Necmettin Erbakan Üniversitesi
Fen ve Mühendislik Bilimleri Dergisi
Journal of Science and Engineering (NEJSE)
DOI: https://doi.org/10.47112/neufmbd.2021.7

Cilt: 3 Sayı:2 Yıl: 2021 | Araştırma Makelesi / Research Article | E-ISSN: 2667- 7989

Transfer Fonksiyonları Kullanarak İkili Güve-Alev Optimizasyonu Algoritmalarının Geliştirilmesi ve Performanslarının Karşılaştırılması

Murat KARAKOYUN [1] **Ahmet ÖZKIŞ** [2]

[1] Necmettin Erbakan Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Konya, Türkiye, mkarakoyun@erbakan.edu.tr
[2] Necmettin Erbakan Üniversitesi, Mühendislik Fakültesi, Adli Bilişim Mühendisliği Bölümü, Konya, Türkiye, ahmetozkis@gmail.com *(Sorumlu Yazar/ Corresponding Author)*

Makale Bilgileri

Makale Geçmişi
Geliş: 14.09.2021
Kabul: 01.11.2021
Yayın: 28.12.2021

Anahtar Kelimeler:
Güve-Alev Optimizasyonu, Kapasitesiz Tesis Yerleştirme Problemi, Metasezgisel Algoritmalar, İkili Optimizasyon Problemleri

ÖZET

Güç sistemleri problemleri, ağ optimizasyonu, sırt çantası problemi gibi çoğu gerçek dünya problemi ikili optimizasyon problemi olarak ifade edilir. İkili optimizasyon problemlerinin klasik matematiksel tekniklerle çözümü çoğu zaman ya uzun zaman almakta ya da mümkün olamamaktadır. Bu sebeple ikili optimizasyon problemlerinin çözümü için metasezgisel algoritmaların kullanımı oldukça yaygındır. Literatürde yer alan metasezgisel algoritmaların çoğu, sürekli problemlerin çözümüne uygun bir yapıya sahip olduğu için bu algoritmaların ikili problemleri çözebilecek şekilde düzenlenmesi gerekir. Transfer fonksiyonları olarak isimlendirilen bazı fonksiyonlar aracılığı ile sürekli algoritmaları ikili algoritmalara dönüştürmek mümkündür. Bu çalışmada, son yıllarda önerilen doğa-esinli bir metasezgisel algoritma olan Güve-Alev Optimizasyonu (GAO) algoritması 8 farklı transfer fonksiyonu ile düzenlenerek 8 ayrı algoritma geliştirilmiştir. Geliştirilen algoritmalar, OR-Lib kütüphanesinden alınan 15 farklı kapasitesiz tesis yerleştirme problemi üzerinde çalıştırılmış ve gap olarak isimlendirilen bir hata metriğine göre değerlendirilmiştir. Elde edilen sonuçlar incelendiğinde, S3 transfer fonksiyonu ile geliştirilen ikili GAO algoritmasının 15 problemin 13'ünde en küçük gap değerini elde ettiği ve diğer transfer fonksiyonları ile geliştirilen algoritmalardan daha başarılı olduğu gözlemlenmiştir.

Development of Binary Moth-Flame Optimization Algorithms using Transfer Functions and Their Performance Comparison

Article Info

Article History
Received: 14.09.2021
Accepted: 01.11.2021
Published: 28.12.2021

Keywords:
Moth-Flame Optimization, Uncapacitated Facility Location Problem, Metaheuristic Algorithms, Binary Optimization Problems

ABSTRACT

Many real-world problems such as power systems problems, network optimization, backpack problems are referred to as binary optimization problems. The solution of binary optimization problems with classical mathematical techniques often takes a long time or is not possible. For this reason, the use of metaheuristic algorithms for the solution of binary optimization problems is quite common. Since most of the metaheuristic algorithms in the literature have a structure suitable for solving continuous problems, these algorithms should be arranged in a way that can solve binary problems. It is possible to convert continuous algorithms to binary algorithms by means of some functions called transfer functions. In this study, 8 different algorithms were developed by arranging the Moth-Flame Optimization (GAO) algorithm, a nature-inspired metaheuristic algorithm proposed in recent years, with 8 different transfer functions. The developed algorithms were run on 15 different uncapacitated facility location problems taken from the OR-Library and evaluated according to an error metric called gap. When the results are examined, it is observed that the binary GAO algorithm developed with the S3 transfer function achieves the minimum gap value in 13 of 15 problems and is more successful than the algorithms developed with other transfer functions.

Atıf/Citation: Karakoyun, M. & Özkış, A. (2021). Transfer fonksiyonları kullanarak ikili güve-alev optimizasyonu algoritmalarının geliştirilmesi ve performanslarının karşılaştırılması, *Necmettin Erbakan Üniversitesi Fen ve Mühendislik Bilimleri Dergisi*, 3(2), 1-10.

GİRİŞ (INTRODUCTION)

İkili optimizasyon problemlerinde, karar değişkenleri 0 ve 1 ile temsil edilen iki değerden birini alabilir. Bu değerler problemin yapısına göre farklı anlamlara gelebilmektedir. Örneğin, güç sistemlerinde 0, "kapalı" durumunu, 1 ise "açık" durumunu temsil ederken [1], ikili görüntü işlemede 0 siyah rengi, 1 beyaz rengi temsil eder [2]. Ağ optimizasyonu, sırt çantası problemi, bazı sınıflandırma ve kümeleme problemleri, kapasitesiz tesis yerleştirme problemleri gibi birçok gerçek dünya problemi ikili optimizasyon problemi olarak kabul edilir [3]. Bu çalışmada, ikili optimizasyon problemlerinden biri olan *kapasitesiz tesis yerleştirme problemi* (KTYP) üzerinde durulacaktır.

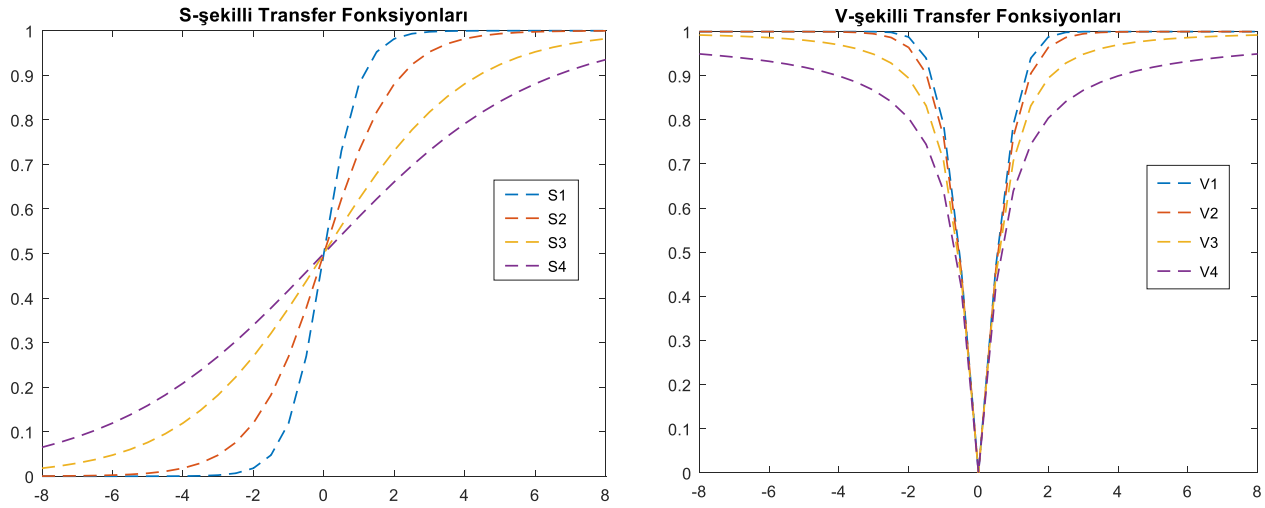
KTYP, bir grup müşteriye minimum maliyetle hizmet verebilmek için kaç adet tesisin açık durumda olması gerektiğine karar verilen bir problem türüdür. KTYP'de açık olup olmayacağına karar verilecek tesis sayısı yani problemin boyutu arttıkça problem daha da zorlaşır. Literatürde önerilen lagrangian teknikleri, indirgeme şemaları ve tamsayılı programlama gibi geleneksel yöntemler düşük boyutlu problemlerin çözümünde başarılı olurken, yüksek boyutlu problemlerde başarılı olamamaktadır [4,5]. Bu nedenle araştırmacılar KTYP problemlerinin çözümünde, makul sürelerde optimuma yakın sonuçlar elde edebilen ve farklı problem türlerine kolaylıkla adapte edilebilen metasezgisel algoritmaları kullanmaya yönelmişlerdir.

Literatürdeki çoğu metasezgisel algoritma arama uzayında sürekli değerler alan optimizasyon problemlerini çözmek amacıyla önerilmiştir. Bu tip algoritmaların ikili problemleri çözebilmesi için arama uzayındaki sürekli değerlerin bazı teknikler kullanılarak ikili değerlere (0 veya 1) dönüştürülmesi gerekir. Bu amaçla yaygın olarak kullanılan tekniklerden biri transfer fonksiyonlarıdır. 1997 yılında Kennedy ve Eberhart [6] tarafından, sigmoid fonksiyonu kullanılarak sürekli Parçacık Sürü Optimizasyonu algoritmasının ikili algoritmaya dönüştürülmesi, transfer fonksiyonlarının kullanımının öncü örneklerinden biridir. Mirjalili ve Lewis [7] ise 2013 yılında 8 farklı transfer fonksiyonu önererek bu alana katkıda bulunmuştur. Mirjalili ve Lewis tarafından önerilen transfer fonksiyonlarının matematiksel modelleri ve şekilleri sırasıyla Tablo 1'de ve Şekil 1'de verilmiştir. Bahsedilen öncü çalışmaların ardından pek çok bilim insanı transfer fonksiyonlarını kullanarak yeni yaklaşımlar önermiştir [4,8–12].

Tablo 1. S-şekilli ve V-şekilli Transfer Fonksiyonları

	S-şekilli		**V-şekilli**
S1:	$\frac{1}{1+e^{-2x}}$	*V1:*	$\left\lvert \operatorname{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
S2:	$\frac{1}{1+e^{-x}}$	*V2:*	$\lvert\tanh(x)\rvert$
S3:	$\frac{1}{1+e^{\frac{-x}{2}}}$	*V3:*	$\left\lvert\frac{x}{\sqrt{1+x^2}}\right\rvert$
S4:	$\frac{1}{1+e^{\frac{-x}{3}}}$	*V4:*	$\left\lvert\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\rvert$

Elimizde gerçel sayılardan oluşan bir sayı dizisi olsun. Bu sayı dizisine Tablo 1'de verilen transfer fonksiyonları uygulandığında sayı dizileri fonksiyonların matematiksel modeline uygun olarak 0-1 aralığında gerçel sayılara dönüştürülür. 0-1 aralığındaki gerçel sayılar S fonksiyonları için Denklem 1, V fonksiyonları için Denklem 2 kullanılarak 0 veya 1'e dönüştürülür.



Şekil 1. *S ve V şekilli transfer fonksiyonları*

$$I_d = \begin{cases} 1 & \text{If } rand < T(x_d)) \\ 0 & \text{If } rand \geq T(x_d) \end{cases} \quad (1)$$

$$If\ rand < T(x_d) \qquad I_d = \begin{cases} 0 & (mod(x_d, 1) > 0.5) \\ 1 & (mod(x_d, 1) \leq 0.5) \end{cases}$$

$$If\ rand > T(x_d) \qquad I_d = \begin{cases} 1 & (mod(x_d, 1) > 0.5) \\ 0 & (mod(x_d, 1) \leq 0.5) \end{cases} \quad (2)$$

Denklem 1 ve Denklem 2'de x_d gerçel sayılardan oluşan sayı dizisinin d. boyutunu, T kullanılan transfer fonksiyonunu, I_d ikili dizinin d. boyutunu, *rand* 0-1 aralığında üretilen rastgele bir gerçel sayıyı ve *mod* matematiksel mod alma işlemini ifade eder.

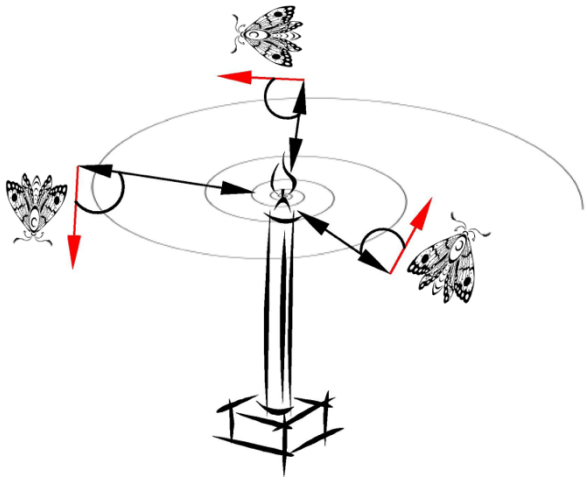
Yukarıda anlatılan yöntem kullanılarak her bir transfer fonksiyonu için farklı bir ikili sayı dizisi oluşturulur. Bir problem için başarılı sonuçlar üreten bir transfer fonksiyonu başka bir problem için başarısız olabilir. Bu sebeple uygun transfer fonksiyonunun belirlenmesi oldukça önemlidir.

Bu çalışmada, Mirjalili tarafından 2015 yılında önerilmiş sürekli bir metasezgisel algoritma olan Güve-Alev Optimizasyonu (GAO) [13] algoritması üzerinde, transfer fonksiyonlarının etkisi analiz edilmiştir. Bunun için Tablo 1'de verilen transfer fonksiyonları orijinal GAO algoritmasına ayrı ayrı uygulanarak 8 farklı ikili GAO algoritması geliştirilmiştir. Geliştirilen algoritmalar OR-Lib kütüphanesinde [14] yer alan 15 farklı KTY problemi üzerinde çalıştırılmış ve elde edilen sonuçlar *gap* olarak isimlendirilen bir hata metriğine göre değerlendirilmiştir.

Çalışmanın diğer kısımları şu şekilde organize edilmiştir: Bölüm 2'de, temel GAO algoritması anlatılmıştır. Bölüm 3'te ikili GAO versiyonlarının nasıl geliştirildiğinden bahsedilmiştir. Bölüm 4'te çalışmada kullanılan problemler, deney ortamı hakkında bilgi ve deney sonuçları yer almaktadır. Bölüm 5'te ise sonuç ve tartışma kısmına yer verilmiştir.

GÜVE-ALEV OPTİMİZASYONU (GAO) ALGORİTMASI

Mirjalili tarafından önerilen GOA [13] algoritması, güvelerin gece uçuş stratejisinden esinlenmiştir. Güveler, ay ışığını sabit bir açıyla kullanan bir uçuş mekanizmasına sahiptir. Yön tayini için kullandıkları uçuş mekanizması stratejisine enine yönlendirme denir. Bu strateji, uzun düz mesafelerde etkili ve rahat bir seyahat sağlar. Öte yandan güveler, tıpkı ay ışığından olduğu gibi yapay ışıklardan da etkilenir ve bu yapay ışıkla açı yaparak benzer davranmaya çalışırlar. Güvelerin ışıkla aralarında sabit bir açı tutarak uçmaları sarmal bir harekete neden olur. Şekil 2, güvelerin ışığın etrafında spiral bir şekilde olan uçuşunu temsilen göstermektedir [13].



Şekil 2. Güvelerin ışık kaynağı etrafındaki sarmal uçuşu

Şekil 2'ye bakıldığında güvelerin spiral uçuş sonunda ışık kaynağına yaklaştıkları görülebilir. GOA, güvelerin ışık kaynağı etrafında sergiledikleri davranışların matematiksel olarak modellenmesi ile geliştirildi. Diğer metasezgiseller gibi, GOA da iteratif ve popülasyon tabanlı bir algoritmadır. Algoritma temelinde güvelerden ve alevlerden oluşur. Popülasyondaki her bir güve olası bir çözümü temsil ederken, güvenin konumunu oluşturan her bir değişken problemin bir boyutunu temsil etmektedir. Daha önce de belirtildiği gibi, GOA popülasyona dayalıdır. *N* ve *D* değişkenlerinin sırasıyla popülasyon büyüklüğü ve problemin boyutu olduğunu düşünürsek, güvelerin oluşturduğu popülasyon aşağıdaki gibi bir matris ile temsil edilebilir:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & \cdots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & \cdots & m_{n,d} \end{bmatrix} \tag{3}$$

M popülasyondaki bireylerin (güve) tutulduğu dizi olmak üzere *OM* popülasyondaki bireylerin konumuna karşılık gelen uygunluk fonksiyonu değerleri dizisini temsil etmektedir.

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{4}$$

Popülasyondaki güvelerin konumlarını iyileştirmek için bir güncelleme sürecine ihtiyaçları vardır. Güncelleme işleminde her güve referans olarak kullanacağı belirli bir aleve ihtiyaç duyar. Güvelerin farklı alevlerden beslenerek konum güncellemesi ile yerel en iyiye takılmasının önüne geçilmesi ve arama uzayında etkin bir arama yapılması hedeflenmiştir. Alevlerin konumu (F), güvelerle aynı boyuta sahip ve benzer şekilde, aşağıdaki gibi temsil edilir:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & \cdots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & \cdots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & m_{n,2} & \cdots & \cdots & F_{n,d} \end{bmatrix} \tag{5}$$

Güvelerde olduğu gibi alevlerin konumuna karşılık gelen uygunluk fonksiyonu değerlerini tutan bir dizi bulunmaktadır. OF notasyonu ile temsil edilen bu dizi aşağıdaki gibi gösterilmektedir.

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{6}$$

Güveler ve alevlerin sunum ve yapı bakımından aynı olduğu unutulmamalıdır. Aralarındaki fark, popülasyon içindeki davranışlarıdır. Alevler mevcut ana kadar bulunan en iyi konumlar dizisi iken; güveler, her iterasyon sonucu değişen konumları temsil etmektedir. Öte yandan, konum güncelleme işlemi sırasında güvelerin konum güncellemesine bir referans noktası olarak bir alev yardım eder [13]. Şekil 2'den esinlenilen konum güncellemesinin matematiksel modeli Denklem 7'de verilmiştir.

$$S(M_i,F_j) = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j \tag{7}$$

$$D_i = |F_j - M_i| \tag{8}$$

Burada $M_i=(m_{i1},m_{i2},\ldots,m_{iD})$ ve $F_j=(f_{j1},f_{j2},\ldots,f_{jD})$ sırasıyla *i*. güve ve *j*. alevin konumlarını göstermektedir. D_i değeri ise Denklem 8 ile hesaplanan ve *i.* güve ile bu güvenin referansı olan *j*. alev arasındaki mesafeyi temsil etmektedir. *t* değeri Denklem 9 ile hesaplanan [-1, 1] aralığında üretilen bir sayı ve *b* logaritmik spiralin şeklini belirleyen sabit bir sayıdır.

$$\begin{aligned} t &= (a-1) * rand + 1 \\ a &= -1 + k * \left(-\frac{1}{K}\right) \end{aligned} \tag{9}$$

Denklem 9'daki *k* değeri mevcut iterasyonu ve *K* değeri ise maksimum iterasyon değerini göstermektedir. Konum güncelleme sürecinin daha başarılı ve efektif olabilmesi için alev sayısı (*as*) her iterasyon adımında azaltılmaktadır. *N* maksimum alev sayısı, ki başlangıçta popülasyon sayısına eşit seçilir, olmak üzere alev sayısındaki azalışın iterasyona bağlı değişimi Denklem 10 ile gösterilmiştir.

$$as = round\left(N - k * \frac{N-1}{K}\right) \tag{10}$$

GAO algoritması, diğer metasezgisel algoritmalara benzer bir işleyiş mekanizmasına sahiptir. Algoritmanın parametreleri ilk adımda ayarlanmalıdır. Daha sonra çözüm uzayının sınırları içinde rastgele bir popülasyon oluşturulur. Popülasyondaki her güve (konum) için uygunluk değerleri hesaplanır ve alevler atanır. Daha sonra algoritmanın ana döngüsü başlatılır. Bu döngüde, pozisyon güncelleme prosedürü her güve için çalışır, alev sayısı güncellenir ve her iterasyon adımı için en iyi konum kaydedilir. Döngü, sonlandırma kriteri karşılanana kadar devam eder [13]. Algoritmanın sözde kodu Şekil 3'te verilmiştir. Algoritma hakkında daha detaylı bilgi için [13,15,16] çalışmaları incelenebilir.

İKİLİ GÜVE-ALEV OPTİMİZASYONU (İGAO) ALGORİTMASI

GAO algoritması sürekli optimizasyon problemlerini çözmek için önerilen bir algoritmadır. Ancak KTYP gibi ikili optimizasyon problemlerinde amaç fonksiyonunu hesaplamak ve problemi ele almak için ikili çözüm yapısına ihtiyaç vardır. Sürekli algoritmaların konum güncelleme stratejisi ikili optimizasyon problemleri için uygun değildir. Bu nedenle, sürekli formdan ikili forma dönüştürmek için özel bir transfer fonksiyonu kullanmak, sorunu çözmek için uygun bir yaklaşımdır. Bir transfer fonksiyonunun temel amacı, sürekli bir çözümün her boyutunu ikili değerlere (0 veya 1) dönüştürmektir. GOA sürekli bir algoritma olduğundan, ikili GOA elde etmek için Tablo 1'de verilen transfer fonksiyonları kullanılmıştır. İkili GOA algoritmasında, güveler konumlarını gerçel sayı biçiminde oluşturur ve konum güncelleme aşaması da sürekli değerler üzerinden gerçekleşir. Ancak, uygunluk fonksiyonunun değerini hesaplamadan önce ikili çözümü üretmek için bir transfer fonksiyonu kullanılır. Bu şekilde algoritma ikili problemlerin yapısına uyumlu hale

getirilerek uygulanır.

```
Alev sayısını güncelle
OM = UygunlukFonksiyonu(M);
if iterasyon == 1
   F = sırala(M);
   OF = sırala(OM);
else
   F = sırala(M_{t-1}, M_t);
   OF = sırala(M_{t-1}, M_t);
end
for i = 1: n      //Her bir güve için
   for j = 1: d // Güvenin her bir boyutu için
      a ve t'yi güncelle
      İlgili güvenin konumuna göre D'yi hesapla
      İlgili güvenin konumuna göre M(i,j)'yi güncelle
   end
end
En iyi güvenin konumunu getir
```

Şekil 3. GAO algoritmasının sözde kodu

DENEY ORTAMI

Çalışmada geliştirilen algoritmalar, 40 birey ve 2000 iterasyon (80,000 maksimum hesaplama sayısı) üzerinden 30 tekrarlı olarak çalıştırılmıştır. Her problem için elde edilen sonuçlar Denklem 11'de verilen *gap* hata metriğine göre değerlendirilmiştir. En az *gap* değerine sahip olan algoritma en başarılı sonucu elde etmiş olur.

$$gap = \frac{ortalama - optimum}{optimum}\ x\ 100 \tag{11}$$

Burada, *ortalama* her bir problemin maliyetinin aynı algoritma ile 30 kez hesaplanmasından sonra elde edilen maliyetlerin ortalamasını, *optimum* her problem için elde edilebilecek en iyi maliyeti ifade eder.

Kapasitesiz Tesis Yerleştirme Problemleri (KTYP)

KTYP'de müşteriler ve tesisler bulunur. Farklı konumlardaki müşterilere en uygun maliyetli hizmeti verebilmek için hangi tesislerin açık hangilerinin kapalı olması gerektiğine karar verilir (En az bir tesis açık olmalıdır). Toplam n tesis olduğunu varsayarsak, tesisler 2^n-1 farklı durumda bulunabilir. Tesis sayısı arttıkça problemin karmaşıklığı arttığından, KTYP NP-Zor problem olarak kabul edilir [17]. F_T tüm tesislerin kümesi ve F_{sub} toplam maliyeti en aza indirgemek için açık olması gereken tesislerin kümesi olmak üzere, bir KTY probleminin amaç fonksiyonu matematiksel olarak Denklem 12'de verilmiştir [18]:

$$Min\ f(x) = \sum_{i \epsilon F_{sub}^1(x)} f_i + \sum_{j \epsilon P} \min\{c_{ij} \mid i \epsilon F_{sub}^1(x) \mid x \in \{0,1\}^q - \{0\}\} \tag{12}$$

Burada f_i açık olan i. tesisin kuruluş maliyetini, c_{ij} i. tesis ve j. müşteri arasındaki hizmetin maliyetini ve P tüm müşterilerin kümesini ifade eder. KTYP'de, olası bir x çözümü ikili vektör ($x \in \{0, 1\}^q$, q toplam tesis sayısı) dizisi ile temsil edilir. Burada i. tesis; $x_i = 1$ ise açık, $x_i = 0$ ise kapalıdır. Denklemdeki $\{0\}$ ifadesi olası çözümde kapalı tesislerin maliyete dâhil edilmeyeceği anlamına gelmektedir. $F_{sub}^1(x)$ aday bir çözümü yani açık tesislerin kümesini göstermektedir.

Çalışmada Kullanılan Problem Seti

Çalışmada kullanılan problemler Tablo 2'de verilmiştir. Problemler, tesis ve müşteri sayısına göre küçük, orta, büyük ve çok büyük olmak üzere 4 ayrı gruba ayrılmaktadır. Örneğin Cap71-74 problemlerinde, 50 müşteriye en az maliyetle hizmet verebilmek için 16 tesisten hangilerinin açık olacağına karar verilmesi gerekmektedir. Problemdeki tesis ve müşteri sayısı arttıkça optimum çözüm değerine ulaşmak daha zor hale gelmektedir.

Tablo 2. *Kapasitesiz Tesis Yerleştirme Problemleri*

Problemin Adı	Problem Tipi	Problemin Boyutu	Problemin Optimum Çözüm Değeri
Cap71	Küçük	16 x 50	932615.750
Cap72		16 x 50	97779.400
Cap73		16 x 50	1010641.450
Cap74		16 x 50	1034976.975
Cap101	Orta	25 x 50	796648.438
Cap102		25 x 50	854704.200
Cap103		25 x 50	893782.113
Cap104		25 x 50	928941.750
Cap131	Büyük	50 x 50	793439.563
Cap132		50 x 50	851495.325
Cap133		50 x 50	893076.713
Cap134		50 x 50	928941.750
CapA	Çok Büyük	100 x 1000	17156454.478
CapB		100 x 1000	12979071.580
CapC		100 x 1000	11505594.330

Deneysel Sonuçlar

Uygulanan transfer fonksiyonlarının isimlerine göre ikili algoritmalar isimlendirilmiştir. Örneğin S1 transfer fonksiyonu kullanılarak ikileştirilen GAO algoritması İGAO-S1 adını almıştır. Her ikili GAO versiyonunun her bir problem için elde ettiği *gap* değerleri Tablo 3'te verilmiştir. Okuyucuların kolayca fark edebilmesi için en iyi değerler koyu olarak vurgulanmıştır. Sonuçlar incelendiğinde, İGAO-S3 algoritmasının 15 problemin 13'ünde en iyi gap değerini elde ederek en başarılı algoritma olduğu görülmüştür. İGAO-S3 algoritmasından sonra İGAO-S2, İGAO-S4, İGAO-S1, İGAO-V4 ve İGAO-V3 algoritmalarının sırasıyla 8, 7, 4 ,4 ve 3 problemde en iyi gap değerlerine sahip olduğu gözlenmiştir. İGAO-S2 algoritmasının çok büyük problem sınıfında yer alan capB ve capC problemlerinde en küçük *gap* değerlerine sahip olması dikkat çekicidir. Küçük boyutlu problemlerde İGAO-V1 ve İGAO-V2 dışındaki algoritmalar genel olarak başarılı olurken, problem boyutu arttıkça İGAO-S2 ve İGAO-S3 algoritmaları başarılarını sürdürebilmiştir. Tablo 4'te ise İGAO-S3 algoritmasının her problem için 30 çalıştırma sonucunda elde ettiği ortalama çözüm değerleri, standart sapma ve hit (optimal çözüme ulaşma) sayıları verilmiştir. Şekil 4'te ise her problem tipinden birer tane seçilerek algoritmaların yakınsama grafikleri verilmiştir.

SONUÇ VE TARTIŞMA (RESULTS AND DISCUSSION)

Bu çalışmada, GAO algoritması 8 farklı transfer fonksiyonu ile düzenlenmiş ve 8 ayrı ikili GAO algoritması geliştirilmiştir. Geliştirilen İGAO versiyonları, 15 farklı kapasitesiz tesis yerleştirme problemi üzerinde çalıştırılmış ve *gap* hata metriğine göre değerlendirilmiştir. Elde edilen sonuçlar incelendiğinde, İGAO-S3 algoritmasının 15 problemin 13'ünde en küçük *gap* değerlerini elde ederek en başarılı ikili algoritmayı oluşturduğu gözlemlenmiştir. İGAO-S3 genel olarak en başarılı algoritma olsa da çok büyük problem sınıfında yer alan capB ve capC problemlerinde İGAO-S2 algoritmasının en başarılı sonuçları elde etmesi ayrıca dikkat çekicidir.

İlerleyen çalışmalarda, bu çalışmada geliştirilen İGAO versiyonları daha farklı ikili problemler üzerinde test edilebilir. Optimizasyon sürecinin başarısına bağlı olarak transfer fonksiyonunu adaptif olarak değiştiren yeni yaklaşımlar geliştirilebilir. Ayrıca transfer fonksiyonları ile ikileştirilen algoritmaların; mantıksal operatörler, çaprazlama teknikleri, Jaccard ve Dice gibi benzerlik metrikleri kullanılarak

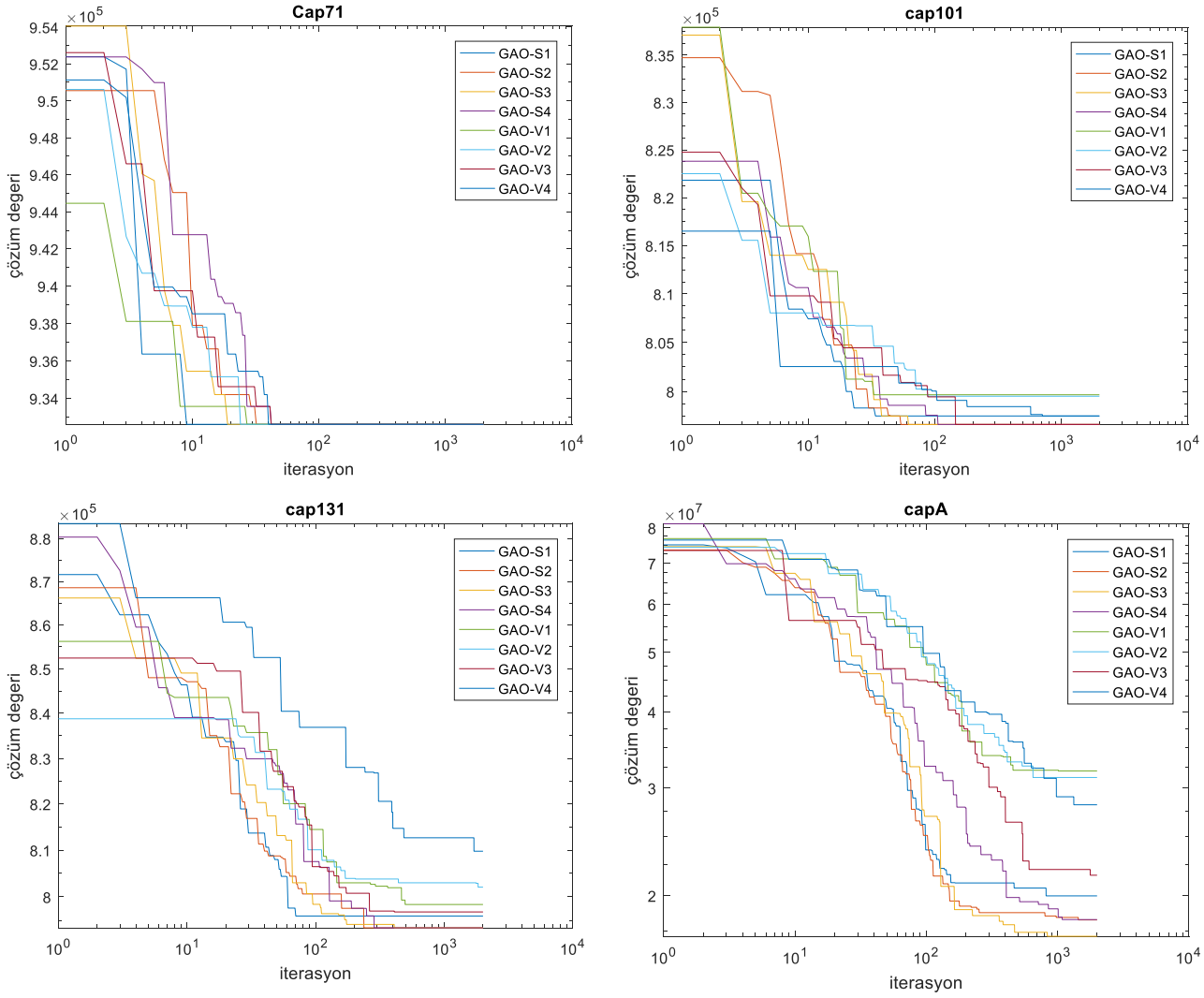
performanslarının iyileştirilmesi sağlanabilir.

Tablo 3. İ*GAO Algoritmalarının Gap Değerleri*

	İGAO-S1	İGAO-S2	İGAO-S3	İGAO-S4	İGAO-V1	İGAO-V2	İGAO-V3	İGAO-V4
Cap71	**0**	**0**	**0**	**0**	0.013627	0.013627	**0**	**0**
Cap72	**0**	**0**	**0**	**0**	0.073387	0.117413	**0**	**0**
Cap73	**0**	**0**	**0**	**0**	0.064631	0.089986	**0**	**0**
Cap74	**0**	**0**	**0**	**0**	0.190343	0.372432	0.026475	**0**
Cap101	0.0036	0.0036	**0**	**0**	0.304095	0.248334	0.080289	0.010799
Cap102	0.0042	**0**	**0**	**0**	0.56523	0.431588	0.146178	0.089585
Cap103	0.013087	0.018154	**0**	0.002529	0.549351	0.503581	0.091563	0.035276
Cap104	0.003893	**0**	**0**	**0**	0.685016	0.7107	0.19885	0.098329
Cap131	0.202464	0.123578	**0.12017046**	0.223196	1.111297	1.534899	1.223978	1.450512
Cap132	0.131689	0.091788	**0.05344299**	0.217825	1.092285	1.289064	1.000876	1.608409
Cap133	0.148517	0.092023	**0.08301139**	0.12737	1.227842	1.333147	1.266733	1.282161
Cap134	0.202033	0.107052	**0.02602482**	0.130148	2.122437	2.235867	1.383747	2.671466
CapA	9.592515	7.109827	**6.99697549**	19.53915	59.72299	53.39312	53.37543	89.19316
CapB	5.22144	**3.845123**	4.1186294	10.28969	20.16379	21.21562	26.33484	35.68414
CapC	5.43781	**4.024399**	4.35089777	8.050159	14.24927	14.82774	17.70531	26.03186
En başarılı olduğu problem sayısı	4	8	**13**	7	0	0	3	4

Tablo 4. İ*GAO-S3 Algoritmasının 30 Çalıştırma Sonucunda Elde Ettiği Ortalama Çözüm Değerleri, Standart Sapma ve Hit Sayıları*

	Problemin Optimum Çözüm Değeri	Ortalama	Std. sapma	Hit
Cap71	932615.750	932615.750	0	30
Cap72	97779.400	97779.400	0	30
Cap73	1010641.450	1010641.450	0	30
Cap74	1034976.975	1034976.975	0	30
Cap101	796648.438	796648.438	0	30
Cap102	854704.200	854704.200	0	30
Cap103	893782.113	893782.113	0	30
Cap104	928941.750	928941.750	0	30
Cap131	793439.563	794393.043	1526.913	16
Cap132	851495.325	851950.389	980.937	17
Cap133	893076.713	893818.068	619.569	7
Cap134	928941.750	929183.505	1129.938	27
CapA	17156454.478	18356887.390	693314.176	1
CapB	12979071.580	13513631.440	213754.521	0
CapC	11505594.330	12006190.980	217140.630	0



Şekil 4. Cap71, cap101, cap131 ve capA problemleri için yakınsama grafikleri

KAYNAKÇA (REFERENCES)

[1] X. Yuan, H. Nie, A. Su, L. Wang, Y. Yuan, An improved binary particle swarm optimization for unit commitment problem, *Expert Systems with Applications*. 36 (2009) 8049-8055. doi:10.1016/j.eswa.2008.10.047.

[2] F. van Beers, A. Lindström, E. Okafor, M. Wiering, Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation:, içinde: *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, *SCITEPRESS - Science and Technology Publications*, Prague, Czech Republic, 2019: ss. 438-445. doi:10.5220/0007347504380445.

[3] A. Banitalebi, M.I.A. Aziz, Z.A. Aziz, A self-adaptive binary differential evolution algorithm for large scale binary optimization problems, *Information Sciences*. 367-368 (2016) 487-511. doi:10.1016/j.ins.2016.05.037.

[4] E. Baş, E. Ülker, A binary social spider algorithm for uncapacitated facility location problem, *Expert Systems with Applications*. 161 (2020) 113618. doi:10.1016/j.eswa.2020.113618.

[5] R. Rizk-Allah, A. Hassanien, M. Elhoseny, G. Manogaran, A new binary salp swarm algorithm: development and application for optimization tasks, *Neural Computing and Applications*. (2018). doi:10.1007/s00521-018-3613-z.

[6] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, içinde: *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, 1997: ss.

4104-4108 c.5. doi:10.1109/ICSMC.1997.637339.

[7] S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization, *Swarm and Evolutionary Computation*. 9 (2013) 1-14. doi:10.1016/j.swevo.2012.09.002.

[8] E. Sonuç, Binary crow search algorithm for the uncapacitated facility location problem, *Neural Computing and Applications*. (2021). doi:10.1007/s00521-021-06107-2.

[9] R. Durgut, Improved binary artificial bee colony algorithm, *Frontiers of Information Technology & Electronic Engineering*. 22 (2021) 1080-1091. doi:10.1631/FITEE.2000239.

[10] R.M. Rizk-Allah, A.E. Hassanien, New binary bat algorithm for solving 0–1 knapsack problem, *Complex & Intelligent Systems*. 4 (2018) 31-53. doi:10.1007/s40747-017-0050-z.

[11] S. Pookpunt, W. Ongsakul, Optimal placement of wind turbines within wind farm using binary particle swarm optimization with time-varying acceleration coefficients, *Renewable Energy*. 55 (2013) 266-276. doi:10.1016/j.renene.2012.12.005.

[12] İ. Babaoğlu, Utilization of Bat Algorithm for Solving Uncapacitated Facility Location Problem, içinde: K. Lavangnananda, S. Phon-Amnuaisuk, W. Engchuan, J.H. Chan (Ed.), *Intelligent and Evolutionary Systems*, *Springer International Publishing*, Cham, 2016: ss. 199-208. doi:10.1007/978-3-319-27000-5_16.

[13] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowledge-Based Systems*. 89 (2015) 228-249. doi:10.1016/j.knosys.2015.07.006.

[14] J.E. Beasley, OR-Library: Distributing Test Problems by Electronic Mail, *Journal of the Operational Research Society*. 41 (1990) 1069-1072. doi:10.1057/jors.1990.166.

[15] H.M. Zawbaa, E. Emary, B. Parv, M. Sharawi, Feature selection approach based on moth-flame optimization algorithm, içinde: *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016: ss. 4612-4617. doi:10.1109/CEC.2016.7744378.

[16] M. Shehab, L. Abualigah, H. Al Hamad, H. Alabool, M. Alshinwan, A.M. Khasawneh, Moth–flame optimization algorithm: variants and applications, *Neural Computing and Applications*. 32 (2020) 9859-9884. doi:10.1007/s00521-019-04570-6.

[17] M. Aslan, M. Gunduz, M.S. Kiran, JayaX: Jaya algorithm with xor operator for binary optimization, *Applied Soft Computing*. 82 (2019) 105576. doi:10.1016/j.asoc.2019.105576.

[18] A simple multi-wave algorithm for the uncapacitated facility location problem, *Frontiers of Engineering Management*. 5 (2018) 451-465. doi:10.15302/J-FEM-2018038.