43
Tablo 3. 12: SFT'de simülasyon sonuçları 44
Tablo 3. 13: LAHT'de simülasyon sonuçları 45

ŞEKİLLER LİSTESİ

Şekil 1.1: Görünürlük grafiği 7
Şekil 1.2: Voronoi grafiği 7
Şekil 1.3: SFT 11
Şekil 1.4: LAHT 12
Şekil 1.5: AAP'yi çözen bir program örneği 13
Şekil 2.1: ABC algoritmasının akış diyagramı 20
Şekil 2.2: Seçilen ağaç T1 22
Şekil 2.3: Rasgele seçilen ağaç T2 22
Şekil 2.4: Üretilen ağaç T3 22
Şekil. 2.5: ABCP algoritmasının temel adamları 24
Şekil. 2.6: shABCP algoritmasının temel adımları 26
Şekil 3.1: Farklı initCS değerleri için ortalama ham uygunluk değerlerinin ortalaması 39
Şekil 3.2: Farklı initCS değerleri için en iyi konfigürasyonun minE'si 39
Şekil 3.3: Parametre ayarlama deneylerinde de elde edilen en iyi çözüm 41
Şekil 3.4: shABCP ve ABCP'nin SFT için yakınsama grafikleri. (a) 20000 değerlendirme sayısı. (b) 100000 değerlendirme sayısı. (c) 200000 değerlendirme sayısı. 48
Şekil 3.5: shABCP ve ABCP'nin LAHT için yakınsama grafikleri. (a) 50000 değerlendirme sayısı. (b) 500000 değerlendirme sayısı 49

GİRİŞ

Robotlar, robotik üretim, insansız sistemler ve araçlar (saha, hava, su altı) gibi birçok farklı alanda kullanılmaktadır. Özellikle insanların erişemediği tehlikeli, küçük veya ulaşılamaz alanlarda önemli rolleri vardır. Bu nedenle, otonom robotlar alanında (insansız sistemler) kapsamlı bir araştırma bulunmaktadır [1]. Yol planlaması, insansız sistemler ve robotlar için en temel meselelerden biridir [2]. Dolayısıyla, robotik yol planlama ve otonom araçlar yapay zekanın iki popüler uygulama alanı olmuştur ve son on yılda yoğun olarak incelenmıştır. Robotik yol planlama mobil bir robotun ortamdaki tüm engellerden kaçarak bir başlangıç noktasından hedef noktasına en uygun (veya en azından uygulanabilen) yolu bulması şekilde tanımlanmaktadır [1-2]. Yol planlaması özellikle birden fazla amacı ve dinamik meseleleri dikkate alarak karmaşık ortamlarda gerçekleştirildiğinde, evrimsel hesaplama tabanlı optimizasyon algoritmaları ve makine öğrenme teknikleri önemli avantajlar sağlamaktadır [3–5].

Yapay karınca problemi (artificial ant problem-AAP) robotik yol planlamanın bir alt problemi olarak görülebilmektedir ve bu problemde yapay bir karınca adım sayısı deninlen sınırlı bir süre içerisinde patikadaki tüm yiyecekleri toplamasını sağlayacak bir yol bulmak zorundadır [3, 6]. Ayrıca, bu problem yeni evrimsel hesaplama tabanlı otomatik programlama (Evolutionary computation based automatic programming-ECBAP) yaklaşımlarının performansını test etmek için bir kıyaslama (benchmark) problemi olarak kullanılmaktadır [3, 7–9]. Problemin patikasına göre, literatürde bulunabilecek birçok versiyonu vardır: Santa Fe, Los Altos Hills, John Muir, San Mateo patikası vb. [3, 6-7, 10]. ECBAP yöntemleri yapay zeka optimizasyon algoritmalarını ve makine öğrenmesi temellerini birlikte kullanmaktadır. Genelde, bu yöntemlerde bilgisayar programı terimi popülasyondaki bireyleri (çözümeleri) ifade eder ve evrimsel hesaplama temelli bir optimizasyon algoritmasının temel araştırma stratejisi optimal bilgisayar programlarını bulmak için kullanılır. Her bir program, sistem çıktılarının

sistem girdilerinden nasıl üretildiğini açıklayan bir sistem modelini vermektedir. Birbilgisayar programını değerlendirmek ve maliyetini bulmak amacıyla, bir girdi-çıktı veri seti kullanılarak toplam hata hesaplanmaktadır. Günümüzde, Genetik Programlama (Genetic Programming-GP) en popüler ECBAP yöntemlerinden biridir, bu algoritma bilinen Genetik Algoritmanın (Genetic Algorithm-GA) bir uzantısı olarak 1992 yılında Koza tarafından geliştirilmiştir [3]. GP ve GA arasındaki temel fark bireylerin gösterimidir. GP bilgisayar programlarının gösterimi için sabit boyutlu çözüm dizeleri yerine ağaçları kullanmaktadır. Sembolik regresyon, optimal kontrol, kombinasyonel devre tasarımı ve robotik yol planlama gibi birçok problem GP uygulanarak çözülmüştür [3, 6]. Ayrıca, Lineer Genetik Programlama [11], Kartezyen Genetik Programlama [12], Gen Ekspresyon Programlama [13] gibi bireylerin yapısını ve gösterimlerini değiştirerek geliştirilen farklı GP versiyonları literatürde bulunabilmektedir.

Yapay Arı Kolonisi (Arificial Bee Colony-ABC) algoritması 2005 yılında Karaboga tarafından geliştirlen bir sürü zekası algoritmasıdır [14]. Bu algoritma bal arılarının davranışlarını simüle etmektedir. ABC sayısal fonksiyonların optimizasyonu [15], yapay sinir ağı [16-17] ve kümeleme [18] gibi farklı alanlarda başarıyla uygulanmıştır [19-21]. Karaboga vd. ABC'den yapay arı koloni programlama (artificial bee colony programming-ABCP) ismi verilen yeni bir ECBAP yöntemi geliştirmişlerdir [22]. GP'ye benzer bir şekilde, ABCP çözümleri göstermek için ağaç yapısını kullanmaktadır. ABCP bir makine öğrenme yöntemidir. ABCP sembolik regresyon problemleri üzerinde test edilmiş ve performansı standart GP ve bazı GP versyonlarıyla karşılaştırılmıştır [22]. Bu çalışmada birçok durumda ABCP'nin performansı GP'den daha iyi çıkmış ve ABCP'nin bu tür problemleri etkili bir şekilde çözebileceği gösterilmiştir. 2016 yılında, Golafshani ve Ashour, SCC'nin elastik modülünü tahmin etmek için ABCP'yi kullanmış ve elde edilen sonuçları biyocoğrafik tabanlı programlama (Biogeographical Based Programming-BBP) ile kıyaslamışlardır [23]. 2018 yılında, Golafshani ve Behnood geri dönüşümlü agrega betonunun (ERAC) elastik modülünün tahmin edilmesi için ABCP, GP ve BBP arasında bir kıyaslama çalışması yapmışlardır [24].

Bildiğimiz kadarıyla ABCP henüz hiçbir robotik yol planlama probleminde kullanılmamıştır. Yukarıda da bahsedilidiği gibi robotların geniş bir şekilde kullanımı ve özellikle mobil robotlara olan ihtiyaç göz önüne alındığında bu problemlerin çözümü için yeni ve daha iyi yöntemlerin bulunması bir gereklilik ve aynı zamanda popüler bir çalışma

alanıdır. Diğer bir yandan, genel olarak bilinen robotik yol planlama problemlerinden biri olan AAP, yeni geliştirilen ECBAP yöntemlerinin performansını gözlemlemek için yaygın bir şekilde bir test problemi olarak kullanılmaktadır. Dolayısıyla, bu çalışmada ABCP AAP'ye uygulanmıştır. Ayrıca, daralan ABCP (shrinking ABCP-shABCP) adı verilen yeni bir ABCP versiyonu geliştirilmiş ve hem ABCP hem de shABCP'nin performansı farklı AAP patikalarının üzerinde incelenmiştir. Sonuçlar bazı rakip yöntemlerin sonuçları ile kıyaslanmıştır.

Bu çalışmanın geri kalanı şu şekilde yapılandırılmıştır: İlk bölüm AAP'ye daha fazla odaklanmakla birlikte robotik yol planlama problemi hakkında genel bilgi vermektedir. Literatür taraması da bu bölümde sunulmaktadır. Önerilen yaklaşımlar ve bu yaklaşımların APP'yi çözmek için temel konfigürasyonları ikinci bölümde açıklanmaktadır. Tezin üçüncü bölümünde ise deneysel çalışmalar ve tartışmalar sunulmaktadır. Son olarak, dördüncü bölümde tez çalışmasının sonuçları ve ilerideki çalışmalar verilmiştir.

1. BÖLÜM

GENEL BİLGİLER VE LİTERATÜR ÇALIŞMASI

Bu bölüm robotik yol planlama ve bu problemi çözmek için kullanılan yöntemler hakkında bilgi vermektedir. İlk olarak, bu problemin tanımı ve çözülmesinin öneminden bahsedilmiştir, sonra bu alanda kullanılan farklı yaklaşımlar açıklanmıştır. Bu tezdeki deneysel çalışmalar robotik yol planlamanın bir alt problemi olan AAP üzerinde yapıldığından dolayı bu problem ayrıntılı olarak tartışılmıştır. Sonunda, AAP'yi çözmede kullanılan farklı yaklaşımlar için literatür taraması sunulmuştur.

1.1. Robotik Yol Planlama

Mobil robotik, otonom veya insansız araçların kontrolünü inceleyen bir alandır [25]. Robotik yol planlama çalışması makina mühendisliği, bilgisayar bilimleri, elektrik mühendisliği, bilişsel psikoloji, algı, sinirbilim, mekatronik gibi birçok disiplini içinde barındırmaktadır [25]. Bu alanlardaki gelişmeler sayesinde robotlar artık daha hızlı, akıllıca ve bazen insandan daha iyi bir şekilde görevleri yapabilmektedir. Otonom robotlar imalat, depo, tıbbi hizmetleri, turist rehberliği, ordu, eğitim, kaynak ve uzay araştırmaları gibi birçok alanda kullanılmaktadır [2]. Robot kullanımı gereken alanların çoğu takip eden özelliklerden en az birine sahiptir: erişilemeyen alanlar, uzun zaman alan, yüksek maliyetli veya insan için madencilik, uzay, sualtı ortamları ve nükleer alanlar gibi kabul edilemez koşullar [25].

Robotların önemli görevlerinden biri harekettir. Robotik alanında mobil bir robotun iki tür hareketi vardır: hareket planlama ve yol planlama [26]. Hareket planlama robotun parçalarının hareketlerine ve koordinasyonlarına odaklanır, insansı (humanoid) ve bacaklı robotlar için bu tür daha önemlidir. Yol planlama veya yörünge planlama ise ortamda robotun hareketine odaklanır [2, 25-26].

Genel olarak robotik yol planlama probleminde, bir robot hareketli bir nokta gibi görünebilir ve ona nokta-robotu (point-robot) denir. Robot başlangıç noktasından hedef noktasına kadar engellerle çarpışmadan (çarpışma serbest yolu) en uygun veya uygulanabilir yolu bulmalıdır. Çevredeki ortamı robotun algılaması yeteneğini göz önüne alınırsa, lokal ve global iki robotik yol planlamanın ana tipi vardır. Lokal yol planlamasında, robot çalışma alanının tüm özelliklerinin hakkında bilgi sahibi değildir. Alanı keşfederken robotun sensörlerinin aracılığıyla bilgileri alır ve engellerden kaçmak için gerçek zamanda kararlarını vermeli ve pozisyonunu güncellemelidir. Global yol planlamasında da robotun amacı aynıdır. Fakat, robot hareket etmeye başlamadan bile ortam hakkında tüm bilgilere sahiptir. Diğer yandan, robotun çalışma alanının boyutuna göre iki boyutlu (2B) ve üç boyutlu (3B) robotik yol planlama problemi ayrılanmıştır. Ek olarak, ortam statik veya dinamik olabilir. İkincisi ise daha karmaşık çünkü engel pozisyonları, hızları ve yörüngeleri değişebilir, bu yüzden robot ortamın yeni konfigürasyonlarını hemen öğrenmelidir. Bu problem versiyonları çözmek için, ayrı ayrı veya birbirine karıştırarak, literatürde birçok çalışma bulunabilir: 2B ortamda global yol planlaması [27], 3B ortamda yol planlaması [28], dinamik ortamda lokal yol planlaması [29] ...vb.

2B ortamda, sürekli kartezyen düzleminde (cartesian plane) R^2, robot bir nokta olarak düşünülebilmektedir. Bu düzleye konfigürasyon alanı (configuration space- C) denir. Robot $q_0(a,b)$ pozisyondan $q_1(c,d)$ pozisyona değiştirerek 2D düzlemde hareket eder. Ancak, konfigürasyon alanında bazı noktalar müsait değildir. Müsait olanlara boş alan (free space C_{free}) denir ve müsait olmayanlara engeller (obstacles B) denir. Robot B'den hiçbir pozisyonda olamaz, ve engellerle çarpışmayı önlemek zorundadır. Yol planlama probleminin uygulanabilir çözümü aşağıdaki gibi tanımlanabilmektedir:

$P = \{q_0, q_1, q_i, \dots, q_n\}$ halbuki $\forall i \in \{0,1 \dots, n\}, q_i \in C_{free}$, ve q_0 başlangıç noktası, q_n hadef noktasıdır.

$$C_{free} = \{q \in C \mid q \cap (B) = \phi\}$$

$B = \bigcup_{i=1}^{m} B_i$. B tüm engellerin birliği ve m çevredeki toplam engel sayısı temsil etmektedir.

Uygulanabilir yolu (P) engelleri önleyerek başlangıç ve hedef noktalarını birbirine bağlamaktadır.

Yol planlama problemini çözmek için birçok yöntem geliştirilmiştir. Yol planlama işlemi "çalışma alanının haritalanması ve modellenmesi" ve "planlama ve sürüş sistemi" olarak ayrılabilmektedir. Her aşamasında uygun bir yöntem seçilmelidir [30]. Haritalama aşamasında grafik ve matris yoğun bir şekilde kullanılmıştır. Probleme uyan grafiği veya matris belirledikten sonra, en uygun yolu bulmak için optimizasyon algoritmaları kullanılmaktadır.

Literatürde iki robotik yol planlama yönteminin kategorisi tartışılmıştır [30]: klasik yöntemleri ve esnek hesaplama teknikleri.

a. Klasik Yöntemler:

Bu kategorinin algoritmalarına deterministik de denebilir. Klasik yöntemleri iki ana grupta özetlenebilir: grafik araması ve yapay potansiyel alanı (artificial potential field APF) yöntemleri [26].

i. Grafik Araması

Bu teknikte grafik gösterimi kullanılmaktadır. Robot hiçbir engelle çarpışmadan bir noktadan başka bir noktaya ulaşabilirse, bu iki nokta birbirine bağlı olmaktadır. Böylece, yol planlama problemi grafik arama problemini olarak çözülebilir. Kullanılan birçok grafik gösterimi vardır, bazıları aşağıda açıklanmıştır.

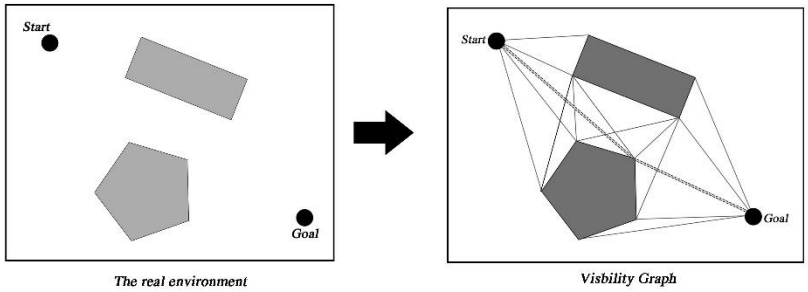
Görünürlük Grafiği (Visibility graph): Başlangıçtan hedef noktasına en kısa yolu oluşturmak için grafik gösterimini kullanan bir tekniktir. Görünürlük grafiği G = (V, E), engellerin köşelerini bağlmaktadır. Yeni grafiğin tepe setinin elemanları, engellerin köşeleriyle aynı ve bağlandığı kenarlar köşelerin bağlantılarıdır. Bu bağlantılar engeller ile çarpışmamasını sağlmaktadır. Dolayısıyla, robotik yol planlama problemi, oluşturulan grafikte en kısa yolu bulma problemine düşürülmüştür [2]. Şekil 1.1 görünürlük grafiğinin gösteriminin bir örneğini göstermektedir.

Voronoi Grafiği: Bu teknik grafik tabanlı gösterimi kullanır ve robotu engellerden uzak tutmaya çalışmaktadır. Grafik engellerin arasından geçen kenarlar oluşturularak

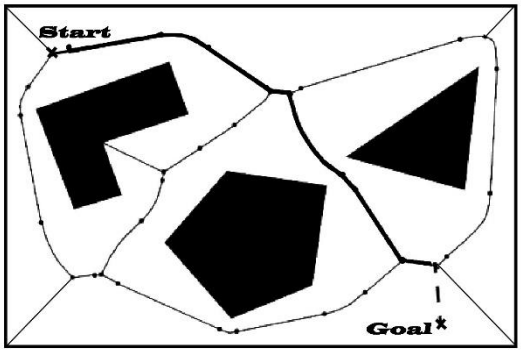
çizilmektedir. Böylece, kenarin her ilgili engelleriyle aynı mesafeyi tutmaya çalışmaktadır. Voronoi grafiğinin gösterimi Şekil 1.2'de sunulmuştur. Oluşturulan grafik başlangıç ve hedef noktası da içermektedir. Problemin grafik gösterimini aldıktan sonra, optimum yolu bulmak için farklı grafik arama algoritmaları kullanılabilmektedir.

ii. APF

APF klasik yöntemlerin ikinci tipi olan Khatib tarafından tanıtılmıştır [31]. APF kavramı iki küvvet kullanımıdır: robotu engellerden uzaklaştıracak kuvvet ve robotu hedef noktaya çektirecek kuvvettir. Engeller yapay itici kuvvete sahip ve hedef noktası yapay çekici bir kuvvete sahiptir. Kuvvetlerin toplamı robotu hedefe yönlendirmekte ve onun hareketini kontrol etmektedir. APF lokal bakışa sahip olduğu için lokal minimaya koyalca takılabilir ve bu yöntem için büyük bir eksiktir. Bu dezavantajı düzeltmek için literatürde bazı APF'nin versiyonları bulanabilmektedir [1, 32].



Şekil 1.1: Görünürlük grafiği



Şekil 1.2: Voronoi grafiği [2, 26]

b. Esnek Hesaplama Yöntemleri:

Klasik yöntemlerin lokal minimaya takılması ve yüksek zaman tüketimi gibi birçok dezavantajı vardır. Bu dezavantajları aşmak için esnek hesaplama yöntemleri kullanılabilmektedir. Genellikle, bu yaklaşımlara olasılıklı yöntemler denir ve kompleks problemleri çözmek için en iyi yöntemlerden biridir. Bu teknikler arasında evrimsel hesaplama tabanlı optimizasyon algoritmaları (evrimsel algoritmalar, sürü zeka algoritmaları, tek çözüm tabanlı meta-sezgisel tarama vb.), sinir ağı tabanlı makine öğrenme yöntemleri ve bulanık mantık tabanlı yaklaşımları içermektedir.

Genetik algoritma (GA), karınca kolonisi optimizasyonu (ant colony optimization-ACO) ve yapay sinir ağları (artificial neural networks ANNs) robot yol planlamasında uygulanan esnek hesaplama yöntemleri örnek olarak aşağıda verilmiştir.

i. GA

GA, Darwin'in evrimi ve doğal seleksiyona dayanan bir sezgisel algoritmasıdır. 1975 yılında Holland tarafından literatüre tanıtılmıştır [33]. GA, mobil robot yolu planlaması dahil çeşitli kompleks preblemleri başarıyla çözmüştür. [4] 'de Achour ve Chaalal, konfigürasyon alanının oluşturulmasından sonra GA'yı kullanmışlardır. Burada GA'nın rolü, konfigürasyon alanının oluşturulan noktalarını birleştirerek ve minimal mesafe dikkate alınarak, çarpışmasız yolu bulmaktadır. AL-Taharwa vd. [34]'da ve Wu vd. [35]' de ortamı temsil etmek için bir matris kullanmışlardır. [34]'da, bireysel dizi boş alandaki düğüm grubunu ifade etmektedir. Bireyin her iki ardışık düğümünün arasındaki mesafenin hesaplanmasıyla en iyi çözümü elde edilmiştir. [35]'de, standart biner şifrelemesi yerine seri numarası kodu kullanılmıştır. Ayrıca, yolun uzunluğun yanında, uygunluk değerleri hesaplamasında yolun düzgünlüğünü (smoothness) de değerlendirildmiştir. GA'nın diğer versiyonları mesela Paralel Elit Genetik Algoritma [36] ve Genetik Programlama [37] bu problemi çözmek için kullanılmıştır.

ii. ACO

ACO, 1990 yıllarda Dorigo ve arkadaşları tarafından önerilen bir sürü algoritmasıdır [38–40]. Bu algoritma, karıncaların besin kaynağı ve yuvanın arasındaki en kısa yolu bilirletmesinin davranışlarını taklit etmektedir. Birçok problemi çözmek için kullanıldı: gezgin satıcı problemi, atama problemi, zamanlama problemi, vb. [41]. Robotik yol planlama da bu problemlerden biridir. [42]'de, Brand vd. matris gösterimi kullanarak

problemi çözmüştür. Önce algoritmayı boş bir ortamda çalıştırarak (hiçbir engel olmadan) çözülmüştür. Bu ortamdaki en kısa yolu bulduktan sonra engeller eklenip ACO'yu yeniden uygulanmıştır. Bazı melez (hybrid) algoritmalar da vardır, örneğin: [43]'de hem global hem de lokal yol planlamasını çözmek için ACO ve APF ile birlikte kullanılmıştır.

iii. ANN

ANN, yapay zeka ve makine öğrenme alanındaki en popüler yaklaşımlardan biridir. İnsan beynindeki biyolojik sinir ağları ve nöronların bilgiyi işlenmesinin davranışları ANN'nin ilham kaynağıdır. Görüntü işleme, konuşma tanıma, bilgisayar görüşü, metin işleme ve sınıflandırma gibi farklı alanda kullanılmaktadır. Onlardan ziyade, ANN mobil yol planlamas problemi çözmek için önemli bir rol oynamaktadır [44-45]. Genel olarak, ANN diğer esnek hesaplama algoritmaları ile birleştirerek bu problemi çözelebilmektedir. [46]'de, bir tarım mobil robotunun en uygun yolu bulmak için ANN ve GA ile birlikte kullanılmıştır. [29]'de ACO, NN'nin ağ yapısının uygun topolojisini sağlamak için kullanılmıştır.

1.2. AAP

AAP literatüre Jefferson ve arkadaşları tarafından tanıtılmıştır [47], ondan sonra Koza tarafından daha ayrıntılı olarak ele alınmıştır. [3]. Robotik yol planlama alt problemi olan AAP, GP vb ECBAP yöntemlerinin performansını test etmek için bir test problemi olarak kullanılmaktadır [3, 6]. Bu problemde, matriste bulunan tüm yiyecek peletlerini toplamak için yapay karınca uygun bir yol bulmaya çalışmaktadır. Karınca tarafından yenen yiyeceklerin sayısı bir çözümün kalitesini (uygunluğu) temsil etmektedir. Problemin aşağıdaki sınırlamaları vardır:

- Yapay karınca çok küçük bir algılama yeteneğine sahiptir, yiyeceğin varlığını ancak karıncanın hemen karşılaştığı ilk bitişik hücrede algılayabilmektedir.
- Karınca sadece üç işlem gerçekleştirebilir: sağa dönüşü (turning right), sola dönüşü (turning left) ve hareket etmesi (moving). Her operasyon bir zaman adımını tüketmektedir.

- Adımların sayısı sınırlıdır: karınca tüm yiyecek peletlerini zaman adımı süresi içinde bulması zorundadır.

Problemin türüne göre matrisin belli bir boyutu vardır. Peletler ve onların matristeki spesifik konumları ile bir "iz" oluşturmaktadır. İz kırılmış (düz değil) ve devamlı değil, boşlukları, zıplamaları ve dönüşleri var, bunun için bu problem zor bir problem haline getirmiştir [10, 47].

Daha önce de belirtildiği gibi, matris önceden tanımlanmış sabit bir boyuta sahiptir. Bu çalışma iki AAP'nin türü incelemektedir: Santa Fe Trail (SFT) ve Los Altos Hills Trail (LAHT). Bu patikalarda beş benzer boşluk vardır:

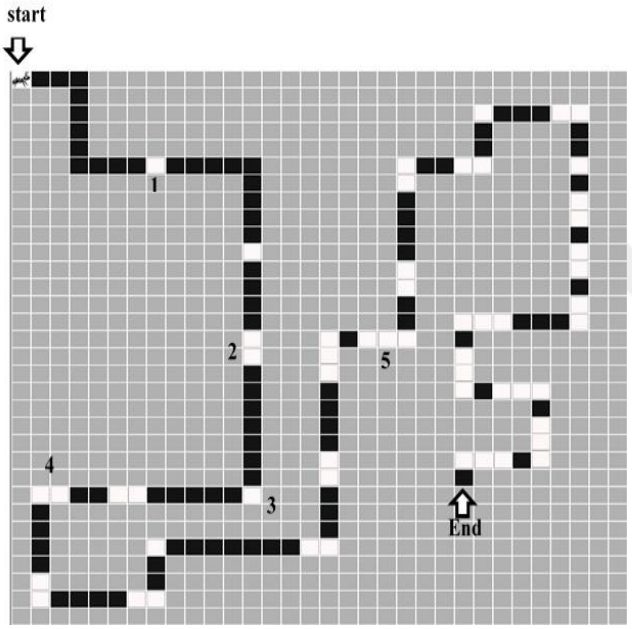
- **Tek boşluk:** Aynı satır veya kolonda iki yiyecek kaynaklarının arasındaki boşluğunda tek bir hücre vardır.
- **Çift boşluk:** İki bitişik boş hücre aynı satır ya da aynı kolondaki iki yiyecek birbirini ayırmaktadır.
- **Köşede tek boşluk:** İzin köşelerinden birinde bir boşluk vardır.
- **Kısa şövalye hareketi (köşede çift boşluk):** aynı yönde olmayan yiyecek kaynakları arasında çift boşluk vardır (bir sonraki yiyecek kaynağını bulmak için bir dönüş gereklidir).
- **Uzun şövalye hareketi (köşede üçlü boşluk):** Kısa şövalye hareketi gibi, fakat iki boşluk yerine uzun şövalye hareketinde üç boş hücre vardır.

a. **SFT**

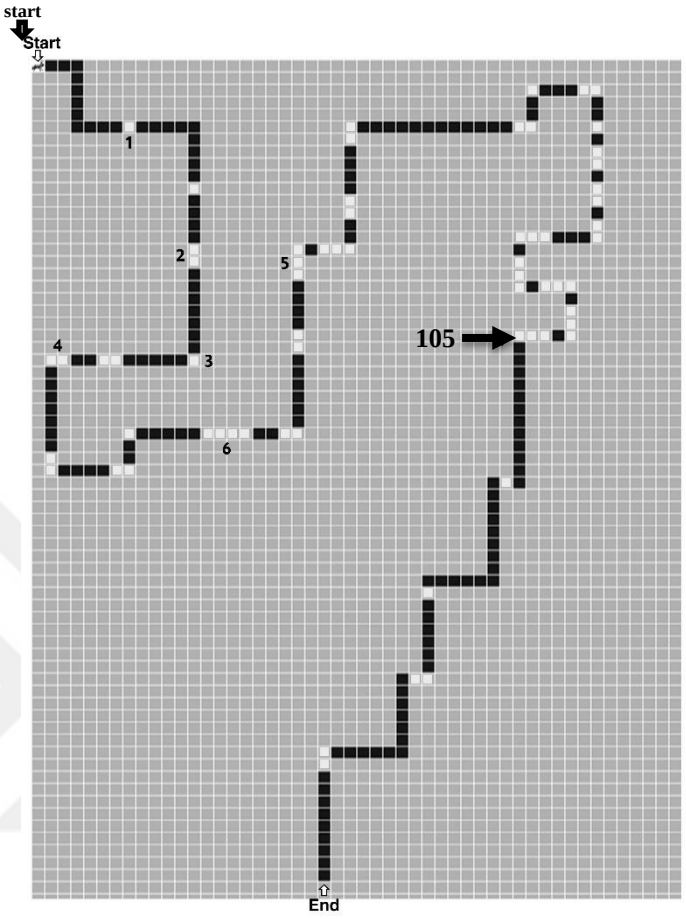
SFT Christopher Langton tarafından tasarlandı ve araştırmacılar tarafından geniş bir şekilde kullanılmaktadır. Bu düz olmayan patika 32 x 32 hücreli toroidal bir matriste tasarlanmıştır ve düzensiz bir şekilde 89 yiyecek peleti üzerine dağılmıştır. İzin uzunluğu 144 hücredir, ve 21 dönüşe sahiptir [3]. SFT Şekil 1.3'te sunulmuştur. İzlerin şekillerindeki kullanılan sembollerin anlamı Tablo 1.1'de verilmiştir.

a. **LAHT**

LAHT, 100 x 100 hücreli bir matriste temsil edilir ve bu patikada 29 dönüş vardır. Üzerinde 157 adet yiyecek parçacığı bulunmaktadır. LAHT, patika uzunluğu 221 olan SFT'nin bir uzantısını sayılmaktadır. Başlangıç noktasından 105 numaralı noktaya kadar patika SFT'ye benzer. Ancak, bundan sonra düzensiz olarak daha fazla yiyecek kaynağı eklenerek patika uzatılmaktadır. Daha önce açıklanan boşluklardan hariç, LAHT'nin aynı satırda dörtlük boşluğu daha vardır. Daha fazla yiyecek peleti sayısı olan LAHT daha geniş bir matrise sahiptir. Matris (100 x 100) çok büyük olduğu ve yolun kalıbının gösterebilmesi için Şekil 1.4'te sadece yiyecek peletlerini içeren patikanın şekli sunulmaktadır. Kalan alan sadece boş (gri rengi) hücrelerdir.



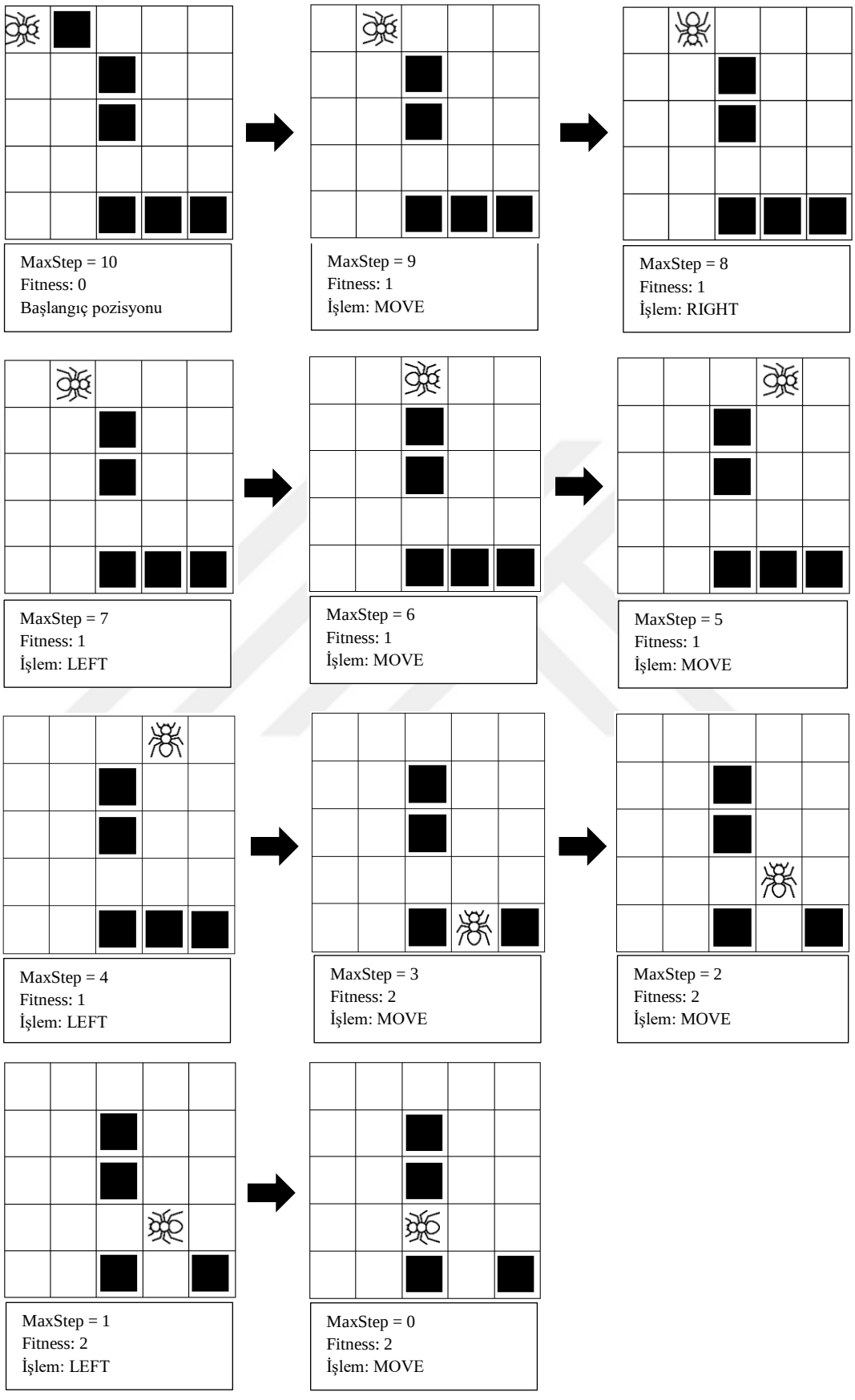
Şekil 1.3: SFT



Şekil 1.4: LAHT

Tablo 1.1: Patikaların şekillerinde kullanılan sembollerin anlamları.

Sembol	**Anlamı**
Start	Başlangıç noktası
1	Tek boşluk
2	Çift boşluk
3	Köşede tek boşluk
4	Kısa şövalye hareketi
5	Uzun şövalye hareketi
6	Dörtlük boşluk
End	Bitiş noktası



Şekil 1.5: AAP'yi çözen bir program örneği

AAP'yi çözmek için bir program örneği ve uygulanan adımlar Şekil 1.5'te gösterilmektedir. Bu örnekte, 5×5 hücreli bir matris üzerinde 6 yiyecek parçacığı vardır. Maximum süre 10 adım olarak verilmektedir. Kullanılan program aşağıda belirtilmiştir:

IF_FOOD_AHEAD (PROGN2 MOVE RIGHT) (PROGN3 LEFT MOVE MOVE).

Program F kümesinden üç fonksiyon tipini ve T kümesinden üç terminal tipini içermektedir. Maksimum adım sayısına ulaşılana veya karınca bütün yiyecek parçacıklarını yiyene kadar program tekrarlanarak uygulanmaktadır. Karınca, programı sol üstteki hücreden (0,0) yürütmeye başlar. İlk fonksiyon, karıncanın tam önündeki hücrede bir yiyecek parçacığının olup olmadığını kontrol eden IF_FOOD_AHEAD'dir, hücrede bir yiyecek olduğu için, ilk parametre (PROGN2 MOVE RIGHT) gerçekleştirilmektedir. PROGN2 iki parametreyi sırayla gerçekleştirir. İlk operasyon MOVE ile karınca önüne doğru bir adım haraket ederken eğer bir yiyecek parçacığı varsa, onu yiyerek ham uygunluk sayısı artırır. İkinci paramtre olan RIGHT ile karınca 90° sağa dönmektedir. Her terminalin çalıştırıldığında kalan adım sayısı bir azaltılır. (PROGN2 MOVE RIGHT) gerçekleştirildikten sonra, $\text{ham uygunluk} = 1 < 6$ ve $\text{kalan adım sayısı} = 8 > 0$ olduğu için program tekrar yürütülecektir. Yeni pozisyonda karınca yeniden IF_FOOD_AHEAD'i gerçekleştirir, bu fonksiyonun dönüşü yanlış olduğu için ikinci parametreyi (PROGN3 LEFT MOVE MOVE) işletecektir. PROGN3 üç parametreyi sırayla yürütür. Buna göre, karınca'nın 90° sola dönüşü ve iki adım ileriye haraket etmesi sağlanmaktadır. Yeni hiç bir yiyecek parçacığı bulunmadığı için, ham uygunluk hala 1'e eşittir, ancak $\text{kalan adım sayısı} = 5$'tir. Karınca aynı mantık ile programı tekrar yürütmektedir. Kalan adım sayısı 0 olduğunda, karınca programı durdurur ve karınca tarafından tüketilen yiyecek parçacığının sayısı bu çözümün ham uygunluğu olarak kaydedilir, bu örnekte çözümün ham uygunluğu 2'dir.

1.3. Literatür Taraması

AAP çözmek için sezgisel ve yapay zeka optimizasyon algoritmaları dahil, literatürde birkaç yöntem kullanılmıştır. Langdon tepe tırmanma (hill climbing) ve ısıl işlem algoritmasını bu probleme uygulamıştır [48].

Evrimsel algoritmalar ve sürü zekası algoritmalar da bu problemi çözmek için çok kullanılmaktadır. Jefferson vd. [47]'de AAP'nin JMT'ine GA uygulamışlardır. Problemi

temsil etmek için sonlu durum otomat (Finite state automata-FSA) ve ANN kullanıldı, ondan sonra FSA'yı kurmak ve ANN'ı tasarlamak için GA kullanılmıştır. [49]'da, gelişmiş bir ACO versiyonu olan MuACOsm AAP'nin beş örneğini: SFT, JMT, LAHT ve iki ek patikayı çözmek için kullanılmıştır. Karıncanın patikadaki tüm yiyecek peletlerini toplamasını sağlayan bir otomat oluşturmak için sonlu durumlu makinelerin (finite-state machines FSM) öğretmesi amacıyla ACO kullanılmıştır.

ECBAP algoritmalarının performansını incelemek için AAP bir kıyaslama problemi olarak kullanılmıştır: GP [3, 6, 50, 51], AP [8], GE [9].

[3]'te, Koza bu problem derin bir şekilde tartıştı ve onun parametrelerini detaylı anlatmıştır. Onun çalışmasında, GP kullanıp AAP'nin SFT'ini çözmüştür. GP, GA'na dayanır, ancak bireyler için ağaç gösterimini kullanmaktadır. Karıncaların yapabileceği işlemleri göz önüne alarak, Koza GP'ya bir terminal kümesi önermiştir [3]. Ayrıca bir fonksiyon kümesi de sağlamıştır [3]. Koza AAP'nin JMT'ini çözmek için GP'yi de kullanılmıştır [6].

Analitik Programlama (AP) SFT için kullanılmıştır [8, 52]. GP GA'ya dayanırken, AP bir optimizasyon aracı olarak herhangi bir evrimsel algoritma ile birleştirilebilir. [8,52]'de Oplatkova ve Zelinka AAP'nin SFT'ini çözmek amacıyla üç farklı evrimsel algoritma: selfOrganizing migrating algoritma , diferansiyel evrim ve ısıl işlem AP ile birlikte kullanmışlardır.

[9]'da Kuroda vd. gramer evriminin geliştirilmiş bir versiyonu önerdi ve AAP'nin üzerinde onun performansı test etmişlerdir. GE backus naur form'u (BNF) kullanıp biner bir dize bazlı genomdan bilgisayar programını (fenotip) oluşturmuştur. GE Arama işlemlerinde genetik işlemlerini taklit etmektedir. Bu geliştirilmiş versiyonda, kurallar seçme aşamasında standart kalanlar seçme metodu yerine GA'nde kullanılan rulet seçmesi kullanılmıştır. Bu önerilen GE daha iyi yakınsama performansı göstermiştir.

2017 yıllında, Dal Piccol vd. tarafından yapılan son çalışmalardan birinde λ-LGP adlı bir linear genetik programlama (LGP) versiyonu geliştirilmiştir. LGP'de bireyler bir turnuva mekanizması kullanarak çiftleşmekte, ama λ-LGP'de popülasyondaki her bireye belli bir sayı lambda (λ) mutasyonu uygulanmaktadır. Seçme sürecinde iki versiyon geliştirildi:

standart koşullu seçme (açgözlü seçim) ve yeni bir versiyonu olan “koşulsuz seçme”. Çalışmalarında AAP’nin SFT, JMT ve LAHT patikalarını kullanmışlardır.

Bu çalışmada, sürü zekası metoduna dayanan ABCP adlı yeni bir ECBAP yaklaşımını AAP için kullanılmıştır. Karaboga vd. ABCP’yi ABC'den geliştirmişlerdir [22]. İlk görünümünde, bazı sembolik regresyon literatür test problemlerine ABCP uygulandı ve onları yarışabilecek bir performansla başarıyla çözmüştür. 2016 yılında, Golafshani ve Ashour, SCC'nin elastik modülünü tahmin etmek için ABCP'yi kullandı ve elde edildiği sonuçları biyocoğrafik tabanlı programlama (BBP) ile kıyasladı [23]. 2018 yılında, Golafshani ve Behnood, geri dönüşümlü agrega betonunun (ERAC) elastik modülünün tahmini için ABCP, GP ve BBP arasında bir kıyaslama çalışması yapmışlardır [24].

Robotik bir yol planlama problem olarak görülen AAP’yi ve onu çözebilen, yeni ya da mevcut ve geliştirebilen yöntem bulmaya çalışan araştırmacılar için ilgi çekici bir noktadır. Langdon ve Poli’nin analitik çalışmalarında bu problemin meşhur algoritmalara bile, GP gibi, ne kadar zor olduğunu gösterdi, ve bu alanda daha ileri çalışmanın gerektiğini göstermişlerdir [10]. ABCP birçok problemin çözmesie başarıyla kullanılmıştır, fakat AAP için ABCP henüz hiç kullanılmamıştır. Bu çalışmada ABCP AAP'ye uygulanmıştır ve bu problemi çözmek için ABCP'ye dayalı yeni bir ECBAP yöntemi önerilmiştir.

2. BÖLÜM

YÖNTEM

Bu bölümde ABCP, shABCP ve bu yöntemlerin AAP'yi çözmek için kullanılan konfigürasyonları ile ilgili ayrıntılı bilgiler verilmektedir. ABCP, ABC'nin bir uzantısı olduğundan dolayı, ilk olarak ABC ve ABCP'nin temel adımları açıklanmaktadır, ondan sonra AAP için ABCP'nin konfigürasyonları detaylı olarak tanımlanmaktadır. Bu bölüm shABCP yani bu tezde geliştirilen yeni ECBAP yöntemi tanıtılarak sonlandırılmaktadır.

2.1. ABC Algoritması

Bal arılarının yiyecek arama davranışlarını taklit eden ABC popüler bir sürü zekası algoritmasıdır [21]. Bu algoritma 2005 yılında Karaboga tarafından literatüre tanıtılmıştır [14]. O zamandan beri, ABC robotik yol planlama dahil farklı problemleri çözmek için kullanılmıştır [19-21, 53].

ABC algoritmasında, koloni üç yapay arı grubundan oluşur: görevli arılar, gözcü arılar ve kaşif arılar. Görevli ve gözcü arılar kovandan ayrıladıklarında hafızalarında bir hedef yiyecek kaynağı vardır ve bu mevcut hedefin komşuluğunda yeni yiyecek kaynağı aramaktadır. Eğer daha iyi bir yiyecek kaynağı bulurlarsa, mevcut olanı terk edip yeni kaynağı tüketmeye başlamaktadır. Aksi halde, mevcut olanı kullanmaya devam etmektedir. Görevli ve gözcü aralarının arasında temel fark, hedef yiyecek kaynaklarını belirleme stratejisidir. Görevli bir arı daha önce ziyaret ettiği kaynağa gider. Ancak, gözcü arı yiyecek kaynakları hakkında bilgi veren görevlilerin danslarını izleyerek hadefini seçmektedir. Bu nedenle, görevli arıların fazında her bir yiyecek kaynağı için sadece bir komşuluk arama uygulanmaktadır. Gözcü arıların fazında ise komşuluk arama sayısı kaynakların kalitesine göre değişmektedir. Hem görevli hem de gözcü arılar var olan

kaynakları iyileştirmeye çalışmaktadır (tüketim-exploitation). Yukarıda da belirtildiği gibi başka bir arı grubu vardır. Bu arılara kaşif denmektedir ve kaşifler yeni yiyecek kaynakları bulmak için rastgele araştırmalar yapmaktadır (keşif-exploration). ABC'de yiyecek kaynakların pozisyonları olası çözümleri ifade etmektedir (popülasyon bireyleri). Popülasyon büyüklüğünü temsil eden kaynak sayısı (food source-FS), görevli arı sayısına eşittir ve bu sayı aynı zamanda gözcü arı sayısı ile de aynıdır. Böylece, koloni büyüklüğü (colony size-CS) görevli ve gözcü arıların toplam sayısına eşittir. Başlangıçta, tüm kaynaklar kaşif arılar tarafından bulunmaktadır. Yani, D boyutlu bir optimizasyon probleminde, bireylerin başlatması Eşitlik (2.1) kullanılarak yapılmaktadır.

$$x_{ij} = x_{min}^{j} + rand[0,1]\ (x_{max}^{j} - x_{max}^{j}) \qquad (2.1)$$

Burada, x_i bir bireyi temsil etmek üzere, x_{min}^{j} ve x_{max}^{j}, sırasıyla j boyutunun alt ve üst sınırıdır. Daha sonra, uygun bir maliyet fonksiyonu kullanılarak her bir kaynağın kalitesi değerlendirilmektedir. Başlangıç fazından durdurma kriteri sağlanana kadar, algoritmada iteratif bir şekilde bazı aşamalar gerçekleştirilmektedir. Her iterasyonda, görevli, gözcü ve kaşif arı fazları sırayla uygulanmaktadır.

Görevli arılar fazında, her bir görevli arı için Eşitlik (2.2)'yi kullanarak mevcut çözümden bir aday çözüm üretilmektedir.

$$v_{ij} = x_{ij} + \phi_{ij}\ (x_{ij} - x_{kj}) \qquad (2.2)$$

Burada j rastgele seçilmiş bir boyut ve ϕ_{ij} [-1, 1] aralığında rastgele bir sayıdır. Bu eşitlikte, x_k popülasyondan rastgele seçilmiş bir yiyecek kaynağıdır ($i \neq k$). Aday çözümü ürettikten sonra, uygunluk değerlerine göre mevcut çözüm ile aday arasında aç gözlü seleksiyon uygulanmaktadır (en uygun kalır). Eğer aday çözüm seçilirse, bu yiyecek kaynağının deneme sayacı sıfırlanır. Yoksa bu yiyecek kaynağının deneme sayacının değeri bir artırılır. Başlangıç fazında, her yiyecek kaynağının deneme sayacının sıfır olarak ayarlandığını unutulmamalıdır.

Görevli arılara benzer bir şekilde, gözcü fazında da her gözcü arı için aday çözümün üretilmesi ve açgözlü seleksiyon işlemleri uygulanmaktadır. Ancak, ondan önce gözcü arının tüketilecek yiyecek kaynağını belirlmesi gerekmektedir. Bunu yapmak için, her

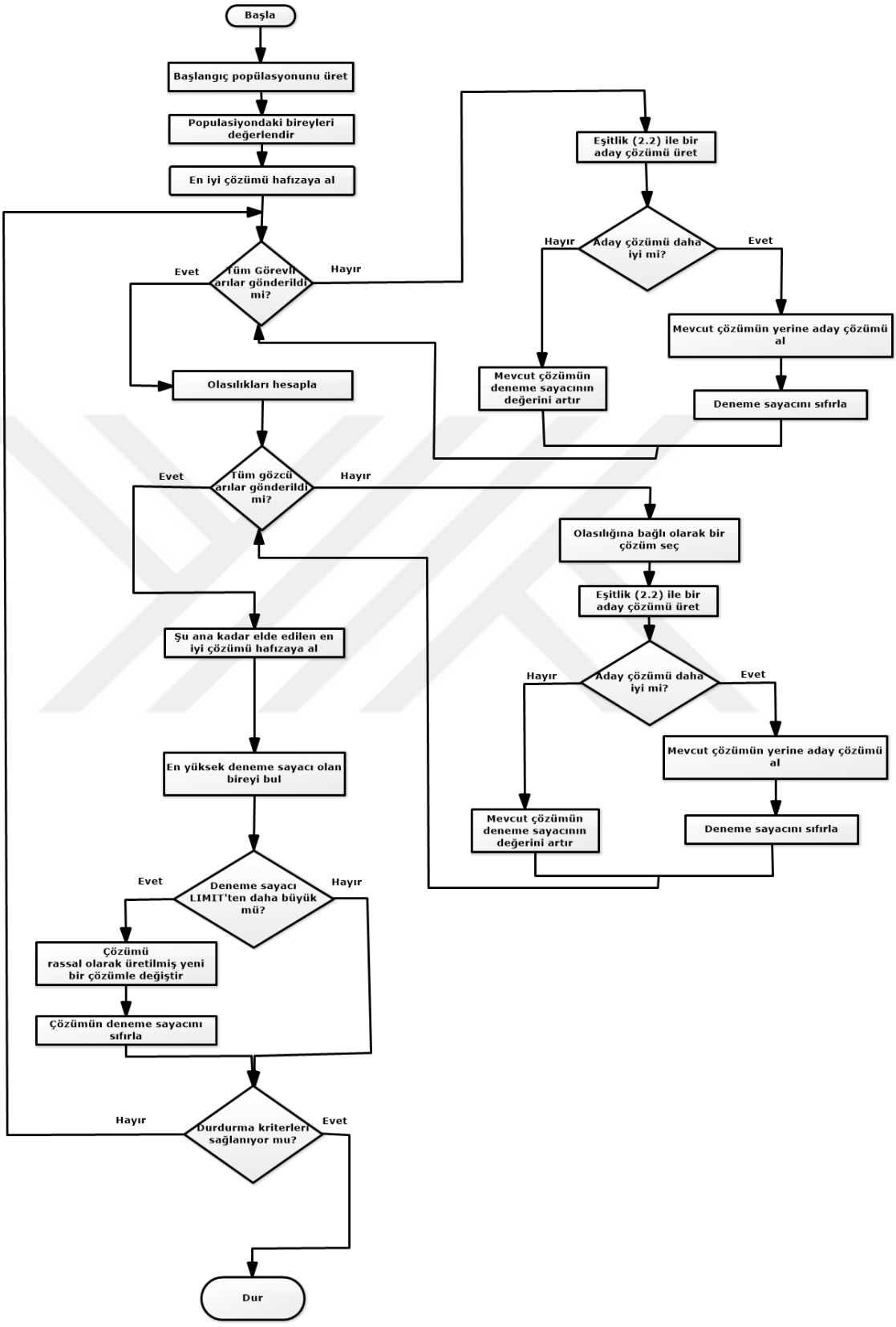
çözümün kalitesine göre bir olasılık değeri p_i hesaplanmaktadır. p_i değerini hesplamak için Eşitlik (2.3) kullanılmaktadır.

$$p_i = \alpha * \frac{fit(x_i)}{fit(x_{best})} + (1 - \alpha) \qquad (2.3)$$

Bu eşitlikte, α [0, 1]'den değerini alan bir parametredir. $fit(x_i)$ çözümün kalitesini temsil etmekte ve değeri Eşitlik (2.4)'de gösterildiği gibi, x_i çözümünün amaç fonksiyonu değeri ($f(x_i)$) kullanılarak belirlenebilmektedir.

$$fit(x_i) = \begin{cases} \frac{1}{1+f(x_i)} & if\ f(x_i) \geq 0 \\ 1 + |f(x_i)| & if\ f(x_i) < 0 \end{cases} \qquad (2.4)$$

ABC algoritmasında, başlangıçta her çözümün deneme sayacı 0 olarak ayarlanmıştır. Görevli veya gözcü arıları fazında açgözlü seleksiyon yapıldığında, eğer aday çözüm seçilirse onun deneme sayacı tekrar sıfırlanır; aksi takdirde, sayacın değeri artmaktadır. İterasyonun sonunda, deneme sayacının değeri en yüksek olan çözüm bulunur ve sayacının değeri "limit" parametresinin değeriyle karşılaştırılır. Eğer limit değerini aşarsa, ilişkili çözümün tükenmiş olduğu bildirilir ve kaşif arı Eşitlik (2.1)'i kullanarak rasgele yeni bir çözüm üretip tükenmiş çözümün yerine onu koyar. Şekil 2.1 ABC algoritmasının akış diyagramını göstermektedir.



Şekil 2.1: ABC algoritmasının akış diyagramı.

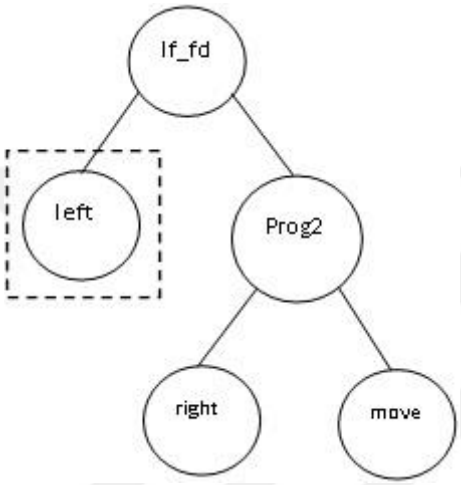
2.2. ABCP ve AAP ile ABCP

ABCP, araştırma uzayını araştırabilmesi için ABC algoritmasının genel yapısını kullanan bir ECBAP yöntemidir. Ayrıca, sabit boyutlu çözüm dizelerinin yerine farklı bir çözüm gösterimi kullanan genişletilmiş bir ABC versiyonu olarak kabul edilmektedir. ABCP algoritmasında çözüm ağaç yapısında temsil edilen bir bilgisayar programıdır. Bir bilgisayar programı problemin özelliklerine göre uygun terminallerden ve fonksiyonlardan oluşmaktadır [3]. Probleme bağlı olarak, bir fonksiyon kümesi standart aritmetik işlemler, programlama işlemleri, mantıksal işlemler, matematiksel işlemler veya diğer alanına özgü fonksiyonları içerebilmektedir. Terminal kümesi ise giriş değişkenleri, sabitler veya problemin özelliklerine göre tanımlanmış bazı değerleri içerebilmektedir. Dolayısıyla, ağacın sadece iki tür düğümü vardır: fonksiyon kümesi F'den değer alan dahili düğümler ve terminal kümesi T'den değerler alan yaprak düğümlerdir. Çözümler ağaç yapısı ile temsil edildiğinden dolayı ABC algoritmasında kullanılan aday çözüm üretme denklemi (Eşitlik 2.2) ABCP algoritmasında geçerli değildir. Bu nedenle, uygun bir bilgi paylaşım mekanizması gerekmektedir.

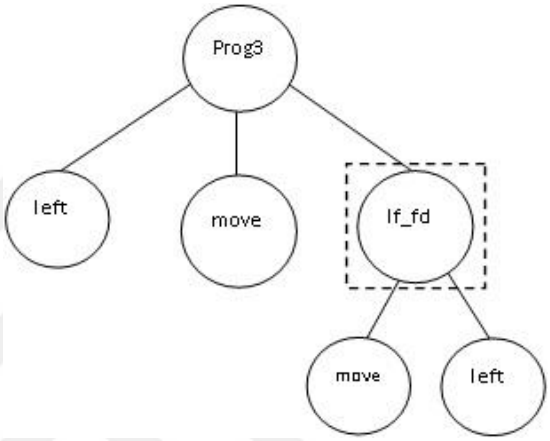
ABCP'nin bilgi paylaşım mekanizması [22]: Bu mekanizmaya göre, x_i çözümünden rasgele bir düğüm seçilecektir. Bir fonksiyon düğümü seçme ihtimali p_f, bir terminal düğümü ise $p_t = 1 - p_f$. [22]'de olduğu gibi, bu çalışmada $p_f = 0.9$ kullanmaktadır. Popülasyondaki mevcut bireylerden bir çözüm x_k $(i \neq k)$ rasgele de seçilecektir. Bu x_k çözümünden p_f ve p_t olasılıkları dikkate alınarak rassal olarak sırasıyla bir fonksiyon ya da bir terminal düğümü seçilmektedir. x_i'ndeki seçilen düğümün bir altı ağacı T_{11} oluşturur, aynı şekilde x_k'deki seçilen düğümü T_{22} altı ağacı oluşturmaktadır. x_i'deki T_{11} yerine T_{22}'yi değiştirerek x_i'den aday bir çözüm v_i üretilmiş olmaktadır. Şekiller 2.2 -2.4 mekanizmayı detayla açıklamaktadır. Şekil 2.2 mecvüt çözümü x_i ve onun rasgele seçilen düğümü göstermektedir. Rasgele seçilen çözüm x_k ve düğümü ise Şekil 2.3'te gösterilmektedir. Oluşturulan aday çözümü v_i Şekil 2.4'te verilmiştir. ABC algoritmasında olduğu gibi, aday çözümün üretilmesinden sonra, mevcut ve aday çözüm arasında açgözlü bir seleksiyon uygulanmaktadır.

ABCP algoritmasında, "grow" ve "full" metotları kombinasyonu, yani "ramped half and half" metodu kullanarak başlangıç popülasyon üretilmektedir. "grow", "full" ve "ramped half and half" metotlarının detayları için [3]'ü incelenebilir. Başlangıç popülasyonda aynı

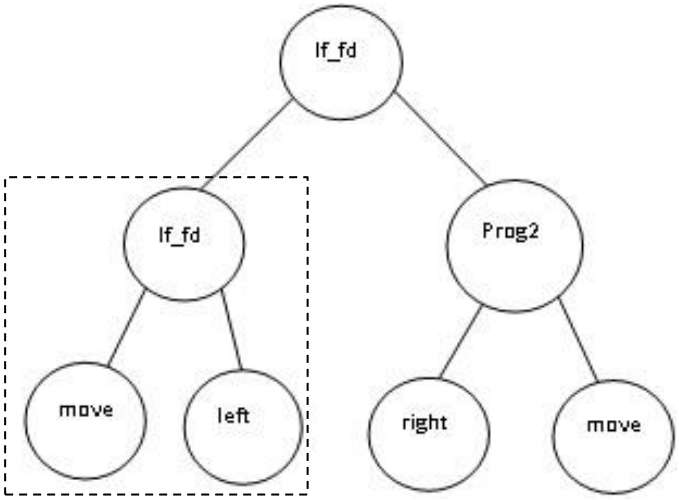
çözüm oluşturulmasına izin verilmez. Kaşif arı fazında, tükenmiş bir yiyecek kaynağı (çözüm) olduğunda, popülasyondaki mevcut çözümlerden farklı olacak şekilde yeni bir çözüm üretilmektedir. Bu fazda "grow" yöntemi kullanılmaktadır. ABCP algoritmasının temel adımlar Şekil. 2.5'te verilmiştir.



Şekil 2.2: Seçilen ağaç T1



Şekil 2.3: Rasgele seçilen ağaç T2



Şekil 2.4: Üretilen ağaç T3

Başlangıç aşaması

- "Ramped half and half " metodu kullanarak başlangıç çözümlerini üret (aynı çözülerin üretimini engelle).
- Eşitlik (2.5)'i kullanarak çözümleri değerlendir.

Repeat

Görevli arılar fazı

Her bir görevli arı için:

- Bilgi paylaşım mekanizmasını kullanarak mevcut çözümden (x_i) bir aday çözüm (v_i) üret.
- Eşitlik (2.5) ile üretilen çözümü (v_i) değerlendir.
- x_i ve v_i arasında en iyi çözümü seç (açgözlü seçim). Eğer v_i seçilirse, v_i ile x_i'yi güncelle ve deneme sayacını sıfırla. Seçilmezse, mevcut çözüm ile devam et ve deneme sayacının değerini artır.

Eşitlik (2.3)'ü kullanarak her çözümün seçilim olasılığını (p_i) hesapla.

Gözcü arılar fazı

Her bir gözcü arı için:

- Seçilim olasılık değerlerini dikkate alarak bir çözüm x_i seç.
- Bilgi paylaşım mekanizmasını kullanarak mevcut çözümden (x_i) bir aday çözüm (v_i) üret.
- Eşitlik (2.5) ile üretilen çözümü (v_i) değerlendir.
- x_i ve v_i arasında en iyi çözümü seç (açgözlü seçim). Eğer v_i seçilirse, v_i ile x_i'yi güncelle ve deneme sayacını sıfırla. Seçilmezse, mevcut çözüm ile devam et ve deneme sayacının değerini artır .

Şimdiye kadar elde edilen en iyi çözümü hafizaya al.

Kaşif arı fazı:

- Tükenmiş yiyecek kaynağını belirle.

- “Grow” metodu kullanarak, rastgele yeni bir çözüm üret (popülasyondaki mevcut çözümlerden farklı olmalıdır).
- Tükenmiş kaynağı üretilen yiyecek kaynağı ile değiştir.

Until (durdurma kriterleri sağlanması)

Şekil. 2.5: ABCP algoritmasının temel adamları

Bu çalışma kapsamında, ABCP AAP 'a uygulanmıştır. [3]’te Koza tarafından bu problem için tanımlanan fonksiyon ve terminal kümeleri kullanılmaktadır. Terminal kümesi yapay karıncanın yapabileceği davranışları içerir. Bu terminaller, RIGHT, LEFT ve MOVE, sırasıyla karıncanın hareket etmeden 90° sağa dönüşünü, hareket etmeden 90° sola dönüşünü ve bir adım ileri doğru hareket etmesini sağlamaktadır. Karınca bir hücreden diğerine hareket ederken, eğer yeni hücrede bir yiyecek parçacığı varsa onu yer, böylece bu yiyecek parçacığını patikadan kaldırır. Fonksiyon kümesinde bir koşullu ve iki koşulsuz değişkenler olan fonksiyonları vardır. Kullanılan foksiyonları aşağıdaki gibi açıklanmaktadır:

IF-FOOD-AHEAD: İki parametresi vardır; karınca yiyecek paraçacığının hemen yanındaki bitişik hücrede olup olmadığını kontrol etmektedir. Eğer bir pelet varsa ilk parametre gerçekleştirilir. Yoksa, ikinci parametre gerçekleştirilmektedir.

PROGN2: Bu koşulsuz bir fonksiyondur. İki parametresi kabul eder ve onlar sıra ile gerçekleştirilmektedir.

PROGN3: Sırayla gerçekleştirilen üç parametresi olan koşulsuz bir fonksiyondur.

Bir çözmün ham uygunluk (raw fitness) değeri, sınırlı bir süre içinde yenen yiyecek parçacıklarının sayısının sayılmasıyla hesaplamaktadır. İz boyunca var olan tüm yiyecek parçacıklarını bulabilen program optimum çözüm sayılmaktadır. Karınca terminaldan bir işlem (RIGHT, LEFT ve MOVE) gerçekleştirdiğinde bir zaman adımı tüketir. Ancak, fonksiyonlar hiçbir zaman adımı tüketmezdir. Karınca, izin verilen tüm zaman adımlarını tüketmesi veya tüm yiyecek parçacıklarını yemesine kadar programı (çözüm) defalarca işletmektedir.

ABCP'de, iyi çözümlere daha küçük değerler veren standartlaştırılmış uygunluk (standardized fitness) kullanılmakta, bir çözümün x_i standartlaştırılmış uygunluğu Eşitlik (2.5) ile hesaplanmaktadır.

$$f(x_i) = MaxFood - rf(x_i) \qquad (2.5)$$

Burada $MaxFood$ patikadaki yiyecek parçacıklarının ilk sayısını ifade eder, $rf(x_i)$ ise x_i çözümünün ham uygunluk değerini ifade etmektedir.

2.3. shABCP

Popülasyon büyüklüğü ECBAP algoritmalarının performansında etkili kontrol parametrelerinden biridir. Ele alınan probleme göre bu parametrenin uygun değerini bulmaya çalışan birçok araştırma yapılmıştır. Fuchs çalışmasında [54], elde edilen sonuçlar ve algoritmanın yakınsama performansı üzerinde GP'nin popülasyon büyüklüğünün etkisini tartışmıştır. Bu çalışmada hem deneysel hem de teorik analizler yapılmıştır. 2003 yılında, Luke vd. araştırma işlemi sürecinde popülasyon büyüklüğünün dinamik güncellenmesi için üç yöntem önermiştir [51]. Bu yöntemler, sembolik regresyon ve AAP gibi farklı kıyaslama problemlerini çözmek üzere GP'ye uygulanmıştır. Tomassini vd. GP'de dinamik popülasyon büyüklüğü için yeni bir teknik önermiştir [55]. Bu çalışmada, popülasyondaki mevcut bireylerin gelişimi dikkate alınarak popülasyon büyüklüğü belirli aralıklarla güncellenmektedir. Yukarıda verilen tüm araştırmalar GP'ye benzer algoritmalarda büyük popülasyonun her zaman faydalı olmadığını göstermektedir.

Bu kısımda, ABCP'nin popülasyon büyüklüğünü (yiyecek kaynağı sayısı) kontrol etmek için yeni bir teknik önerilmiştir. Bu tekniğe göre, başlangıçta popülasyon büyüklüğü araştırma uzayını keşfetmek için yeterli miktarda alınmakta, daha sonra, kalan bireylerin üzerinde daha yoğun bir araştırma gerçekleştirmek için birey sayısı periyodik olarak azaltılmaktadır. Her periyotta, uygun olmayan bireylerden bir miktarı elimine edilmektedir. ABCP'de olduğu gibi, koloni büyüklüğü popülasyondaki birey sayısının iki katına eşittir. Bu tekniği kullanan ABCP'nin yeni versiyonuna shABCP adı verilmiştir. Daha önce ifade edildiği gibi, shABCP'deki bireylerin sayısı periyodik olarak değişmektedir. Her periyotta, önceden belirlenmiş miktarda uygun olmayan (unfit) birey popülasyondan kaldırılmaktadır. Burada shABCP'deki CS'nin her zaman yiyecek

kaynağı sayısıyla orantılı olduğu unutulmamalıdır ($CS = \times FS$). Bu nedenle, bu yöntem ABCP'ye daralma oranı (shrinking rate-shR) ve daralma periyodu (shrinking period-shP) olmak üzere iki ek kontrol parametresi getirmektedir. shR, her shP'te popülasyondan elimine edilecek uygun olmayan bireylerin oranını ifade etmektedir. her shP'de elenecek bireylerin tam sayısını belirlemek için bu oran (shR) başlangıç FS değeri ile çarpılmaktadır. FS, her shP'de alt sınıra, yani her periyotta elimine edilen birey sayısına ($başlangıç\ FS\ \times\ shR$) ulaşana kadar azaltılmaktadır. ABCP'nin çalışması için en az iki bireye ihtiyaç duyması nedeniyle (lütfen bilgi paylaşım mekanizmasına bakınız), alt sınır ($başlangıç\ FS\ \times\ shR$) hiçbir zaman ikiden az olamaz, yani ($initCS\ \times\ shR$) ≥ 4 olmalıdır. shABCP algoritmasının adımları ABCP'ninki ile yaklaşık olarak aynıdır. Başlangıç popülasyonun rastgele üretiminden sonra, görevli arılar fazı, gözcü arılar fazı ve kaşif arı fazı sıra ile tekrarlanmaktadır. Her çevrimin sonunda, kaşif arı fazından sonra, eğer önceden belirlenmiş shP değerine ulaşılırsa, CS'nin değeri $initCS\ \times\ shR$ kadar azaltılır (aynı zamanda FS de $(initCS\ \times\ shR)/2$ kadar azaltılır). shABCP algoritmasının temel adımları Şekil. 2.6'da gösterilmektedir.

Başlangıç
Çevrim sayacını sıfırla (cycleCounter = 0)
Repeat
    Görevli arılar fazı
    Gözcü arılar fazı
    Kaşif arı fazı
    Çevrim sayacının değerini artır (cycleCounter++).
    If (cycleCounter mod shP = 0) then
        If ($CS > (initCS\ \times\ shR)$ ve $(initCS\ \times\ shR)\ \geq\ 4$) then
            Uygun olmayan $(initCS \times shR)/2$ bireyi popülasyondan çıkar.

Until (durdurma kriterleri sağlanılması)

Şekil 2.6: shABCP algoritmasının temel adımları.

3. BÖLÜM

DENEYSEL ÇALIŞMALAR VE TARTIŞMA

Bu bölümde, AAP'yi çözmek için ABCP ve shABCP ile yapılan deneyler sunulmaktadır. İlk olarak, shABCP algoritmasının bazı kontrol parametrelerinin uygun değerlerinin bulunması için bazı deneysel analizler gerçekleştirilmiştir. Daha sonra, shABCP ve ABCP'nin performansları GP, etkili bir mutasyon kullanan LGP (LGP effmut) ve λ-LGP dahil olmak üzere diğer bazı ECBAP yöntemlerinin performansları ile AAP üzerinde kıyaslanmıştır. Deneysel ayarlar, simülasyon sonuçları ve tartışmalar da bu bölümde verilmektedir.

3.1 Patikalarla İlgili Parametrelerin Ayarları

Bu çalışmada, AAP'nin iki farklı patikası olan SFT ve LAHT için deneyler yapılmıştır. Her patika, algoritmaların performansını etkileyebilecek bazı özelliklere sahiptir. Dolayısıyla, adil bir kıyaslama için [7]'de verilen ayarların aynısı kullanılmaktadır, çünkü GP, LGP effmut ve λ-LGP'nin sonuçları [7]'den alınmıştır. İzlerle ilgili parametrelerin ayarları da Tablo 3.1'de sunulmaktadır. Bu tabloda "Max steps", karıncanın programı çalıştırması için izin verilen maksimum zaman adımını göstermektedir.

Tablo 3. 1: Patikalarla ilgili parametrelerin ayarları.

Parametre	Değeri	
	SFT	LAHT
Max yiyecek parçacığı sayısı	89	157
Matris büyüklüğü	32x32	100x100
Max steps	600	3000
Fonkisyon kümesi	F= {IF_FOOD_AHEAD, PROGN2, PROGN3}	
Terminal kümesi	T= {RIGHT, LEFT, MOVE}	

3.2 shABCP'nin Başlangıç Koloni Büyüklüğü, Daralma Periyodu ve Daralma Oranı Parametreleri ile İlgili Deneyler

Daralma periyotu (shP) ve daralma oranı (shR) olmak üzere ABCP'nin parametreleri dışında shABCP'nin iki yeni ek parametresi vardır ve koloni büyüklüğü (CS), shABCP'de başlangıç koloni büyüklüğü (initCS) olmuştur. Bu parametrelerin algoritmanın performansına etkilerini gözlemlemek ve en uygun değerlerini belirlemek amacıyla, diğer ECBAP yöntemleriyle kıyaslama çalışmasına geçmeden önce bazı deneyler yapılmıştır. Bu deneylerde AAP'nin bir patikası olan SFT kullanılmıştır. Durdurma kriteri olarak maksimum değerlendirme sayısı (MaxEval) alınmış ve değeri 20000 olarak sabit tutulmuştur.

shABCP'nin yeni iki paramtresi shP ve shR, sırasıyla maksimum çevrim (MaxCyc) ve initCS ile ilişkilidir ve sabit bir MaxEval değeri için MaxCyc ve initCS arasında bir takas (trade off) vardır. Kaşif arılar tarafından yapılan değerlendirmeler dikkate alınmazsa (çünkü bir çevrimde en fazla bir kaşif çıkabilir), shABCP için initCS ile elde edilebilecek MaxCyc (minMaxCyc) değeri arasındaki ilişki Eşitlik (3.1) ile formüle edilebilir.

$$MaxEval = initCS \; x \; minMaxCyc \qquad (3.1)$$

InitCS ve minMaxCyc arasındaki takasın shABCP'nin performansına etkisi dikkate alınarak, deneylerde farklı ($initCS/minMaxCyc$) oranları kullanılmıştır. InitCS'nin belirlenmesini etkili olan CS ile yiyecek kaynağı sayısı arasındaki ilişki ($CS = 2 \times FS$) veya başlangıç derinliği seviyesi sayısı gibi bazı durumlar dikkate alarak, yedi farklı initCS değeri seçilmiştir, öyle ki Tablo 3.2'de verilen $initCS/minMaxCyc$ oranları 20000 değerlendirme için yaklaşık olarak yakalanmıştır. Bu oranlara ve initCS değerlerine göre, minMaxCyc değerleri elde edilir, daha sonra $initCS \times minMaxCyc$'i MaxEval'a mümkün olunca yakın tutmak için ayarlanmaktadır ($initCS \times minMaxCyc \leq MaxEval$). Bu initCS ve minMaxCyc değerleri Tablo 3.2'de de sunulmaktadır.

Her initCS ve minMaxCyc çifti için, shABCP'nin performansını görmek için dört farklı shP ve beş farklı shR değeri kullanılmıştır. Bu test durumları ve algoritmanın genel konfigürasyonu Tablo 3.3'te gösterilmektedir. Tablo 3.3'te verilen shP ve shR değerleri uygulandığında, sırasıyla periyot değerlerinin ve her bir periyotta popülasyondan elimine edilecek birey sayısının tamsayı kısımlarını almak için "floor function" kullanılmıştır. "Ramped half and half" yöntemini uygularken, eğer her bir derinlik seviyesinde üretilecek birey miktarı bir tek sayı ise, "full" yöntemiyle üretilen birey sayısı "grow" yöntemiyle üretilen birey sayısından daha fazla olacak şekilde yöntem çalıştırılmaktadır.

Table 3. 2: Test durumları için initCS ve minMaxCyc değerleri

initCS	minMaxCyc	Oran ($initCS/minMaxCyc$)
50	400	1/8
70	285	1/4
100	200	1/2
140	142	1
200	100	2
280	71	4
400	50	8

Tablo 3.3: initCS, shR ve shP deneylerinde kullanılan shABCP'nin parametre değerleri

Parametre	Değeri
Başlatma yöntemi	Ramped half and half
Koloni büyüklüğü başlangıç değeri (initCS)	50, 70, 100, 140, 200, 280, 400
Maksimum değerlendirme sayısı (MaxEval)	20 000
Maksimum çevrim sayısı elde edilebilir minimum değeri (minMaxCyc)	400, 285, 200, 142, 100, 71, 50
Başlangıç ağaç derinliği	6
Maksimum ağaç derinliği	17
Limit	initCS
Daralma periyodu (shP)	minMaxCyc/40, minMaxCyc/20, minMaxCyc/8, minMaxCyc/4
Daralma oranı (shR)	4%, 10%, 20%, 50%, 80%
Ham uygunluk	Maximum süre içerisinde tüketilen yiyecek parçacıklarının toplam miktarı.
Koşma sayısı	100.

Her durum için, 100 bağımsız koşma sonucunda ortalama, standart sapma (standard deviation-Std. Dev.), en iyi ve en kötü ham uygunluk değerleri açısından shABCP'nin sonuçları Tablo 3.4-3.10'da verilmiştir. Ayrıca, minimum hesaplama eforu (minimum computational effort-minE) ve başarı oranı (success rate-SR) 100 koşma üzerinden

hesaplanmış ve bu tablolarda sunulmuştur. Her tablo, sabit bir initCS değeri ve farklı shP ve shR değerleri ile shABCP'nin performansını göstermektedir. Her initCS değeri için en iyi performans vurgulanmış ve sonuçları koyu renkle yazılmıştır. Bu çalışmada, karıncayı tüm yiyecek parçacıklarına yönlendirebilecek bir program varsa, bu koşma başarılı kabul edilir, SR tüm başarılı koşma sayısı ve toplam koşma sayısının oranıdır. minE, z olasılıklı optimum bir çözümü bulmak için gereken minimum değerlendirme sayısını tahmin etmektedir. minE hesaplanması için formülasyonlar Eşitlik (3.2) ve Eşitlik (3.3)'te verilmiştir.

$$minE = \underset{eval \in S}{Min}\ I(eval) \tag{3.2}$$

$$S = \{eval \mid eval = initCS \times k \wedge eval \leq MaxEval\ \wedge k \in N\}$$

$$R(z) = [\frac{log(1-z)}{log(1-P(eval)}]$$

$$\mathrm{I(eval)} = \mathrm{M} \times R(\mathrm{z}) \times (k+1) \tag{3.3}$$

Bu eşitliklerde, eval şu anki değerlendirme sayısını temsil etmektedir ve $P(eval)$ ilgili değerlendirme sayısı için elde edilen başarı oranıdır. Bu çalışmada, [3, 7]'de olduğu gibi z olasılığı 0.99 olarak alınmıştır.

Tablo 3. 4: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 50)

shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc /40	4	78.63	**89**	61	9.42	19	155625
	10	78.54	**89**	58	10.04	15	231000
	20	76.83	**89**	64	8.54	18	160650
10	50	77.13	**89**	61	8.52	23	119900
	80	77.39	**89**	64	8.29	21	144275
minMaxCyc /20	4	79.25	**89**	62	8.81	18	239400
	10	79.59	**89**	60	9.02	21	160200
	20	77.97	**89**	62	8.94	22	119700
20	50	76.05	**89**	63	7.66	11	264600
	80	**79.82**	**89**	64	8.02	15	103275
minMaxCyc /8	4	76.72	**89**	62	8.17	17	178200
	10	77.28	**89**	62	8.20	15	147075
	20	76.13	**89**	61	8.72	18	201400
50	50	76.76	**89**	65	7.53	13	250600
	80	77.25	**89**	64	8.63	**23**	**88350**
minMaxCyc /4	4	76.09	**89**	59	8.16	17	154000
	10	75.66	**89**	64	7.79	14	269500
	20	77.38	**89**	**66**	8.07	19	215600
100	50	78.05	**89**	61	8.35	21	140000
	80	77.94	**89**	60	8.80	21	115475

Tablo 3. 5: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 70)

shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc/40 7	4	77.12	**89**	62	9.70	20	80325
	10	78.55	**89**	63	8.74	24	86240
	20	77.04	**89**	63	8.29	24	120190
	50	77.01	**89**	66	7.92	21	135660
	80	76.97	**89**	62	8.56	21	101080
minMaxCyc/20 14	4	77.59	**89**	**65**	8.47	25	128520
	10	**80.38**	**89**	63	8.59	31	77000
	20	79.09	**89**	58	9.27	**32**	66045
	50	78.95	**89**	**65**	8.01	23	92400
	80	77.40	**89**	59	8.80	24	128520
minMaxCyc/8 35	4	76.96	**89**	62	9.22	25	96390
	10	78.85	**89**	**65**	8.44	29	123200
	20	77.17	**89**	63	8.27	23	125125
	50	77.84	**89**	62	8.77	21	132370
	80	77.01	**89**	61	8.61	20	123480
minMaxCyc/4 71	4	77.75	**89**	60	8.85	25	114800
	10	76.91	**89**	60	8.79	17	95760
	20	78.56	**89**	61	8.75	28	**64260**
	50	77.52	**89**	62	7.90	22	138250
	80	78.98	**89**	64	78.98	30	111300

Tablo 3. 6: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 100)

shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc /40	4	79.36	**89**	62	9.12	26	180000
	10	78.33	**89**	63	8.71	33	125800
	20	77.92	**89**	59	9.13	32	**114400**
5	50	76.94	**89**	59	8.26	22	259900
	80	78.54	**89**	63	9.01	27	203500
minMaxCyc /20	4	77.42	**89**	63	8.56	21	275400
	10	76.05	**89**	63	7.50	13	313600
	20	77.93	**89**	**64**	8.29	26	192200
10	50	77.51	**89**	63	7.81	21	240100
	80	76.92	**89**	62	8.92	24	183600
minMaxCyc /8	4	77.76	**89**	62	8.24	20	345600
	10	78.51	**89**	61	8.54	27	258400
	20	77.75	**89**	63	8.31	27	187200
25	50	78.15	**89**	63	8.37	29	189600
	80	76.81	**89**	60	8.93	26	229500
minMaxCyc /4	4	78.51	**89**	63	9.05	34	232500
	10	79.05	**89**	63	8.54	27	216000
	20	79.28	**89**	61	8.53	33	177600
50	50	**79.87**	**89**	63	8.41	**34**	204000
	80	77.30	**89**	62	8.91	26	261800

Tablo 3. 7: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 140)

shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc/40 3	4	79.21	**89**	64	8.98	29	95480
	10	77.12	**89**	64	7.65	20	103600
	20	78.93	**89**	60	8.65	33	73780
	50	75.75	**89**	60	8.89	22	180880
	80	76.03	**89**	60	9.54	25	154980
minMaxCyc/20 7	4	78.94	**89**	63	8.93	32	120540
	10	78.81	**89**	63	8.35	29	105560
	20	78.90	**89**	63	8.03	27	106190
	50	78.90	**89**	64	8.43	31	107730
	80	75.69	**89**	62	8.66	15	246400
minMaxCyc/8 17	4	77.87	**89**	61	9.33	29	129150
	10	**81.08**	**89**	**65**	8.24	**41**	**83930**
	20	78.08	**89**	64	8.33	30	101500
	50	77.62	**89**	63	7.79	21	198800
	80	77.40	**89**	61	9.31	31	127400
minMaxCyc/4 35	4	78.13	**89**	63	9.37	29	120190
	10	78.94	**89**	61	9.03	32	117600
	20	80.31	**89**	64	8.55	40	84000
	50	79.82	**89**	64	8.45	37	98700
	80	79.41	**89**	63	9.10	35	107800

Tablo 3. 8: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 200)

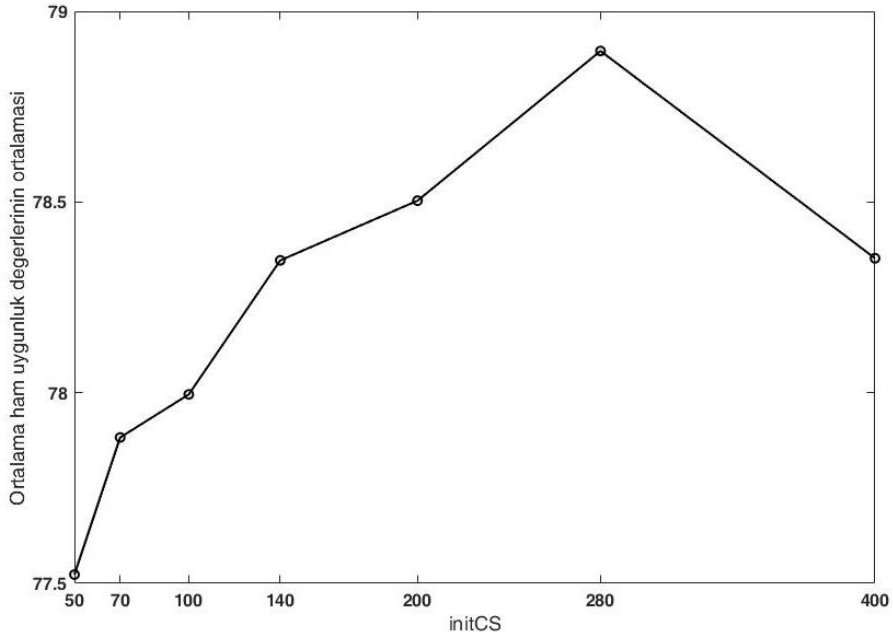
shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc /40 2	4	79.48	**89**	61	8.46	34	**64600**
	10	79.18	**89**	62	8.42	37	54000
	20	75.77	**89**	61	7.88	17	172500
	50	75.92	**89**	61	9.03	20	186000
	80	75.78	**89**	59	9.60	20	191400
minMaxCyc /20 5	4	79.85	**89**	64	8.37	36	84000
	10	79.06	**89**	**65**	8.01	31	91800
	20	78.51	**89**	61	8.13	28	89300
	50	76.54	**89**	59	9.90	26	161600
	80	78.16	**89**	58	9.81	34	105400
minMaxCyc /8 12	4	78.31	**89**	61	9.43	30	131300
	10	80.97	**89**	64	8.38	44	78000
	20	**81.48**	**89**	61	7.98	**46**	64900
	50	78.39	**89**	62	9.29	31	99000
	80	77.65	**89**	60	10.21	30	109800
minMaxCyc /4 25	4	77.61	**89**	55	10.51	30	129000
	10	78.82	**89**	62	9.08	25	154700
	20	80.35	**89**	60	8.80	44	78300
	50	79.69	**89**	61	8.61	32	85000
	80	78.55	**89**	61	9.70	32	115500

Tablo 3. 9: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 280)

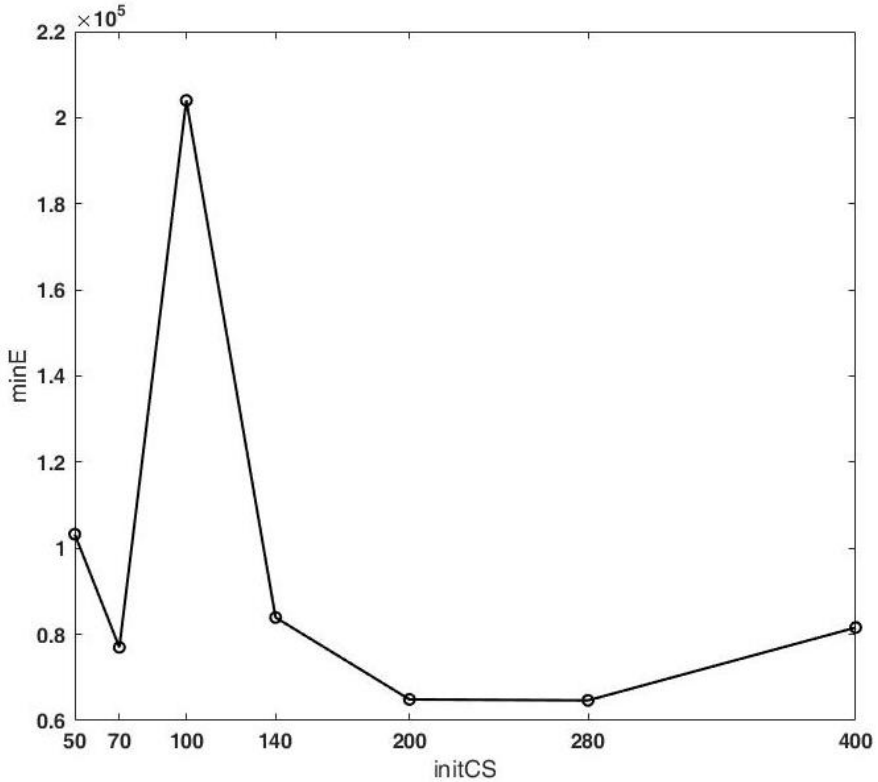
shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc /40 1	4	78.52	**89**	58	9.11	34	78540
	10	77.37	**89**	62	8.15	26	97440
	20	76.50	**89**	60	9.32	25	102900
	50	76.34	**89**	60	9.62	23	181440
	80	78.25	**89**	61	10.15	31	129220
minMaxCyc /20 3	4	81.48	**89**	61	8.91	47	66080
	10	78.95	**89**	61	8.54	34	76160
	20	77.69	**89**	58	9.04	31	96320
	50	77.86	**89**	61	9.79	26	157080
	80	76.89	**89**	61	9.74	16	271320
minMaxCyc /8 8	4	80.81	**89**	**63**	9.37	36	107800
	10	81.59	**89**	**63**	8.57	46	75600
	20	80.25	**89**	60	8.42	35	100100
	50	78.67	**89**	59	9.78	34	104160
	80	76.76	**89**	60	9.90	18	192780
minMaxCyc /4 17	4	79.01	**89**	62	9.44	25	157080
	10	79.48	**89**	**63**	9.35	32	120960
	20	**82.68**	**89**	62	8.51	**52**	**64680**
	50	80.75	**89**	57	8.90	37	100800
	80	78.07	**89**	**63**	9.67	26	163520

Tablo 3. 10: Farklı shP ve shR değerleri için shABCP sonuçları (initCS = 400)

shP	shR (%)	Ortalama	En iyi	En kötü	Std. Dev.	SR (%)	minE
minMaxCyc /40	4	78.33	**89**	61	8.88	30	92000
	10	76.93	**89**	58	8.61	23	161200
	20	76.26	**89**	57	9.69	25	156400
1	50	76.15	**89**	58	10.93	20	211200
	80	74.23	**89**	58	10.29	21	192000
minMaxCyc /20	4	81.53	**89**	61	8.45	**46**	**64000**
	10	79.09	**89**	63	8.69	32	88200
	20	78.25	**89**	58	9.33	35	98400
2	50	76.82	**89**	60	10.07	22	171600
	80	76.75	**89**	59	10.24	19	215600
minMaxCyc /8	4	77.55	**89**	60	10.64	23	180000
	10	**81.78**	**89**	**65**	8.27	44	81600
	20	81.37	**89**	63	8.53	43	75600
6	50	77.05	**89**	61	9.99	20	214200
	80	79.76	**89**	61	10.31	24	173400
minMaxCyc /4	4	78.56	**89**	59	9.79	19	224400
	10	78.72	**89**	60	10.21	25	173400
	20	81.36	**89**	64	8.63	41	91800
12	50	78.62	**89**	61	9.80	26	149600
	80	77.94	**89**	61	10.12	21	192000



Şekil 3.1: Farklı initCS değerleri için ortalama ham uygunluk değerlerinin ortalaması



Şekil 3. 2: Farklı initCS değerleri için en iyi konfigürasyonun minE'si

Tablo 3.4’ten, minE değeri en iyisi olmasa bile, initCS = 50 olduğunda shR = 80% ve shP = minMaxCyc/20 konfigürasyonuyla en iyi ortalama ham uygunluk değeri elde edilmiştir.

Tablo 3.5 initCS = 70 olduğunda shABCP’nin sonuçlarını göstermektedir. Periyot minMaxCyc / 20'ye eşit olduğunda en iyi ortalama ham uygunluk değeri de elde edildiği görülmektedir. Ancak, bu konfigürasyonda karşılık gelen shR 10% olmuştur. En iyi SR değeri de bu ayarlar ile sağlanmıştır.

Tablo 3.6’da gösterildiği gibi, initCS = 100 için, shP = minMaxCyc/4 ve shR=50% ayarlandığında daha iyi sonuçlar bulunmuştur. Aynı parametrelerle en iyi SR'nin elde edildiğine dikkat edilmelidir.

initCS = 140 olduğunda, tüm ölçümler dikkate alarak, Tablo 3.7'nin gösterdiği gibi, shP=minMaxCyc/8 ve shR = 10% en iyi sonuçları veren konfigürasyondur.

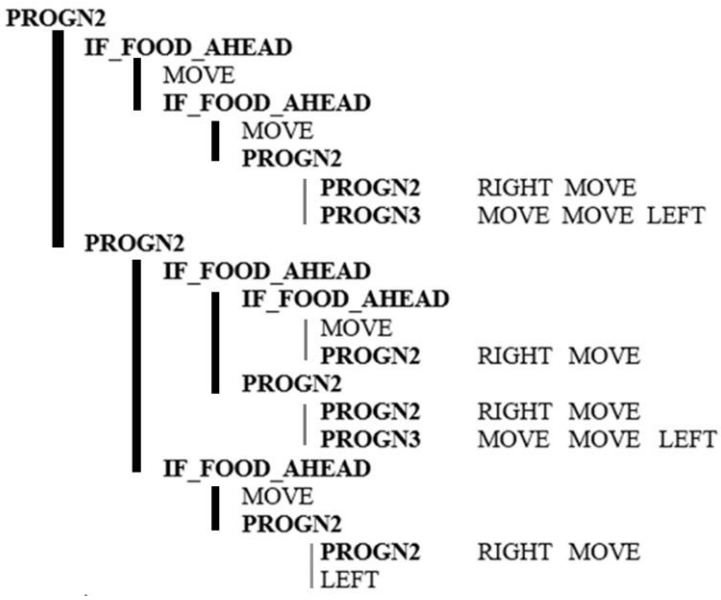
Tablo 3.8’de verilen sonuçlara göre, initCS = 200 durumunda, shR = 20% ve shP=minMaxCyc/8 olduğunda en iyi ortalama ham uygunluk değeri ve en yüksek başarı oranına ulaşılmıştır.

Tablo 3.9 initCS = 280 sonuçları göstermektedir. Koşmaların en kötüsü hariç, shABCP, shP=minMaxCyc/4 ve shR initCS'nin 20%'si olduğunda tablonun tüm kolonlarında en iyi sonuçları vermiştir. Ayrıca, bu parametreler ayarları kullanılarak ortalama ham uygunluk ve SR değerleri sadece bu initCS değeri değil, aynı zamanda elde edilen tüm initCS değerleri için bu konfigürasyonla en iyi sonuçları bulunmuştur.

Son olarak, Tablo 3.10'da verilen sonuçlar için initCS 400'e eşittir. Ortalama ham uygunluk değeri ve 100 koşmanın en kötüsü açısından, shP = minMaxCyc/8 ve shR = 10% konfigürasyonu en iyi sonuçları vermiştir.

Bu deneyler sırasında, sabit bir değerlendirme sayısı için farklı initCS değerleri kullanılmış ve her initCS için minMaxCyc hesaplanmıştır. Her initCS ve minMaxCyc çifti için, shABCP'nin parametreleri değiştirilerek (shP ve shR) deneyler gerçekleştirilmiştir. Önerilen parametreler minMaxCyc ve initCS ile orantılı olduğu için genel bir sonuç çıkarılabilmektedir. Tablo 3.4-3.10'da gösterilen sonuçlardan shP ve shR parametrelerinin değerler seçimlerini önemi ve initCS'nin shABCP’nin performansına

etkisi kolayca gözlenebilmektedir. shABCP'nin AAP'de performansını görmek amacıyla, belirli değerlendirme sayısının değeri sabit tutularak farklı initCS ve minMaxCyc oranları kullanıldığında her initCS değeri için Tablo 3.4-3.10'daki ortalama ham uygunluk kolonunda bulunan değerlerinin ortalaması hesaplanmıştır. Bu ortalama değerler Şekil 3.1'de gösterilmiştir. Şekil 3.1 incelendiğinde, initCS = 280'e kadar initCS artarken ortalama değerin de artışı görülebilir. 280 değerinden sonra, ortalama tekrar azalmıştır. Her initCS değeri için en iyi konfigürasyonun minE'sinin değerleri Şekil 3.2'de gösterilmiştir. initCS=280 ile en düşük, initCS = 100 olduğunda ise en yüksek minE değeri bulunmuştur. Aslında, initCS = 100 hariç, Şekil 3.1 ve Şekil 3.2'de benzer eğilimler görülmektedir. Sonuç olarak, deneylere göre, initCS = 280 olduğunda en iyi shABCP performansı elde edilmiştir, bu initCS/minMaxCyc = 4 oranı ile temsil edilebilmektedir. Bu şartlar altında (initCS = 280 ve initCS/minMaxCyc = 4) shABCP'nin AAP'nin SFT'de elde edilen en iyi sonuçlarına shP = minMaxCyc/4 ve shR = 20% konfigürasyonu ile ulaşılmıştır. Parametre ayarlama deneylerinde elde edilen en iyi çözüm Şekil 3.3'te gösterilmektedir.



Şekil 3.3: Parametre ayarlama deneylerinde elde edilen en iyi çözüm

3.3. Kıyaslama Çalışması

Bu kısımda, AAP'nin SFT ve LAHT patikaları üzerinde shABCP, ABCP, GP, LGP effmut ve λ-LGP algoritmaları arasında bir kıyaslama çalışması yapılmıştır. GP, LGP effmut ve λ-LGP'nin sonuçları [7] 'den alındığı için, ABCP ve shABCP için [7] 'ye benzer parametre ayarları kullanılmıştır. ABCP ve shABCP başlangıç popülasyonu için "ramped half and half" yöntemi kullanılmıştır. [22]'de olduğu gibi bu çalışmada da ABCP için "limit" değeri CS'ye eşit tutulmaktadır, shABCP için ise "limit" değeri initCS'ye eşittir. Önceki bölümde yapılan deneysel çalışmalar dikkate alınarak, shABCP'nin ek parametreleri shP=minMaxCyc/4, shR = 20% ve initCS/minMaxCyc = 4 olarak ayarlanmıştır. Tablo 3.11 kıyaslama çalışmasında kullanılan algoritmaların ortak parametrelerinin değerilerini göstermektedir.

Adil bir kıyaslama için, maximum değerlendirme sayısı durduma kriteri olarak kullanılmaktadır ve [7]'deki gibi her konfigürasyon için 100 bağımsız koşma gerçekleştirilmiştir. Sonuçlar altı farklı performans ölçütü ile sunulmuştur. 100 koşma sonucunda elde edilen ortalama, en iyi ve en kötü ham uygunluk değeri, standart sapma, başarı oranı ve minE değerleri Tablo 3.12 ve Tablo 3.13'te, sırasıyla SFT ve LAHT için gösterilmektedir. En iyi değerler koyu ile renk yazılmıştır.

Tablo 3. 11: Algoritmaların ortak parametrelerinin ayarları

Parametre		Değeri
MaxEval	SFT	20 000, 100 000, 200 000
	LAHT	50 000, 500 000
Başlangıç ağaç derinliği		6
Maksimum ağaç derinliği		17
Ham uygunluk		Maximum süre içerisinde tüketilen yiyecek parçacıklarının toplam miktarı.
Koşma sayısı		100.

Tablo 3.12'ye göre, MaxEval = 20000 olduğunda, ortalama ham uygunluk ve minE ölçütleri dikkate alındığında ABCP, GP ve LGP effmut dahil olmak üzere kullanılan bütün algoritmalar arasında shABCP en iyi sonuçları göstermektedir. shABCP'nin SR'sine yakın olmasına rağmen, LGP effmut bu durumda en yüksek SR'ye ulaşmayı başarmıştır. MaxEval 100000 veya 200000'e eşit olduğu zaman, λ–LGP, ABCP ve shABCP, SFT'de benzer ortalama ham uygunluk değeri elde etmektedir. Bu durumlarda, ortalama maliyet değerleri bakımından, shABCP ABCP'den biraz daha iyi, λ–LGP ise shABCP'den biraz daha iyidir. En iyi minE'ler ve SR'lere de λ–LGP kullanarak ulaşılmaktadır.

Tablo 3. 12: SFT'de simülasyon sonuçları

MaxEval	CS (initCS, NP)	Yöntem	Ortalama	Std. Dev.	En İyi	En kötü	SR (%)	minE
20 000	100	ABCP	78.60	9.35	**89**	63	34	234000
		LGP effmut	80.98	-	**89**	-	**59**	440000
		GP	64.06	-	**89**	-	5	-
	280	shABCP	**82.68**	8.51	**89**	62	52	**64680**
100000	100	λ–LGP λ=5	**88.95**	-	**89**	-	**98**	**71000**
	500	ABCP	87.58	4.00	**89**	69	83	232000
	632	shABCP	88.11	2.91	**89**	73	87	221830
200 000	100	λ–LGP λ=10	**89**	-	**89**	**89**	**100**	**47000**
	1000	ABCP	88.75	1.55	**89**	77	95	288000
	1000	GP	79.74	-	**89**	-	34	-
	894	shABCP	88.88	1.01	**89**	79	98	239056

Tablo 3. 13: LAHT'de simülasyon sonuçları

MaxEval	CS (initCS, NP)	Yöntem	Ortalama	Std. Dev.	En İyi	En kötü	SR (%)	minE
50 000	500	GP	101.7	9.35	-	-	0	Inf
		LGP effmut	113.48	-	-	-	0	Inf
		ABCP	135.43	11.49	155	**116**	0	Inf
	100	λ–LGP λ=5	138.48	-	**157**	-	**12**	Inf
	444	shABCP	**142.97**	13.84	**157**	**116**	9	**2480184**
500000	500	λ–LGP λ=10	149.14	-	**157**	-	22	11745000
	500	ABCP	154.43	6.31	**157**	116	48	3596000
	500	GP	124.94	-	**157**	-	4	-
	1412	shABCP	**156.24**	1.16	**157**	**149**	**59**	**2702568**

LAHT, AAP'nin diğer patikalarına göre en zor patika olarak bilinmektedir. Bu patikanın zorluğu Tablo 3.13'teki sonuçlardan görülebilmektedir. Tablonun gösterdiği gibi, pek çok algoritma ya hiç bir optimum çözümü bulamamakta ya da bulmak için yüksek bir hesaplama eforuna ihtiyaç duymaktadır. MaxEval = 50000 olduğunda, sadece shABCP ve λ –LGP 100 koşma boyunca bazı optimum çözümler bulabilmektedir. Yani, diğer tüm algoritmaların SR'si 0%'dır. λ–LGP'nin SR'si, shABCP'ninden büyük olmasına rağmen, shABCP en iyi ortalama ham uygunluk değerine sahiptir (142.97 vs 138.48), ve minimum

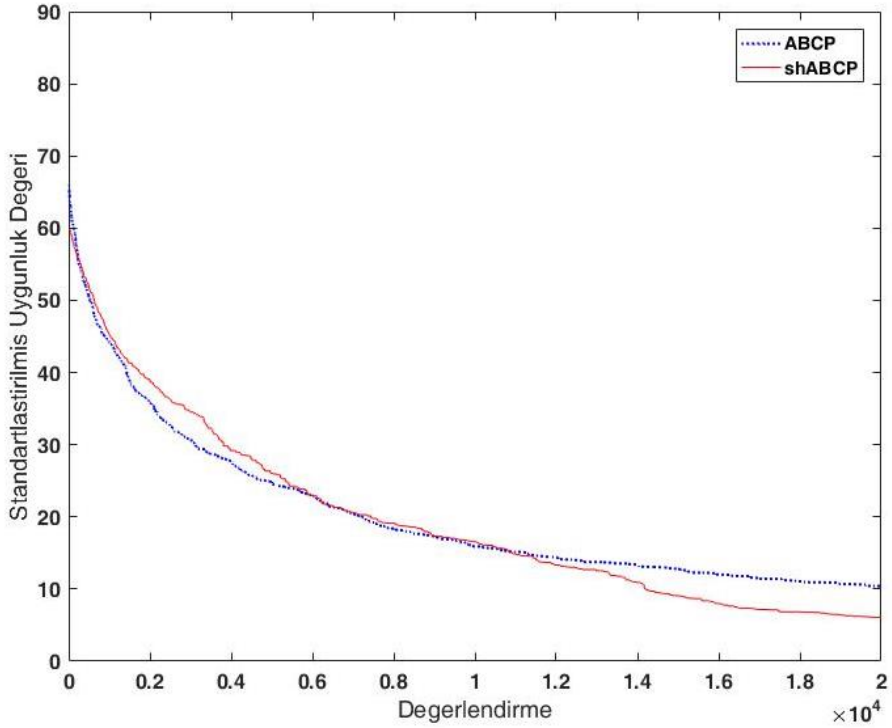
hesaplama eforu daha kabul edilebilirdir. SR değeri sıfır olan algoritmalar arasında ABCP λ–LGP ve shABCP'nin sonuçlarına daha yakın olan en yüksek ortalama ham uygunluk değerine sahiptir. MaxEval = 500000 olan diğer durumda, shABCP ve ABCP, diğer tüm algoritmalardan büyük ölçüde daha iyi performans göstermektedir. Bir kez daha, shABCP tüm metriklerin mevcut değerlerini (ortalama ham uygunluk, Std. Dev., SR ve minE) göz önünde bulundurarak en iyi performansı göstermektedir.

GP kıyaslamaya dahil edildiğinde hem SFT hem de LAHT'deki ortalama ham uygunluk ve SR değerleri için diğer algoritmalar GP'den oldukça iyi performans göstermektedir.

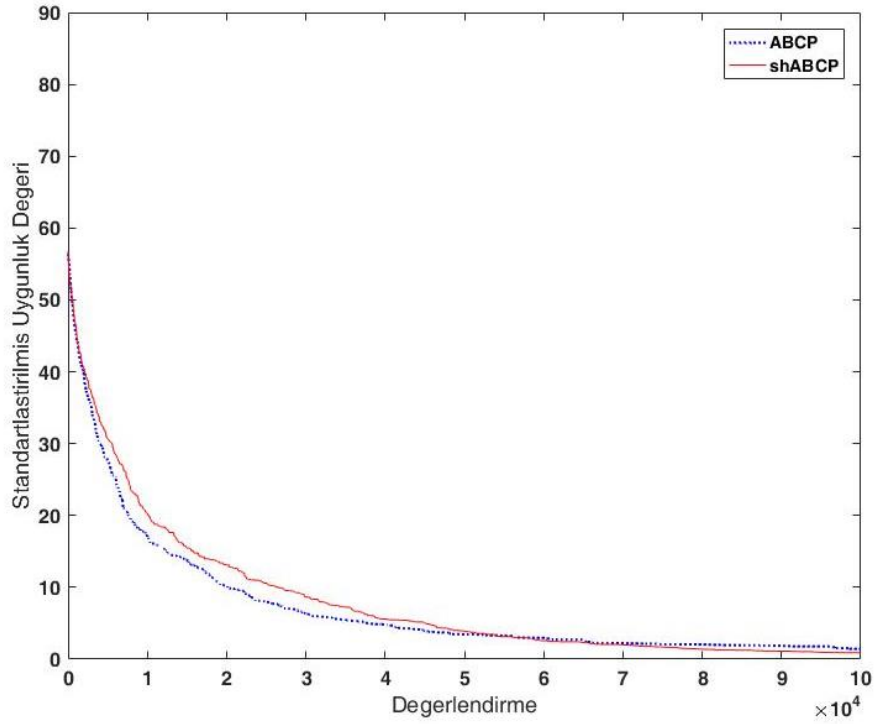
Genel olarak, ABCP GP'den çok daha iyi sonuçlar bulmakta ve ayrıca bu temel versiyon diğer ECBAP yöntemleriyle kıyaslama yapıldığında da rekabetçi bir performans göstermektedir. shABCP, beş farklı durumun üçünde en iyi sonuçları elde etmekte, diğer iki durumda ise iyi sonuçlara çok yakın sonuçlar bulmaktadır. Yani, simülasyon sonuçlarına göre, ABCP ve shABCP'nin AAP'yi çözmek için etkin bir şekilde kullanılabileceği ve shABCP'de kullanılan önerilen daralma mekanizmasının, ABCP'nin performansını AAP'de önemli ölçüde iyileştirdiği görülmektedir.

Şekil 3.4 ve Şekil 3.5'te shABCP ve ABCP'nin beş farklı durum için yakınsama grafikleri verilmiştir. Şekil 3.4 SFT için üç farklı değerlemdrirme sayısı kullanılarak elde edilen sonuçlarının yakınsama grafiklerini sunmaktadır. Şekil 3.5 ise LAHT'nin iki farklı durumu için yakınsama grafiklerini göstermektedir. Bu grafiklerde ham uygunluk yerine standartlaştırılmış uygunluk değeri kullanılmaktadır. Standartlaştırılmış uygunluk değeri ham uygunluk değerinden Eşitlik (2.5) ile hesaplanmaktadır. Şekiller incelendiğinde shABCP'nin optimuma yakınsama performansının ABCP'den daha iyi olduğu görülmektedir.

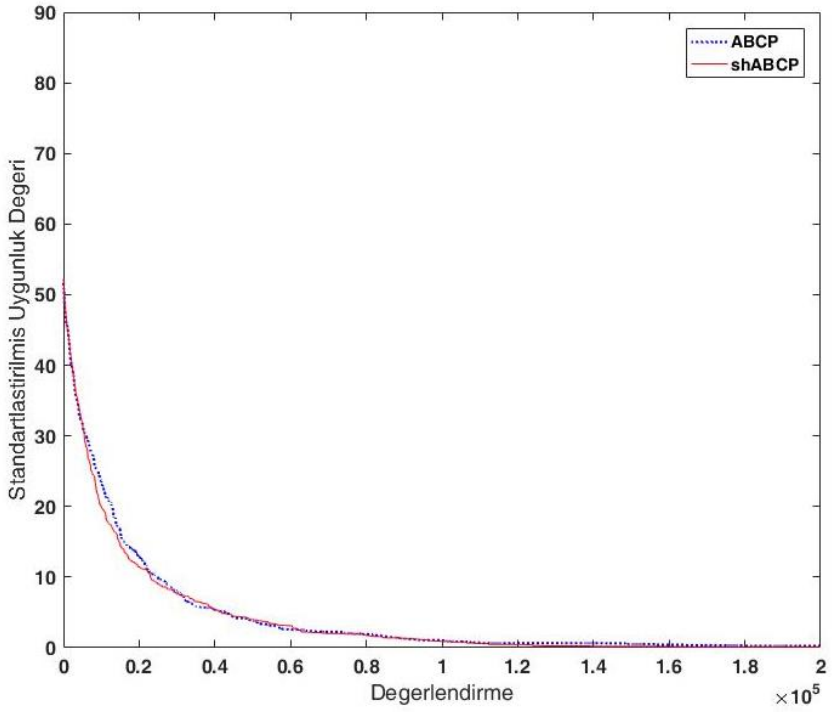
Grafiklerde genel olarak başlangıçta ABCP'nin yakınsaması daha hızlıdır. Burada bu deneylerde shABCP'nin ABCP'den daha fazla birey ile araştırmaya başlamasının etkisi olduğu düşünülmektedir. Değerlendirmeler ilerledikçe, bir noktadan sonra genellikle shABCP'nin yakınsama performansı ABCP'den daha iyi hale gelmektedir.


ABCP
shABCP
Standartlastirilmis Uygunluk Degeri
Degerlendirme
×10^4

(a)

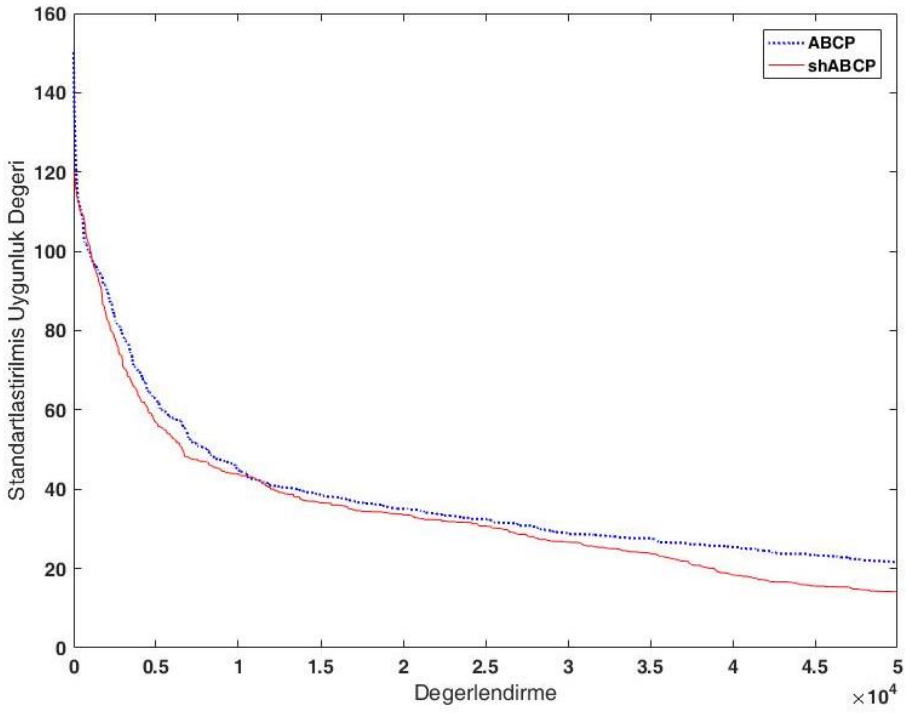

ABCP
shABCP
Standartlastirilmis Uygunluk Degeri
Degerlendirme
×10^4

(b)

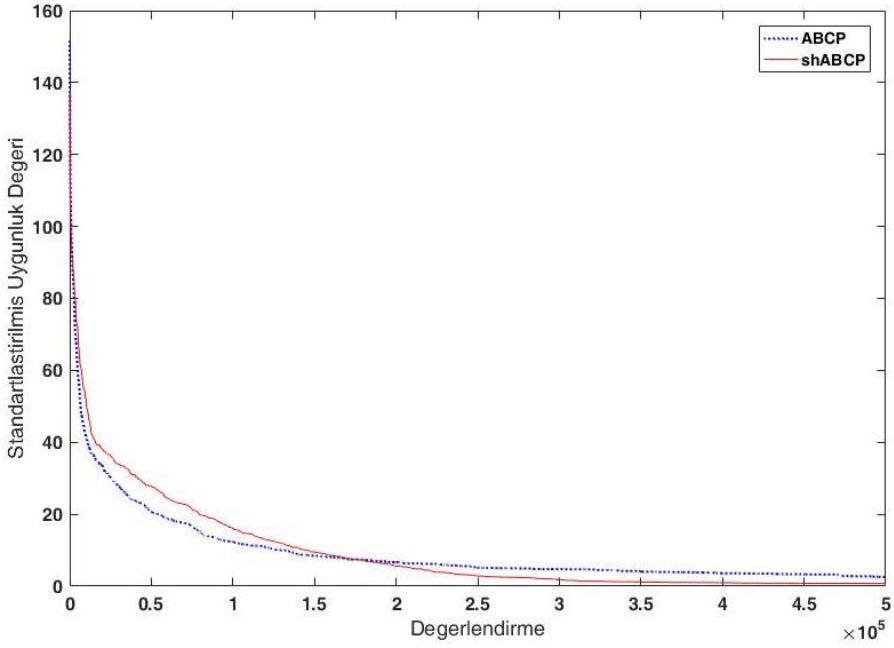


(c)

Şekil 3.4: shABCP ve ABCP'nin SFT için yakınsama grafikleri. (a) 20000 değerlendirme sayısı. (b) 100000 değerlendirme sayısı. (c) 200000 değerlendirme sayısı.



(a)



(b)

Şekil 3.5: shABCP ve ABCP'nin LAHT için yakınsama grafikleri. (a) 50000 değerlendirme sayısı. (b) 500000 değerlendirme sayısı

4. BÖLÜM

SONUÇ VE ÖNERİLER

Bu çalışmada, robotik yol planlama probleminin bir alt türü olan AAP çözülmüştür. AAP'nin patikalarına göre birçok versiyonu vardır. Bu çalışma SFT ve LAHT adı verilen iki farklı patika üzerinde gerçekleştirilmiştir. Her bir patikanın kendine ait özellikleri bulunmakta ve bu özellikler patikanın zorluğunu da göstermektedir. Bu tez çalışmasında AAP için ABCP ve shABCP olmak üzere iki yeni ECBAP yöntemi kullanılmaktadır. ABCP 2012 yılında literatüre tanıtılmıştır [22], ancak henüz AAP'ye uygulanmamıştır. Literatürde ECBAP yöntemlerinin performansını değerlendirmek için AAP'nin bir kıyaslama problemi olarak kabul edilmesi nedeniyle, sembolik regresyon problemlerinde başarılı olan ABCP'nin bu alanda da başarısını test etmek amacıyla bu probleme uygulaması yapılmıştır. Aynı zamanda, daha önce de belirtildiği gibi ABCP'nin sembolik regresyon gibi diğer alanlardaki ümit verici sonuçları dikkate alınarak, çok önemli ve popüler bir araştırma ve uygulama alanı olan robotik yol planlama problemi için bu algoritma yeni bir çözüm yöntemi olarak önerilmiştir. Bu çalışmada ayrıca, ABCP'nin optimum çözüme yakınsama performansını artırmak için yeni bir versiyonu, shABCP, geliştirilmiş ve yukarıda belirtilen benzer motivasyonlarla AAP'ye uygulanmıştır.

İlk olarak, robotik yol planlama ile ilgili bazı genel bilgiler, tanımlar ve bu alanda özellikle de AAP için kullanılan çözüm yöntemleri açıklanmaktadır. Bu çalışmada APP'nin çözümü için önerilen ECBAP yöntemleri bal arılarının yiyecek arama davranışlarından ilham alınarak geliştirilmiş bir sürü zekası algoritması olan ABC'ye dayanmaktadır. Önerilen yöntemlerin ayrıntılı olarak açıklanmasından sonra, ABCP parametrelerine eklenen ve shABCP'de kullanılan yeni kontrol parametreleri üzerinde bazı parametre ayarlama çalışmaları yapılmıştır. Son olarak, bu yöntemlerin AAP'de performansını görmek için literatürde yer alan bazı diğer ECBAP teknikleri (GP, LGP effmut ve λ-LGP) ile bir kıyaslama çalışması yapılmıştır. shABCP ABCP temellidir

ve periyodik bir şekilde FS'yi azaltarak popülasyon büyüklüğü güncelleme tekniğine sahiptir. shP parametresi kaç çevrimde bir popülasyonda daraltma yapılacağını ve kaç adet daralma periyodunun olacağını, shR parametresi ise her bir periyotta popülasyondan elimine edilecek uygun olmayan (unfit) bireylerin yüzdesini kontrol etmektedir. shABCP'de FS ile birlikte CS de azaldığı için CS yerine initCS kullanılmaktadır. Uygun olmayan bireylerin periyodik olarak elenmesi, algoritmanın kalan bireylerinin etrafında daha fazla araştırma yapmasına izin vermektedir (tüketim-exploitation). Parametre ayarlama çalışmaları initCS ile minMaxCyc'nin sabit bir MaxEval değeri için uygun bir oranı, shP ve shR için yapılmıştır. Bu deneylere göre, bu parametreler kıyaslama çalışmasında initCS/minMaxCyc = 4, shP=minMaxCyc/4 ve shR = 20% olarak ayarlanmıştır.

Kıyaslama çalışmasında üçü SFT, ikisi LAHT kullanmak üzere beş farklı durum dikkate alınmıştır. Genel olarak ABCP, GP'den çok daha iyi sonuçlar bulmakta ve ayrıca bu temel versiyon diğer ECBAP yöntemleriyle karşılaştırıldığında da rekabetçi bir performans göstermektedir. shABCP ise beş durumun üçünde en iyi sonuçları elde etmekte, diğer iki durumda da en iyi sonuçlara oldukça yakın değerler üretmektedir. Sonuç olarak, simülasyon çalışmalarına göre, ABCP ve shABCP'nin AAP'nin çözümünde etkin bir şekilde kullanılabileceği ve önerilen daralma mekanizmasının shABCP'de işletilmesinin ABCP'nin AAP'deki performansını önemli bir şekilde iyileştirdiği tespit edilmiştir.

Dinamik popülasyon büyüklüğü yaklaşımı ABCP'ye ilk kez bu çalışmada uygulanmış ve bu çalışmadan önce ABCP ve shABCP robotik yol planlama probleminin çözümü için hiç kullanılmamıştır. Bu nedenle, ileride yapılacak çalışmalar için birçok öneride bulunmak mümkündür. İlk olarak, dinamik popülasyon büyüklüğü yaklaşımını iyileştirmek için bir çalışma yapılabilir. Bu tez çalışmasında sadece daralma davranışı uygulanmaktadır. Gelecekte, popülasyon büyüklüğü ihtiyaçlara göre dinamik olarak (arttırılarak veya azaltılarak) güncellenebilir. İkinci olarak, AAP için shABCP ve ABCP'nin tüm kontrol parametreleri için deney tasarımı yapılabilir. shABCP, sembolik regresyon gibi diğer alanlarda da test edilebilir. Ek olarak, diğer robotik yol planlama problemlerini çözmek için ABCP ve shABCP kullanılabilir. Son olarak, ABCP ve shABCP yöntemlerinin performansı robotik yol planlama problemlerini çözmek için kullanılabilecek daha farklı yöntemler de dikkate alınarak daha geniş kapsamda bir kıyaslama çalışması gerçekleştirilebilir.

KAYNAKÇA

[1] P. Vadakkepat, K. C. Tan, and W. Ming-liang, "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning," in ***Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)***, 2000, pp. 256–263.

[2] J. C. Latombe, *Robot Motion Planning*, 1st ed. New York: Springer US, 1991, pp. 651.

[3] J. R. Koza, *Genetic Programming On the Programming of Computers by Means of Natural Selection*, 6th ed. Cambridge, MA, USA: MIT Press, 1992, pp. 609.

[4] N. Achour and M. Chaalal, "Mobile Robots Path Planning using Genetic Algorithms," in ***The Seventh International Conference on Autonomic and Autonomous Systems ICAS***, 2011, pp. 111–115.

[5] P. Junjie and W. Dingwei, "An Ant Colony Optimization Algorithm for Three dimensional Path Planning," in ***2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)***, 2017, pp. 564–568.

[6] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. London, England: MIT Press, 1994, pp. 746.

[7] L. F. Dal Piccol Sotto, V. V. De Melo, and V. V. Basgalupp, "λ -LGP : an improved version of linear genetic programming evaluated in the Ant Trail problem," ***Knowledge and Information Systems*, vol. 52, no. 2**, pp. 445–465, 2017.

[8] Z. K. Oplatkova and I. Zelinka, "Santa Fe Trail for Artificial Ant with Analytic Programming and Three Evolutionary Algorithms," in ***First Asia International Conference on Modelling & Simulation (AMS'07)***, 2007, pp. 334–339.

[9] T. Kuroda, H. Iwasawa, T. Awgichew, and E. Kita, "Application of Improved Grammatical Evolution to Santa Fe Trail Problems," in ***Natural Computing. Proceedings in Information and Communications Technology*, vol. 2**, 2010, pp. 218–225.

[10] W. B. Langdon and R. Poli, “Why Ants are Hard,” in ***Genetic Programming 1998: Proceedings of the Third Annual Conference***, 1998, pp. 193–201.

[11] M. Brameier and B. Wolfgang, *Linear Genetic Programming*, 1st ed. New York: Springer US, 2007, pp. 322.

[12] Julian F. Miller, “Cartesian Genetic Programming,” in *Natural Computing Series*, Springer, Berlin, Heidelberg, 2011, pp. 17–34.

[13] C. Ferreira, “Gene Expression Programming : A New Adaptive Algorithm for Solving Problems,” ***Complex Systems*, vol. 13, no. 2**, pp. 87–129, 2001.

[14] D. Karaboga, “An idea based on Honey Bee Swarm for Numerical Optimization,” *Technical Report TR06, Erciyes University*, no. TR06, pp. 10, 2005.

[15] D. Karaboga and B. Akay, “A comparative study of Artificial Bee Colony algorithm,” ***Applied Mathematics and Computation*, vol. 214, no. 1**, pp. 108–132, 2009.

[16] J. A. Bullinaria and K. AlYahya, “Artificial Bee Colony Training of Neural Networks,” ***Nature Inspired Cooperative Strategies for Optimization (NICSO 2013). Studies in Computational Intelligence*, vol. 52**, pp. 191–201, 2014.

[17] D. Karaboga, B. Akay, and C. Ozturk, “Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks,” ***Modeling Decisions for Artificial Intelligence*, vol. 4617**, pp. 318–329, 2007.

[18] D. Karaboga and C. Ozturk, “A novel clustering approach : Artificial Bee Colony (ABC) algorithm,” ***Applied Soft Computing*, vol. 11, no. 1**, pp. 652–657, 2011.

[19] K. S. Kaswan, S. Choudhary, and K. Sharma, “Applications of Artificial Bee Colony Optimization Technique : Survey,” in ***2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)***, 2015, pp. 1660–1664.

[20] J. C. Bansal, H. Sharma, and S. S. Jadon, “Artificial bee colony algorithm: a survey,” ***International Journal of Advanced Intelligence Paradigms (IJAIP),***

vol. 5, no. 1/2, pp. 123–159, 2013.

[21] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," ***Artificial Intelligence Review*, vol. 42, no. 1**, pp. 21–57, 2014.

[22] D. Karaboga, C. Ozturk, N. Karaboga, and B. Gorkemli, "Artificial bee colony programming for symbolic regression," ***Information Sciences*, vol. 209**, pp. 1–15, 2012.

[23] E. M. Golafshani and A. Ashour, "Prediction of self-compacting concrete elastic modulus using two symbolic regression techniques," ***Automation in Construction*, vol. 64**, pp. 7–19, 2016.

[24] E. M. Golafshani and A. Behnood, "Automatic regression methods for formulation of elastic modulus of recycled aggregate concrete," ***Applied Soft Computing*, vol. 64**, pp. 377–400, 2018.

[25] D. Gregory and J. Michael, *Computational Principles of Mobile Robotics*, 2nd ed. New York: Cambridge University Press, 2010, pp. 407.

[26] N. Leena and K. K. Saju, "A survey on path planning techniques for autonomous mobilerobots," ***IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)***, pp. 76–79, 2014.

[27] V. Pshikhopov, *Path Planning for Vehicles Operating in Uncertain 2D Environments*. Elsevier Inc., 2017, pp. 303.

[28] J. Liang, "A path planning algorithm of mobile robot in known 3D environment," ***Procedia Engineering*, vol. 15**, pp. 157–162, 2011.

[29] Z. W. Wen, M. K. Li, and L. J. Wang, "Path Planning for UUV in Dynamic Environment," in ***2016 9th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1**, 2016, pp. 211–215.

[30] N. Sariff and N. Buniyamin, "An Overview of Autonomous Mobile Robot Path Planning Algorithms," in ***4th Student Conference on Research and Development,***

2006, pp. 183–188.

[31] O. Khatib, “Real-Ttime Obstacle Avoidance For Manipulators And Mobile Robots.,” in ***Proceedings. 1985 IEEE International Conference on Robotics and Automation***, 1985, pp. 500–505.

[32] S. Chen, Z. Yang, Z. Liu, and H. Jin, “An Improved Artificial Potential Field based Path Planning Algorithm for Unmanned Aerial Vehicle in Dynamic Environments,” in ***2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)***, 2017, pp. 591–596.

[33] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press Cambridge, MA, USA, 1992, pp. 228.

[34] I. AL-Taharwa, A. Sheta, and M. Al-Weshah, “A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment,” ***Computer Science*, vol. 4, no. 4**, pp. 341–344, 2008.

[35] M. Wu, E. Chen, Q. Shi, L. Zhou, Z. Chen, and M. Li, “Path planning of mobile robot based on improved genetic algorithm,” in ***2017 Chinese Automation Congress (CAC)*, 2017, vol. 12, no. 11**, pp. 6696–6700.

[36] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, “Parallel Elite Genetic Algorithm and Its Application to Global Path Planning for Autonomous Robot Navigation,” ***IEEE Transactions on Industrial Electronics*, vol. 58, no. 10**, pp. 4813–4821, 2011.

[37] H. Yamamoto, “Robot path planning by genetic programming,” ***Artificial Life and Robotics*, vol. 2, no. 1**, pp. 28–32, 1998.

[38] Marco Dorigo, “Optimization, learning and natural algorithms,” Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[39] M. Dorigo, V. Maniezzo, and A. Colorni, “Positive Feedback as a Search Strategy,” *Tech rep., 91-016, Dip Elettronica, Politecnico di Milano, Italy*, pp. 20, 1999.

[40] M. Dorigo, V. Maniezzo, and A. Colorni, “The Ant System: Optimization by a Colony of Cooperating Agents,” ***IEEE Transactions of Systems, Man and Cybernetics-Part B*, vol. 26, no. 1**, pp. 29–41, 1996.

[41] M. Dorigo and G. Di Caro, “Ant colony optimization: a new meta-heuristic,” in ***Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2**, 1999, pp. 1470–1477.

[42] B. Michael, M. Michael, W. Nicole, and Y. Xiao-Hua, “Ant Colony Optimization Algorithm for Robot Path Planning,” in ***2010 International Conference On Computer Design And Appliations (ICCDA 2010)*, vol. 3**, 2010, pp. 436–440.

[43] H. Mei, Y. Tian, and L. Zu, “A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment,” ***International Journal of Information Technology*, vol. 12, no. 3**, pp. 78–88, 2006.

[44] I. Engedy and G. Horvath, “Artificial neural network based mobile robot navigation,” in ***2009 IEEE International Symposium on Intelligent Signal Processing*,** 2009, pp. 241–246.

[45] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: A survey,” ***Robotics and Autonomous Systems*, vol. 86**, pp. 13–28, 2016.

[46] N. Noguchi and H. Terao, “Path planning of an agricultural mobile robot by neural network and genetic algorithm,” ***Computers and Electronics in Agriculture*, vol. 18**, no. 2–3, pp. 187–204, 1997.

[47] D, Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, and R. Korf, “The Genesys System: Evolution as a Theme in Artificial Life,” *Technical Report CA 90024-1596*, pp. 15, 1991.

[48] W. B. Langdon, “Fitness Causes Bloat: Simulated Annealing, Hill Climbing and Populations,” *Technical Report: CSRP-97-22, School of Computer Science, The University of Birmingham*, pp. 24, 1997.

[49] D. S. Chivilikhin, V. I. Ulyants ev, and A. A. Shalyto, “Solving five

instances of the artificial ant problem with ant colony optimization," in ***IFAC Proceedings Volumes*, 2013, vol. 46, no. 9**, pp. 1043–1048.

[50] S. Christensen and F. Oppacher, "Solving the artificial ant on the Santa Fe trail problem in 20,696 fitness evaluations," ***Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, no. 1**, pp. 1574, 2007.

[51] S. Luke, C. G. Balan and L. Panait, "Population Implosion in Genetic Programming," in ***GECCO 2003, Genetic and Evolutionary Computation Conference*, vol. 2724** 2003, pp. 1729–1739.

[52] Z. K. Oplatkova and I. Zelinka, "Investigation on Evolutionary Synthesis of Movement Commands," ***Modelling and Simulation in Engineering***, pp. 12, 2009.

[53] M. A. Contreras-cruz, V. Ayala-ramirez, and U. H. Hernandez-belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," ***Applied Soft Computing Journal*, vol. 30**, pp. 319–328, 2015.

[54] M. Fuchs, "Large Populations Are Not Always The Best Choice In Genetic Programming," in *GECCO'99* ***Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, vol. 2**, 1997, pp. 1033–1038.

[55] M. Tomassini, L. Vanneschi, J. Cuendet, and F. Fernandez, "A New Technique for Dynamic Size Populations In Genetic Programming," in ***Proceedings of the 2004 Congress on Evolutionary Computation*, vol. 1 and 2**, 2004, pp. 486–493.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı:	Fateh BOUDARDARA.
Uyruğu:	Cezayir.
Doğum Tarihi ve Yeri:	30.05.1991 - Setif, Cezayir.
Medeni Durum:	Bekar.
e-mail:	Fateh.boudardara@gmail.com.
Yazışma Adresi:	Köşk Mahallesi, Talas Blv. No:79, 38030 Melikgazi/Kayseri.

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Lisans	Higher School of Computer Science, Computing Systems	2015
Lise	Maazou Said , Setif-Agleria	2010

YABANCI DİL

İngilizce

Fransızca

Türkçe

YAYINLAR

1. T. Zerrouki, M. M. Abu Shquier, A. Balla, N. Bousbia, I. Sakraoui, F Boudardara, “Adapting eSpeak to Arabic Language:Converting Arabic Text to Speech Language using eSpeak”, International Journal of Reasoning-based Intelligent Systems 11(1):pp76-89, 2019.

2. Boudardara F., Görkemli B., "Application of Artificial Bee Colony Programming to Two Trails of the Artificial Ant Problem", 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT 2018), ANKARA, TÜRKIYE, 19-21 Ekim 2018, pp.1-6.

3. Boudardara F., Görkemli B., "Solving the Artificial Ant on the Santa Fe Trail Problem Using Artificial Bee Colony Programming", International Conference on Mathematical Advances and Applications (ICOMAA2018) , İSTANBUL, TÜRKIYE, 11-13 Mayıs 2018, pp-55.