Journal of the Faculty of Engineering and Architecture of Gazi University 38:4 (2023) 2435-2449

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

A binary Jaya algorithm with selection-based local search mechanism for large-scale optimization problems

Ahmet Özkış[1]*, Murat Karakoyun[2]

[1]Department of Digital Forensic Engineering, Engineering Faculty, Necmettin Erbakan University, 42090, Meram, Konya, Türkiye
[2]Department of Computer Engineering, Engineering Faculty, Necmettin Erbakan University, 42090, Meram, Konya, Türkiye

Highlights:

- A novel local search module named as ELSM has been proposed to improve the search ability of the binary optimization algorithms.
- The proposed ELSM has been implemented on binary Jaya algorithm.
- The proposed algorithm has been applied on Cap and M* problem sets that included in UFLPs

Keywords:

- Jaya algorithm
- Binary optimization
- Uncapacitated facility location problem
- Local search mechanism
- Metaheuristic algorithms

Article Info:

Research Article
Received: 29.04.2022
Accepted: 10.12.2022

DOI:

10.17341/gazimmfd.1111302

Correspondence:

Author: Ahmet Özkış
e-mail: ahmetozkis@gmail.com
phone: +90

Graphical/Tabular Abstract

In this study, the JayaX-ELSM algorithm was suggested by applying an enhanced local search mechanism to the binary Jaya algorithm. The flowchart of the suggested algorithm is given in Figure A.

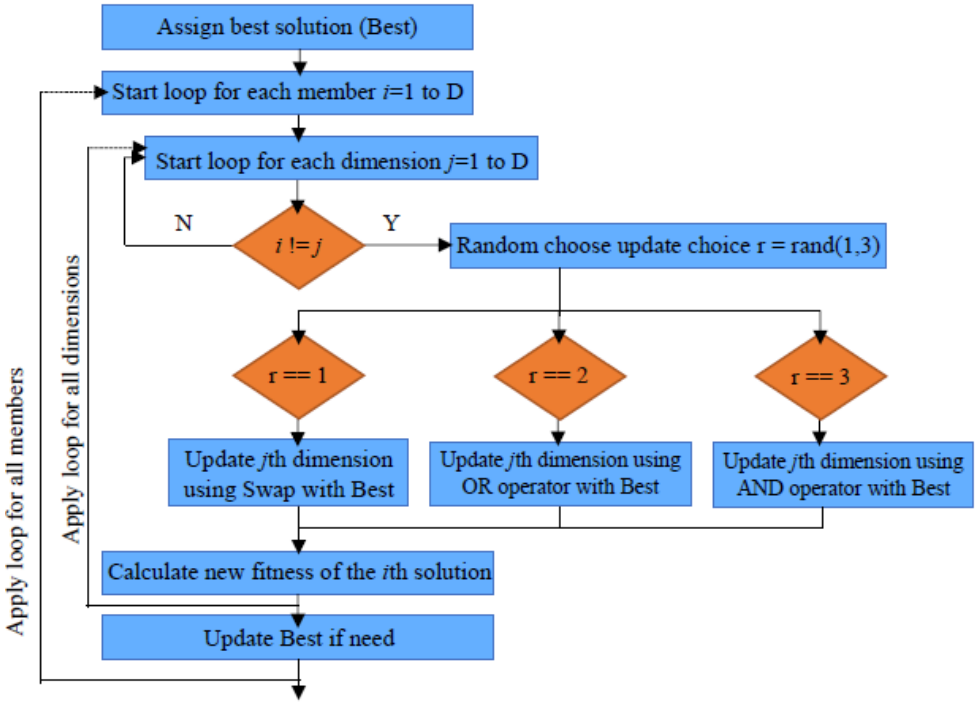


Figure A. The Flow chart of the JayaX-ELSM algorithm

Purpose: The ELSM mechanism aims to search the solution space in binary optimization problems effectively. The ELSM mechanism contributes to the global best solution to get rid of the local optimum position and obtain a better solution.

Theory and Methods: In the JayaX-ELSM algorithm, a random number in the range of (0, 1) is generated at the end of each iteration, and if this number is less than the Plocal value, the ELSM mechanism is applied. In the ELSM mechanism, one of the SWAP, OR, AND status update strategies is randomly selected and a candidate position is generated for the global best solution. If the objective function value of the candidate position is better than the current position's, the position update is performed.

Results: The proposed ELSM mechanism has greatly contributed to getting rid of the JayaX algorithm from the local optimum, especially in M* problems, and has increased the convergence performance of the JayaX algorithm.

Conclusion: The proposed ELSM mechanism makes an essential contribution to the optimization of uncapacitated facility location problems. It also has a significant potential for optimizing other binary problems such as feature selection, knapsack problems, and wind turbine placement.

Journal of the Faculty of Engineering and Architecture of Gazi University 38:4 (2023) 2435-2449

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Büyük ölçekli optimizasyon problemleri için seçime dayalı yerel arama mekanizmasına sahip ikili Jaya algoritması

Ahmet Özkış[1]*, Murat Karakoyun[2]

[1]Necmettin Erbakan Üniversitesi, Mühendislik Fakültesi, Adli Bilişim Mühendisliği Bölümü, 42090, Meram, Konya, Türkiye
[2]Necmettin Erbakan Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 42090, Meram, Konya, Türkiye

ÖNEÇIKANLAR

- ELSM adlı yeni bir yerel arama modülü önerilmiştir
- ELSM, ikili Jaya algoritması üzerinde uygulanmıştır
- Önerilen algoritma UFLP'lere uygulanmıştır

Makale Bilgileri
Araştırma Makalesi
Geliş: 29.04.2022
Kabul: 10.12.2022

DOI:
10.17341/gazimmfd.1111302

Anahtar Kelimeler:
İkili optimizasyon,
Jaya algoritması,
kapasitesiz tesis yerleştirme problemi,
metasezgisel algoritmalar,
yerel arama mekanizması

ÖZ

Jaya, yakın zamanda sürekli optimizasyon problemlerinin çözümü için önerilen popülasyon tabanlı metasezgisel bir algoritmadır. Literatürde ikili optimizasyon problemlerinin çözümü için çeşitli Jaya varyantları geliştirilmiştir. Bunlardan biri olan JayaX-LSM algoritması CAP problemlerinin çözümünde kullanılmış ve başarılı sonuçlar üretmiştir. Ancak CAP problemlerinden daha yüksek boyutlu ve kompleks bir yapıya sahip olan M* problemleri üzerinde test ettiğimizde algoritmanın oldukça başarısız sonuçlar ürettiği görülmüştür. Bu çalışmada, ikili optimizasyon problemlerinde çözüm uzayının etkili bir şekilde aranmasını sağlayan seçime dayalı yeni bir yerel arama modülü (ELSM) geliştirilmiştir. Bu modül ikili JayaX algoritmasına eklenerek JayaX-ELSM algoritması önerilmiştir. Önerilen JayaX-ELSM algoritmasının performansı öncelikle JayaX-LSM algoritmasıyla CAP ve M* problem setleri üzerinde karşılaştırmalı olarak analiz edilmiştir. Daha sonra, önerilen algoritma, literatürde yakın zamanda yayınlanmış toplam 11 farklı algoritmayla performans karşılaştırmasına tabi tutulmuştur. Elde edilen sonuçlar, önerilen JayaX-ELSM'nin JayaX-LSM algoritmasının CAP problemlerinde sergilediği performansı devam ettirdiğini, M* problemlerinde de JayaX-LSM'den çok daha başarılı sonuçlar ürettiğini göstermektedir. Bu durum algoritmaya eklenen ELSM mekanizmasının algoritmanın yerel arama başarısını arttırdığını göstermektedir. Önerilen algoritmanın M* problemleri üzerindeki performansının, karşılaştırılan algoritmalara göre rekabetçi olması ELSM mekanizmasının katkısını göstermektedir.

A binary Jaya algorithm with selection-based local search mechanism for large-scale optimization problems

HIGHLIGHTS

- A novel local search module named ELSM has been proposed
- The ELSM has been implemented on the binary Jaya algorithm
- The proposed algorithm has been applied on UFLPs

Article Info
Research Article
Received: 29.04.2022
Accepted: 10.12.2022

DOI:
10.17341/gazimmfd.1111302

Keywords:
Binary optimization,
Jaya algorithm,
local search mechanism,
metaheuristic algorithms,
uncapacitated facility location problem

ABSTRACT

Jaya is a population-based metaheuristic algorithm recently proposed for solving continuous optimization problems. Various Jaya variants have been developed in the literature to solve binary optimization problems. One of them, the JayaX-LSM algorithm, has been used in solving CAP problems and has produced successful results. However, when we tested it on M* problems, which have a higher dimensional and complex structure than CAP problems, it was seen that the algorithm produced relatively unsuccessful results. In this study, a new selection-based local search module (ELSM) has been developed, which provides an efficient search of the solution space in binary optimization problems. By adding this module to the binary JayaX algorithm, the JayaX-ELSM algorithm is proposed. The performance of the proposed JayaX-ELSM algorithm was first analyzed comparatively with the JayaX-LSM algorithm on CAP and M* problem sets. Then, the proposed algorithm was compared with 11 different algorithms recently published in the literature. The results show that the proposed JayaX-ELSM maintains the performance of the JayaX-LSM algorithm in CAP problems and produces much more successful results than JayaX-LSM in M* problems. This shows that the ELSM mechanism added to the algorithm increases the local search success of the algorithm. The fact that the performance of the proposed algorithm on M* problems is competitive compared to the compared algorithms shows the contribution of the ELSM mechanism.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *aozkis@erbakan.edu.tr, mkarakoyun@erbakan.edu.tr / Tel: +90 506 619 3368

1. Giriş (Introduction)

Optimizasyon; bir problemi amaç durumuna göre maksimize veya minimize edecek parametleri, belirli bir aralıkta seçip problemi sistematik olarak inceleme veya çözme sürecini ifade etmektedir. Çözülmesi için bu tür süreçler gerektiren problemler ise optimizasyon problemleri (OP) olarak ifade edilmektedir. Optimizasyon problemlerinin, geleneksel yöntemler ile çözülmesi durumunda genellikle uzun süreçlere ihtiyaç duyulurken sezgisel yaklaşımlar ile daha kısa sürelerde ve kabul edilebilir başarım sonuçları ile çözülebilmektedir. Optimizasyon problemlerini, çözüm kümesini temsil eden karar değişkenlerinin yapısına göre sürekli, kategorik, ayrık vb. gibi farklı kategorilere ayırmak mümkündür. Karar değişkenlerinin "0" ve "1" ile temsil edildiği ikili optimizasyon, ayrık kategorinin özel bir durumu olarak düşünülebilir. İkili optimizasyon problemlerinde her çözüm 0 veya 1 alan bir dizi karar değişkeni ile temsil edilmektedir. Bu temsilde 0 değeri "yokluk" anlamına gelirken 1 değeri "varlık" durumunu göstermektedir [1-4]. Sırt çantası problemi [5, 6], graf renklendirme [7, 8], görev çizelgeleme [9-11], özellik seçimi [12-14], kapasitesiz tesis yerleştirme [4, 15, 16], rüzgar tribünü yerleştirme [17-19] gibi problemler ikili optimizasyon kategorisinde ele alınmakta ve endüstri, ekonomi, iletişim, ulaşım, teknoloji gibi alanlarda çözülmeyi beklemektedir [4, 20, 21]. İkili optimizasyon problemlerinde karar değişkenlerinin 0 ve 1 değerlerini alabilmesi olası çözümlerin farklı kombinasyonlardan oluştuğunu göstermektedir. Bu durumda karar değişkenlerinin sayısının artması problem için büyük sayıda olası çözümün olması ve daha karmaşık bir yapı anlamına gelmektedir. Bu açıdan ele alındığında, ikili optimizasyon problemleri NP-Zor sınıfına dahil edilmektedir. İkili optimizasyon problemleri üzerinde çalışan araştırmacılar bu problemlerin çözümü için literatüre farklı geleneksel yöntemler (langragian metotları, numaralandırma şemaları, dal/sınır yöntemleri, gevşetme ve indirgeme metotları vb.) kazandırmışlardır. Ancak bu yöntemler küçük boyutlu problemlerde işlevsel ve başarılı olmasına rağmen büyük boyutlu problemlerde çok uzun süreler dahilinde sonuç vermektedir [4, 15, 16, 22, 23]. Bu bağlamda ikili optimizasyon problemlerinde geleneksel yaklaşımların yaşadığı sorunların üstesinden gelebilmek amacıyla metasezgisel algoritmaların kullanımı yaygınlaşmıştır.

Bu çalışmada, Rao tarafından 2016 yılında sürekli optimizasyon problemlerinin çözümü için literatüre kazandırılan Jaya algoritması [24], ikili bir optimasyon problemi olan kapasitesiz tesis yerleştirme (uncapacitated facility location problem, UFLP) problemine uygulanmıştır. UFL probleminde iki temel bileşen mevcuttur: tesisler ve müşteriler. Problemde, en az bir tesisin açık olması koşuluyla farklı konumlarda bulunan tesisler içerisinde hangi tesislerin açık hangilerinin kapalı olmasına karar verilmektedir. Bu duruma karar verilirken tesis kurulum maliyetinin en az olması hedeflenirken müşteriler ile tesisler arasındaki taşıma maliyetinin de minimize edilmesi amaçlanmaktadır. UFL probleminde toplam tesis sayısının n olduğu düşünülürse çözüm için 2^n-1 olası durumun söz konusu olduğu görülmektedir. Buradan da anlaşılacağı üzere problem boyutunun (tesis ve müşteri sayısı) artması, geleneksel yaklaşımların olası tüm çözümleri ele alması ve en uygun seçeneği bulması oldukça zorlaşmaktadır. Bu nedenle araştırmacılar, makul süreler içerisinde kabul edilebilir sonuçlar elde eden metasezgisel algoritmaları kullanmaya yönelmişlerdir. Bu yaklaşıma dayanarak bu çalışmada metasezgisel algoritmalardan olan Jaya algoritması UFL problemine uygulanmıştır. Diğer metasezgisel algoritmalar gibi Jaya da populasyon tabanlı iteratif bir algoritmadır. Algoritmanın saf hali sürekli problemlerin çözümü için önerildiğinden karar değişkenlerinin yapısı da buna uygun olarak oluşturulmaktadır. Ancak bu çalışmada üzerinde çalışılan UFL problemi, karar değişkenleri "0" ve "1" değerlerinden oluşan ikili bir optimizasyon problemidir. Bu nedenle Jaya algoritmasının ikili problemlere uygulanabilmesi için karar değişkenlerinin ikili olarak temsil edilmesi gerekmektedir. Aslan vd. [25] çalışmalarında konum güncelleme işleminin de ikili yapıya uygun olması için XOR ("özel veya") lojik kapısı kullanılmışlardır. Ayrıca yerel arama noktasında algoritmanın başarısını arttırmak amacıyla yerel arama modülü (local search module, LSM) kullanmışlardır. Geliştirdikleri JayaX-LSM algoritmasını CAP problemlerine uygulayıp farklı algoritmaların performansıyla karşılaştırmışlardır. Deneysel sonuçlar önerdikleri algoritmanın CAP problemlerinde başarılı olduğunugöstermiştir. Ayrıca yazarlar, algoritmalarının farklı ikili problemlere uygulanıp peformansının değerlendirilebileceğini belirtmişlerdir. Yazarların bu önerisine istinaden bu çalışmada CAP problemlerinden, problem boyutunun büyük olması açısından daha karmaşık ve çözülmesi zor olan M-Yıldız (M*) problem seti kullanılmıştır. Ancak JayaX-LSM algoritmasının bu problemlerde optimal çözüm değerlerinden oldukça uzak yerel minimum noktalara takılarak başarısız olduğu gözlemlenmiştir. Bu çalışmada, bu sorunun üstesinden gelmek için geliştirilmiş yerel arama modülü (enhanced local search module, ELSM) adı verilen yeni bir mekanizma önerilmiştir. ELSM mekanizması, JayaX algoritmasına uygulanarak JayaX-ELSM algoritması önerilmiştir.

Çalışmanın geri kalanı aşağıdaki gibi organize edilmiştir: Bölüm 1.1'de, literartür araştırmasına yer verilmiştir. Bölüm 1.2'de, çalışmanın temel motivasyonuna ve literatüre olan katkısına değinilmiştir. Bölüm 2'de, orijinal Jaya algoritması ve çalışmada kullanılan CAP ve M* problem setleri sunulmuştur. Bölüm 3'te, literatürde yer alan ikili Jaya varyantları ve bu çalışmada önerilen JayaX-ELSM algoritması anlatılmış ve ELSM mekanizmasının katkısını araştırmak için deneysel çalışmalar yapılmıştır. Bölüm 4'te, önerilen JayaX-ELSM algoritması literatürde yer alan çeşitli çalışmalarla performans karşılaştırmasına tabi tutulmuştur. Bölüm 5'te, deneysel sonuçlarla ilgili değerlendirmelere ve gelecek çalışmalarla ilgili önerilere yer verilmiştir.

1.1. Literatür Araştırması (Literature Review)

Literatüre bakıldığında ikili optimizasyon problemlerinin çözümü için kullanılmış bir çok metasezgisel algoritmanın olduğu görülmektedir. Bu çalışmaların tamamını ele almak ve incelemek çok uzun zaman ve efor gerektirmektedir. Bu nedenle, literatür taraması kapsamında temel bazı metasezgisel algoritmalar ve bu algoritmaların ikili türevleri ele alınarak özet bir şekilde sunulmuştur.

Metasezgisel algoritmalar içerisinde temel bir algoritma olan parçacık sürü algoritması (particle swarm optimization, PSO) [26] [27], ilham kaynağı olarak kuş ve balık sürülerinin besin arama davranışlarını ilham alarak modellenmiştir. Algoritma 1995 yılında Kennedy ve Eberhart tarafından modellenmiş olup ilk ikili versiyonu olan ikili PSO (binary PSO, BPSO) [28] da bu araştırmacılar tarafından 1997 yılında önerilmiştir. Önerilen BPSO algoritmasında, orijinalinde sürekli olan, karar değişkenleri sigmoid fonksiyonu kullanılarak ikili değerlere çevrilmiştir. Ancak elde edilen deneysel sonuçlar, BPSO algoritmasının keşif yeteneğinin büyük boyutlu problemlerde yetersiz kaldığını göstermiştir. Bu sorunun üstesinden gelmek ve algoritmanın performansını arttırmak amacıyla Khanesar vd. [29] PSO'daki hız vektörünü değiştirerek kullanmışlardır. Yuan vd. [3] ise önerdikleri ikili IBPSO yaklaşımı ile birim taahhüt problemlerine çözüm getirmeyi amaçlamışlardır. IBPSO algoritmasının orijinalinden farklı olarak karar değişkenlerini başlangıçtan itibaren ikili olarak oluşturması ve konum güncellemelerini ikili karar değişkenleri üzerinde yapması göze çarpmaktadır. Bu yaklaşımların yanı sıra, ikili optimizasyon algoritmalarını çözmek için literatüre kazandırılan diğer bazı algoritmalar şu şekilde özetlenebilir: Bahesti vd. [30], Güner ve

Şevkli [31], Nezamabadi-pour vd. [32] ve Saha vd. [33]. Storn ve Price tarafından 1997 yılında önerilen diferansiyel evrim (differential evolution, DE) [34] algoritması evrimsel tabanlı popüler bir algoritmadır. Pampara [35], açı modülasyonlu ikili bir DE (angel modulated DE, MADE) algoritması önererek klasik test problemleri üzerinde performans değerlendirmesi gerçekleştirmiştir. Engelbrecht ve Pampara [36], DE algoritmasının iki farklı ikili versiyonunu önererek literatüre katkıda bulunmuşlardır. Bu yaklaşımlardan ilki olan binDE, ikili PSO yaklaşımından faydalanırken; normDE yaklaşımı ise arama uzayının alt ve üst sınırlarının arasındaki sürekli boşluklarda normalizasyon stratejisi uygulamışlardır. Bu stratejide normalize edilen sürekli bir değer 0,5 değerinden küçük iken 0 aksi takdirde 1 olarak ikili hale dönüştürülmektedir. Su ve Yang [37], quantum tabanlı bir diferansiyel evrim (QDE) algoritması önermişlerdir. Chen vd. [38] eski çözümlerden öğrenme yapan bir BLDE yaklaşımı önererek sırt çantası problemlerine uygulamışlardır. He vd. [39] önerdikleri ikili DE (binary DE, BDE) algoritmasını UCI veri ambarından aldıkları farklı veri setleri üzerinde özellik seçimi problemi için uygulamışlardır. Deng vd. [40] haritalama operatörü tabanlı bir ikili DE algoritması önerip sırt çantası probleminde kullanmışlardır. Literatürde ikili optimizasyon problemlerinin çözümü için önerilen Yang [41], Wang vd. [42] ve Kashan vd. [43] çalışmaları da DE algoritmasının diğer ikili varyasyonları olarak göze çarpmaktadır.

Yapay arı kolonisi (artificial bee colony, ABC) [44] algoritması 2005 yılında Karaboğa tarafından literatüre kazandırılmıştır. Bal arılarının yiyecek arama davranışlarını model alarak geliştirilen ABC, sürekli optimizasyon problemlerinin çözümü için oldukça başarılı sonuç veren bir algoritmadır. Kashan vd. [45] algoritmanın karar değişkenlerini doğrudan ikili değerler alacak şekilde başlatmış konum güncelleme sürecinde ise Jaccard'ın benzerliğini kullanmıştır. DisABC olarak isimlendirilen bu yaklaşım kapasitesiz tesis yerleştirme problemlerine uygulanmıştır. Bununla beraber Kıran ve Gündüz [46] çalışmalarında XOR mantıksal operatörü tabanlı binABC algoritmasını önererek kapasitesiz tesis yerleştirme problemlerine uygulamışlardır. Benzer şekilde, Kıran [47] stigmergic davranış tabanlı bir ikili ABC algoritması önermiş ve algoritmanın performansını CEC 2015 ve kapasitesiz tesis yerleştirme problemleri üzerinde test etmiştir. Öztürk vd. [48] genetik operatör tabanlı, GB-ABC olarak adlandırdıkları yeni bir ikili algoritmayı sırt çantası problemlerine uygulamak üzere önermişlerdir.

Yukarıda ikili optimizasyon problemlerini çözmek amacıyla ikili varyasyonları verilen metasezgisel algoritmalar dışında literatüre son zamanlarda kazandırılan ve kapasitesiz tesis yerleştirme problemine uygulanan diğer çalışmalar aşağıdaki gibi özetlenebilir: Aslan vd. [25] sürekli problemler için önerilen Jaya algoritmasının iki farklı ikili varyasyonunu önermişlerdir. İlk çalışmalarında XOR mantık operatörü tabanlı JayaX algoritmasını önerirken ikinci versiyonunda JayaX algoritmasına yerel arama mekanizması ekleyerek JayaX-LSM algoritmasını geliştirmişlerdir. Algoritmanın her iki versiyonunu da kapasitesiz tesis yerleştirme problemlerine uygulamışlardır. Çınar ve Kıran [16] ise ağaç tohum algoritmasını (tree-seed algorithm, TSA) üç farklı şekilde modifiye ederek ikili problemlere uygulamışlardır. İlk versiyon olarak mantık kapısı tabanlı LogicTSA algoritmasını önermişlerdir. İkinci versiyonda ise benzerlik ölçümü tabanlı SimTSA algoritmasını önermişlerdir. Üçüncü ve son versiyon da ise ilk iki versiyonun bir hibrit yaklaşımı olan SimLogicTSA algoritmasını önermişlerdir. TSA'nın bu farklı versiyonlarını kapasitesiz tesis yerleştirme problemlerine uygulayarak performans karşılaştırması yapmışlardır. Haklı ve Ortaçay [49] dağınık arama (scatter search, SS) algoritmasını geliştirerek kapasitesiz tesis yerleştirme problemlerine uygulamış ve literatürdeki diğer çalışmalar ile performans karşılaştırması yapmışlardır. Baş ve Ülker [15] çalışmalarında sosyal örümcek algoritmasını (social spider algorithm, SSA) [50] ikili olarak kullanarak kapasitesiz tesis yerleştirme problemlerine uygulamışlardır. Korkmaz vd. [51] yapay alg algoritmasını (artificial algae algorithm, AAA) ikili olarak kullanan bir yaklaşım önermiş ve kapasitesiz tesis yerleştirme problemlerine uygulamışlardır. Ayrıca elde ettikleri deneysel sonuçları literatüredeki diğer çalışmalar ile karşılaştırmışlardır.

1.2. Motivasyon ve Katkı (Main Motivation and Contribution)

Literatür araştırmasında elde edilen sonuçlar göz önüne alındığında, ikili optimizasyon problemlerinin çözümüne yönelik birçok metasezgisel algoritmanın geliştirildiği ve bu yöndeki araştırmaların günümüzde de devam ettiği görülmektedir. Bu durum her ne kadar bazı bilim insanları tarafından eleştirilse de [52], No Free Lunch (NFL) teoremine göre yeni tekniklerin geliştirilmesi kaçınılmaz ve gereklidir [53]. Bu teoreme göre herhangi bir algoritmanın belirli bir problem türündeki yüksek performansı, başka bir problem türündeki performansını dengelemektedir. Yani hiçbir algoritma her türdeki problemler için en uygun çözümü bularak daimi bir başarı garanti edemez. Bu yaklaşım, araştırmacıları, mevcut önerilmiş algoritmalardan farklı türdeki problemler için daha iyi sonuçlar üreten yeni teknikler önermeye teşvik etmiştir. Bu motivasyonla, bu çalışmada, yakın tarihte CAP problemleri üzerinde uygulanarak başarılı sonuçlar elde eden JayaX-LSM algoritmasının M* problem setindeki başarısı araştırılmıştır. NFL teoremine uygun olarak algoritmanın, CAP problemlerindeki başarısının aksine M* problem setinde oldukça başarısız olduğu gözlenmiştir.

Bu çalışmada, JayaX algoritmasının yerel arama kabiliyetini iyileştirerek M* problem setinde başarılı sonuçlar elde etmesini sağlamak için ELSM adı verilen yeni bir mekanizma önerilmiştir. Öncelikle ELSM mekanizmasının CAP problemlerindeki etkisi analiz edilmiştir. Önerilen JayaX-ELSM algoritmasının, bu problemlerde zaten iyi sonuç veren JayaX-LSM algoritmasının performansına denk bir performans göstererek algoritmanın performansını olumsuz yönde etkilemediği görülmüştür. Daha sonra JayaX-LSM ve JayaX-ELSM algoritmalarının M* problem setinde performans karşılaştırması yapılmıştır. Karşılaştırmalı deneysel sonuçlar incelendiğinde önerilen algoritmanın JayaX-LSM algoritmasından açık bir şekilde daha başarılı olduğu gözlemlenmiştir. Son olarak önerilen algoritmanın performansı hem CAP hem de M* problemleri için literatürdeki öne çıkan bazı çalışmalarla kıyaslanmıştır. Önerilen algoritmanın hem CAP hem de M* problem setinde başarılı sonuçlar elde ettiği ve başarısı açısından literatüre katkı sağladığı gözlemlenmiştir.

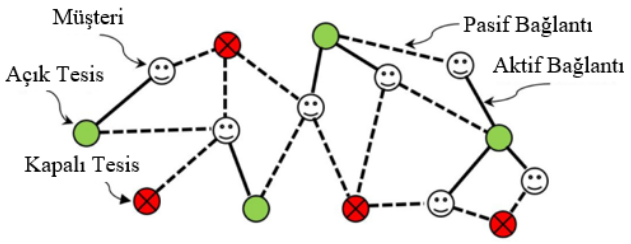
2. Materyal ve Metot (Material and Methods)

Bu bölümde UFL problemi hakkında detaylı bilgi verilmiştir. Bu çalışmada kullanılan kapasitesiz tesis yerleştirme problemlerinden CAP ve M* problemleri hakkında bilgi verilmiştir. Son olarak bu çalışmada kullanılan Jaya algoritmasının orijinali hakkında bilgi verilmiştir.

2.1. Kapasitesiz Tesis Yerleştirme Problemi (Uncapacitated Facility Location Problem)

Temel ikili optimizasyon problemlerinden biri olan UFLP, gerçek dünyada karşılaşılan ve çözümü oldukça zor olan bir problem türüdür. Problem en basit haliyle ele alındığında hizmet sunan tesisler ve bu tesislerden hizmet bekleyen müşterilerden oluşmaktadır denilebilir. Her bir tesisin kurulum maliyeti ve her bir müşterinin en yakın olduğu (açık) tesisten hizmet alma maliyeti problemin toplam maliyetini oluşturmaktadır. Problemin çözümüne gidilirken temel hedef, müşterilerin tamamına hizmet verilmesini sağlayan en uygun tesis yerleştirme seçimini minimum maliyet ile sağlamaktır. Problemdeki toplam tesis sayısının n (kapalı ve açık tesislerin tamamı) olduğu

düşünülürse açık olacak tesislerin durumunu temsil eden olası çözüm sayısının 2^n olduğu görülmektedir. Bununla beraber UFL probleminde en az bir tesisin açık olması şartı arandığından bu sayı 2^n-1 olmaktadır. Bu basit hesaplama ile tesis sayısının (n) problemin karmaşıklığı üzerinde doğrudan etkili olduğu görülmektedir. Bu karmaşıklığa ek olarak tesislerin kurulum maliyeti de hesaba dahil edildiğinde, problemin NP-Zor sınıfına dahil olduğu görülmektedir [54-57]. UFL probleminin görsel sunumu Şekil 1 ile verilmiştir. Görsel sunumda açık tesisler ve kapalı tesisler sırasıyla yeşil ve kırmızı renkli daireler ile temsil edilmiştir. Probleme dahil olan müşteriler ise beyaz daire ile gösterilmiştir. Müşteriler ile tesisler arasındaki yollar aktif ve pasif durumunda olacak şekilde belirtilmiştir. Burada aktif yol müşteri ile kendisine en yakın olan tesis arasındaki yoldur ve düz çizgi ile belirtilmiştir. Kesikli çizgi ile gösterilen pasif yol ise müşteri ile kapalı tesis arasındaki yol veya müşteri ile açık tesis arasında ama daha az maliyete sahip bir alternatifi olan yol olarak ele alınmıştır.



Şekil 1. UFL probleminin görsel sunumu
(The graphical presentation of the UFLP) [25]

UFL probleminde toplam tesis sayısı biliniyorken hangi tesislerin açık veya kapalı olacağı bilgisi mevcut değildir. Problemdeki her bir tesis için sabit bir kurulum maliyeti vardır. Bununla beraber, müşterilerin tesislerden hizmet alması da maliyet gerektiren bir durumdur. Her müşteri açık olan tesisler içerisinden kendisine en yakın (maliyeti en az) tesisten hizmet almak durumundadır. UFL probleminde temel amaç, tesislerin kurulumundan ve açık tesislerin müşterilere sunduğu hizmetten kaynaklanan maliyetlerden oluşan toplam maliyetin en aza indirgenmesidir [55, 58, 59]. Mevcut tesislerin tamamını temsil eden kümeye F_T denilsin; amaç bu kümeden öyle bir alt küme (F_{alt}) seçilsin ki toplam maliyet minimum olsun. $F_{alt} \subseteq F_T$ olmak üzere Eş. 1 problemin toplam maliyetini matematiksel olarak sunmaktadır. UFL probleminde amaç $min(f(F_{alt}))$ durumunu sağlamaktır.

$$f(F_{alt}) = \sum_{i \epsilon F_{alt}} f_i + \sum_{j \epsilon M} \min\{C_{ij} \mid i \epsilon F_{alt}\} \quad (1)$$

Eş. 1'de; f_i açık bir tesisin kurulum maliyeti, C_{ij} i.tesis ile j.müşteri arasındaki maliyet, F_{alt} açık tesislerin tutulduğu alt küme vektörü ve M müşterilerin kümesidir. UFL probleminde her bir çözüm (x) ikili formattaki bir vektör ($x \in \{0, 1\}^n, n$ burada toplam tesis sayısıdır) ile temsil edilmektedir. Burada tesis kümesindeki i.tesis açık ise çözüm vektöründeki $x_i = 1$ iken kapalı olduğunda $x_i = 0$ olmaktadır. Bu çalışmada, önerilen algoritmanın performansını ölçmek ve diğer algoritmalar ile performans karşılaştırması yapmak amacıyla OR-Library [60] veri ambarından alınan iki farklı problem seti kullanılmıştır.

2.1.1. CAP Problemleri (CAP Problems)

OR-Library veri ambarından alınan ve bu çalışmada kullanılan ilk veri seti, farklı özelliklere sahip problemleri içerisinde barındıran CAP problem setidir. Bu problem ailesini oluşturan 15 farklı problemin özellikleri Tablo 1 ile verilmiştir.

Tablo 1. CAP problemlerinin özellikleri
(The properties of the Cap problems)

Problem	Tip	Boyut	En iyi maliyet
Cap71	Küçük	16 × 50	932615,750
Cap72		16 × 50	97779,400
Cap73		16 × 50	1010641,450
Cap74		16 × 50	1034976,975
Cap101	Orta	25 × 50	796648,438
Cap102		25 × 50	854704,200
Cap103		25 × 50	893782,113
Cap104		25 × 50	928941,750
Cap131	Büyük	50 × 50	793439,563
Cap132		50 × 50	851495,325
Cap133		50 × 50	893076,713
Cap134		50 × 50	928941,750
CapA	Çok büyük	100 × 1000	17156454,478
CapB		100 × 1000	12979071,580
CapC		100 × 1000	11505594,330

CAP problemleri boyut (tesis × müşteri) özelliklerine göre kategorize edildiğinde 4 farklı tip ile temsil edilebilir. Bu açıdan bakıldığında; 16 tesis ve 50 müşteriye sahip Cap71-74 arasındaki problemleri *küçük* problem tipine, 25 tesis ve 50 müşteriye sahip Cap101-104 problemlerini *orta* problem tipine, 50 tesis ve 50 müşteriye sahip Cap131-134 problemlerini *büyük* problem tipine ve 100 tesis ve 1000 müşteriye sahip CapA, CapB, CapC problemlerini de *çok büyük* problem tipine dahil olduğu görülmektedir. Tabloda ayrıca her problem için en iyi çözüme ulaşıldığında bu çözüme ait maliyet değeri verilmiştir.

2.1.2. M-yıldız problemleri (M-star (M) problems)*

Bu çalışmada algoritmaların performansını değerlendirmek üzere kullanılan bir diğer problem seti M* problemleridir. İçerisinde 20 farklı problem barındıran bu veri seti de OR-Library veri ambarından alınmıştır. M* veri setindeki problemlerin özellikleri Tablo 2'de verilmiştir. Tabloda her problem için ideal çözüme ait maliyet değeri verilmiştir. Bu veri setindeki problemler sahip olduğu tesis sayısı bakımından CAP problemlerinden daha büyüktür. Bu nedenle çözümleri nispeten daha zor olmaktadır.

Tablo 2. M* problemlerinin özellikleri
(The properties of the M* problems)

Problem	Boyut	En iyi maliyet
MO1	100 × 100	1305,98
MO2	100 × 100	1432,36
MO3	100 × 100	1516,77
MO4	100 × 100	1442,24
MO5	100 × 100	1408,77
MP1	200 × 200	2686,48
MP2	200 × 200	2904,86
MP3	200 × 200	2623,71
MP4	200 × 200	2938,75
MP5	200 × 200	2932,33
MQ1	300 × 300	4091,01
MQ2	300 × 300	4028,33
MQ3	300 × 300	4275,43
MQ4	300 × 300	4235,15
MQ5	300 × 300	4080,74
MR1	500 × 500	2608,15
MR2	500 × 500	2654,73
MR3	500 × 500	2788,25
MR4	500 × 500	2756,04
MR5	500 × 500	2505,05

2.2. Temel Jaya Algoritması (Basic Jaya Algorithm)

Rao tarafından 2016 yılında önerilen Jaya [24], diğer metasezgisel algoritmalar gibi popülasyon tabanlı iteratif bir algoritmadır. Jaya sürekli optimizasyon problemlerinin çözümü için önerilen parametre bağımsız bir algoritmadır. Bu özelliği sayesinde farklı optimizasyon problemlerine rahatlıkla adapte edilip uygulanabilmektedir. Jaya algoritması konum güncelleme sürecinde popülasyondaki en iyi konum ile beraber en kötü konumu da kullanmaktadır. Bu sayede iyi konuma yönelirken kötü konumdan da uzaklaşma amacı gütmektedir. Bu durum algoritmaya yerel en iyi konuma takılma problemini çözmekte avantaj sağlamaktadır. Ayrıca keşif ve sömürü işlevleri bakımından arada bir denge olması sağlanmaktadır. Bu durum algoritmaya arama uzayında geniş bir alanda etkili bir tarama yapma imkanı sunmaktadır [24, 25, 61].

Jaya algoritması, başlangıç parametrelerinin belirlenmesi ile başlamaktadır. Başlangıç popülasyonu, aksi belirtilmediği sürece diğer metasezgisel algoritmalar gibi arama uzayında rasgele konumlarda üretilir ve bu konumlar için uygunluk değerleri hesaplanır. İlk popülasyon içerisindeki en iyi ve en kötü konuma sahip bireyler işaretlenir. Belirlenen bitirme kriterine göre bir döngü başlatılır ve popülasyondaki bireylerin konum güncelleme süreçleri başlatılır. Popülasyondaki her birey için Eş. 2 kullanılarak aday bir çözüm elde edilir. Elde edilen aday çözüm mevcut çözümden iyiyse birey yeni konum ile güncellenir değilse mevcut konumunu muhafaza eder. Her bir iterasyon adımında tüm bireyler için konum güncellemesi yapıldıktan sonra yeni konumlar için en iyi ve en kötü konum bulunarak işaretlenir. Konum güncelleme döngüsü belirlenen bitirme kriteri sağlanana kadar devam eder. Bitirme kriteri sağlandığında ise bulunan en iyi konum algoritmanın çıktısı olarak verilir.

$$X'_{i,j} = X_{i,j} + r_1 * \left(Best_j - \left|X_{i,j}\right|\right) - r_2 * \left(Worst_j - \left|X_{i,j}\right|\right) \quad (2)$$

N popülasyon büyüklüğü olmak üzere; i (i=1, 2, …, N) popülasyon içerisinde konumu güncellenecek olan çözümün indeks değerini, j (j=1, 2, …, D, D: problem boyutu) güncellemeye tabi tutulan boyutun indeks değerini, *Best* en iyi çözümün konumunu ve *Worst* en kötü çözümün konumunu temsil etmektedir. Eşitlikte kullanılan r_1 ve r_2 değerleri ise [0, 1] aralığında rassal olarak üretilen reel sayılardır. Jaya algoritmasının sözde kodu Şekil 2 ile verilmiştir [24, 25].

3. Jaya Algoritmasının İkili Varyantları (Binary Variants of Jaya Algorithm)

3.1. JayaX Algoritması (JayaX Algorithm)

Aslan vd. [25] temel Jaya algoritması üzerinde çeşitli düzenlemeler yaparak ikili problemlerin çözümünde kullanılabilen JayaX algoritmasını önermişlerdir. JayaX algoritmasında yapılan düzenlemeler aşağıda verilen 3 başlık altında ele alınabilir [25]:

- *Başlangıç çözümlerinin oluşturulması*

Temel Jaya algoritmasında aday çözümler, sürekli arama uzayında oluşturulurlar. JayaX algoritması ikili problemlerin çözümü için geliştirildiğinden Eş. 3'deki yöntem ile aday çözümleri doğrudan ikili değerlerle başlatmaktadır.

$$X_{i,j} = \begin{cases} 1 & \text{if rand} > 0{,}5 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

- *Güncellenecek boyut sayısının belirlenmesi*

Temel Jaya algoritmasının konum güncelleme fazında aday çözümlerin tüm karar değişkenleri güncellenmektedir. Bu durum rastsallığın artmasına ve konumu güncellenen çözümlerin en iyi çözümün etrafından uzaklaşmasına yol açabilmektedir. Bu sorunu ortadan kaldırmak ve yerel-global arama arasındaki dengeyi sağlamak için JayaX algoritmasına yeni bir mekanizma eklenmiştir. Bu mekanizmada, 0-1 aralığında değer alan bir P parametresi problemin boyut sayısıyla çarpılarak güncellenecek boyut sayısı belirlenir. Örneğin 10 boyutlu bir problemde P = 0,5 değeri için yalnızca 5 boyut güncellenir. Bu mekanizmanın sözde kodu Şekil 3'te verilmiştir.

- *İkili konum güncelleme stratejisi*

Temel Jaya algoritması, Eş.2'de verilen sürekli problemlere uygun konum güncelleme stratejisine sahiptir. Bu strateji, ikili problemlerin çözümüne uygun olmadığından, JayaX algoritmasında konum güncellemesi için mantıksal kapıların kullanılması düşünülmüştür. Temel mantıksal kapılardan VE, VEYA ve XOR'un doğruluk tablosu Tablo 3'te verilmiştir.

Şekildeki A ve B değerleri lojik kapılara verilen girdileri, "&" simgesi VE kapısını, "+" simgesi VEYA kapısını ve ⊕ simgesi XOR kapısını

```
Başla
Başlangıç parametrelerini belirle (N, D, MaxIter)
Başlangıç popülasyonunu arama uzayında rasgele oluştur
Başlangıç popülasyonu için uygunluk değerleri hesapla (Eş. 1)
Uygunluk değerine göre en iyi (Best) ve en kötü (Worst) çözümü işaretle
FOR iter=1:MaxIter
   FOR i=1:N
      Eş. 2 kullanarak i. çözüm (Xi) için aday çözüm (X'i) oluştur
      IF f(X'i) < f(Xi)    //Aday çözüm mevcut çözümden daha iyi ise
         Mevcut çözümü aday çözümün konumu ile güncelle
      ELSE
         Mevcut çözümü muhafaza et
      END IF
   END FOR
   Uygunluk değerine göre en iyi (Best) ve en kötü (Worst) çözümü işaretle
END FOR
En iyi çözümü (Best) çıktı olarak ver
Bitir
```

Şekil 2. Temel Jaya algoritmasının sözde kodu (The pseudo code of the basic Jaya algorithm)

Adım 1: Konum güncelleme aşamasında seçilecek boyut sayısını belirle
Olasılık (Probability - P) parametresini [0,1] arasında olacak şekilde belirle,
Problemin boyutsallığına (D) bağlı olarak güncelleme işlemi için boyut sayısını (d) belirle,
Güncellenecek karar değişkenlerinin sayısını belirle ($d = round(D * P)$),

Adım 2: Konum güncelleme aşamasında aday çözümün güncellenecek boyutlarını belirle
d Adet karar değişkenlerini rastgele seç,

Adım 3: Güncellenecek boyutları getir

Şekil 3. Konum güncelleme aşamasında aday çözümün güncellenecek boyutlarının belirlenmesi
(Determining the dimensions of the candidate solution to be updated during the position update phase)

Tablo 3. VE, VEYA ve XOR kapılarının doğruluk tablosu (Truth table of AND, OR, and XOR gates)

Karar Değişkenleri		Lojik Kapılar		
A	B	A&B (VE)	A+B (VEYA)	A⊕B (XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0
		%75 olasılıkla 0 %25 olasılıkla 1	%25 olasılıkla 0 %75 olasılıkla 1	%50 olasılıkla 0 %50 olasılıkla 1

IF $rand()<P_{local}$ // $rand()$: [0,1] arasında oluşturulan rastgele bir değer, P_{local} = 0,1
  $Best$ = En iyi uygunluk değerine sahip çözüm
  FOR j = 1 to D // D problemin boyutunu gösterir.
    FOR jj = 1 to D
      IF ($Best_j$! = $Best_{jj}$)
        // $Best$ çözümünün j. ve jj. boyutlarının değerlerini karşılıklı olarak değiştir,
        $x = swap(Best, j, jj)$; // x geçici çözümü temsil etmektedir.
        x çözümü için uygunluk (fitness) değerini hesapla,
        IF (x çözümünün uygunluk değeri $Best$ çözümünün uygunluk değerinden iyi ise)
          // En iyi uygunluk değerine sahip çözümün ($Best$) konumunu güncelle,
          $Best = x$;
        END IF
        // Fonksiyon değerlendirme sayısını güncelle,
        $FEs = FEs + 1$;
      END IF
    END FOR
  END FOR
END IF

Şekil 4. LSM mekanizmasının sözde kodu (Pseudo code of the LSM mechanism)

ifade etmektedir. Lojik kapıların olası 4 durum için elde ettiği çıktılar incelendiğinde VE kapısının %75 olasılıkla 0, VEYA kapısının %75 olasılıkla 1 ürettiği görülmektedir. VE ile VEYA kapıları, konum güncellemelerini %75 olasılıkla aynı tarafa yönlendireceğinden arama uzayının ideal bir şekilde aranmasını sağlayamaz. XOR kapısında 0 ve 1 değerlerinin üretilmesi eşit olasılıklara sahip olduğundan, JayaX algoritmasında konumlar XOR kapısı kullanılarak Eş. 4'teki gibi güncellenmektedir.

$$X'_{k,j} = \{X_{k,j} \oplus (Best_j \oplus Worst_j) \quad (4)$$

X'_k popülasyondaki k. ajan için yeni üretilecek konum vektörünü, X_k popülasyondaki k. ajanın mevcut konum vektörünü, j konumu güncellenecek boyutun indisini, $\oplus$ mantıksal XOR operatörünü, Best ve Worst ise sırasıyla popülasyondaki en iyi ve en kötü uygunluk değerine sahip ajanların konum vektörlerini ifade etmektedir [25].

3.2. JayaX-LSM Algoritması (JayaX-LSM Algorithm)

Aslan vd., JayaX algoritmasına yerel arama modülü (Local Search Module, LSM) adı verilen bir mekanizma ekleyerek JayaX-LSM algoritmasını [25] önermiştir. Bu mekanizma sayesinde JayaX algoritmasının yerel arama kabiliyeti artırılmıştır. LSM mekanizması aşağıda detaylı bir şekilde anlatılmaktadır.

3.3. Yerel Arama Modülü (Local Search Module, LSM)

Zhang vd. 2016 yılında önerdikleri ikili yapay alg algoritmasının (BAAA) [62] performansını iyileştirmek amacıyla EliteLocalSearch adı verilen açgözlü bir yerel arama mekanizması kullanmışlardır. BAAA'da, EliteLocalSearch mekanizması her iterasyonun sonunda kullanılmıştır.

Kashan vd. tarafından 2011 yılında önerilen BinABC algoritmasında [45] ise farklı bir yerel arama mekanizması kullanılmıştır. BinABC'de, her iterasyonun sonunda yerel arama mekanizması uygulanmamıştır. Her iterasyonun sonunda 0-1 aralığında rastgele bir sayı üretilmiş, bu sayı Plocal adını verdikleri bir parametrenin değerinden küçükse yerel arama mekanizması uygulanmıştır. Aslan vd., BAAA'daki EliteLocalSearch mekanizmasını ve BinABC algoritmasındaki Plocal parametresini birleştirerek JayaX algoritmasına uygulamıştır. LSM'nin sözde kodu Şekil 4'te verilmiştir.

3.4. Önerilen JayaX-ELSM Algoritması
(Proposed JayaX-ELSM Algorithm)

Bu çalışmada, orijinal JayaX algoritmasına geliştirilmiş yerel arama modülü (Enhanced LSM, ELSM) adı verilen yeni bir mekanizma eklenerek JayaX-ELSM algoritması önerilmiştir. ELSM mekanizması ve bu mekanizmanın konum güncelleme üzerindeki etkisi aşağıdaki bölümde detaylı olarak açıklanmıştır.

3.4.1. Geliştirilmiş Yerel Arama Modülü
(Enhanced local search module (ELSM))

LSM mekanizmasında, takas edilen boyutların değerleri, mantıksal NOT kapısında olduğu gibi önceki durumlarına göre terslenir. UFL problemlerini düşünürsek, bu bir tesisin açılmasını diğerinin kapanmasını sağlar. Ancak, optimal çözüme ulaşmak için her iki tesisin de açık olması veya her ikisinin de kapalı olması gerekiyorsa, LSM mekanizması bu imkanı sağlayamaz. Bu durum, çözümlerin yerel optimumda takılıp kalmasına neden olabilir. Bu çalışmada, boyutların tüm olası durumları almasına izin veren ELSM mekanizması önerilmiştir. ELSM mekanizmasının sözde kodu Şekil 5'te verilmiştir.

ELSM'de SWAP, OR veya AND durum güncelleme stratejilerinden biri rastgele seçilir ve konum güncellenir. ELSM mekanizmasının arama çeşitliliğine nasıl katkıda bulunduğu Tablo 4'te LSM mekanizmasıyla karşılaştırmalı olarak verilmiştir.

Tablo 4'te görüldüğü gibi, ELSM'de LSM mekanizmasından farklı olarak her iki konumun da 1 veya 0 olması mümkündür. UFLP açısından bu durum, iki tesisin aynı anda kapatılmasına veya açılmasına izin verir. Böylece ELSM sayesinde daha detaylı yerel arama yapılmasına imkan sağlanmış olur.

```
IF rand()<Plocal     // rand(): [0,1] arasında oluşturulan rastgele bir değer, Plocal = 0,1
    Best = En iyi uygunluk değerine sahip çözüm
    FOR i = 1 to D
      FOR j = 1 to D
          IF(i != j and Best_i != Best_j)
             // en iyi çözümün i. ve j. boyutlarının değerlerini karşılıklı olarak değiştir
          secim = rand(1,3) //secim degiskenine 1-3 arasında rastgele bir değer ata
             IF(secim == 1)
            x = swap(Best, i, j);  // swap operatörünü uygula

          ELSE IF (secim == 2)
             x = OR(Best, i, j);   //  mantıksal OR operatörünü uygula

          ELSE IF (secim == 3)
             x = AND(Best, i, j); //  mantıksal VE operatörünü uygula

          END IF
              x çözümü için uygunluk (fitness) değerini hesapla,
          IF(x çözümünün uygunluk değeri Best çözümünün uygunluk değerinden iyi ise)
              //En iyi uygunluk değerine sahip çözümün (Best) konumunu güncelle,
              Best = x;
          END IF
          // Fonksiyon değerlendirme sayısını güncelle,
            FEs = FEs + 1;
        END IF
      END FOR
    END FOR
END IF
```

Şekil 5. ELSM mekanizmasının sözde kodu (Pseudo code of the ELSM mechanism)

Tablo 4. LSM ve ELSM mekanizmalarının konum güncellemeleri (Position updates of LSM and ELSM mechanisms)

	i. boyutun mevcut değeri	j. boyutun mevcut değeri	i. boyutun yeni değeri	j. boyutun yeni değeri
LSM	0	1	1	0
	1	0	0	1
ELSM-SWAP	0	1	1	0
	1	0	0	1
ELSM-OR	0	1	1	1
	1	0	1	1
ELSM-AND	0	1	0	0
	1	0	0	0

3.4.2. JayaX-LSM ve JayaX-ELSM algoritmalarının performans kıyaslaması için kullanılan parametre değerleri
(Parameter values used for performance comparison of JayaX-LSM and JayaX-ELSM algorithms)

Önerilen ELSM mekanizmasının etkisini gösterebilmek amacıyla bu bölümde CAP ve M* problem setleri üzerinde JayaX-LSM ve JayaX-ELSM algoritmalarının performans karşılaştırması yapılmıştır. Orijinal JayaX-LSM algoritmasının CAP problemlerindeki performansı Aslan vd. [25] tarafından analiz edilmiştir. İlgili çalışmada kullanılan değerlere sadık kalınarak JayaX-ELSM algoritması popülasyon genişliği 50, maksimum uygunluk hesaplama sayısı (maximum fitness evalution size, MaxFES) 80000 için 30 tekrarlı olarak çalıştırılmıştır. Orijinal JayaX-LSM'nin sonuçları doğrudan ilgili çalışmadan alınmıştır.

Aslan vd. kendi çalışmalarında [25] M* problemlerine yer vermediği için orijinal JayaX-LSM algoritması M* problemleri için ilk kez bu çalışmada test edilmiştir. M* problemleri için Haklı ve Ortaçay [49] tarafından kullanılan çalıştırma kriterlerine sadık kalınarak MaxFES 80000, çalıştırma tekrarı 100 olarak ayarlanmıştır. Çalışmanın en önemli kısımlarından biri de, LSM ve ELSM mekanizmalarında kullanılan Plocal parametresinin ayarlanmasıdır. Aslan vd. [25] JayaX-LSM algoritmasının Plocal parametresini analiz etmiş ve en uygun değerin 0,01 olduğuna karar vermiştir. Bu sebeple CAP problemleri için JayaX-LSM algoritmasının Plocal değeri 0,01 olarak kullanılmıştır. JayaX-ELSM algoritmasının CAP problemlerindeki en uygun Plocal değerini belirlemek amacıyla JayaX-ELSM Plocal parametresinin 0 - 0,01 - 0,02 - ... - 0,1 değerleri için çalıştırılmıştır. Burada Plocal değeri 0 seçildiğinde yerel arama mekanizması kullanılmadan algoritma çalıştırılmış olur. Elde edilen sonuçlar, matematiksel ifadesi Eş. 5'te verilen gap metriği üzerinden değerlendirilmiştir.

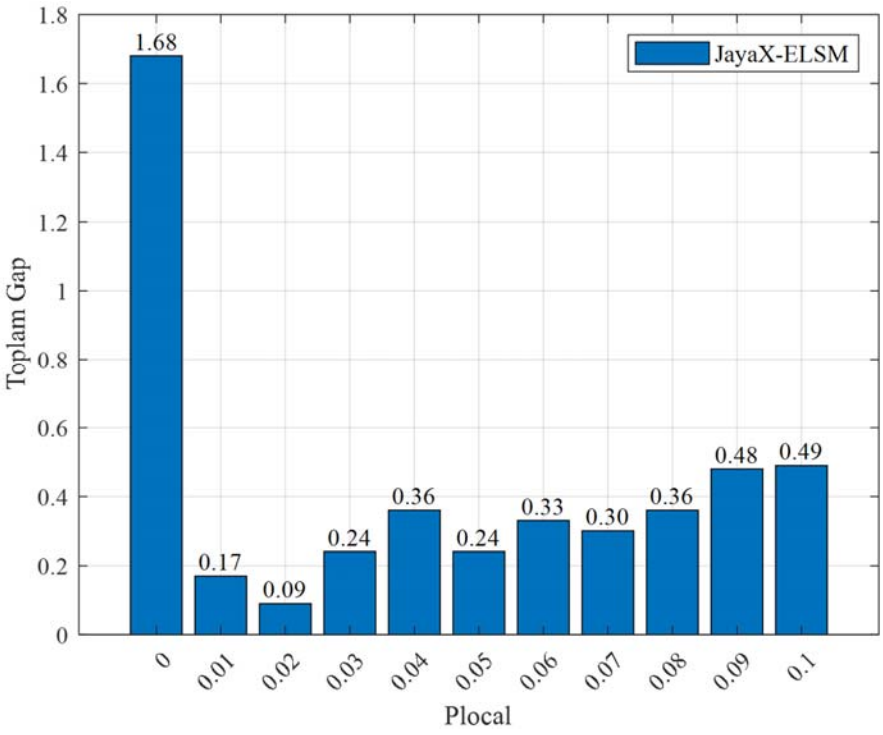
$$\text{gap} = \frac{f^{\text{ort}} - f^{\text{opt}}}{f^{\text{opt}}} \times 100 \tag{5}$$

Burada, f^{ort} tüm çalıştırmaların ortalama maliyetini, f^{opt} ise ilgili problem için optimal çözüm değerini ifade etmektedir. Gap değerinin küçülmesi optimal çözüme yaklaşıldığını gösterir. Yukarıda bahsedilen parametre ayarlama yöntemi JayaX-LSM ve JayaX-ELSM algoritmalarının M* problemleri için en uygun Plocal değerlerinin belirlenmesinde de kullanılmıştır. İlgili sonuçlar bölümün devamında verilmiştir.

3.4.2.1. JayaX-LSM ve JayaX-ELSM'nin CAP problemlerinde performans karşılaştırması
(Performance comparison of JayaX-LSM and JayaX-ELSM on CAP problems)

Bu bölümde, öncelikle Plocal parametresi 0 - 0,01 - 0,02 - ... - 0,1 değerleri için ayarlanmış ve JayaX-ELSM algoritması CAP problemleri üzerinde çalıştırılmıştır. Şekil 6'da verilen toplam gap değerleri incelendiğinde JayaX-ELSM algoritması için Plocal parametresinin en uygun değerinin 0,02 olduğu görülmektedir. Bu sebeple JayaX-ELSM algoritması için Plocal değeri 0,02 olarak ayarlanmıştır.

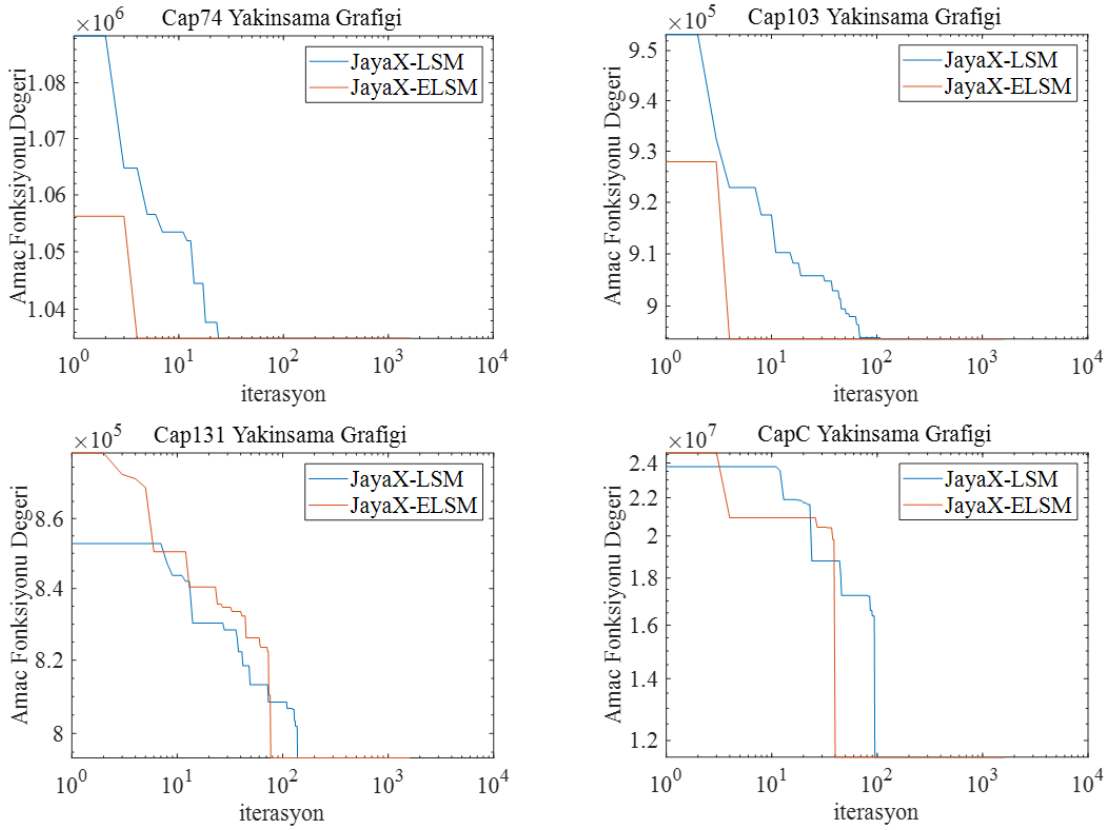
Tablo 5'te JayaX-LSM ve JayaX-ELSM algoritmalarının CAP problemleri üzerinde elde ettiği ortalama, standart sapma, gap ve hit değerleri verilmiştir. Burada hit, bir problemin optimal çözümüne ulaşma sayısını ifade etmektedir. JayaX-LSM algoritmasının performans sonuçları doğrudan Aslan vd. [25] tarafından yapılan çalışmadan alınmıştır. Tablo 5'teki sonuçlar incelendiğinde, CapB ve CapC dışındaki problemler için her iki algoritmanın da optimal çözümlere ulaştığı gözlenmektedir. CapB probleminde önerilen JayaX-ELSM algoritması JayaX-LSM algoritmasından daha iyi sonuçlar elde ederken, CapC probleminde ise tam tersi bir durum söz konusudur. Özetle CAP problem ailesinde her iki algoritmanın birbirine yakın bir performansa sahip olduğu görülmektedir. Şekil 7'de JayaX-LSM ve JayaX-ELSM algoritmalarının bazı CAP problemleri üzerinde elde ettiği amaç fonksiyon değerlerinin



Şekil 6. JayaX-ELSM algoritmasında 0 – 0,1 aralığındaki Plocal değerleri için CAP problemlerinden elde edilen toplam gap değerleri
(Total gap values obtained from CAP problems for Plocal values in the range of 0 – 0,1 in the JayaX-ELSM algorithm)

Tablo 5. JayaX-LSM ve JayaX-ELSM algoritmalarının CAP problemlerinde elde ettiği sonuçlar
(Obtained results of JayaX-LSM and JayaX-ELSM algorithms for CAP problems)

		JayaX-LSM (Plocal = 0,01) [25]				JayaX-ELSM (Plocal = 0,02)			
Problem	*Optimal*	*Ort*	*SS*	*Gap*	*Hit*	*Ort*	*SS*	*Gap*	*Hit*
Cap71	932615,75	932615,75	0,00	0,00	30	932615,75	0,00	0,00	30
Cap72	977799,40	977799,40	0,00	0,00	30	977799,40	0,00	0,00	30
Cap73	1010641,45	1010641,45	0,00	0,00	30	1010641,45	0,00	0,00	30
Cap74	1034976,98	1034976,98	0,00	0,00	30	1034976,98	0,00	0,00	30
Cap101	796648,44	796648,44	0,00	0,00	30	796648,44	0,00	0,00	30
Cap102	854704,20	854704,20	0,00	0,00	30	854704,20	0,00	0,00	30
Cap103	893782,11	893782,11	0,00	0,00	30	893782,11	0,00	0,00	30
Cap104	928941,75	928941,75	0,00	0,00	30	928941,75	0,00	0,00	30
Cap131	793439,56	793439,56	0,00	0,00	30	793439,56	0,00	0,00	30
Cap132	851495,33	851495,33	0,00	0,00	30	851495,33	0,00	0,00	30
Cap133	893076,71	893076,71	0,00	0,00	30	893076,71	0,00	0,00	30
Cap134	928941,75	928941,75	0,00	0,00	30	928941,75	0,00	0,00	30
CapA	17156454,48	17156454,48	0,00	0,00	30	17156454,48	0,00	0,00	30
CapB	12979071,58	12989379,44	27033,02	0,08	26	12985670,45	21979,12	0,05	27
CapC	11505594,33	11508089,97	5455,95	0,02	17	11510332,36	9597,01	0,04	15



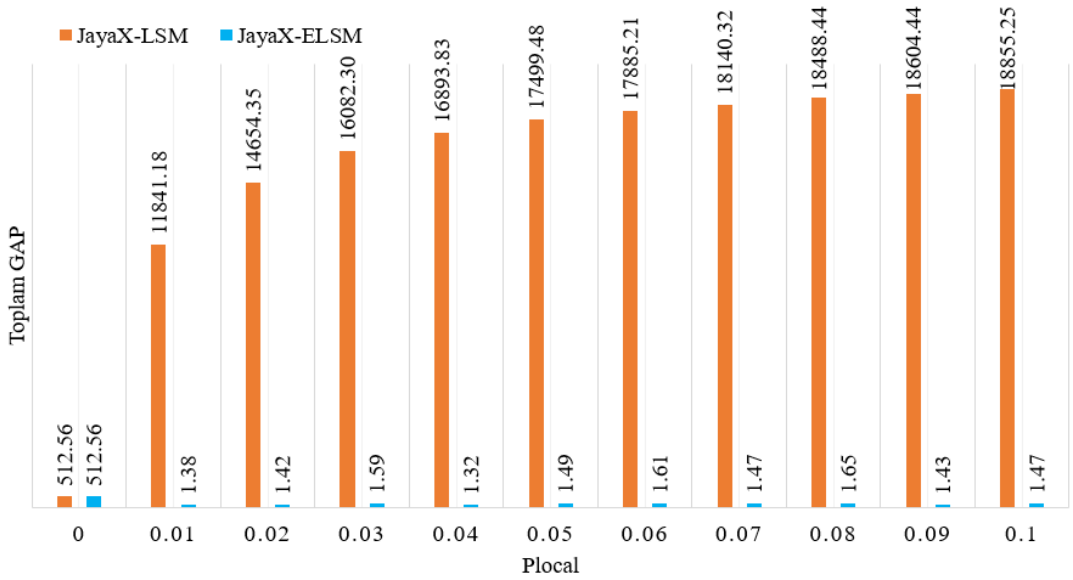
Şekil 7. JayaX-LSM ve JayaX-ELSM algoritmalarının bazı CAP problemleri için yakınsama grafikleri
(Convergence graphics of JayaX-LSM and JayaX-ELSM algorithms for some CAP problems)

iterasyonlara göre yakınsama grafikleri verilmiştir. Bu grafikler incelendiğinde önerilen JayaX-ELSM algoritmasının genellikle daha iyi bir yakınsama performansına sahip olduğu gözlenmektedir.

3.4.2.2. JayaX-LSM ve JayaX-ELSM'nin M problemlerinde performans karşılaştırması*
(Performance comparison of JayaX-LSM and JayaX-ELSM on M problems)*

Bu bölümde, Plocal parametresi 0 - 0,01 - 0,02 - ... - 0,1 değerleri için ayarlanmış, JayaX-LSM ve JayaX-ELSM algoritmaları M* problemleri üzerinde çalıştırılmıştır. Şekil 8'de verilen toplam gap değerlerine bakıldığında, JayaX-LSM algoritması için Plocal parametresinin en uygun değerinin 0 olduğu görülmektedir. Yani LSM mekanizması uygulanmadığı takdirde algoritma daha başarılı olmuştur. JayaX-LSM algoritması için Plocal parametresi 0,1'e yaklaştıkça toplam gap değeri artmıştır. Bu durum, LSM mekanizmasının M* problemleri üzerinde yerel aramaya katkı sunmadığını gösteren bir diğer işarettir. JayaX-ELSM algoritması tarafından elde edilen sonuçlar incelendiğinde ELSM mekanizmasının arama sürecine katkısı çok açık bir şekilde görülmektedir. JayaX-ELSM, Plocal parametresinin 0,01 ile 0,1 arasındaki tüm değerleri için birbirine yakın ve JayaX-LSM

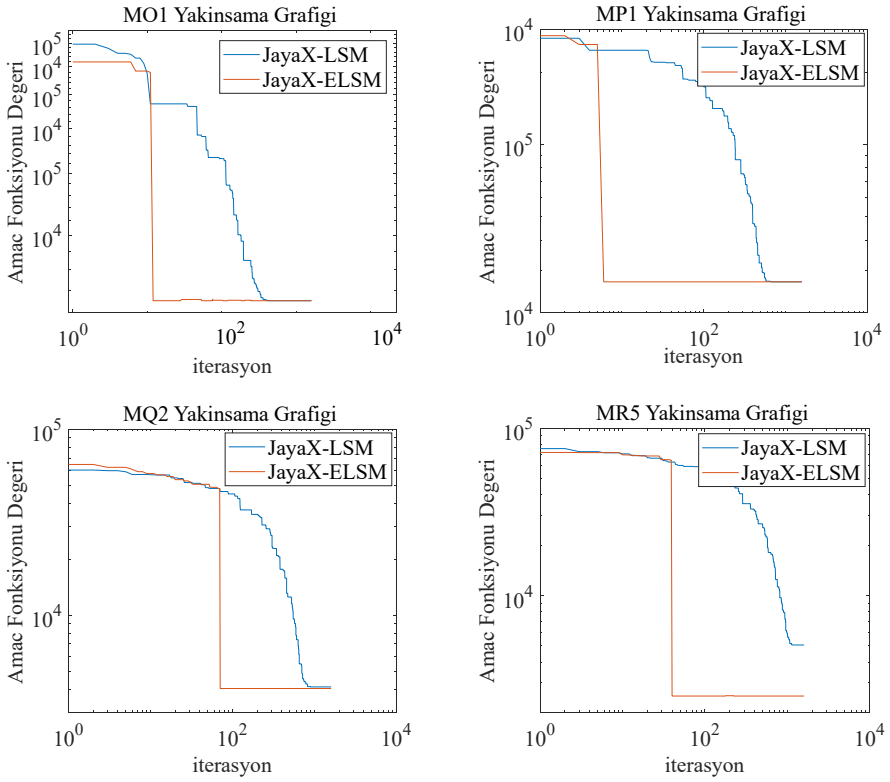
algoritmasına göre oldukça başarılı toplam gap değerleri elde etmiştir. M* problemlerinde Plocal parametresinin 0,04 değeri için en küçük toplam gap değeri (1,329) elde edilmiş olsa da CAP problemleriyle standart oluşturmak amacıyla M* problemlerinde de Plocal parametresinin 0,02 olarak ayarlanmasına karar verilmiştir. Tablo 6'da JayaX-LSM (Plocal = 0) ve JayaX-ELSM (Plocal = 0,02) algoritmalarının M* problemleri üzerinde elde ettiği ortalama, standart sapma, gap ve hit değerleri verilmiştir. Sonuçlar incelendiğinde JayaX-ELSM algoritmasının tüm problemlerde JayaX-LSM algoritmasından çok açık bir şekilde üstün olduğu gözlenmektedir. Hit sayıları incelendiğinde, JayaX-LSM algoritmasının hiçbir problemde 100 hit sayısına ulaşamadığı gözlenirken, önerilen JayaX-ELSM algoritması 9 farklı problemde 100 hit sayısına ulaşarak tüm çözümlerde optimal çözümleri elde edebilmiştir. Ayrıca MR4 ve MR5 problemlerinde de 99 hit sayısına ulaşmıştır. Problemler için elde edilen ortalama, standart sapma ve gap değerleri incelendiğinde önerilen JayaX-ELSM algoritmasının JayaX-LSM algoritmasına göre optimal çözümlere çok daha yakın ve



Şekil 8. JayaX-LSM ve JayaX-ELSM algoritmalarında 0 – 0,1 aralığındaki Plocal değerleri için M* problemlerinden elde edilen toplam gap değerleri
(Total gap values obtained from M* problems for Plocal values in the range of 0 – 0,1 in JayaX-ELSM algorithm)

Tablo 6. JayaX-LSM ve JayaX-ELSM algoritmalarının M* problemlerinde elde ettiği sonuçlar
(Obtained results of JayaX-LSM and JayaX-ELSM algorithms for M* problems)

		JayaX-LSM (Plocal = 0)				JayaX-ELSM (Plocal = 0,02)			
Problem	Optimal	Ort	SS	Gap	Hit	Ort	SS	Gap	Hit
MO1	1305,95	1306,86	3,50	0,07	92	1305,95	0,00	0,00	100
MO2	1432,36	1434,80	5,73	0,17	84	1434,08	5,08	0,12	89
MO3	1516,77	1521,17	7,62	0,29	55	1517,88	3,09	0,07	81
MO4	1442,24	1447,77	12,71	0,38	83	1442,24	0,00	0,00	100
MO5	1408,77	1411,01	7,99	0,16	71	1409,45	1,07	0,05	71
MP1	2686,48	2726,56	44,83	1,49	29	2686,48	0,00	0,00	100
MP2	2904,86	2958,13	52,97	1,83	11	2904,86	0,00	0,00	100
MP3	2623,71	2683,38	65,59	2,27	34	2623,71	0,00	0,00	100
MP4	2938,75	2984,63	53,73	1,56	20	2941,59	2,76	0,10	44
MP5	2932,33	2968,03	48,20	1,22	17	2937,82	5,02	0,19	37
MQ1	4091,01	4483,78	302,60	9,60	2	4091,01	0,00	0,00	100
MQ2	4028,33	4373,98	256,85	8,58	2	4028,33	0,00	0,00	100
MQ3	4275,43	4685,77	304,13	9,60	4	4275,43	0,00	0,00	100
MQ4	4235,15	4691,95	322,57	10,79	0	4235,15	0,00	0,00	100
MQ5	4080,74	4412,48	269,40	8,13	3	4087,05	11,60	0,15	76
MR1	2608,15	5225,45	1002,33	100,35	0	2609,76	3,39	0,06	34
MR2	2654,73	5197,28	1017,52	95,77	0	2671,63	22,68	0,64	63
MR3	2788,25	5346,33	946,12	91,74	0	2789,41	2,02	0,04	75
MR4	2756,04	5144,37	906,18	86,66	0	2756,22	1,78	0,01	99
MR5	2505,05	4778,71	882,17	90,76	0	2505,11	0,57	0,00	99



Şekil 9. JayaX-LSM ve JayaX-ELSM algoritmalarının bazı M* problemleri için yakınsama grafikleri
(Convergence graphics of JayaX-LSM and JayaX-ELSM algorithms for some M* problems)

Tablo 7. Karşılaştırma-1: JayaX-ELSM ve diğer ikili algoritmaların CAP problemleri üzerinde elde ettikleri gap değerleri
(Comparison-1: Performance comparison of JayaX-ELSM and other binary algorithms on CAP problems with gap metric)

	ABC_{bin}	DisABC	binABC	DisDE	BinDE	BinSSA	BPSO	CPSO	binAAA	JayaX-ELSM
Cap71	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	5,00E-02	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	1	2	1	1
Cap72	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	7,00E-02	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	1	2	1	1
Cap73	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	2,42E-02	6,00E-02	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	2	3	1	1
Cap74	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	8,82E-03	7,00E-02	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	2	3	1	1
Cap101	0,00E+00	0,00E+00	0,00E+00	7,20E-03	0,00E+00	0,00E+00	4,32E-02	1,40E-01	0,00E+00	0,00E+00
BS	1	1	1	2	1	1	3	4	1	1
Cap102	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	9,89E-03	1,50E-01	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	2	3	1	1
Cap103	5,10E-03	0,00E+00	0,00E+00	8,40E-04	0,00E+00	0,00E+00	4,94E-02	1,60E-01	0,00E+00	0,00E+00
BS	3	1	1	2	1	1	4	5	1	1
Cap104	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	4,05E-02	1,80E-01	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	2	3	1	1
Cap131	2,00E-01	6,20E-01	0,00E+00	0,00E+00	3,60E-03	0,00E+00	1,71E-01	7,50E-01	0,00E+00	0,00E+00
BS	4	5	1	1	2	1	3	6	1	1
Cap132	2,00E-02	9,50E-02	0,00E+00	0,00E+00	5,00E-03	0,00E+00	5,83E-02	7,80E-01	0,00E+00	0,00E+00
BS	3	5	1	1	2	1	4	6	1	1
Cap133	7,50E-02	3,10E-02	1,20E-01	1,50E-02	1,40E-02	0,00E+00	8,29E-02	7,30E-01	0,00E+00	0,00E+00
BS	5	4	7	3	2	1	6	8	1	1
Cap134	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	1,95E-01	8,90E-01	0,00E+00	0,00E+00
BS	1	1	1	1	1	1	2	3	1	1
CapA	3,20E+00	1,50E-01	3,00E+00	3,70E-02	1,30E+00	0,00E+00	1,70E+00	2,20E+01	0,00E+00	0,00E+00
BS	7	3	6	2	4	1	5	8	1	1
CapB	2,80E+00	3,30E+00	2,50E+00	6,70E-02	1,50E+00	2,55E-01	1,40E+00	1,10E+01	2,50E-01	5,08E-02
BS	8	9	7	2	6	4	5	10	3	1
CapC	2,00E+00	4,70E+00	2,60E+00	5,80E-02	1,60E+00	4,34E-01	1,60E+00	9,70E+00	2,90E-01	4,12E-02
BS	6	8	7	2	5	4	5	9	3	1
OrtBS	2,93	2,87	2,53	1,47	1,67	1,40	3,13	5,00	1,27	1,00

daha kararlı sonuçlar ürettiği gözlenmiştir. Özetle, ELSM mekanizmasının JayaX algoritmasının yerel arama kabiliyetine olan katkısı M* problemlerinde çok açık bir şekilde görülmektedir. Şekil 9'da JayaX-LSM ve JayaX-ELSM algoritmalarının bazı M* problemleri üzerinde elde ettiği amaç fonksiyon değerlerinin iterasyonlara göre yakınsama grafikleri verilmiştir. Bu grafikler

incelendiğinde önerilen JayaX-ELSM algoritmasının çok daha üstün bir yakınsama performansına sahip olduğu gözlenmektedir.

4. Deneysel Sonuçlar (Experimental Results)

Bu bölümde, önerilen JayaX-ELSM algoritmasının yakın zamanda yayınlanmış veya bu alanda iyi bilinen diğer algoritmalarla Cap ve M* problem setleri üzerinde performans karşılaştırması yapılmıştır. Karşılaştırma için kullanılan algoritmaların sonuçları doğrudan ilgili çalışmalardan alınmıştır.

4.1. Parametre Ayarları (Parameter Tuning)

JayaX-ELSM algoritması, karşılaştırma için kullanılan algoritmalara paralel olarak CAP problemlerinde 80000 MaxFES için 30 tekrarlı, M* problemlerinde 80000 MaxFES için 100 tekrarlı çalıştırılmıştır. JayaX-ELSM algoritmasının Plocal parametresi yapılan deneysel çalışmaların ışığında 0,02 olarak ayarlanmıştır. Güncellenecek boyut sayısını belirleyen P parametresi ise JayaX-LSM algoritmasına paralel olarak 0,5 değeriyle kullanılmıştır.

4.2. Karşılaştırma-1: JayaX-ELSM ve Diğer Algoritmaların CAP Problemleri Üzerindeki Performans Karşılaştırması (Comparison-1: Performance Comparison of JayaX-ELSM and Other Binary Algorithms on CAP Problems)

Tablo 7'de literatürde iyi bilinen ABCbin, DisABC, binABC, DisDE, BinDE, BinSSA, BPSO, CPSO ve binAAA isimli 9 farklı algoritma ile önerilen JayaX-ELSM algoritmasının CAP problemlerindeki performansları gap metriği kullanılarak karşılaştırılmıştır. Karşılaştırma için kullanılan algoritmaların sonuçları Kıran [63], Baş ve Ülker [15], Korkmaz ve Kıran [64] tarafından yapılan çalışmalardan doğrudan alınmıştır. Her problemin altında algoritmaların o problem için elde ettiği başarı sıraları (BS) verilmiştir. Tablonun altında ise algoritmaların tüm problem setindeki ortalama başarı sırası (OrtBS) verilmiştir. Tablo 7'deki sonuçlar incelendiğinde önerilen JayaX-ELSM algoritmasının tüm problemlerde karşılaştırılan algoritmalarla eşit veya bu algoritmalardan daha üstün bir performans elde ettiği gözlenmiştir. Bu duruma paralel olarak önerilen algoritma, hem problem bazında hem de tüm problem setinde ortalama başarı sırası açısından ilk sırayı almıştır.

Tablo 8. Karşılaştırma-2: JayaX-ELSM ve diğer ikili algoritmaların M* problemleri üzerindeki performans karşılaştırması (Comparison-2: Performance comparison of JayaX-ELSM and other binary algorithms on M* problems)

	LS		ISS		JayaX-ELSM	
	Ortalama	Gap	Ortalama	Gap	Ortalama	Gap
MO1	1305,95	0,00	1305,95	0,00	1305,95	0,00
BS	1		1		1	
MO2	1432,70	0,01	1432,36	0,00	1434,08	0,12
BS	2		1		3	
MO3	1520,27	0,23	1516,77	0,00	1517,88	0,07
BS	3		1		2	
MO4	1442,24	0,00	1442,24	0,00	1442,24	0,00
BS	1		1		1	
MO5	1409,17	0,03	1408,77	0,00	1409,45	0,05
BS	2		1		3	
MP1	2688,50	0,08	2686,66	0,01	2686,48	0,00
BS	3		2		1	
MP2	2904,86	0,00	2904,86	0,00	2904,86	0,00
BS	1		1		1	
MP3	2624,77	0,04	2624,34	0,02	2623,71	0,00
BS	3		2		1	
MP4	2939,53	0,03	2940,80	0,07	2941,59	0,10
BS	1		2		3	
MP5	2933,46	0,04	2932,60	0,01	2937,82	0,19
BS	2		1		3	
MQ1	4091,01	0,00	4091,01	0,00	4091,01	0,00
BS	1		1		1	
MQ2	4028,33	0,00	4030,08	0,04	4028,33	0,00
BS	1		2		1	
MQ3	4275,43	0,00	4275,43	0,00	4275,43	0,00
BS	1		1		1	
MQ4	4235,47	0,01	4236,46	0,03	4235,15	0,00
BS	2		3		1	
MQ5	4086,53	0,14	4095,46	0,36	4087,05	0,15
BS	1		3		2	
MR1	2608,24	0,00	2647,03	1,49	2609,76	0,06
BS	1		3		2	
MR2	2654,73	0,00	2691,54	1,39	2671,63	0,64
BS	1		3		2	
MR3	2789,04	0,03	2832,33	1,58	2789,41	0,04
BS	1		3		2	
MR4	2756,04	0,00	2807,90	1,88	2756,22	0,01
BS	1		3		2	
MR5	2505,48	0,02	2549,97	1,79	2505,11	0,00
BS	2		3		1	
OrtBS	1,55		1,90		1,70	

4.3. Karşılaştırma-2: JayaX-ELSM ve Diğer Algoritmaların M Problemleri Üzerindeki Performans Karşılaştırması*
(Comparison-2: Performance Comparison of JayaX-ELSM and Other Binary Algorithms on M Problems)*

Tablo 8'de LS [65] ve ISS [49] algoritmaları ile önerilen JayaX-ELSM algoritmasının M* problemlerindeki performansları ortalama ve gap metriği kullanılarak karşılaştırılmıştır. Karşılaştırma için kullanılan algoritmaların sonuçları ilgili çalışmalardan doğrudan alınmıştır. Her problemin altında algoritmaların o problem için elde ettiği başarı sıraları (BS) verilmiştir. Tablonun altında ise algoritmaların tüm problem setindeki ortalama başarı sıraları (OrtBS) verilmiştir. Elde edilen sonuçlar incelendiğinde, tüm MO problemlerinde ISS algoritmasının optimal çözümlere ulaşarak en başarılı algoritma olduğu gözlenmektedir. MP problemlerinin 3'ünde önerilen JayaX-ELSM algoritması optimal çözümlere ulaşırken, LS ve ISS algoritmaları 2'şer problemde en başarılı sonuçları elde etmiştir. MQ problemlerinin 4'ünde JayaX-ELSM ve LS algoritmaları ilk sırayı elde ederken, ISS algoritması 2 problemde ilk sırayı almıştır. MR problemlerinin 4'ünde LS birinde önerilen algoritma ilk sırayı alırken, ISS algoritması tüm MR problemlerinde üçüncü yer almıştır. Tüm problem setindeki ortalama başarı sıralamasına bakıldığında LS algoritması birinci olurken, önerilen JayaX-ELSM algoritması ikinci sırada yer almıştır.

5. Sonuçlar ve Tartışmalar (Conclusions and Discussions)

Bu çalışmada, JayaX algoritmasının yerel arama kabiliyetini artırmak amacıyla ELSM adı verilen yeni bir mekanizma önerilmiştir. ELSM mekanizması JayaX algoritmasına eklenerek JayaX-ELSM algoritması önerilmiştir. JayaX-ELSM algoritmasının performansını değerlendirmek için iki aşamalı olarak deneysel çalışmalar yapılmıştır. Çalışmanın ilk kısmında JayaX-LSM ve JayaX-ELSM algoritmaları CAP problemleri üzerinde ikili olarak karşılaştırılmıştır. Elde edilen sonuçlar her iki yaklaşımın da CAP problemlerinde üstün bir başarıya sahip olduğu göstermiştir. Ardından, önerilen JayaX-ELSM algoritması CAP problemleri üzerinde literatürdeki 9 farklı algoritma ile karşılaştırılmıştır. Elde edilen sonuçlar incelendiğinde, önerilen algoritmanın karşılaştırma yapılan algoritmaların tamamından daha başarılı sonuçlar üreterek birinci sırada yer aldığı gözlemlenmiştir. Çalışmanın ikinci kısmında JayaX-LSM ve JayaX-ELSM algoritmaları boyut sayısı ve zorluk derecesi yüksek M* problemleri üzerinde ikili olarak karşılaştırılmıştır. Önerilen JayaX-ELSM algoritması, M* problem setinde JayaX-LSM algoritmasından çok daha başarılı sonuçlar elde etmiştir. Bu durum ELSM mekanizmasının yerel aramaya olan katkısını göstermektedir. Daha sonra JayaX-ELSM algoritması literatürdeki 2 farklı algoritma ile karşılaştırılmış ve ortalama başarı sırası açısından ikinci sırada yer almıştır. Sonuç olarak, LSM mekanizmasının konum güncelleme aşamasında ortaya çıkarılamayan kombinasyonlar önerilen, seçime dayalı ELSM mekanizması sayesinde oluşturularak çözüm uzayının daha etkili bir şekilde aranması sağlanmaktadır. Algoritmadaki bireylerin farklı çözüm seçeneklerini deneme şansı bulması ve bu seçeneklerden yeni çözümler elde edebilmesi popülasyonu daha başarılı konumlara yönlendirmektedir. Elde edilen sonuçlar LSM mekanizmasında ortaya çıkan yerel optimuma takılma sorununun giderildiğini göstermektedir.

Sonraki çalışmalarda, önerilen ELSM mekanizmasının diğer metasezgisel algoritmalar üzerinde nasıl bir etki yapacağı araştırılabilir. Ayrıca UFLP dışındaki rüzgar tribünü yerleşimi, bulut hesaplamadaki kaynak tahsisi, sırt çantası, öznitelik seçimi, iş akış çizelgeleme gibi ikili problemlerde de ELSM mekanizmasının yerel aramaya olan etkisi araştırılabilir.

Kaynaklar (References)

1. Murty K., Optimization models for decision making, 2003.
2. Gould N., An introduction to algorithms for continuous optimization, Oxford University Computing Laboratory Notes, 2006.
3. Yuan X., Nie H., Su A., Wang L., Yuan Y., An improved binary particle swarm optimization for unit commitment problem, Expert Systems with applications, 36 (4), 8049-8055, 2009.
4. He Y., Zhang F., Mirjalili S., Zhang T., Novel binary differential evolution algorithm based on Taper-shaped transfer functions for binary optimization problems, Swarm and Evolutionary Computation, 101022, 2021.
5. Hakli H., BinEHO: a new binary variant based on elephant herding optimization algorithm, Neural Computing and Applications, 32 (22), 16971-16991, 2020.
6. Sahinkoc H.M.,Bilge Ü., A reference set based many-objective co-evolutionary algorithm with an application to the knapsack problem, European Journal of Operational Research, 2021.
7. Tongur V.,Ülker E., Migrating Birds Optimization (MBO) Algorithm to Solve Graph Coloring Problem, International Journal of Engineering Science, 14545, 2017.
8. Aslan M.,Baykan N.A., A performance comparison of graph coloring algorithms, International Journal of Intelligent Systems and Applications in Engineering, 1-7, 2016.
9. Ibrahim I.M., Task scheduling algorithms in cloud computing: A review, Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12 (4), 1041-1053, 2021.
10. Abualigah L.,Diabat A., A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, Cluster Computing, 24 (1), 205-223, 2021.
11. Külahlı S., Engin O., İsmail K., A New Hybrid Scatter Search Algorithm for Solving the Flexible Job Shop Scheduling Problems, Celal Bayar University Journal of Science, 17 (4), 347-359, 2021.
12. Özbay F.A.,Özbay E., A new approach for gender detection from voice data: Feature selection with optimization methods, Journal of the Faculty of Engineering and Architecture of Gazi University, 38 (2), 1179-1192, 2023.
13. Inan O., Uzer M.S., Yılmaz N., A new hybrid feature selection method based on association rules and PCA for detection of breast cancer, International Journal of Innovative Computing, Information and Control, 9 (2), 727-729, 2013.
14. Dhiman G., Oliva D., Kaur A., Singh K.K., Vimal S., Sharma A., Cengiz K., BEPO: A novel binary emperor penguin optimizer for automatic feature selection, Knowledge-Based Systems, 211, 106560, 2021.
15. Baş E.,Ülker E., A binary social spider algorithm for uncapacitated facility location problem, Expert Systems with Applications, 161, 113618, 2020.
16. Cinar A.C.,Kiran M.S., Similarity and logic gate-based tree-seed algorithms for binary optimization, Computers & Industrial Engineering, 115, 631-646, 2018.
17. Koc I., A comprehensive analysis of grid-based wind turbine layout using an efficient binary invasive weed optimization algorithm with levy flight, Expert Systems with Applications, 198, 116835, 2022.
18. Gao X., Yang H., Lin L., Koo P., Wind turbine layout optimization using multi-population genetic algorithm and a case study in Hong Kong offshore, Journal of Wind Engineering and Industrial Aerodynamics, 139, 89-99, 2015.
19. Aslan M., Gunduz M., Kiran M.S., A Jaya-based approach to wind turbine placement problem, Energy Sources, Part A: Recovery, Utilization, and Environmental Effects, 1-20, 2020.
20. Sbihi A., Adaptive perturbed neighbourhood search for the expanding capacity multiple-choice knapsack problem, Journal of the Operational Research Society, 64 (10), 1461-1473, 2013.
21. Ghezelsoflu A., Di Francesco M., Frangioni A., Zuddas P., A set-covering formulation for a drayage problem with single and double container loads, Journal of Industrial Engineering International, 14 (4), 665-676, 2018.
22. Rizk-Allah R.M.,Hassanien A.E., New binary bat algorithm for solving 0–1 knapsack problem, Complex & Intelligent Systems, 4 (1), 31-53, 2018.
23. Banitalebi A., Abd Aziz M.I., Aziz Z.A., A self-adaptive binary differential evolution algorithm for large scale binary optimization problems, Information Sciences, 367, 487-511, 2016.
24. Rao R., Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, International Journal of Industrial Engineering Computations, 7 (1), 19-34, 2016.
25. Aslan M., Gunduz M., Kiran M.S., JayaX: Jaya algorithm with xor operator for binary optimization, Applied Soft Computing, 82, 105576, 2019.
26. Eberhart R.,Kennedy J., A new optimizer using particle swarm theory, MHS'95. Proceedings of the sixth international symposium on micro machine and human science, 39-43, 1995.

27. Hasanoğlu M.S., Dölen M., Comparison of multi-objective and single-objective approaches in feasibility enhanced particle swarm optimization, Journal of the Faculty of Engineering and Architecture of Gazi University, 35 (2), 887-900, 2020.

28. Kennedy J.,Eberhart R.C. A discrete binary version of the particle swarm algorithm, 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation, 4104-4108, 1997.

29. Khanesar M.A., Teshnehlab M., Shoorehdeli M.A. A novel binary particle swarm optimization, 2007 Mediterranean conference on control & automation, 1-6, 2007.

30. Beheshti Z., Shamsuddin S.M., Hasan S., Memetic binary particle swarm optimization for discrete optimization problems, Information Sciences, 299, 58-84, 2015.

31. Guner A.R.,Sevkli M., A discrete particle swarm optimization algorithm for uncapacitated facility location problem, Journal of Artificial Evolution and Applications, 2008, 2008.

32. Nezamabadi-pour H., Rostami-Shahrbabaki M., Maghfoori-Farsangi M., Binary particle swarm optimization: challenges and new solutions, CSI J Comput Sci Eng, 6 (1), 21-32, 2008.

33. Saha S., Kole A., Dey K. A modified continuous particle swarm optimization algorithm for uncapacitated facility location problem, International Conference on Advances in Information Technology and Mobile Communication, 305-311, 2011.

34. Storn R.,Price K., Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization, 11 (4), 341-359, 1997.

35. Pampara G., Engelbrecht A.P., Franken N. Binary differential evolution, 2006 IEEE International Conference on Evolutionary Computation, 1873-1879, 2006.

36. Engelbrecht A.P.,Pampara G. Binary differential evolution strategies, 2007 IEEE Congress on Evolutionary Computation, 1942-1947, 2007.

37. Su H.,Yang Y. Quantum-Inspired Differential Evolution for Binary Optimization, 2008 Fourth International Conference on Natural Computation, 341-346, 2008.

38. Chen Y., Xie W., Zou X., A binary differential evolution algorithm learning from explored solutions, Neurocomputing, 149, 1038-1047, 2015.

39. He X., Zhang Q., Sun N., Dong Y. Feature selection with discrete binary differential evolution, 2009 international conference on artificial intelligence and computational intelligence, 327-330, 2009.

40. Deng C., Zhao B., Yang Y., Deng A. Novel binary differential evolution algorithm for discrete optimization, 2009 Fifth International Conference on Natural Computation, 346-349, 2009.

41. Yang Q. A comparative study of discrete differential evolution on binary constraint satisfaction problems, 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 330-335, 2008.

42. Wang L., Fu X., Menhas M.I., Fei M., A modified binary differential evolution algorithm, Life System Modeling and Intelligent Computing, Springer. 49-57, 2010.

43. Kashan M.H., Kashan A.H., Nahavandi N., A novel differential evolution algorithm for binary optimization, Computational Optimization and Applications, 55 (2), 481-513, 2013.

44. Karaboga D., An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

45. Kashan M.H., Nahavandi N., Kashan A.H., DisABC: A new artificial bee colony algorithm for binary optimization, Applied Soft Computing, 12 (1), 342-352, 2012.

46. Kiran M.S.,Gündüz M., XOR-based artificial bee colony algorithm for binary optimization, Turkish Journal of Electrical Engineering & Computer Sciences, 21 (Sup. 2), 2307-2328, 2013.

47. Kiran M.S., A binary artificial bee colony algorithm and its performance assessment, Expert Systems with Applications, 175, 114817, 2021.

48. Ozturk C., Hancer E., Karaboga D., A novel binary artificial bee colony algorithm based on genetic operators, Information Sciences, 297, 154-170, 2015.

49. Hakli H.,Ortacay Z., An improved scatter search algorithm for the uncapacitated facility location problem, Computers & Industrial Engineering, 135, 855-867, 2019.

50. James J.,Li V.O., A social spider algorithm for global optimization, Applied soft computing, 30, 614-627, 2015.

51. Korkmaz S., Babalik A., Kiran M.S., An artificial algae algorithm for solving binary optimization problems, International Journal of Machine Learning and Cybernetics, 9 (7), 1233-1247, 2018.

52. Sörensen K., Metaheuristics—the metaphor exposed, International Transactions in Operational Research, 22 (1), 3-18, 2015.

53. Wolpert D.H.,Macready W.G., No free lunch theorems for optimization, IEEE transactions on evolutionary computation, 1 (1), 67-82, 1997.

54. Cornuéjols G., Nemhauser G., Wolsey L., The uncapicitated facility location problem, Cornell University Operations Research and Industrial Engineering, 1983.

55. Glover F., Hanafi S., Guemri O., Crevits I., A simple multi-wave algorithm for the uncapacitated facility location problem, Frontiers of engineering management, 5 (4), 451-465, 2018.

56. Jakob K.,Pruzan P.M., The simple plant location problem: Survey and synthesis, European journal of operational research, 12, 36-81, 1983.

57. Monabbati E.,Kakhki H.T., On a class of subadditive duals for the uncapacitated facility location problem, Applied Mathematics and Computation, 251, 118-131, 2015.

58. Kole A., Chakrabarti P., Bhattacharyya S., An ant colony optimization algorithm for uncapacitated facility location problem, 2013.

59. Tuncbilek N., Tasgetiren F., Esnaf S., Artificial bee colony optimization algorithm for uncapacitated facility location problems, Journal of Economic and Social Research, 14 (1), 1, 2012.

60. Beasley J.E., OR-Library: distributing test problems by electronic mail, Journal of the operational research society, 41 (11), 1069-1072, 1990.

61. Ingle K.K.,Jatoth R.K., An efficient JAYA algorithm with lévy flight for non-linear channel equalization, Expert Systems with Applications, 145, 112970, 2020.

62. Zhang X., Wu C., Li J., Wang X., Yang Z., Lee J.-M., Jung K.-H., Binary artificial algae algorithm for multidimensional knapsack problems, Applied Soft Computing, 43, 583-595, 2016.

63. Kiran M.S., The continuous artificial bee colony algorithm for binary optimization, Applied Soft Computing, 33, 15-23, 2015.

64. Korkmaz S.,Kiran M.S., An artificial algae algorithm with stigmergic behavior for binary optimization, Applied Soft Computing, 64, 627-640, 2018.

65. Cura T., A parallel local search approach to solving the uncapacitated warehouse location problem, Computers & Industrial Engineering, 59 (4), 1000-1009, 2010.