Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

İkili Bal Porsuğu Algoritmasının Küme Birleşimli Sırt Çantası Problemine Uygulanması

Gülşen ORUCOVA BÜYÜKÖZ [a*], Hüseyin HAKLI [b]

[a] *Matematik ve Bilgisayar Bilimleri Bölümü, Fen Fakültesi, Necmettin Erbakan Üniversitesi, Konya, TÜRKİYE*

[b] *Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Necmettin Erbakan Üniversitesi, Konya, TÜRKİYE*

* *Sorumlu yazarın e-posta adresi: gorucova@erbakan.edu.tr*

DOI: 10.29130/dubited.1205144

ÖZET

NP-zor problem sınıfından olan küme birleşimli sırt çantası (KBSÇ) problemi, 0-1 sırt çantası probleminin (0-1 KP) genelleştirilmiş halidir. Literatürde bu probleme çeşitli sezgisel yaklaşımlar uygulanmasına rağmen, çözüm kalitesinin iyileştirilmesi için çalışmalar devam etmektedir. Son zamanlarda önerilmiş olan Bal Porsuğu Algoritması (Honey Badger Algorithm (HBA)) sürekli problemleri çözmek için tasarlanmıştır. Bu çalışmada ikili yapıya sahip olan küme birleşimli sırt çantası problemine, transfer fonksiyonları yardımıyla ikili yapıya uyarlanan BPA algoritması uygulanmıştır. Transfer fonksiyonları olarak S-şekilli, V-şekilli, U-şekilli, Taper-şekilli fonksiyonlar kullanılmıştır ve elde edilen sonuçlar karşılaştırılmıştır. Çözümlerin iyileştirilmesi için onarım algoritması ve onarım algoritması ile birlikte iyileştirme algoritması kullanılmıştır.

Anahtar Kelimeler: *Küme birleşimli sırt çantası problemi, Bal porsuğu algoritması, Transfer fonksiyonları, İkili optimizasyon*

Implementation of Binary Honey Badger Algorithm to Set Union Knapsack Problem

ABSTRACT

Set Union Knapsack Problem (SUKP), which is one of the NP-hard problem class, is the generalization of the 0-1 Knapsack Problem (0-1 KP). Although various heuristic approaches have been applied to this problem in the literature, the improvement of the solution quality continues. The recently proposed Honey Badger Algorithm (HBA) is designed to solve continuous problems. In this study, HBA algorithm, which is adapted to the binary structure with the help of transfer functions, is applied to the SUKP problem, which has a binary structure. S-shaped, V-shaped, U-shaped, Taper-shaped functions were used as transfer functions and the results obtained were compared. In order to improve the solutions, the repair algorithm and the improvement algorithm together with the repair algorithm were used.

Keywords: *Set union knapsack problem, Honey badger algorithm, Transfer functions, Binary optimization*

Geliş: 18/11/2022, Düzeltme: 08/04/2022, Kabul: 25/07/2023

I. GİRİŞ

Sırt çantası problemi sınırlı kapasiteye sahip bir sırt çantasını, elimizde bulunan n nesnenin alt kümesi ile doldurma problemidir. Burada amaç çantanın kapasitesini aşmadan maksimum fayda sağlanacak şekilde nesneleri seçmektir. Sırt çantası probleminin 0-1 KP, bir boyutlu KP (tek kısıt olduğunda), çok boyutlu KP (birden çok kısıt olduğunda) gibi farklı versiyonları bulunmaktadır [1].

Küme Birleşimli Sırt Çantası (KBSÇ) problemi [2,3] standart 0-1 sırt çantası probleminin [4] genişletilmiş versiyonudur ve NP-zor sınıfındandır. KBSÇ problemleri veri tabanı bölümleme [5], finansal karar verme [1-3], esnek üretim makineleri [2,6], geniş siber sistemlerde [7], veri akışı sıkıştırma [7,8] gibi birçok gerçek dünya probleminde kullanılmaktadır.

Son zamanlarda bazı meta-sezgisel optimizasyon algoritmaları, KBSÇ problemlerini çözmek için kullanılmıştır. Sürekli problemler için sunulan optimizasyon algoritmalarının, ikili yapıya sahip olan KBSÇ problemine uygulanması için öncelikle algoritmanın ikili uzaya adapte edilmesi gerekmektedir. Bu adaptasyon süreci transfer fonksiyonları [9], açı modülasyonu [10], kuantum ilhamlı bitler [11] vb. yollar ile yapılabilmektedir. Transfer fonksiyonları ile algoritmanın genel işleyişine hiç müdahale edilmeden, algoritma ikili arama uzayında kullanılabilmektedir. Yapay arı kolonisi algoritmasının ikili versiyonu (BABC) [12] çalışmasında sunulmuş ve KBSÇ problemlerine uygulanmıştır. Sürekli problemler için geliştirilmiş olan yapay arı kolonisi algoritmasının ikili versiyonu için görüntüsü 0 veya 1 olacak şekilde örten bir fonksiyon tanımlanmıştır. Arama uzayının dışında kalan mümkün olmayan çözümleri ortadan kaldırmak için açgözlü stratejiye sahip onarım algoritması (S-GROA) kullanılmıştır. Ayrık güve arama algoritması, sürekli optimizasyon problemlerini çözmek için tasarlanan güve arama (MS) algoritmasının transfer fonksiyonları ile düzenlenmesiyle elde edilmiştir [9]. Çözüm kalitesini iyileştirmek için onarım algoritmasından faydalanılmıştır. 15 KBSÇ problemi ayrık MS algoritması ile S-şekilli, V-şekilli, O-şekilli 12 transfer fonksiyonu yardımıyla çözüldükten sonra sonuçlar karşılaştırılmıştır. Elde edilen sonuçlardan O4 transfer fonksiyonunun en iyi olduğu kanısına varılmıştır. Sürü zekasına dayalı optimizasyon algoritmaları, problem çözmede dikkate değer fırsatlar sunmaktadır. Genetik algoritma ve parçacık sürü optimizasyonuna dayalı basit ama etkili bir ikili sürü zekâsı tekniği olan gPSO algoritması transfer fonksiyonlarına ihtiyaç duyulmayacak şekilde tasarlanmıştır [13]. Onarım algoritması kullanımı sayesinde mümkün olmayan çözümlerden kaçınılmıştır. KBSÇ problemleri üzerinde test edilen algoritmanın daha önce elde edilen çözümlerden daha etkili olduğu görülmüştür. Hibrit ikili parçacık sürü optimizasyon algoritması tabu arama algoritması ile (HBPSO/TS) geliştirildikten sonra KBSÇ problemi için test edilmiştir [14]. Arama süreci boyunca çözümlerin kalitesini değerlendirmek için ceza fonksiyonu kullanılmıştır. Böylece uygun çözüm uzayının sınırları keşfedilmeye çalışılmıştır. Algoritmada tabu tabanlı mutasyon prosedürü ile aç gözlü arama yapılmaktadır. Sömürü yeteneğini geliştirmek için tabu arama prosedürü tasarlanmıştır. HBPSO/TS, 30 kıyaslama örneğinden oluşan üç sette test edilmiştir. Yinelenen iki fazlı yerel arama algoritması (Iterated two-phase local search) NP-zor problemleri çözmek için sunulmuştur [15]. Algoritmanın birbirini tamamlayan arama ve sömürü aşamaları arama boyunca yinelenmektedir. Bu durum son zamanlarda tanıtılan algoritmalara kıyasla algoritmanın rekabet gücünü artırmaktadır. KBSÇ problemini çözmek için geliştirilmiş güve arama algoritması (EMS), diferansiyel mutasyonu global harmoni aramasına entegre ederek gelişmiş bir etkileşim operatörü (EIO) sunmaktadır [16]. Keşif yeteneğini geliştirmek için diferansiyel evrim algoritması ile jaya algoritmasının birleştirilmesi ile hibrit jaya algoritması (DHJaya) elde edilmiştir [17]. Sömürü yeteneğini geliştirmek için Cauchy mutasyonu bazı bireylerde gerçekleştirilmiştir. Uygun olmayan çözümleri onarmak ve uygun çözümleri optimize etmek için geliştirilmiş bir onarım ve iyileştirme algoritması (MSGROA) önerilmiştir. Tabu arama algoritmasına dayanan çok başlangıçlı çözüm tabanlı algoritma, aday çözümlerin etkili bir şekilde incelenmesi için çoklu başlangıç arama stratejisine sahiptir [18]. Grup teorisine dayalı bir optimizasyon algoritması (GTOA), evrim sürecine cebirsel grup işlemlerini dahil etmektedir [19]. GTOA genetik algoritma, ikili parçacık sürü optimizasyonu, ikili yapay arı kolonisi algoritması ile karşılaştırılmıştır. Orijinal çekirdek tabanlı arama özellikleri ile etkili lokal arama

sürecinin birleştirildiği çekirdek tabanlı tabu arama algoritması KBSÇ problemlerine uygulanmıştır [20].

KBSÇ problemlerinin yapısı 0-1 KP problemlerinden daha karmaşık olduğundan literatürde çok fazla çalışılmadığı görülmüştür. Bu problemlerin çözümünde bazı meta-sezgisel optimizasyon yöntemi kullanılmış olmasına rağmen, son zamanlarda önerilmiş olan bal porsuğu algoritması (BPA) henüz bu probleme uygulanmamıştır. BPA, sürekli optimizasyon problemleri için başarılı bir performans sunar, ancak literatürde BPA'nın dikkate değer ikili versiyonu görülmemektedir [21]. Bu sebeple bu çalışmada sürekli problemler için önerilen BPA'nın transfer fonksiyonu yardımıyla ikili versiyonu önerilmiştir. Çalışmada 16 transfer fonksiyonu kullanılarak hangi fonksiyonun daha etkili çözümler verdiği incelenmiştir. Arama uzayında bulunan mümkün olmayan çözümleri ortadan kaldırmak için onarım algoritması ve onarım algoritması ile birlikte iyileştirme algoritması kullanılmıştır. Önerilen algoritma KBSÇ problemleri ile test edilmiştir. Bu makalenin geri kalanı şu şekilde oluşturulmuştur: Bölüm 2'de KBSÇ ve orijinal BPA tanıtılmıştır. Bölüm 3'te onarım ve iyileştirme algoritmaları, Bölüm 4'te önerilen algoritmanın uygulanması, deney sonuçları ve transfer fonksiyonlarının karşılaştırılması yapılmıştır. Son bölüm, çalışmanın sonucunu ve gelecekte yapılabilir önerilerden oluşmaktadır.

II. MATERYAL VE METOTLAR

A. KÜME BİRLEŞİMLİ SIRT ÇANTASI PROBLEMİ (SET-UNION KNAPSACK PROBLEM (SUKP))

Küme birleşimli sırt çantası (KBSÇ) problemi 0-1 sırt çantası probleminin genelleştirilmiş halidir. Bu problemde nesnelerle ilişkili olan elemanlar kümesi dikkate alınmaktadır.

$U = \{1,2,\dots,n\}$ elemanlar kümesi, $S = \{1,2,\dots,m\}$ nesneler kümesi olmak üzere her $i \epsilon S$ $(i = 1,2,\dots,m)$ nesnesi $p_i > 0$ faydasına sahip $U_i \subseteq U$ alt kümesi karşılık gelmektedir. Her bir $j \in U (j = 1,2,\dots,n)$ elemanının $w_j > 0$ ağırlığı vardır. Boş olmayan keyfi bir $A \subseteq S$ kümesi için A'nın faydası (profit):

$$P(A) = \sum_{i \in A} p_i \tag{1}$$

A'nın ağırlığı

$$W(A) = \sum_{j \in \bigcup_{i \in A} U_i} w_j \tag{2}$$

denklemleri ile tanımlanır. KBSÇ problemlerinde ilişki matrisleri nesnelerle elemanların ilişkisini göstermektedir. Nesnedeki eleman kullanıldıysa matriste değeri 1, kullanılmadıysa 0 olarak verilmektedir.

KBSÇ probleminin amaç fonksiyonu $P(S^*)$ maksimum olacak şekilde $S^* \subseteq S$ alt kümesini aramaktadır. Burada kısıt K sırt çantasının kapasitesi olmak üzere $W(S^*) \leq K$ dır [1,2].

B. BAL PORSUĞU ALGORİTMASI (HONEY BADGER ALGORITHM)

Bal Porsuğu Algoritması (BPA), bal porsuğunun yiyecek arama davranışından esinlenerek matematiksel olarak geliştirilen, sürekli optimizasyon problemlerini çözmek için kullanılan arama stratejisine sahip bir algoritmadır [21]. BPA bal porsuğunun yiyecek arama davranışını taklit eder. Besin kaynağını bulmak için, bal porsuğu ya kokuyu takip eder ve kazar ya da bal kılavuzu kuşunu

takip eder. İlk duruma kazma durumu, ikinci duruma bal durumu denmektedir. Kazma durumunda, avın konumunu tahmin etmek için koku alma yeteneğini kullanır; ulaştığında avı yakalamak için avın etrafında hareket eder. Bal durumunda bal porsuğu, arı kovanını doğrudan bulmak için bal kılavuzu kuşunun rehberini alır.

Bal Porsuğu Algoritması aşamaları aşağıdaki gibidir [21].

$$x_i = lb_i + r_1(ub_i - lb_i) \tag{3}$$

eşitliği ile rassal olarak popülasyonun oluşturulmasıyla algoritma başlar. Burada x_i bal porsuğunun i. konumunu, lb_i ve ub_i arama uzayının sırasıyla alt ve üst sınırlarını göstermektedir. r_1 0 ve 1 arasında rassal olarak üretilen sayıdır. Algoritmadaki önemli notasyonlardan biri I yoğunluk yani avın konsantrasyon gücüdür ve av ile arasındaki mesafe ile ilişkilidir. I_i avın koku yoğunluğudur; koku yüksekse, bal porsuğunun hareketi hızlı olacaktır ve bunun tersi de geçerlidir. Bu koku yoğunluğu ise bal porsuğunun ava olan uzaklığı ile ters orantılıdır. Burada S kaynak gücü veya konsantrasyon gücüdür, d_i av ile bal porsuğu arasındaki uzaklık, r_2 0, 1 arasında rasgele sayıdır.

$$I_i = r_2 \frac{S}{4\pi d_i^2} \tag{4}$$

$$S = (x_i - x_{i+1})^2 \tag{5}$$

$$d_i = x_{prey} - x_i \tag{6}$$

Yoğunluk faktörü α, aramadan sömürüye geçişin pürüzsüz olmasını sağlamak için zaman değişim rassallığını kontrol eder. Azalan α faktörü, rassal zamanda azalan iterasyonlarla Eşt. 7'de olduğu gibi güncellenir. Burada $C \geq 1$ sabittir (varsayılan değeri 2'dir).

$$\alpha = C.exp\left(\frac{-t}{t_{max}}\right) \tag{7}$$

Lokal optimumdan kaçınmak için aramanın yönünü değiştiren F bayrağı kullanır. Eşt. 9'da F bayrağının özelliği verilmiştir.

Algoritmanın temel aşamaları kazı ve bal aşamasıdır. Kazı aşamasında bal porsuğu Cardioid şeklinde yol çizer bu aşamanın matematiksel gösterimi Eşt. 8'de verilmiştir.

$$x_{new} = x_{prey} + F.\beta.I.x_{prey} + F.r_3.\alpha.d_i.|cos(2\pi r_4).[1 - cos(2\pi r_5)]| \tag{8}$$

Burada x_{prey} avın pozisyonu, x_{new} bal porsuğunun yeni pozisyonu, $\beta \geq 1$ (varsayılan 6) bal porsuğunun bala ulaşma becerisi, d_i bal ile arasındaki uzaklık, r_3, r_4, r_5 ise 0,1 arasında birbirinden farklı rassal sayılardır.

$$F = \begin{cases} 1, & r_6 \leq 0.5 \\ -1, & \text{diğer} \end{cases} \tag{9}$$

Burada r_6 0,1 arasında rassal sayıdır.

Bal aşamasında bal porsuğu kılavuz kuşunun takip ederek bala ulaşır. Bu olay aşağıdaki denklemle gösterilmiştir.

$$x_{new} = x_{prey} + F.r_7.\alpha.d_i \tag{10}$$

Burada r_7 0,1 arasında rassal sayıdır. Algoritma hakkında daha detaylı bilgi için [21] çalışması incelenebilir.

III. BAL PORSUĞU ALGORİTMASININ KBSÇ PROBLEMİNE UYGULANMASI

Bal porsuğu algoritması sürekli problemlere uygulanmak üzere modellenmiştir. Literatürde bu algoritmanın kayda değer ikili versiyonuna rastlanmamıştır. Bu sebeple ikili yapıya sahip olan KBSÇ problemini çözmek için transfer fonksiyonları yardımıyla BPA'nın ikili versiyonu önerilmiştir. Çalışmada farklı dönüşümlere sahip S-şekilli [22], V-şekilli [22], U-şekilli [23], Taper-şekilli [24] transfer fonksiyonları kullanılmıştır.

A. ONARIM ALGORİTMASI (REPAIR ALGORITHM)

KBSÇ problemi kısıtlı optimizasyon problemi olduğu için bu probleme BPA'nın ikili versiyonu uygulandığında kullanılması mümkün olmayan (infeasible) çözümler de elde edilmektedir. Bu çözümleri onarmak, arama uzayına dahil etmek için onarım algoritması kullanılmıştır. Açgözlü stratejiye [3] göre d_j $(j = 1,2, \dots, n)$, $U_1, U_2, \dots, U_m$ alt kümelerindeki $j \in U$ elemanının frekansı olsun. $R_i = \sum_{j \in U_i} \frac{w_j}{d_j}$ $(i = 1,2, \dots, m)$ olmak üzere $\frac{p_i}{R_i}$, i. nesnenin yoğunluk değeri olarak gösterilsin. S kümesindeki bütün nesneler için yoğunluk değerine göre azalan olarak sıralandığında nesnelerin indisleri $H[1,2, \dots, m]$ dizisinde tutulsun. m boyutlu 0-1'lerden oluşan bir $Y = [y_1, y_2, \dots, y_m] \in \{0,1\}^m$ aday çözümü için nesnelerin indis numaralarından oluşan $A_Y = \{i | y_i \in Y \; ve \; y_i = 1, \leq i \leq m\}$ gösterimi kullanılsın. Aç gözlü stratejiye dayanarak onarım algoritmasında yoğunluk değeri en büyük olandan küçük olan nesneye kadar boş sırt çantasına nesneler kapasiteyi aşmayacak şekilde eklenmektedir. Böylece yoğunluğu daha düşük olan nesneler çantadan çıkarılmıştır. Algoritmanın adımları aşağıdaki gibidir [12]:

Girdi: KBSÇ problemi aday çözümü $\boldsymbol{Y = [y_1, y_2, \dots, y_m] \in \{0,1\}^m}$ ve $\mathbf{H[1,2,\dots,m]}$ dizisi

Çıktı: Mümkün çözüm $\mathbf{Y = [y_1, y_2, \dots, y_m] \in \{0,1\}^m}$ ve mümkün çözümün $\boldsymbol{f(Y)}$ uygunluk değeri

---

if $(\boldsymbol{W(A_Y) > C})$

  $\boldsymbol{Z \leftarrow [0,0,\dots,0]}$ $\%\boldsymbol{Z\ m - boyutlu}$

  for $\boldsymbol{i \leftarrow 1}$ to $\boldsymbol{m}$ do

    If $\boldsymbol{\left(Y_{H[i]} = 1 \text{ and } W(A_Z \cup \{H[i]\}) \leq C\right) \text{ then } Z_{H[i]} \leftarrow 1 \; and \; A_Z \leftarrow A_Z \cup \{H[i]\}}$.

  end for

$\boldsymbol{Y \leftarrow Z}$

return $(\boldsymbol{Y, f(Y)})$.

Algoritma 1. *Onarım algoritmasının kaba kodu*

B. İYİLEŞTİRME ALGORİTMASI (IMPROVEMENT ALGORITHM)

Aday çözümdeki seçili nesneler çantaya eklendiğinde kapasite aşılmıyorsa, çantaya daha fazla fayda sağlanacak şekilde nesne eklenebilme stratejisi iyileştirme algoritması olarak adlandırılır. Eklenen nesnedeki elemanlar daha önce eklenmiş nesnenin içinde var ise bu elemanların ağırlıkları dikkate alınmadığı için iyileştirme algoritması ile ağırlık artırılmadan fayda artırılabilmektedir [17]. Onarım algoritmasında kapasiteyi aşma durumunda çantadan bazı nesneler çıkarılarak maksimum verimlilik sağlanmaya çalışıldı. Bu çalışmada onarım algoritmasından gelen aday çözüme de iyileştirme algoritması uygulandı. İyileştirme algoritmasının kaba kodu aşağıdaki gibidir [17]:

Girdi: KBSÇ problemi aday çözümü $\boldsymbol{Y} = [\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_m}] \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{m}}$ ve $\mathbf{H}[\mathbf{1}, \mathbf{2}, \dots, \mathbf{m}]$ dizisi
Çıktı: Mümkün çözüm $\mathbf{Y} = [\mathbf{y_1}, \mathbf{y_2}, \dots, \mathbf{y_m}] \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{m}}$ ve mümkün çözümün $\boldsymbol{f}(\boldsymbol{Y})$ uygunluk değeri

1 if $(\mathbf{W}(\mathbf{A_Y}) < \mathbf{C})$ then go to 7
2 $\boldsymbol{Z} \leftarrow [\mathbf{0}, \mathbf{0}, \dots, \mathbf{0}]$ $\%\boldsymbol{Z}\ \boldsymbol{m} - \boldsymbol{boyutlu}$
3 for $\boldsymbol{i} \leftarrow \mathbf{1}$ to $\boldsymbol{m}$ do % onarım aşaması
4 If $\left(\boldsymbol{Y_{H[i]}} = \mathbf{1}\ \mathbf{and}\ \boldsymbol{W}(\boldsymbol{A_Z} \cup \{\boldsymbol{H}[\boldsymbol{i}]\}) \leq \boldsymbol{C}\right)$ $\mathbf{then}\ \boldsymbol{Z_{H[i]}} \leftarrow \mathbf{1}\ \boldsymbol{and}\ \boldsymbol{A_Z} \leftarrow \boldsymbol{A_Z} \cup \{\boldsymbol{H}[\boldsymbol{i}]\}$.
5 end for
6 $\boldsymbol{Y} \leftarrow \boldsymbol{Z}$
7 for $\boldsymbol{i} \leftarrow \mathbf{1}$ to $\boldsymbol{m}$ do % iyileştirme aşaması
8 If $\left(\boldsymbol{Y_{H[i]}} = \mathbf{0}\ \mathbf{and}\ \boldsymbol{W}(\boldsymbol{A_Z} \cup \{\boldsymbol{H}[\boldsymbol{i}]\}) \leq \boldsymbol{C}\right)$ $\mathbf{then}\ \boldsymbol{Z_{H[i]}} \leftarrow \mathbf{1}\ \boldsymbol{and}\ \boldsymbol{A_Z} \leftarrow \boldsymbol{A_Z} \cup \{\boldsymbol{H}[\boldsymbol{i}]\}$.
9 end for
return $(\boldsymbol{Y}, \boldsymbol{f}(\boldsymbol{Y}))$.

Algoritma 2. *Onarım ve iyileştirme algoritmasının kaba kodu*

Çalışmada kullanılan problemlerden biri üzerinde onarım ve iyileştirme aşamaları Şekil 1'de gösterilmiştir. SUKP_85_100_0.10_0.75 problemi dikkate alınırsa, bu problemin ismindeki ilk sayı problemdeki nesnelerin toplam sayısını, ikinci sayı ise her bir nesnenin içindeki toplam eleman sayısını göstermektedir. Dolayısıyla bu problemde 85 nesne, kapasitesi 12180 olan bir çantaya yerleştirilecektir. Eşt. 3 ile rasgele oluşturulan bir aday çözüm reel değerlerden oluşan 1x85 boyutunda bir vektördür. Bu reel değerler transfer fonksiyonu yardımıyla ikili yapıya dönüştürüldükten sonra bazı bileşenleri Şekil 1 (a)' da verilmiştir. Burada 1. satır aday çözümün bileşenlerinin indis numaralarını göstermektedir. 2. satır ise aday çözümdeki nesnelerin seçilip seçilmediği gösterilmektedir. Bu aday çözümün ağırlıkları toplamı hesaplandığında 15970 olduğu görülür bu ise 12180 çanta kapasitesini aşmaktadır, bu sebeple çözüm onarım işlemine uğrar. Onarım işlemi sonrası yeni aday çözüm Şekil 1 (b)'de verilmiştir. Kapasite aşıldığı için çantadan bazı nesneler çıkarılmıştır. Elde edilen aday çözümün ağırlığı 12179'dur ve çanta kapasitesini aşmamaktadır. Kapasiteyi aşmadığı için bu çözüm iyileştirilebilir, o yüzden iyileştirme işlemine uğrar. İyileştirme işlemi sonrası elde edilen çözüm Şekil 1 (c)'deki gibidir. Bu çözümün ağırlığı 12179'dur kapasiteyi aşmamaktadır. Ancak dikkat edilirse 41, 64, 82 ve 84. nesne çantaya eklenmiştir. Dolayısıyla iyileştirme algoritması sayesinde kapasite aşmadan daha fazla fayda sağlanmıştır.

1	…	12	…	14	…	19	…	32	…	41	42	43	…	64	…	70	…	79	…	82	83	84	85
0	…	1	…	1	…	1	…	1	…	0	0	1	…	0	…	1	…	1	…	0	1	0	0

(a)

Onarım işlemi

1	…	12	…	14	…	19	…	32	…	41	42	43	…	64	…	70	…	79	…	82	83	84	85
0	…	0	…	0	…	0	…	0	…	0	0	0	…	0	…	0	…	0	…	0	1	0	0

(b)

İyileştirme işlemi

1	…	12	…	14	…	19	…	32	…	41	42	43	…	64	…	70	…	79	…	82	83	84	85
0	…	0	…	0	…	0	…	0	…	1	0	0	…	1	…	0	…	0	…	1	1	1	0

(c)

Şekil 1. (a) *ikili aday çözüm,* ***(b)*** *onarım algoritması sonrası aday çözüm,* ***(c)*** *iyileştirme algoritması sonrası aday çözüm.*

IV. DENEYSEL ÇALIŞMA

Bu çalışmada KBSÇ problemini çözmek için S-şekilli, V-şekilli, U-şekilli, Taper-şekilli transfer fonksiyonları yardımıyla BPA ikili yapıya uyarlanmıştır. KBSÇ problemindeki nesne sayısı boyutunda N adet reel değerli aday çözüm oluşturulduktan sonra transfer fonksiyonları yardımıyla ikili değere dönüştürülmüştür. Daha sonra uygunluk değerleri hesaplanarak onarım veya iyileştirme algoritmaları yardımıyla çözümler iyileştirilmiştir. Bu işlemler Tablo 1'de verilen 6 adet KBSÇ problemine uygulanarak elde edilen sonuçlar karşılaştırılmıştır. Literatürde KBSÇ problemleri 100 tekrar, popülasyon boyutu 20, iterasyon sayısı ise KBSÇ probleminin içeriğinde bulunan nesne sayı m, eleman sayısı n olmak üzere $\max(m, n)$ şeklinde belirlenmiştir.

KBSÇ probleminin verileri http://www.info.univ-angers.fr/pub/hao/SUKP_KBTS.html adresinden alınmıştır.

Çalışmada kullanılan KBSÇ problemlerinin listesi Tablo 1'de verilmiştir.

Tablo 1. *Kullanılan problem isimleri*

PROBLEM NO	PROBLEM ADI
P1	SUKP_85_100_0.10_0.75
P2	SUKP_85_100_0.15_0.85
P3	SUKP_100_85_0.10_0.75
P4	SUKP_100_85_0.15_0.85
P5	SUKP_100_100_0.10_0.75
P6	SUKP_100_100_0.15_0.85

BPA ile elde edilen çözümler içinde çanta kapasitesini aşan mümkün olmayan çözümler, onarım algoritmasına gönderilerek mümkün çözümler haline getirildi. Bu işlemler 100 tekrarlı olarak her bir transfer fonksiyonu için hesaplandıktan sonra minimum, maksimum, ortalama değer ve standart sapmalarından oluşan Tablo 2 elde edildi. Tablo 2'den kullanılan transfer fonksiyonları içinde Taper-şekilli transfer fonksiyonu kullanımının daha iyi sonuçlar verdiği görülmektedir.

BPA ile elde edilen çözümler içinde çanta kapasitesini aşan mümkün olmayan çözümler, onarım algoritmasına gönderilerek mümkün çözümler haline getirildikten sonra iyileştirme algoritması ile çantaya daha fazla nesne eklenmeye çalışıldı. Çünkü problemin yapısı gereği çantanın içerdiği nesnelerin içindeki elemanlar birden fazla kez çantada bulunuyorsa bu elemanların ağırlıkları sadece bir kez dikkate alınmaktadır. Bunun yanı sıra eğer aday çözümdeki nesneler çanta kapasitesini aşmıyorsa yine iyileştirme algoritması ile fayda artırılmaya çalışıldı. Bu işlemler 100 tekrarlı olarak her bir transfer fonksiyonu için hesaplandıktan sonra minimum, maksimum, ortalama değer ve standart sapmalarından oluşan Tablo 3 elde edildi. Tablo 3'ten 6 problemin 4'ünde kullanılan transfer fonksiyonları içinde Taper-şekilli transfer fonksiyonu kullanımının daha iyi sonuçlar verdiği görülmektedir. Diğer 2 problem için ise en iyi çözüm S-şekilli transfer fonksiyonu kullanılarak elde edilmiştir. Çözülen 6 problemin sonuçlarından U-şekilli transfer fonksiyonunun V-şekilli transfer fonksiyonundan daha etkili olduğu gözlemlenmiştir.

Tablo 2. *BPA ve onarım algoritması ile elde edilen sonuçlar*

Problem	Sonuçlar	S1	S2	S3	S4	V1	V2	V3	V4	U1	U2	U3	U4	T1	T2	T3	T4
	En iyi	10950	10970	11026	10941	11215	11306	11215	11115	11215	11215	11240	11283	11484	11283	11456	11215
	Ort	10156.15	10079.69	9941.61	9836.12	10619.33	10562.09	10568.81	10537.42	10639.38	10725.97	10718.02	10756.94	10783.53	**10866.36**	10767.2	10702.34
P1	**En kötü**	8737	9216	9220	8943	9660	9503	9710	9825	9545	9408	9511	9856	9878	10307	9990	9744
	StD	3.99E+02	3.58E+02	3.80E+02	3.40E+02	3.17E+02	3.59E+02	3.52E+02	3.40E+02	3.30E+02	3.33E+02	3.25E+02	3.22E+02	3.33E+02	2.32E+02	3.01E+02	3.45E+02
	En iyi	11689	11645	10906	10561	11235	12025	11224	11113	11804	11531	11513	11504	11595	12217	12091	11844
	Ort	9760.05	9823.18	9586.46	9416.64	10098.86	10203.18	10106.27	10017.05	10342.42	10394.1	10421.3	10443.92	10677.25	**10743.24**	10602.52	10505.71
P2	**En kötü**	8082	8755	8608	8540	8436	9117	8348	9015	9111	8810	8993	8841	9604	9493	9210	9175
	StD	6.43E+02	5.43E+02	4.54E+02	4.49E+02	5.18E+02	4.83E+02	5.24E+02	4.44E+02	5.61E+02	5.16E+02	5.66E+02	5.34E+02	4.30E+02	4.76E+02	5.48E+02	5.15E+02
	En iyi	12703	12603	12286	12163	13003	13044	13041	13044	13044	13044	13044	13044	13044	13044	13044	13044
	Ort	11418.5	11398.48	11178.99	10921.52	12411.08	12359.65	12409.52	12340.84	12420.31	12430.67	12401.02	12467.19	12522.22	**12564.35**	12486.69	12447.08
P3	**En kötü**	9062	10380	10110	9974	11878	11465	11813	11501	11629	11850	11838	11874	11899	11992	11693	11702
	StD	5.88E+02	4.57E+02	4.81E+02	4.58E+02	2.81E+02	2.96E+02	2.88E+02	2.94E+02	2.91E+02	2.86E+02	3.11E+02	3.23E+02	3.22E+02	3.08E+02	3.16E+02	3.18E+02
	En iyi	11484	11660	11009	11278	11878	11985	11985	11842	12077	12074	12074	12049	12065	12130	12130	12065
	Ort	10454.1	10496.6	10117.98	10034.26	11071.47	11107.9	11124.04	11029.31	11260.6	11199.12	11236.51	11184.21	11319.94	**11408.58**	11360.08	11226.46
P4	**En kötü**	9105	9397	9157	8804	9467	9793	10042	10007	10266	9475	10203	10593	10757	10773	10092	10007
	StD	5.84E+02	4.78E+02	4.79E+02	4.54E+02	3.98E+02	3.46E+02	3.34E+02	3.76E+02	3.94E+02	3.74E+02	3.37E+02	3.14E+02	3.28E+02	2.95E+02	4.24E+02	4.49E+02
	En iyi	13368	13122	13333	12670	13629	13701	13721	13648	13748	13814	13674	13748	13748	13748	13849	13814
	Ort	12202.14	12165.04	11913.49	11602.51	13026.73	13035.27	13017.76	12966.06	13137.76	13164.66	13137.39	13203.03	13283.82	**13347.8**	13151.55	13091.82
P5	**En kötü**	10854	11028	11031	10439	12007	11767	11938	11989	12195	12098	12218	11936	12489	12660	12261	12218
	StD	5.34E+02	4.43E+02	4.94E+02	4.39E+02	3.28E+02	4.02E+02	3.54E+02	3.34E+02	3.44E+02	3.31E+02	3.39E+02	3.02E+02	2.95E+02	2.70E+02	3.69E+02	3.28E+02
	En iyi	12892	12365	11978	11550	13145	13145	13145	13044	13145	13407	13407	13407	13407	13407	13161	13407
	Ort	11001.53	10954.34	10599.87	10402.73	11807.95	11945.04	11887.54	11699.62	11864.5	12064.01	11998.83	12029.05	12291.9	**12430.43**	12196.47	12023.94
P6	**En kötü**	9146	9466	9336	9504	10371	10590	10586	10061	10589	10441	10287	10384	11118	11490	10872	10545
	StD	7.87E+02	5.59E+02	5.75E+02	4.53E+02	5.44E+02	6.04E+02	5.87E+02	6.08E+02	5.60E+02	5.85E+02	6.17E+02	6.57E+02	6.07E+02	5.68E+02	6.10E+02	6.29E+02

Tablo 3. *BPA, onarım ve iyileştirme algoritması ile elde edilen sonuçlar*

Problem	Sonuçlar	S1	S2	S3	S4	V1	V2	V3	V4	U1	U2	U3	U4	T1	T2	T3	T4
	En iyi	11604	11484	11947	11421	11752	12020	11752	11350	11922	11713	12020	12020	12045	11912	11922	11752
	Ort	11000.8	10875.21	10970.22	10939.64	10925.09	10981.92	10884.83	10832.49	11110.41	11015.52	11118	11069.99	**11137.09**	11107.5	**11137.41**	11031.2
P1	**En kötü**	10389	10217	10396	10458	10213	10350	10244	10204	10231	10348	10513	10231	10059	10627	9877	9877
	StD	2.26E+02	2.61E+02	2.47E+02	1.93E+02	3.31E+02	3.11E+02	3.00E+02	2.54E+02	3.53E+02	3.11E+02	3.19E+02	3.40E+02	3.00E+02	2.16E+02	3.20E+02	3.51E+02
	En iyi	12369	12369	12369	12369	12369	12369	12299	12224	12369	12369	12369	12369	12369	12369	12369	12369
	Ort	11490.52	11483.43	11459.96	11448.79	11426.8	11422.63	11451.26	11389.52	11626.18	11430.34	11560.73	11566.32	**11729.32**	11624.59	**11723.66**	11612.13
P2	**En kötü**	10059	10352	10347	10590	10483	10426	10483	10483	10050	10000	10603	10483	10314	10641	10603	10374
	StD	5.14E+02	4.61E+02	4.00E+02	3.72E+02	4.68E+02	4.63E+02	4.75E+02	3.86E+02	4.88E+02	5.30E+02	4.54E+02	4.87E+02	4.77E+02	4.75E+02	4.47E+02	5.06E+02
	En iyi	13283	13167	13167	13082	13089	12888	13003	12943	12910	13089	12935	13167	13167	13044	13020	13167
	Ort	12712.77	12772.72	**12811.62**	12750.15	12534.22	12544.09	12589.3	12588.13	12537.48	12584.89	12556.33	12560.95	12649.12	12779.27	12551.35	12559.71
P3	**En kötü**	11982	11867	12314	12199	12253	12256	12296	12268	12296	12276	12296	12256	12296	12296	12253	12296
	StD	2.87E+02	2.42E+02	1.83E+02	2.13E+02	2.00E+02	1.84E+02	2.09E+02	1.97E+02	2.07E+02	2.05E+02	2.13E+02	2.22E+02	2.72E+02	2.49E+02	2.18E+02	1.97E+02
	En iyi	12272	12348	12348	12479	12130	12077	12153	12238	12273	12272	12274	12272	12348	12479	12238	12235
	Ort	11907.38	11915.72	11972.24	**12003.85**	11431.91	11491.63	11494.58	11590.08	11477.2	11532.27	11471.02	11436.2	11873.47	11919.71	11503.03	11491.14
P4	**En kötü**	11272	11327	11346	11298	10994	11119	11119	11119	10962	11100	10797	10781	11119	11024	11041	10797
	StD	2.29E+02	2.35E+02	2.01E+02	2.09E+02	3.64E+02	3.66E+02	3.49E+02	3.47E+02	3.73E+02	3.82E+02	3.95E+02	3.72E+02	3.64E+02	2.67E+02	4.08E+02	3.93E+02
	En iyi	13860	13937	13860	13860	14044	14044	14044	13673	14044	14044	13963	13950	14044	13860	14044	14044
	Ort	13443.58	13505.29	13495.2	13479.37	13551.65	13547.98	13527.86	13515.78	13567.52	13551.24	13547.49	13546.56	13570.47	**13604.24**	13515.84	13564.6
P5	**En kötü**	12710	12626	12944	12837	12953	13077	13077	13169	12827	12827	12827	12863	13018	12877	12827	12874
	StD	2.62E+02	2.42E+02	2.02E+02	2.30E+02	1.85E+02	1.93E+02	1.96E+02	1.57E+02	2.01E+02	2.22E+02	1.94E+02	2.04E+02	1.82E+02	1.47E+02	2.26E+02	2.17E+02
	En iyi	13508	13508	13498	13407	13508	13508	13508	13407	13508	13508	13508	13508	13508	13508	13508	13508
	Ort	12714.09	12700.96	12852.26	12888.16	12768.12	12750.98	12705.48	12565.48	12805.76	12748.81	12849.76	12891.46	12854.94	**13027.41**	12761.25	12767.44
P6	**En kötü**	11006	11201	11970	11834	11635	11586	11812	11689	11939	11219	11217	11648	11970	11814	11552	11271
	StD	5.17E+02	5.25E+02	4.04E+02	4.44E+02	4.56E+02	4.30E+02	4.00E+02	3.88E+02	4.00E+02	4.06E+02	4.46E+02	3.95E+02	3.89E+02	3.89E+02	4.52E+02	4.92E+02

BPA onarım ve iyileştirme algoritmaları ile elde edilen sonuçları, literatürde bulunan [12] A-SUKP, ABC-Bin ve GA algoritmaları ile elde edilen sonuçlarla kıyaslamak amacıyla Tablo 4 oluşturulmuştur. Bu tabloda çözülen 6 problem için ortalama değerler Friedman testine tabi tutulmuştur. Elde edilen sıralamada T1 ve T2 transfer fonksiyonu kullanımının BPA'nın performansını artırdığı, A-SUKP, ABC-Bin ve GA algoritmalarının önüne geçerek 1 ve 2. sıraya taşıdığı görülmüştür.

Tablo 4. *BPA algoritması ile A-SUKP, ABC-Bin, GA algoritmalarının karşılaştırılması*

					BPA Onarım ve İyileştirme Algoritması			
Problem	**Sonuçlar**	**A-SUKP**	**GA**	**ABCbin**	**T1**	**T2**	**T3**	**T4**
P1	**En iyi**	10231	11454	11206	12045	11912	11922	11752
	Ort	10231	11092.7	10879.5	**11137.09**	11107.5	**11137.41**	11031.2
	En kötü				10059	10627	9877	9877
	StD	0.00	171.22	163.62	3.00E+02	2.16E+02	3.20E+02	3.51E+02
P2	**En iyi**	10483	12124	12006	12369	12369	12369	12369
	Ort	10483	11326.3	11485.3	**11729.32**	11624.59	11723.66	11612.13
	En kötü				10314	10641	10603	10374
	StD	0.00	417.00	248.33	4.77E+02	4.75E+02	4.47E+02	5.06E+02

P3	**En iyi**	12459	13044	13044	13167	13044	13020	13167
	Ort	12459	**12956.4**	12818.5	12649.12	12779.27	12551.35	12559.71
	En kötü				12296	12296	12253	12296
	StD	0.00	130.66	153.06	2.72E+02	2.49E+02	2.18E+02	1.97E+02
P4	**En iyi**	11119	12066	12238	12348	12479	12238	12235
	Ort	11119	11546	**12049.3**	11873.47	11919.71	11503.03	11491.14
	En kötü				11119	11024	11041	10797
	StD	0.00	214.94	96.11	3.64E+02	2.67E+02	4.08E+02	3.93E+02
P5	**En iyi**	13634	14044	13860	14044	13860	14044	14044
	Ort	13634	**13806**	13547.2	13570.47	13604.24	13515.84	13564.6
	En kötü				13018	12877	12827	12874
	StD	0.00	144.91	119.11	1.82E+02	1.47E+02	2.26E+02	2.17E+02
P6	**En iyi**	11325	13145	13498	13508	13508	13508	13508
	Ort	11325	12234.8	**13103.1**	12854.94	13027.41	12761.25	12767.44
	En kötü				11970	11814	11552	11271
	StD	0.00	388.66	343.46	3.89E+02	3.89E+02	4.52E+02	4.92E+02
Friedman değeri		6.17E+00	3.67E+00	3.50E+00	2.83E+00	2.67E+00	4.33E+00	4.83E+00
Son sıralama		**7**	**4**	**3**	**2**	**1**	**5**	**6**

IV. SONUÇ VE ÖNERİLER

Bu çalışmada, BPA algoritması transfer fonksiyonları yardımıyla ikili bir optimizasyon problemi olan KBSÇ problemine uygulandı. Bu çözümlerin ortalaması, minimum, maksimum ve standart sapması hesaplanarak değerler tablolaştırıldı. Elde edilen sonuçlara göre Tablo 2'den BPA ve onarım algoritması birlikte kullanıldığında 4 tip transfer fonksiyonu kümesi içinde T2-şekilli transfer fonksiyon yardımıyla bulunan değerlerin daha iyi olduğu görüldü. Tablo 2 ve Tablo 3 karşılaştırılırsa hem onarım hem iyileştirme algoritmasının BPA ile kullanılması sonuçları önemli ölçüde iyileştirdiği görülmektedir. Ayrıca Tablo 3'den 6 problemin 4'ünde yine Taper-şekilli transfer fonksiyonların en iyi çözümleri verdiği söylenebilir. Tablo 4' den daha önce KBSÇ problemini çözmede kullanılan A-SUKP, ABC-Bin, GA algoritmaları ile BPA'nın karşılaştırılmasında BPA'nın dikkate değer ölçüde sıralamada ilk sıralarda bulunduğu ve karşılaştırılabilir seviyede olduğu gözlemlendi.

BPA'nın arama stratejisindeki başarısı ile onarım algoritmasının mümkün olmayan çözümleri mümkün çözüme dönüştürmesi, iyileştirme algoritmasının var olan çözümün kalitesini artırması özellikleri birleştirildiğinde karmaşık olan KBSÇ problemlerini çözen kayda değer ikili algoritma elde edilmiş oldu.

Farklı transfer fonksiyonları kullanılarak BPA algoritması KBSÇ problemi için test edilebilir ve Taper-şekilli transfer fonksiyonundan daha iyi transfer fonksiyonunun olup olmadığı araştırılabilir. Ayrıca farklı optimizasyon algoritmaları BPA ile birleştirilerek BPA'dan daha etkili yeni hibrit yapılar elde edilebilir.

V. KAYNAKLAR

[1] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," *Knapsack Problems*, pp. 235–283, 2004.

[2] O. Goldschmidt, D. Nehme, and G. Yu, "Note: on the set-union knapsack problem," Naval Research Logistics (NRL), vol. 41, no. 6, pp. 833–842, 1994.

[3] A. Arulselvan, "A note on the set union knapsack problem," Discrete Appl Math, vol. 169, pp. 214–218, 2014.

[4] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for knapsack problems," Appl Math Comput, vol. 218, no. 22, pp. 11042–11061, Jul. 2012.

[5] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou, Vertical Partitioning Algorithms for Database Design, pp. 680–710, 1984.

[6] C. S. Tang and E. v. Denardo, "Models arising from a flexible manufacturing machine, part II: minimization of the number of switching instants," Operations Research, vol. 36, no. 5, pp. 778–784, Oct. 1988.

[7] M. Tu and L. Xiao, "System resilience enhancement through modularization for large scale cyber systems," in *2016 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2016, pp. 1-6.

[8] X. Yang, A. Vernitski, and L. Carrea, "An approximate dynamic programming approach for improving accuracy of lossy data compression by bloom filters," Eur J Oper Res, vol. 252, no. 3, pp. 985–994, 2016.

[9] Y. Feng, H. An, and X. Gao, "The importance of transfer function in solving set-union knapsack problem based on discrete moth search algorithm," Mathematics, vol. 7, no. 1, 2018.

[10] G. Pampara, N. Franken, and A. P. Engelbrecht, "Combining particle swarm optimisation with angle modulation to solve binary problems," in *2005 IEEE Congress on Evolutionary Computation, IEEE CEC,* 2005, vol. 1, pp. 89–96.

[11] H. Nezamabadi-Pour, "A quantum-inspired gravitational search algorithm for binary encoded optimization problems," Eng Appl Artif Intell, vol. 40, pp. 62–75, Apr. 2015.

[12] Y. He, H. Xie, T. L. Wong, and X. Wang, "A novel binary artificial bee colony algorithm for the set-union knapsack problem," Future Generation Computer Systems, vol. 78, pp. 77–86, Jan. 2018.

[13] F. B. Ozsoydan and A. Baykasoglu, "A swarm intelligence-based algorithm for the set-union knapsack problem," Future Generation Computer Systems, vol. 93, pp. 560–569, Apr. 2019.

[14] G. Lin, J. Guan, Z. Li, and H. Feng, "A hybrid binary particle swarm optimization with tabu search for the set-union knapsack problem," Expert Syst Appl, vol. 135, pp. 201–211, Nov. 2019.

[15] Z. Wei and J. K. Hao, "Iterated two-phase local search for the set-union knapsack problem," Future Generation Computer Systems, vol. 101, pp. 1005–1017, Dec. 2019.

[16] Y. Feng, J. H. Yi, and G. G. Wang, "Enhanced moth search algorithm for the set-union knapsack problems," IEEE Access, vol. 7, pp. 173774–173785, 2019.

[17] C. Wu and Y. He, "Solving the set-union knapsack problem by a novel hybrid jaya algorithm," Soft comput, vol. 24, no. 3, pp. 1883–1902, Feb. 2020.

[18] Z. Wei and J. K. Hao, "Multistart solution-based tabu search for the set-union knapsack problem," Appl Soft Comput, vol. 105, Jul. 2021.

[19] Y. He and X. Wang, "Group theory-based optimization algorithm for solving knapsack problems," Knowl Based Syst, vol. 219, May 2021.

[20] Z. Wei and J. K. Hao, "Kernel based tabu search for the set-union knapsack problem," Expert Syst Appl, vol. 165, Mar. 2021.

[21] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey badger algorithm: new metaheuristic algorithm for solving optimization problems," Math Comput Simul, vol. 192, pp. 84–110, Feb. 2022.

[22] S. Mirjalili and A. Lewis, “S-shaped versus v-shaped transfer functions for binary particle swarm optimization,” Swarm Evol Comput, vol. 9, pp. 1–14, Apr. 2013.

[23] S. Mirjalili, H. Zhang, S. Mirjalili, S. Chalup, and N. Noman, “A novel u-shaped transfer function for binary particle swarm optimisation,” in Advances in Intelligent Systems and Computing, vol. 1138, pp. 241–259, 2020.

[24] Y. He, F. Zhang, S. Mirjalili, and T. Zhang, “Novel binary differential evolution algorithm based on taper-shaped transfer functions for binary optimization problems,” Swarm Evol Comput, vol. 69, Mar. 2022.