Journal of the Faculty of Engineering and Architecture of Gazi University 36:4 (2021) 2333-2348

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

Adaptive binary artificial bee colony for multi-dimensional knapsack problem

Rafet Durgut[1]*, Mehmet Emin Aydın[2]
[1]Karabük University, Engineering Faculty, Computer Engineering Department, 78050, Karabük
[2]University of the West of England, Computer Science and Creative Technologies, Bristol, UK

Highlights:

- Adaptive ABC is used to MKP problem
- The proposed version is superior than metaheuristic literature
- The Adaptive version is efficient and robust for different problem dimensions.

Keywords:

- Artificial Bee Colony
- Multi-dimensional knapsack problem
- Adaptive ABC
- Binary ABC

Article Info:

Research Article
Received: 04.10.2020
Accepted: 01.05.2021

DOI:

10.17341/gazimmfd.804858

Correspondence:

Author: Rafet Durgut
e-mail: rafetdurgut@karabuk.edu.tr
phone: 0370 4187270

Graphical/Tabular Abstract

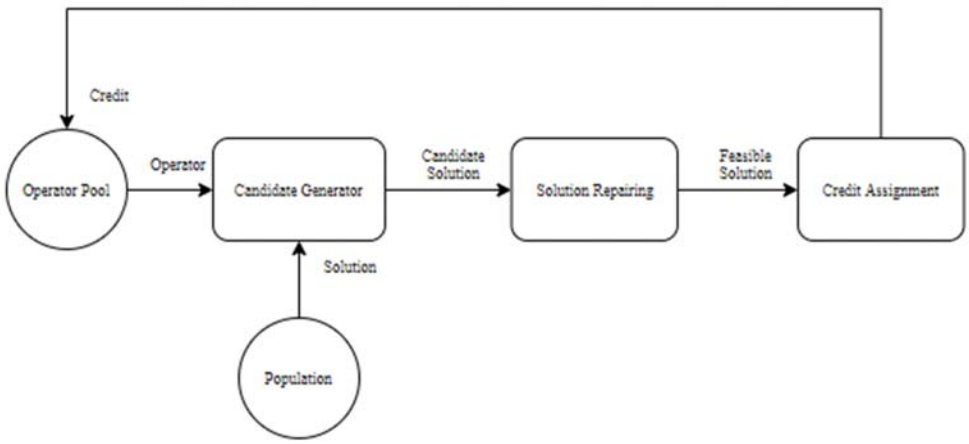


Figure A. General overview of the proposed algorithm

Purpose: The purpose of the study is to investigate how to solve for multi-dimensional knapsack problems better with higher robustness using binary artificial bee colony algorithms.

Theory and Methods:
The efficiency and effectiveness of metaheuristic optimization algorithms is managed with diverse search and fast approximation in the solution space. A balanced "exploration" and "exploitation" capability is required to achieve by the neighborhood operators towards the aimed efficiency. The majority of metaheuristic algorithms use either single operator or limited to genetic operators, which impose serious boundaries upon performance. In order to avoid this limitation, multiple neighborhood operators can be used within the search process orchestrated by a selection scheme. In this study, an adaptive operator selection scheme is studied with multiple binary operators embedded within artificial bee colony algorithm to solve the multidimensional backpack problem.

Results:
The performance gained with proposed artificial bee colony algorithm is compared with four different state-of-art metaheuristics approaches worked in the same circumstances. Three different benchmarking datasets are used for detailed comparisons. The statistical results including rank and Wilcoxon signed rank test values has been presented.

Conclusion:
Statistical analysis demonstrated that the proposed algorithm, adaptive binary artificial bee colony, has outperformed the state-of-art approaches with significant results over three benchmarking datasets. It has also been observed that the proposed algorithm produces more robust results too.

Journal of the Faculty of Engineering and Architecture of Gazi University 36:4 (2021) 2333-2348

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

Çok boyutlu sırt çantası problemi için adaptif ikili yapay arı kolonisi algoritması (AİYAK)

Rafet Durgut[1]*, Mehmet Emin Aydın[2]

[1]Karabük Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 78050, Karabük, Türkiye
[2]University of the West of England, Computer Science and Creative Technologies, Bristol, UK

Ö N E Ç I K A N L A R

- Çok boyutlu sırt çantası probleminin çözümü için adaptif ikili yapay arı kolonisi önerilmiştir
- Önerilen yöntem, literatürdeki meta sezgisel yöntemlerden daha iyi sonuçlar sunmaktadır
- Yöntemin etkililiği ve gürbüzlüğü istatistiksel olarak gösterilmiştir

Makale Bilgileri

Araştırma Makalesi
Geliş: 04.10.2020
Kabul: 01.05.2021

DOI:
10.17341/gazimmfd.804858

Anahtar Kelimeler:
Sırt çantası problemi,
yapay arı kolonisi,
ikili yak,
adaptif yak

ÖZ

Optimizasyon algoritmalarının etkinlik ve verimliliği çözüm uzayında aktif arama/keşif ve hızlı hareket etme kabiliyetlerine bağlıdır. Bir algoritmada "arama" ve "kullanma" kabiliyetleri kullanılan komşuluk operatörleri ile doğrudan ilgilidir. Bu kabiliyetleri arttırmak için birden fazla komşuluk operatörü arama süreci içerisinde dâhil edilebilir. Bu çalışmadan çok boyutlu sırt çantası probleminin çözümü için üç adet komşuluk operatörü içeren adaptif ikili yapay arı kolonisi kullanımı önerilmiştir. Çok boyutlu sırt çantası problemi birçok uygulama alanına sahip olan bir NP-zor problemdir. Özellikle büyük boyutlu problem örneklerinin makul sürelerde çözülmesi oldukça güçtür. Önerilen algoritmaya ait en iyi parametre yapılanmasının belirlenmesi için ilk olarak parametre ayarlama deneysel çalışmaları gerçekleştirilmiştir. Önerilen algoritmanın başarısı ve literatürdeki dört farklı yöntem ile üç farklı problem kümesi üzerinde istatistiksel karşılaştırmaları yapılmıştır. Önerilen algoritmanın literatürdeki diğer yöntemlerden daha başarılı sonuçlar ürettiği gösterilmiştir.

Adaptive binary artificial bee colony for multi-dimensional knapsack problem

H I G H L I G H T S

- Adaptive ABC is used to MKP problem
- The proposed version is superior than metaheuristic literature
- The Adaptive version is efficient and robust for different problem dimensions

Article Info

Research Article
Received: 04.10.2020
Accepted: 01.05.2021

DOI:
10.17341/gazimmfd.804858

Keywords:
Multi-dimensional knapsack problem,
artificial bee colony,
binary abc,
adaptive abc

ABSTRACT

The efficiency and effectiveness of metaheuristic optimization algorithms is managed with diverse search and fast approximation in the solution space. A balanced "exploration" and "exploitation" capability is required to achieve by the neighborhood operators towards the aimed efficiency. The majority of metaheuristic algorithms use either single operator or limited to genetic operators, which impose serious boundaries upon performance. In order to avoid this limitation, multiple neighborhood operators can be used within the search process orchestrated by a selection scheme. In this study, an adaptive operator selection scheme is studied with multiple binary operators embedded within artificial bee colony algorithm to solve the multidimensional knapsack problem (MKP) as a renown NP-Hard combinatorial problem. It is implemented for modelling and solving many real-world problems, while it is not trivial to offer a good solution within a reasonable timeframe. A parametric study has been conducted for the approach proposed in this study. The success of the proposed approach has been demonstrated and discussed with comparative analysis using three different classes of benchmark problem sets.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *rafetdurgut@karabuk.edu.tr, mehmet.aydin@uwe.ac.uk /
Tel: +90 370 418 7270

1. GİRİŞ (INTRODUCTION):

Son yıllarda metasezgisel optimizasyon algoritmalarına olan ilgi hızla artmaktadır [1]. Zor problemlere makul sürelerde makul çözümler üretebilmesi, problemden bağımsız çalışabilmesi ve kolay uygulanabilmesi bu ilginin temel nedenleri arasında gösterilebilir [2]. Bu alanda en çok kullanılanlar arasında Genetik Algoritma [3], Parçacık Sürüsü Optimizasyon [4], Yapay Arı Kolonisi [5], Evrimsel Gelişim [6] algoritmaları gösterilmesine karşın, doğal fenomenlerden esinlenen yeni bir metasezgisel algoritma her geçen gün literatüre eklenmektedir [7, 8].

Metasezgisel algoritmalar, başlangıç çözümü veya çözümlerin oluşturduğu popülasyon ile arama sürecine başlar. Küresel en iyiye ulaşabilmek için bu çözümlerden yeni çözümler üretir. Algoritmalar, arama sürecinde yeni çözümlerin daha iyiye gidemeyip sıkışıp kaldığı durumlarda (yerel minimuma takılma) arama uzayının farklı konumlarındaki çözümleri kullanmaya çalışmalıdır [9]. Böylece tüm arama uzayı etkin biçimde kullanılmış olur. Arama uzayındaki her çözümün kalitesi uygunluk fonksiyonu ile belirlenir. Nihai amaç etkin bir algoritma geliştirerek, küresel en uygun çözüme ulaşılabilmesidir.

Metasezgisel optimizasyon algoritmalarının etkinliği problem çözümünde büyük önem arz eder [10]. Bir algoritmanın etkinliği çözüm uzayındaki iki temel hareket etme kabiliyetine bağlıdır. Bunlardan ilki arama süreçlerindeki yeni ve iyi çözümleri keşfedebilme (exploration) kabiliyetidir. İkincisi ise eldeki bilgi ve çözümleri kullanarak (exploitation) geliştirebilme kabiliyetidir. Bu iki kabiliyet arasında denge kurulması gerekmektedir. Aksi takdirde algoritmadan iyi bir performans elde edilemeyecektir. Eğer arama/keşif fazı baskın olursa, algoritma var olan çözümleri geliştirmek için yeterli fırsat bulamayabilir ve yakınsama hızı oldukça yavaşlar. Eğer kullanma/geliştirme fazı baskın olursa, daha iyi çözümler sunabilecek bölgelere ulaşmakta zorlanılır ve algoritma yerel minimuma takılır [11].

Yapay Arı Kolonisi (YAK) algoritması Karaboğa vd. tarafından 2005 yılında önerilmiş metasezgisel optimizasyon algoritmasıdır [12]. Bal arılarının yiyecek bulma davranışı modellenerek, numerik optimizasyon problemlerine çözüm üretilmesi amacıyla geliştirilmiştir. YAK algoritması en önemli dezavantajı yakınsama hızının yavaş olmasıdır [13]. Bunun sebebi, her bir iterasyonda sadece bir karar değişkeninin güncellenmesi sonucunda oluşan arama/keşif fazının kullanma/geliştirme fazına olan baskınlığıdır. Ayrıca, çözüm kalitesine bağlı olarak gerçekleştirilen aday çözüm seçim işlemi arama sürecinin sonlarına doğru rassal hale gelmektedir [14].

Metasezgisel algoritmalarda tüm arama süreci genellikle tek bir arama/komşuluk operatörü ile yürütülür [15]. Seçilen çözüm üzerine komşuluk operatörü uygulanarak en iyi olmaya aday yeni bir çözüm oluşturulur. Tek operatör kullanımı çeşitli sorunlar oluşturmaktadır. Bunlardan ilki, algoritmanın başarısı kullanılan operatörün başarısı ile belirlenir [16]. Bu yüzden başarıyı arttırmak farklı arama operatörlerinin kullanılması veya algoritmaların birleştirilmesi önerilmiştir [17, 18]. Başarının arttırılması için farklı metasezgisellerin farklı operatörler kullanılarak geliştirilmesi ile ilgili literatür oldukça zengindir [19-22]. Literatüre bakıldığında başarının arttırıldığı görünse de, arama/keşif ve kullanma/geliştirme (sömürü) fazının dengelenmesine ait problemler devam etmektedir [23, 24].

Yapay arı kolonisi ve diğer metasezgisel algoritmalarda tüm arama süreci tek bir arama operatörü kullanılmasının yerine, farklı operatörlerin bir arada kullanılması adaptif mekanizma vasıtası ile sağlanmaktadır [25]. Burada amaç arama uzayının durumuna ve operatörlerin başarısına göre en etkili operatörün süreç içerisinde belirlenmesidir. Sürekli problemler için önerilen yaklaşımlarda başarılı çözüm üretme sayısına [26] veya çözümün kalitesine [27] bağlı olarak kullanılacak operatör belirlenmektedir. Bu sayede arama sürecinin farklı noktalarında daha başarılı olabilecek operatörlere imkan verilmektedir [28]. Böylece hem arama/keşif, hem de kullanma/geliştirme fazlarında denge sağlanmaya çalışılmaktadır.

Çok-boyutlu Sırt Çantası (ÇBSÇ) problemi, fen [29] ve mühendislik [30] alanında birçok uygulama alanı olan NP-Zor problemlerdendir [31]. Klasik 0-1 Sırt çantası problemini de içerisinde barındıran ÇBSÇ problemi Eş.1-3'teki gibi tanımlanır.

$$maksimize \ \sum_{j=1}^{n} p_j x_j \tag{1}$$

s.t.

$$\sum_{j=1}^{n} r_{i,j} x_j \leq b_i \ , i = 1,2,\dots,m \tag{2}$$

$$x_j \in \{0,1\}, j = 1,2,\dots,n \tag{3}$$

Burada p nesnelerin paha değerini, x nesnelerin durumunu (1 çantada, 0 dışarıda), r kısıt bilgisini, b kapasite bilgisini, n nesne sayısını, m ise boyut sayısını ifade eder. Problemin çözüm amacı en yüksek paha değerine sahip çantanın m boyutlu kısıtların tamamının kapasite sınırının aşılmadan elde edilmesidir.

ÇBSÇ probleminin çözümünün etki ettiği birçok günlük yaşamda problem bulunmaktadır. n adet projenin j adet kaynak ile sermaye bütçelendirilmesi için ÇBSÇ uygulanabilir [32]. Benzer şekilde çok bölmeli taşıma araçları ile yapılan market teslimatları [33] da ÇBSÇ kullanımına örnek olarak verilebilir. n sınırlı bölmeye sahip araçlara m adet paketlerin maksimum fayda getirecek şekilde atanması ÇBSÇ problemidir. Genel olarak n adet görevin, j adet sınırlı kaynak ile maksimum fayda ile tamamlanması gereken tüm problemlere uygulanabilmektedir.

ÇBSÇ günümüzde hala makul sürelerde kesin çözümü bulunmayan bir problemdir[34]. Kesin çözüm sunan algoritmalar küçük boyutlu problemler (eleman sayısı 250,

500 ve kısıt sayısı 5, 10 olan) için dal ve sınır algoritması [33], dinamik programlama veya hibrit algoritmalar [34] ile kesin çözüme ulaşabilmektedir. Fakat, NP-Zor problemlerin doğası gereği problem boyutu büyüdükçe çözüm için gerekli olan süre çok yüksek oranda (üssel olarak) artmaktadır [35, 36]. Gerçek yaşam problemlerinde çok daha büyük boyutlu problemler (eleman sayısının 500'den fazla kısıtların da 30'dan fazla olması durumu) için kesin çözüm oldukça uzun sürelerde üretilebilmektedir. Bu yüzden, kesin çözüm yerine en iyiye yakın çözümler sunan yaklaşımlar ön plana çıkmaktadır [37].

ÇBSÇ problemi için yaklaşık çözüm sunan algoritmalar ise, yerel arama ve popülasyon tabanlı metasezgisel optimizasyon algoritmaları kullanılmaktadır [38]. Yerel arama temelli tabu arama [39, 40], benzetilmiş tavlama [41] ve çekirdek arama [42] algoritmaları ile CBSÇ problemine yaklaşım çözümler sunulmaktadır. Popülasyon temelli algoritmalar ise, arama işlemini birden çok çözüm üzerinden gerçekleştirmektedir. Parçacık Sürüsü optimizasyonu [43, 44], genetik [45] ve memetik [46] algoritmalar, karınca kolonisi [47], gri kurt [36], harmonik arama [48], meyve sineği [49], guguk kuşu [50] gibi birçok popülasyon temelli optimizasyon algoritması ÇBSÇ problemleri için kullanılmıştır. Bu çalışmaların tamamında, tek bir komşuluk operatörü kullanılarak probleme kaliteli çözümler üretilmeye çalışılmaktadır. Bu sebeple, kaliteli çözüm üretilmesi amacıyla, literatürdeki çalışmalar halen geliştirme arayışına devam etmektedir [51, 52].

Önerilen çalışma, popülasyon temelli yaklaşık çözüm literatürüne katkı sağlamaktadır. Var olan literatüre katkısı, mevcut yöntemlerden daha kaliteli çözümler üretilmesi için tek bir komşuluk operatörü yerine birden çok komşuluk operatörünün adaptif olarak ÇBSÇ probleminde uygulanmasıdır. Çalışmada üretilen sonuçlar güncel literatürde yer alan metasezgiseller ile karşılaştırılmıştır. Önerilen yöntemin literatürdeki yöntemler ile karşılaştırmalarının istatistiksel olarak anlamlılığı da ortaya konulmuştur.

Yapay arı kolonisinin "0" ve "1" karar değişkenlerine sahip ikili optimizasyon problemlerine uygulanabilmesi için yapılmış birçok çalışma ve algoritmik düzenleme literatürde mevcuttur [53-56]. Bu düzenlemeler kullanılan karar değişkenine göre iki çatı altında toplanabilir. Birincisi, sürekli karar değişkeni kullanarak probleme uygulama öncesi ikili uzaya dönüşüm gerçekleştiren yöntemler literatürde mevcuttur [57, 58]. Bu yaklaşımlar algoritmalara ek yük getirdiği için diğer grup yaklaşımlara yönelme olmuştur. İkinci olarak; tamamen ikili karar değişkenleri üzerinde çalışan düzenlemeler üzerinde durulmuştur. Literatürde mevcut örnekler aşağıda dikkate alınmaktadır. Kashan vd. önerdiği çalışmada (disABC) yeni bir komşu çözüm üretirken aday çözüme benzer çözüm üretmeye çalışmaktadır [56]. Bu yöntemin içerdiği doğrusal optimizasyon problemi nedeniyle büyük boyutlu problemler için yavaş çalışmasıdır. Kıran vd. tarafından önerilen çalışmada (binABC) komşu çözüm üretme mekanizmasında aday çözüm ile seçili çözümü mantıksal işleme alınmaktadır [59]. Bu yöntemde ise, her güncelleme tek bir karar değişkeni üzerinde gerçekleştiği için yakınsama hızı yavaştır. Durgut, binABC algoritmasını geliştirerek önerdiği çalışmada (ibinABC) her iterasyondaki karar değişkeni sayısını dinamik olarak ayarlayarak geliştirme fazının katkısını arttırmıştır [60]. Her ne kadar bu yaklaşımlar belli başarılar ile uygulanmış olsa da tek operatörlü çözüm geliştirme yaklaşımlarının başarısı operatörün teknik sınırları ile kısıtlı kalır. Bunu yanında çok operatörle çalışan hibrit yaklaşımların söz konusu kısıtları aşma şansı daima daha yüksektir.

Bu çalışmada, literatürde kullanılan disABC, binABC ve ibinABC operatörleri birlikte kullanılarak geliştirilen adaptif YAK algoritması, çok boyutlu sırt çantası probleminin çözümü için önerilmiştir. Geliştirilen algoritmada kredi atama probleminin çözümü için üç farklı ödül atama test edilmiştir. Operatör seçimi için adaptif takip /kovalama (adaptive pursuit) [61] tekniği kullanılmıştır. Önerilen algoritma için detaylı parametre analizi gerçekleştirilmiş olup, en iyi yapılandırma belirlenmiştir. Algoritmanın başarısı ölçülmesi üç farklı veri kümesi üzerinde test edilmiştir.

2. MATERYAL VE METOT (MATERIAL AND METHOD):

2.1. Yapay Arı Kolonisi (Artificial bee colony)

Yapay arı kolonisi, Karaboğa vd. tarafından geliştirilmiş, bal arılarının yiyecek bulma davranışından esinlenilmiş sürü zekasına dayalı meta sezgisel optimizasyon algoritmasıdır [5]. Bal arıları yiyecek bulma sürecinde işçi, gözcü ve kâşif arı olmak üzere farklı görevlere sahiptir. Bu görev paylaşımı optimizasyon algoritmasında üç faz olacak şekilde modellenmiştir. İşçi arı fazında her arı üzerinde çalıştığı besin kaynağını geliştirmektedir. Gözcü arı fazında ise, kaliteli olan besin kaynakları daha da geliştirilmeye çalışılmaktadır. Besin kaynaklarının kalitesi sahip oldukları gerçek yaşamda nektar miktarına, optimizasyon açısından ise amaç fonksiyonuna göre belirlenmektedir. Geçerli çözümü ifade eden besin kaynaklarının sayısı işçi ve gözcü arı sayısına eşittir.

Yöntem içerisinde ilk çözümler Eş. 4'e göre belirlenmektedir.

$$x_{i,j} = sınır_{alt}^{j} + \left(sınır_{üst}^{j} - sınır_{alt}^{j}\right) * rast[0,1] \quad (4)$$

i.besin kaynağının j. boyutuna ait rastgele değer ($x_{i,j}$) verilen sınır aralığında rastgele olarak belirlenir. Tüm besinlerin tüm boyutlarına geçerli değerler atanarak ilk çözümler oluşturulur. Elde edilen çözümler amaç fonksiyonu $f(x_i)$ değerine göre uygunluk değeri Eş. 5'e göre belirlenir.

$$uygunluk_i = \begin{cases} \frac{1}{1+f(x_i)}, f(x_i) \geq 0 \\ 1 + |f(x_i)|, f(x_i) < 0 \end{cases} \quad (5)$$

En iyi besin kaynaklarının seçilebilmesi için her besin kaynağına bir olasılık değeri atanır. Bu sayede gözcü arı fazında geliştirilecek besin kaynakları belirlenir. Olasılık değerleri Eş. 6'ya göre atanır.

$$olasilik_i = \frac{uygunluk_i}{\sum_{j=1}^{SN} uygunluk_i} \quad (6)$$

Burada SN besin kaynağı sayısını ifade etmektedir. Bu olasılık değerleri rulet tekeri, turnuva seçimi gibi yöntemlerle birlikte kullanılabilir. Orijinal YAK (ABC) algoritmasında rastgele bir sayı üretip olasılık değerinden küçük olup olmamasına bakarak güncelleme yaptırmaktadır. Güncelleme operatörü olarak Eş. 7 kullanılmaktadır.

$$aday_{i,j} = secili_{i,j} + \theta_{i,j}(secili_{i,j} - komsu_{i,j}) \quad (7)$$

İşçi arı fazında her bir besin kaynağı, gözcü arı fazında yalnızca seçili olanlar için yeni çözüm üretilmektedir. Aday çözüm üretilirken kullanılan $\theta_{i,j}, [-1, 1]$ aralığındadır. Rastgele belirlenen j hanesi (bit) için çözüm güncellemesi yapılır.

2.2. İkili Yapay Arı Kolonisi (Binary artificial bee colony)

YAK sürekli optimizasyon problemleri çözme amacıyla geliştirildiği için, ikili optimizasyon problemlerine doğrudan uygulanamaz. Bunun uygulanabilmesi için bazı düzenlemeler yapılmalıdır. Bu düzenlemeler iki sınıfa ayrılabilir; Bunlar probleme uygulama aşamasında sürekli karar değişkenlerinin ikili uzaya yerleştirilmesi [56] veya karar değişkenlerinin ikili formda [57,58] olmasıdır. İlk sınıftaki düzenleme kullanıldığında yöntemin temel taşlarında değişiklik olmaz ve sürekli uzaydan ikili uzaya haritalama fonksiyonu yeterlidir. İkinci sınıftaki düzenlemeler de ise, mantıksal karşılaştırma ve ifadeler kullanılmaktadır. Bu çalışmada literatürde sıklıkla kullanılan binABC, disABC ve yeni önerilmiş olan ibinABC operatörleri adaptif bir mekanizma içerisinde kullanılmaktadır.

binABC algoritması Kiran vd tarafından kapasite kısıtsız tesis yerleştirme problemi üzerine uygulanarak geliştirilmiştir [59]. Arama operatörü olarak mantıksal kapıları kullanan bu geliştirme Eş. 4 yerine Eş. 8 kullanılarak çözüm üzerinde güncelleme yapılmaktadır. 0 veya 1 formunda olan karar değişkenlerinin değerleri θ değişkene göre belirlenmektedir.

$$aday_{i,j} = aday_{i,j} \oplus \theta_{i,j}(aday_{i,j} \oplus komsu_{i,j}) \quad (8)$$

İkili (Binary) arama uzayında çalışabilen bir diğer ikili YAK versiyon Kashan vd. tarafından önerilmiştir [56]. Bu yaklaşımda Eş. 4 yerine yeni çözüm üretme mekanizması kullanılmaktadır. Üretilen yeni çözümün, seçili ve komşu çözüm arasındaki farklılık değerine göre belirlenmektedir. Yöntem ilk olarak aday ve komşu çözüme ait Jaccard benzerlik katsayısını Eş. 9'a göre hesaplamaktadır. Ardından Eş. 10'a göre farklılık değerini sayısal olarak belirlemektedir.

$$benzerlik(secili, komsu) = \frac{M_{11}}{M_{01}+M_{10}+M_{11}} \quad (9)$$

$$farklılık(secili, komsu) = 1 - benzerlik(secili, komsu) \quad (10)$$

Burada M_{11} seçili ve aday çözümde aynı konumda 1 olanların sayısı, M_{10} seçili çözümde 1 iken komşu çözümde 0 olanların sayısı, M_{01} ise seçili çözümde 0 iken komşu çözümde 1 olanların sayısını ifade etmektedir. Farklılık katsayısı kullanılarak aday çözüm Eş. 11 ve Eş. 12-15'deki tam sayılı doğrusal olmayan modele göre belirlenir.

$$farklilik(aday, seçili) = \phi \times farklilik(secili, komsu) \quad (11)$$

$$\min|farklilik(aday, seçili) - \phi \times farklilik(secili, komsu)| \quad (12)$$

s. t.

$$M_{11} + M_{01} = n_1 \quad (13)$$

$$M_{10} \leq n_0 \quad (14)$$

$$\{M_{10} + M_{01} + M_{11}\} \geq 0 \; ve \in \mathbb{Z} \quad (15)$$

Burada ϕ, pozitif ölçeklendirme katsayısıdır. Model sonucunda n_1 ve n_0 parametreleri elde edilir. n_1 aday çözümde 1 olacak hanelerin (bitlerin) sayısını, n_0 ise 0 olacak hanelerin (bitlerin) sayısını belirler. Böylece, aday çözüm ile seçili çözüm arasındaki farklılık en aza indirilmeye çalışılır. Seçili çözümde 1 olan hanelerden (bitlerden) n_1 tanesi seçilir ve aday çözüme aktarılır ve diğer haneler (bitler) sıfırlanır.

Durgut, binABC operatörünü iyileştirerek ibinABC algoritmasını önermiştir [60]. ibinABC algoritması iki yeni düzenleme içermektedir. Bunlardan ilki komşuluk operatörünün birden fazla hane (bit) üzerinde uygulanmasıdır. Operatörün uygulanacağı hane (bit) sayısı Eş. 16'ya göre belirlenmektedir.

$$d_t = rast[0, \alpha] + e^{-\left(\frac{t}{tmax}\right)} 0{,}1D + 1 \quad (16)$$

Burada α yöntem parametresi olup, ölçeklendirme için kullanılmaktadır. D, problem boyutunu, t o anki iterasyon değerini t_{max} ise maksimum iterasyon sayısını ifade eder. Bir diğer düzenleme ise θ değişkenin seçili ve komşu çözümün uygunluk değerine göre belirlenmesidir. binABC algoritmasında θ değeri rastgele olarak belirlenmektedir. Böyle bir durumda, iki çözümün uygunluk değerlerinin bir önemi yoktur. ibinABC'de ise eğer komşu çözüm daha iyi ise komşuluk operatöründe komşu çözümün ilgili hanesi (biti) işleme daha fazla etkili olacak şekilde

belirlenmektedir. Aksi durumda ise, θ iterasyon değerine göre Eş. 17'deki gibi belirlenmektedir.

$$\theta = \begin{cases} \theta_{max} - \frac{(\theta_{max}-\theta_{min})}{\theta_{max}} t, uygunluk(secili) < \\ \qquad uygunluk(komsu) \\ 0, \ aksi\ halde \end{cases} \quad (17)$$

2.2. Adaptif ikili yapay arı kolonisi
(Adaptive binary artificial bee colony)

Birden çok operatör bir arada kullanılması için bir seçim mekanizmasına ihtiyaç vardır. Bu çalışmada söz konusu mekanizma adaptif bir yapı olarak önerilmektedir. Adaptif yapının genel çalışma mantığı şekil 1'de verilmiştir.

Başarılı bir adaptif yapı oluşturmak için iki temel problemin çözülmesi gerekmektedir. Bunlardan ilki operatörlerin başarısını ödüllendirmek için kredilendirmek ve bir kredi atama yaklaşımı belirlemektir [63]. İkincisi ise, yeni çözüm üretilmesi için kullanılacak operatör seçiminde her operatörün sahip olduğu krediye dayalı bir seçim yaklaşımını belirleme problemidir [64]. Kredi atama probleminde; ödül bilgisi olarak operatörlerin başarılı olma sayıları ve çözümü ne kadar iyileştirdikleri ele alınmaktadır. Burada bir alt problem olan kredi belirlenirken ödüllerin hangi aralıkta değerlendirileceğidir. Son iterasyona ait ödül, Son n iterasyon ödülünün ortalaması veya son n iterasyonun en yüksek ödül değeri kullanılabilir. En iyi konfigürasyon önceden kestirilemeyeceği için, probleme göre deneysel olarak belirlenebilir. Bu çalışmada operatörlere kredi atama için Eş. 18 kullanılmaktadır.

$$kredi_{i,t} = (1-\alpha) kredi_{i,t} + \alpha\ \text{ödül}_{i,t}, i = 1,2,\dots,K \quad (18)$$

Eş. 18'deki *ödül* miktarı bu çalışmada Eş. 19 kullanılarak hesap edilir. Başlangıç iterasyonlarında iyileşme daha fazla olduğu için küresel en iyi değerine göre ölçeklendirme yapılmıştır.

$$\text{ödül}_{i,t} = \frac{f(aday)}{küresel\_eniyi}(f(aday) - f(secili)) \quad (19)$$

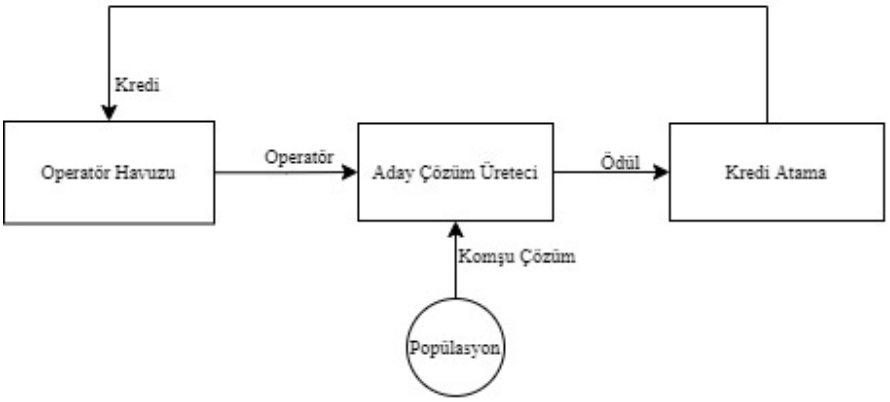
Operatör seçimi için ise olasılık tabanlı seçim yöntemleri kullanıldığı gibi pekiştirmeli öğrenme yöntemlerinden de faydalanılmaktadır. Rulet tekeri (roulette wheel), olasılıklı eşleme (probability matching), adaptif takip (adaptive pursuit) ve üst güvenilirlik sınırı (upper confidence bounds) yaklaşımları kullanılmaktadır. Gerçekleştirdiğimiz kapsamlı çalışma sonucunda adaptif takip ile ikili YAK algoritması diğerlerinden daha üstün sonuçlar üretmiştir. Bu yüzden bu çalışmada adaptif takip yöntemi kullanılmıştır. Bu yaklaşım "kazanan tümünü alır" prensibine göre çalışmaktadır. İlgili iterasyonda en yüksek krediye sahip olan operatör maksimum olasılık değerini alırken, diğer operatörlere minimum olasılık değeri verilir. $t+1$ anında operatörlerin olasılık değeri Eş. 20'ye göre belirlenir.

$$p_{i,t+1} = \begin{cases} p_{i,t} + \beta(p_{max} - p_{i,t}), & eğer\ i = i_t^* \\ p_{i,t} + \beta(p_{min} - p_{i,t}), & aksi\ halde \end{cases} \quad (20)$$

Burada $\beta \in [0,1]$ öğrenme katsayısı, i^* en iyi operatörü belirtmektedir.

Bu çalışmada kullanılan adaptif ikili YAK algoritması şekil 2'de verilmiştir. Algoritma, başlangıç parametrelerinin (limit, popülasyon boyutu vs.) belirlenmesiyle başlar. Geçerli ve rastgele çözüm içeren bireyler ile ilk popülasyon oluşturulur ve sahip oldukları çözümün amaç fonksiyonundaki karşılıkları atanır. Popülasyon içerisindeki en iyi birey global en iyi olarak tutulur. Yöntem, esas döngü içerisinde durdurma kriteri sağlanana kadar çalışmasını sürdürür. Durdurma kriteri olarak maksimum çalışma süresi, maksimum iterasyon sayısı veya maksimum fonksiyon değerlendirme sayısı olabilir. Adil bir karşılaştırma için yöntemden bağımsız olarak maksimum fonksiyon değerlendirme sayısı sıklıkla kullanılmaktadır [64].

Üç temel fazdan ilki olan işçi arı fazında ilk olarak her bireyin için operatör seçimi yapılır. Başlangıçta tüm krediler aynı olduğu için bu seçim rastgele olarak yapılır. Kredi değerleri güncellendikçe operatör seçimi de anlamlı hale gelecektir. Her birey için operatör atanması yapıldıktan sonra, seçili çözüm üzerine uygulanarak aday çözüm elde edilir. Eğer aday çözüm, seçili olan çözümden daha iyi ise ilk olarak aday çözüm seçili çözüme kopyalanır, ardından



Şekil 1. Adaptif mekanizma (Adaptive mechanism)

```
Başlangıç parametrelerini belirle.
İlk popülasyonu oluştur.
Popülasyonu değerlendir.
while durdurma kriteri sağlanana kadar do
        İşçi arı fazı:
        Uygulanacak operatörleri belirle.
        for i=1 to N do
                Op_i kullanarak aday çözümü (x_a) üret.
                if f(x_a) daha iyiyse f(x_i)'den then
                        x_i = x_a
                        Ödülü hesapla.
                        Ödül sayacını arttır.
                        Limit sayacını sıfırla.
                else
                        limit sayacını arttır.
                end if
        end for
        Gözcü arı fazı:
        Çözümlerin olasılık değerlerini belirle.
        for i=1 to N do
                Komşu çözümü seç (x_k)
                Op_i kullanarak aday çözümü (x_a) üret.
                if f(x_a) daha iyiyse f(x_k)'den then
                        x_k = x_a
                        Ödülü hesapla.
                        Ödül sayacını arttır.
                        Limit sayacını sıfırla.
                else
                        limit sayacını arttır.
                end if
        end for
        Kredileri ata.
        En iyi çözümleri sakla.
        Kaşif arı fazı:
        if limit değerini aşan çözüm varsa then
                limit değerini aşan ilk çözüm yerine rastgele yeni çözüm üret.
        end if
end while
```

Şekil 2. Adaptif İkili YAK sözde kodu (The pseudo code of binary adaptive ABC)

ödül miktarı belirlenir ve deneme sayacı sıfırlanır. Aksi durumda deneme sayacı arttırılır. Deneme sayacı, bireyin kaç iterasyon boyunca iyi çözüme ulaşamadığı bilgisini tutar ve kâşif arı fazında kullanılır.

Gözcü arı fazı da işçi arı fazına benzer, fakat güncellenecek çözümler kalitesine göre seçilir. Bu yüzden ilk olarak her çözümün kalitesine bağlı seçilme olasılığı hesaplanır. Eğer olasılık gerçekleşirse, çözüm üzerinde güncelleme yapılır. Güncelleme yapılması için seçili operatör uygulanır. Eğer iyileşme olursa çözümler ve ödül bilgileri güncellenir ve deneme sayacı sıfırlanır. Aksi durumda deneme sayacı arttırılır. Birey konum güncellemeleri gerçekleştikten sonra, bilgi güncellemeleri gerçekleşir. İlk olarak elde edilen ödül bilgilerine göre her operatör için kredi atamaları yapılır. Yöntem içerisindeki en iyi çözüm saklanır. Kaşif arı fazında ise, limit değerini aşan ilk bireye ait çözüm yeni, rastgele ve geçerli bir çözüm ile değiştirilir.

2.3. Çok boyutlu sırt çantası problemi için Adaptif ikili yapay arı kolonisi algoritması

(Adaptive binary artificial bee colony algorithm for multi dimensional knapsack problem)

ÇBSÇ Problemine uygulanabilecek çözüm vektörü $x = (x_1, x_2, \dots, x_m)$ formunda tutulur. Eğer i. nesne çantaya eklenecekse $x_i = 1$ aksi durumda $x_i = 0$ olur. Her x vektörü m boyutlu olup, YAK içerisindeki her bir birey için konum değeridir. En iyi çözüme ulaşabilmek için x vektörü üzerinde komşuluk operatörleri uygulanmaktadır.

Elde edilen çözümler kapasite kısıtlarını aşması durumunda mümkün olmayan çözümler isimlendirilir ve farklı biçimde ele alınır. Mümkün olmayan çözümlerin kullanılması için üç farklı yaklaşım vardır. Bunlardan ilki ve en basiti olan çözümü göz ardı etmek. Çözüm kapasite kısıtını sağlamadığında işleme alınmaz ve güncelleme yapılmaz. İkinci yaklaşım ise; aşılan kısıtlara göre ceza puanı

uygulamaktır. Bu durumda ise ceza puanı düzgün belirlenmelidir aksi durumda algoritma hiç mümkün çözüme ulaşamayabilir. Üçüncü yaklaşım ise, mümkün olmayan çözümleri mümkün hale getirmek için onarma işlemi yapılmasıdır. Kısıtları aşmayacak hale getirmek için çözüm üzerinde ekleme ve silme işlemleri yapılır.

Bu çalışmada, mümkün olmayan çözümleri onarmak için Chu ve Beasley tarafından önerilen fayda tabanlı onarma tekniği [65] kullanılmıştır. Sadece teknik içerisinde kullanılacak her bir nesne için fayda bilgisi gereklidir. Bu fayda bilgisi kullanılarak silme fazında, çantada bulunan nesneler çanta dışarısına çıkarılırken faydası en küçük olan nesnelerden başlanır. Benzer şekilde çantaya nesne eklenirken faydası en büyük olan nesneler çantaya eklenmeye devam eder. Fayda (u) değeri aşağıdaki formül yardımıyla hesaplanır.

$$u_j = \frac{p_j}{\sum_{i=1}^{m} w_i r_{i,j}} \quad ,\forall j = 1,2,\dots,n \tag{21}$$

Burada p_j j. nesnenin paha değerini, $r_{i,j}$ j. nesnenin i. boyuttaki kısıt değerini w_i ise tam sayılı doğrusal programlamada temsili gevşetme katsayılarını (ağırlıklarını) ifade eder.

Teknikte, nesneler fayda değerine göre artan sırada sıralanır. İlk adımda her bir kısıt için toplam kısıt değerleri hesaplanır. Silme fazında çantada ekli olan nesne, herhangi bir boyutta kısıt değerinin aşılmasına neden oluyorsa çantadan çıkarılır ve birikimli kısıt değeri çıkarılan nesnenin tüm kısıt boyutları için güncellenir. Ekleme fazında ise çanta dışında olan nesnenin çantaya eklenmesi durumunda tüm kısıtlar sağlanmaya devam ediyorsa, çantaya eklenir. Teknik sonucunda onarılan çözüm tüm kısıtları sağlar ve geçerlidir.

Geliştirilen algoritmanın test edilmesi ve literatürdeki yöntemler ile karşılaştırılması için 2 farklı problem kümesi oluşturulmuştur. İlk problem kümesinde çözümü daha kolay olan Chu [65] tarafından oluşturulan 100 nesne ve 5 boyut içeren 30 adet ve Weing [66] tarafından oluşturulan 28 nesne 2 boyut ve 28 nesne 105 boyut içeren 8 adet problem örneği kullanılmıştır. İkinci problem kümesinde ise çözümü daha zor olan Chu tarafından oluşturulan 500 nesne ve 30 problem boyutu içeren 30 adet ve Glover ve Kochenberg [67] tarafından oluşturulan farklı boyutlarda 11 problem kullanılmıştır. Yöntemlerin karşılaştırılması için en iyi çözüme yüzdelik olarak uzaklığı ifade eden *gap* kullanılacak olup Eş. 22'deki gibi hesaplanır.

$R_i = \sum_{j=1}^{n} \mathrm{r_{i,j}} \times x_j \quad ,\forall i = 1,2,\dots,m$
Silme Fazı:
for j = n to 1 do
  if x_j==1 && $R_i > b_i$ herhangi bir $i \in I$ then
    $x_j = 0$
    $\mathrm{R_i} = \mathrm{R_i} - \mathrm{r_{i,j}} \quad ,\forall i \in I$
  end if
end for
Ekleme Fazı:
for j = 1 to n do
  if x_j==0 && $R_i + r_{i,j} \leq b_i \, ,\forall i \in I$ then
    $x_j = 1$
    $\mathrm{R_i} = \mathrm{R_i} + \mathrm{r_{i,j}} \quad ,\forall i \in I$
  end if
end for

Şekil 3. Fayda temelli onarma operatörü
(The pseudo-utility based repair operator)

$$gap = 100 * \frac{f_{eniyi} - f_{bulunan}}{f_{eniyi}} \tag{22}$$

Çalışmada önerilen algoritmanın parametrelerinin belirlenmesi için parametre ayarlama işlemi gerçekleştirilmiştir. Bu adımda, adaptif mekanizma içerisinde kullanılacak konfigürasyon belirlenmiştir. Parametre havuzunda popülasyon boyutu, α (alpha), β (beta) ve kredi atamasında kullanılacak ödül türleri (anlık, ortalama ve maksimum) bulunmaktadır.

İlk parametre ayarlaması popülasyon boyutu 100 için gerçekleştirilmiş ve 30 farklı çalıştırma sonucunda elde edilen ortalama sonuçlar Tablo 1'de verilmiştir.

Tablo 1. SN=100 için GK4 problem örneği üzerinde parametre ayarlama
(The parameter tuning of the algorithm with SN=100 on GK4 problem instance)

			Anlık			Ortalama				Maksimum	
			W			W				W	
	α	Pmin	1	5	10	25	50	5	10	25	50
N=100	0,10	0,05	0,38	0,35	0,36	0,34	0,37	0,37	0,34	*0,33*	0,35
		0,10	0,38	0,35	0,35	0,36	0,34	0,37	0,37	0,35	0,36
		0,20	0,38	0,36	0,37	0,36	0,36	0,37	0,38	0,38	0,35
	0,50	0,05	*0,34*	0,36	*0,32*	0,34	0,35	*0,33*	0,35	0,33	0,34
		0,10	0,36	0,35	0,37	0,34	0,36	0,36	0,35	0,35	0,35
		0,20	0,38	0,37	0,35	0,38	0,36	0,37	0,37	0,36	0,34
	0,90	0,05	*0,34*	0,34	0,37	0,35	0,37	0,35	0,34	*0,33*	0,35
		0,10	*0,34*	0,33	0,34	0,37	0,37	0,33	0,35	0,34	0,35
		0,20	0,36	0,36	0,35	0,35	0,37	0,37	0,36	0,36	0,36
		Min:	*0,34*			*Min:*	*0,32*			*Min:*	*0,33*

Farklı α değerleri $(0{,}1, 0{,}5, 0{,}9)$, Farklı p_{min} değerleri $(0{,}05, 0{,}1, 0{,}2)$ Farklı ödül türleri ve bu türlere ait pencere boyutları için elde edilmiş sonuçlar görülmektedir. Tablodan görüleceği üzere, en düşük *gap* değeri 10 pencere boyutu için ortalama ödül kullanıldığında elde edilmiştir. Diğer ödül türlerine göre ortalama ödül daha üstün sonuçlar sunmaktadır. Anlık ödül en kötü sonuçları sunmaktadır. Pencere boyutuna göre ise ortalama ödül için 10 ve 25, maksimum ödül için 5 ve 25 aralarında rekabet etmektedir.

Popülasyon boyutu (N) 40 için parametre ayarlama işlemi 30 farklı çalıştırma sonucunda elde edilen en iyi değerlerin ortalaması Tablo 2'de verilmiştir. 40 popülasyon boyutu ile elde edilen sonuçlar açıkça görüldüğü üzere 100 popülasyon boyutundan daha iyidir. Tablodaki değerlere göre, en iyi ödül türü 25 pencere boyutundaki ortalama ödül miktarının kullanmasıdır. Bu sonuçlar arasında $\alpha = 0{,}9$ olması ve $p_{min} = 0{,}05$ olması en iyi sonuçlara ulaşılmasını sağlamıştır. Bu iki tablo ışığında en iyi parametre yapılandırması olarak popülasyon boyutunun 40, α parametresi 0,9, p_{min} parametresi 0,05, Pencere boyutu 25 ve ortalama ödül türü yöntemi ifade etmekte ve çalışmanın devamındaki tablolarda kullanılacaktır.

3. DENEYSEL SONUÇLAR VE TARTIŞMALAR (EXPERIMENTAL RESULTS AND DISCUSSIONS)

Bu bölümde, önerilen yöntemin literatürdeki diğer 4 yöntem (BGWO, BFOA, HHS, QPSO) ile karşılaştırmalı tabloları verilecektir. Yöntemlerin adil bir şekilde karşılaştırılabilmesi için tüm yöntemler 30 farklı çalıştırma ile 100,000 maksimum fonksiyon değerlendirmesi yapmışlardır. BGWO 20, HHS 40, BFOA 40, QPSO 20, AİYAK 40 popülasyon boyutuyla çalıştırılmıştır. Diğer yöntemlere ait sonuçlar önceki çalışmadan doğrudan alınmıştır [68]. AİYAK algoritması için çalıştırma parametreleri $limit = SN * \frac{D}{2}, p_{max} = 0{,}8, \beta = 0{,}5$ olarak verilmiştir.

Tablo 3'de 38 problem içeren ilk problem kümesi için elde edilen sonuçlar verilmiştir. Boyut sütunu kısıt ve nesne sayısını, Eniyi sütunu o problem örneği için bilinen en yüksek çözüm değerini, Ortalama sütunu 30 farklı çalıştırma sonucunda yöntemin elde ettiği çözüm değerini, oran sütunu ise 30 çalıştırmanın yüzde kaçında en iyi sonuca ulaşıldığını belirtmektedir. Tabloda en iyi yöntemin belirlenmesi için ortalama sütunları kullanılmış olup, en iyi çözüm italik yazı tipiyle belirtilmiştir.

Tablodaki bilgiler ortalama değerlere göre incelendiğinde BGWO ve AİYAK 38 problem örneğinin 24'ünde en iyi sonuca ulaşmıştır. BFOA 6, HHS 7, QPSO ise 15 örnekte en iyi sonuca ulaşabilmişlerdir. %100 başarılı problem kümelerinin sayısı ise sırasıyla 14, 6, 7, 14, 14 tümünde başarısız olunan problem kümelerinin sayısı sırasıyla 5, 8, 10, 7 ve 4'dür. En başarılı algoritma AİYAK'dır.

Yöntemlerin detaylı karşılaştırılması ve istatistiksel olarak yorumlanabilmesi için her yöntemin başarı sırası ve Wilcoxon işaretli test p-değeri Tablo 4'de verilmiştir. Başarı sırası belirlenirken ortalama değerler kullanılmıştır. p-değeri AİYAK ile karşılaştırılan yöntem sonuçları arasında istatistiksel güvenilirlik değerini boş hipotezine göre vermektedir. Eğer p-değeri 0,05'den küçük ise boş hipotez reddedilir ve istatistiksel olarak sonuçlar güvenilirdir. p-değerinin 1 çıkması ise sonuçların tamamen aynı olduğunu gösterir. Tabloya sonuçların aynı olmadığı durumların çoğunda istatistiksel olarak da anlamlıdır. Başarı sıralarına bakıldığında ise AİYAK ortalama 1,5 ile en başarılı algoritma olmuştur. Sonraki sıralama BGWO, QPSO, HHS ve BFOA şeklindedir.

İkinci problem kümesinde Chu'ya ait 30 problem kümesi ve Gloover ve Kochenberger'e ait 11 problem kümesi kullanılmıştır. Problemlerin boyutu ve en iyi değerleri Tablo 5'de verilmiştir. Tabloda her yöntemin ulaştığı en iyi çözüm ve 30 farklı çalıştırma için elde edilen *gap* değeri Eş. 22'de olduğu gibi hesap edilerek verilmiştir. Yöntemlerin

Tablo 2. SN=40 için GK4 problem örneği üzerinde parametre ayarı
(Parameter tuning of the algorithm with SN=40 on GK4 problem instance)

			Anlık		Ortalama			Maksimum			
			W			W				W	
	α	p_{min}	1	5	10	25	50	5	10	25	50
N=40	0,10	0,05	*0,29*	0,29	0,29	0,29	0,30	0,31	0,29	0,30	0,30
		0,10	0,32	0,30	0,32	0,32	0,32	0,31	0,29	0,30	0,31
		0,20	0,34	0,33	0,32	0,33	0,33	0,33	0,33	0,32	0,31
	0,50	0,05	0,31	0,29	0,31	0,31	0,29	0,30	0,29	0,29	0,31
		0,10	0,30	0,33	0,31	0,31	0,29	0,30	0,32	0,31	0,31
		0,20	0,32	0,31	0,33	0,33	0,30	0,33	0,30	0,32	0,33
	0,90	0,05	*0,29*	0,31	0,31	*0,27*	0,30	*0,28*	*0,28*	0,31	0,31
		0,10	*0,29*	0,29	0,31	0,30	0,30	0,31	0,31	0,32	0,31
		0,20	0,31	0,33	0,28	0,32	0,31	0,31	0,33	0,32	0,32
		Min:	*0,29*			*Min:*	*0,27*			*Min:*	*0,28*

Tablo 3. İlk problem kümesi için yöntemlerin karşılaştırılması (1-30 Chu ve 31-38 Weing problemleri)
(The comparison of methods for the first problem set (1-30 Chu and 31-38 Weing problems))

	Problem		BGWO		BFAO		HHS		QPSO		AİYAK	
No	Boyut	Eniyi	Ortalama	Oran	Ortalama	Oran	Ortalama	Oran	Ortalama	Oran	Ortalama	Oran
1	5×100	24381	24381,0	100%	24352,4	53%	24365,2	77%	*24381,0*	100%	*24381,0*	100%
2	5×100	24274	24270,7	97%	24224,4	37%	24239,8	40%	*24274,0*	100%	24269,5	83%
3	5×100	23551	23525,5	0%	23520,6	0%	23521,0	0%	23534,5	0%	*23540,3*	33%
4	5×100	23534	23494,2	3%	23458,5	0%	23463,5	0%	23487,9	0%	*23497,5*	3%
5	5×100	23991	*23970,1*	37%	23944,0	7%	23940,4	0%	23960,8	3%	23960,2	17%
6	5×100	24613	*24608,7*	87%	24560,9	7%	24556,5	0%	24599,5	47%	24599,9	50%
7	5×100	25591	*25591,0*	100%	25539,7	30%	25528,0	10%	25539,7	27%	25584,0	90%
8	5×100	23410	23369,4	7%	23348,2	10%	23348,0	0%	23375,0	7%	*23396,8*	63%
9	5×100	24216	24206,8	23%	24200,3	37%	24197,8	23%	*24216,0*	100%	24204,0	0%
10	5×100	24411	*24406,6*	63%	24309,2	3%	24336,4	10%	24379,4	50%	24365,3	20%
11	5×100	42757	42705,8	7%	42699,3	3%	42706,6	3%	42705,0	0%	*42706,7*	3%
12	5×100	42545	42473,7	10%	42463,2	0%	42461,7	3%	42469,4	3%	*42477,8*	10%
13	5×100	41968	41952,8	0%	41934,0	0%	41943,7	0%	41953,1	0%	*41959,5*	0%
14	5×100	45090	*45070,9*	0%	45020,8	0%	45017,6	0%	45057,1	0%	45066,9	0%
15	5×100	42218	*42218,0*	100%	42202,3	77%	42204,2	80%	42216,4	93%	*42218,0*	100%
16	5×100	42927	*42927,0*	100%	42852,0	23%	42876,3	47%	*42927,0*	100%	*42927,0*	100%
17	5×100	42009	*42009,0*	100%	41994,5	90%	42006,4	97%	*42009,0*	100%	*42009,0*	100%
18	5×100	45020	45008,7	0%	44983,7	50%	44991,8	53%	45018,3	93%	*45019,0*	97%
19	5×100	43441	43377,3	3%	43353,1	20%	43338,1	10%	43397,2	50%	*43402,6*	37%
20	5×100	44554	*44527,0*	10%	44480,0	0%	44491,8	0%	44511,4	0%	44512,6	0%
21	5×100	59822	*59822,0*	100%	59808,2	40%	59819,7	90%	*59822,0*	*100%*	*59822,0*	100%
22	5×100	62081	62012,7	13%	61973,9	13%	61971,2	10%	62015,3	33%	*62068,6*	87%
23	5×100	59802	*59785,0*	63%	59731,8	3%	59735,6	7%	59749,0	3%	59768,6	23%
24	5×100	60479	*60479,0*	100%	60434,3	17%	60428,4	0%	60452,9	40%	60458,9	23%
25	5×100	61091	61089,8	97%	61057,5	17%	61074,0	60%	61087,4	90%	*61091,0*	100%
26	5×100	58959	58937,7	3%	58922,8	0%	58922,0	3%	*58938,2*	27%	58930,4	10%
27	5×100	61538	*61537,5*	97%	61493,7	10%	61503,1	13%	61514,3	27%	61515,5	23%
28	5×100	61520	*61512,2*	73%	61468,0	7%	61477,3	17%	61504,7	50%	61507,2	60%
29	5×100	59453	*59453,0*	100%	59430,4	83%	59447,4	97%	*59453,0*	100%	*59453,0*	100%
30	5×100	59965	*59960,0*	0%	59950,5	0%	59950,9	0%	59958,8	0%	*59960,0*	3%
31	2×28	141278	*141278,0*	100%	*141278,0*	100%	*141278,0*	100%	*141278,0*	100%	*141278,0*	100%
32	2×28	130883	*130883,0*	100%	*130883,0*	100%	*130883,0*	100%	*130883,0*	100%	*130883,0*	100%
33	2×28	95677	95670,3	87%	*95677,0*	100%	*95677,0*	100%	*95677,0*	100%	*95677,0*	100%
34	2×28	119337	*119337,0*	100%	*119337,0*	100%	*119337,0*	100%	*119337,0*	100%	*119337,0*	100%
35	2×28	98796	*98796,0*	100%	*98796,0*	100%	*98796,0*	100%	*98796,0*	100%	*98796,0*	100%
36	2×28	130623	*130623,0*	100%	*130623,0*	100%	*130623,0*	100%	*130623,0*	100%	*130623,0*	100%
37	2×105	1095445	*1095442,9*	97%	1095384,1	3%	1095384,1	3%	1095388,3	10%	1095407,2	40%
38	2×105	624319	*624319,0*	100%	623564,6	77%	*624319,0*	100%	*624319,0*	100%	*624319,0*	100%

sağlamlık (robust) açısından başarısı dikkate almak için karşılaştırma *gap* değeri üzerinden yapılmış olup en iyi *gap* değeri italik yazı tipiyle yazılmıştır. Tablodan açıkça görüldüğü üzere AİYAK ilk 36 problem için en iyi çözümlere ulaşmıştır. Son 5 problemde ise BGWO en iyi çözümlere ulaşmıştır. BGWO Eniyi çözümlere bazı çalıştırmalarda ulaşabilmesine karşın sağlam/gürbüz bir performansa sahip olmadığı gözlenmektedir. Nitekim, *gap* cinsinden AİYAK sonuçları ile BGWO sonuçları arasında ihmal edilebilecek bir fark olduğu açıktır.

Tablo 4. İlk problem kümesi için istatistiksel karşılaştırmalar (The statistical comparison for the first problem set)

		BGWO		BFOA		HHS		QPSO	AİYAK
No	Sıra	p-değeri	Sıra	p-değeri	Sıra	p-değeri	Sıra	p-değeri	Sıra
1	1	*1,00E+00*	5	8,02E-05	4	9,77E-04	1	*1,00E+00*	1
2	2	3,44E-01	5	1,29E-04	4	8,73E-04	1	6,25E-02	3
3	3	5,89E-05	5	5,86E-06	4	2,77E-05	2	8,75E-03	1
4	2	7,63E-01	5	8,21E-06	4	1,44E-04	3	2,93E-04	1
5	1	1,20E-01	4	5,86E-03	5	7,35E-05	2	8,34E-01	3
6	1	3,67E-02	4	1,32E-05	6	5,10E-06	3	8,96E-01	2
7	1	2,50E-01	3	2,73E-06	5	9,63E-07	3	9,63E-07	2
8	3	1,01E-05	4	2,36E-06	5	2,25E-06	2	1,36E-05	1
9	2	3,13E-02	4	6,49E-02	5	3,69E-02	1	4,32E-08	3
10	1	1,01E-04	5	3,20E-03	4	3,34E-02	2	1,09E-01	3
11	3	1,39E-05	6	7,42E-02	2	1,48E-01	4	*1,00E+00*	1
12	2	2,14E-03	4	1,45E-05	5	1,23E-04	3	4,25E-01	1
13	3	2,97E-05	5	1,14E-05	4	4,23E-04	2	*1,00E+00*	1
14	1	4,88E-04	4	5,16E-06	5	3,72E-06	3	8,58E-04	2
15	1	*1,00E+00*	6	4,88E-04	5	6,25E-02	3	*1,00E+00*	1
16	1	*1,00E+00*	5	2,85E-04	4	1,22E-04	1	1,25E-01	1
17	1	*1,00E+00*	6	3,91E-03	4	2,50E-01	1	*1,00E+00*	1
18	3	1,72E-05	5	9,42E-04	4	8,17E-04	2	7,50E-01	1
19	3	5,44E-04	4	6,10E-04	5	7,41E-03	2	9,22E-01	1
20	1	3,99E-04	5	7,71E-05	4	1,82E-04	3	7,38E-01	2
21	1	*1,00E+00*	5	1,19E-05	4	1,56E-02	1	*1,00E+00*	1
22	3	2,96E-03	5	1,29E-04	6	7,43E-06	2	9,77E-06	1
23	1	1,68E-01	5	3,62E-04	4	4,31E-06	3	2,30E-05	2
24	1	1,78E-05	4	1,19E-04	5	1,12E-03	3	4,31E-01	2
25	2	*1,00E+00*	5	9,94E-06	4	5,73E-05	3	2,50E-01	1
26	2	1,32E-02	4	3,41E-02	5	2,45E-02	1	9,29E-02	3
27	1	1,17E-05	5	7,80E-03	4	5,10E-04	3	9,98E-01	2
28	1	8,37E-02	5	1,47E-05	4	4,96E-04	3	9,66E-01	2
29	1	*1,00E+00*	5	1,25E-01	4	*1,00E+00*	1	*1,00E+00*	1
30	1	*1,00E+00*	5	5,86E-03	4	1,56E-01	3	7,50E-01	1
31	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1
32	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1
33	5	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1
34	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1
35	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1
36	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1	*1,00E+00*	1
37	1	3,74E-05	4	4,88E-04	4	1,29E-02	3	3,42E-03	2
38	1	*1,00E+00*	5	1,56E-02	1	*1,00E+00*	1	*1,00E+00*	1
Ort:	1,63		4,13		3,76		2,03		1,50

Tablo 5. İkinci problem kümesi için yöntemlerin karşılaştırılması (1-30 Chu, 31-42 Gloover ve Kochenberger) (The comparison of methods for the second problem set (1-30 Chu, 31-42 Gloover-Kochenberger))

Problem			BGWO		BFAO		HHS		QPSO		AİYAK	
P,No	Boyut	Eniyi	Eniyi	Gap	Eniyi	Gap	Eniyi	Gap	Eniyi	Gap	Eniyi	Gap
1	30×500	116056	115814	0,82%	115759	1,10%	115759	1,01%	115288	1,46%	115813	*0,36%*
2	30×500	114810	114701	0,70%	114418	1,11%	114524	0,98%	114156	1,36%	114582	*0,33%*
3	30×500	116741	116531	0,80%	116329	1,17%	116472	1,09%	116087	1,47%	116454	*0,29%*
4	30×500	115354	115125	0,82%	115011	1,15%	115014	1,06%	114860	1,43%	115102	*0,31%*
5	30×500	116525	116312	0,74%	116098	1,11%	116103	1,08%	115742	1,50%	116270	*0,32%*
6	30×500	115741	115586	0,76%	115329	1,14%	115389	1,05%	115016	1,49%	115498	*0,35%*
7	30×500	114181	113939	0,77%	113930	1,13%	113759	1,00%	113491	1,39%	113834	*0,42%*
8	30×500	114403	114118	0,74%	113987	1,03%	114012	0,97%	113824	1,28%	114037	*0,42%*
9	30×500	115419	115153	0,74%	115023	1,12%	115007	1,05%	114568	1,48%	115050	*0,38%*
10	30×500	117116	116939	0,75%	116647	1,17%	116698	1,07%	116424	1,46%	116873	*0,32%*
11	30×500	218104	217938	0,35%	217803	0,55%	217891	0,49%	217496	1,01%	217930	*0,10%*
12	30×500	214648	214506	0,32%	214333	0,46%	214347	0,48%	213669	1,00%	214359	*0,17%*
13	30×500	215978	215817	0,33%	215662	0,46%	215629	0,43%	215089	1,02%	215766	*0,14%*
14	30×500	217910	217746	0,31%	217623	0,48%	217552	0,47%	217114	1,20%	217726	*0,16%*
15	30×500	215689	215495	0,32%	215248	0,51%	215355	0,45%	214751	1,13%	215484	*0,14%*
16	30×500	215919	215734	0,30%	215450	0,51%	215491	0,50%	215048	1,15%	215697	*0,17%*
17	30×500	215907	215761	0,33%	215673	0,46%	215648	0,44%	214951	0,97%	215798	*0,11%*
18	30×500	216542	216379	0,33%	216197	0,52%	216256	0,48%	215784	1,13%	216217	*0,19%*
19	30×500	217340	217251	0,33%	217170	0,46%	217132	0,44%	216666	0,99%	217208	*0,12%*
20	30×500	214739	214597	0,34%	214443	0,51%	214422	0,49%	213749	1,06%	214594	*0,11%*
21	30×500	301675	301627	0,18%	301440	0,29%	301449	0,28%	300962	0,51%	301627	*0,08%*
22	30×500	300055	299987	0,20%	299808	0,31%	299835	0,28%	299443	0,53%	299828	*0,10%*
23	30×500	305087	305028	0,20%	304859	0,31%	304884	0,29%	304695	0,46%	304921	*0,08%*
24	30×500	302032	301897	0,22%	301797	0,31%	301742	0,31%	301454	0,53%	301857	*0,10%*
25	30×500	304462	304411	0,20%	304146	0,31%	304287	0,28%	304021	0,52%	304350	*0,07%*
26	30×500	297012	296883	0,19%	296893	0,32%	296766	0,32%	296407	0,55%	296784	*0,10%*
27	30×500	303364	303232	0,19%	303131	0,29%	303156	0,28%	302729	0,55%	303139	*0,11%*
28	30×500	307007	306892	0,19%	306817	0,27%	306741	0,27%	306455	0,48%	306823	*0,10%*
29	30×500	303199	303077	0,20%	302950	0,29%	303007	0,27%	302590	0,53%	303032	*0,09%*
30	30×500	300572	300493	0,21%	300424	0,31%	300380	0,29%	299946	0,56%	300436	*0,06%*
31	15×100	3766	3766	0,47%	3766	0,60%	3766	0,60%	3759	0,51%	3763	*0,19%*
32	25×100	3958	3948	0,80%	3958	0,86%	3955	0,78%	3957	0,69%	3955	*0,26%*
33	25×150	5656	5647	0,50%	5643	0,68%	5647	0,70%	5640	0,72%	5643	*0,32%*
34	50×150	5767	5764	0,82%	5752	1,02%	5753	1,01%	5750	0,96%	5758	*0,27%*
35	25×200	7560	7552	0,40%	7542	0,68%	7544	0,70%	7538	0,75%	7549	*0,21%*
36	50×200	7677	7656	0,78%	7652	0,93%	7652	0,92%	7649	0,90%	7653	*0,42%*
37	25×500	19220	19213	*0,15%*	19163	0,63%	19136	0,75%	19122	0,83%	19194	0,17%
38	50×500	18806	18783	*0,32%*	18712	0,92%	18691	0,98%	18689	1,06%	18752	0,34%
39	25×1500	58087	58079	*0,05%*	57816	0,80%	57631	0,92%	57838	0,80%	58049	0,09%
40	50×1500	57295	57272	*0,10%*	56912	0,92%	56836	0,99%	57044	0,72%	57195	0,20%
41	100×2500	95237	95161	*0,14%*	94394	1,09%	94321	1,14%	94675	0,79%	95010	0,26%

Tablo 6. İkinci problem kümesi için istatistiksel karşılaştırmalar (The statistical comparison for the second problem set)

	BGWO		BFOA		HHS		QPSO	AİYAK
No	Sıra	p-değeri	Sıra	p-değeri	Sıra	p-değeri	Sıra	p-değeri	Sıra
1	2	0,734318532	5	4,73E-06	4	2,35E-06	6	1,73E-06	1
2	2	5,75E-06	5	1,73E-06	4	2,13E-06	6	1,73E-06	1
3	2	3,34E-01	5	1,73E-06	4	1,73E-06	6	1,73E-06	1
4	2	3,29E-01	5	1,73E-06	4	2,34E-06	6	1,73E-06	1
5	2	2,41E-04	5	2,35E-06	4	4,73E-06	6	1,73E-06	1
6	2	2,13E-06	5	8,46E-06	4	5,07E-02	6	1,73E-06	1
7	2	4,73E-06	5	1,92E-06	4	6,98E-06	6	1,73E-06	1
8	2	1,03E-03	5	3,52E-06	4	7,69E-06	6	1,73E-06	1
9	2	1,07E-03	5	1,73E-06	4	2,35E-06	6	1,73E-06	1
10	2	1,90E-02	5	1,92E-06	4	1,73E-06	6	1,73E-06	1
11	2	9,38E-04	5	1,73E-06	4	1,73E-06	6	1,73E-06	1
12	2	2,12E-06	4	5,22E-06	5	3,88E-06	6	1,73E-06	1
13	2	8,29E-01	5	3,18E-06	3	1,92E-06	6	1,73E-06	1
14	2	2,60E-06	5	1,24E-05	4	2,37E-05	6	1,73E-06	1
15	2	1,50E-01	5	2,35E-06	4	2,13E-06	6	1,73E-06	1
16	2	3,87E-06	5	3,88E-06	4	2,35E-06	6	1,73E-06	1
17	2	2,41E-01	5	1,92E-06	4	2,12E-06	6	1,73E-06	1
18	2	2,13E-06	5	1,04E-04	4	1,36E-04	6	1,73E-06	1
19	2	7,27E-03	5	1,24E-05	4	1,92E-06	6	1,73E-06	1
20	2	4,95E-02	5	3,88E-06	4	7,69E-06	6	1,73E-06	1
21	2	4,41E-04	5	1,92E-06	4	1,73E-06	6	1,73E-06	1
22	2	5,20E-06	5	4,29E-06	4	5,08E-05	6	1,73E-06	1
23	2	1,28E-02	5	2,35E-06	4	4,07E-05	6	1,73E-06	1
24	2	5,21E-06	4	3,51E-06	4	2,73E-06	6	1,73E-06	1
25	2	7,97E-01	5	3,51E-06	4	1,73E-06	6	1,73E-06	1
26	2	1,71E-06	4	1,73E-06	4	2,13E-06	6	1,73E-06	1
27	2	1,73E-06	5	5,78E-05	4	2,22E-04	6	1,73E-06	1
28	2	4,28E-06	4	3,11E-05	4	4,39E-03	6	1,73E-06	1
29	2	3,56E-05	5	5,47E-06	4	1,88E-05	6	1,73E-06	1
30	2	8,29E-01	5	1,73E-06	4	2,13E-06	6	1,73E-06	1
31	2	4,84E-01	4	5,43E-06	4	6,40E-06	3	8,14E-06	1
32	4	5,85E-04	5	8,42E-04	3	2,69E-02	2	1,65E-02	1
33	2	9,56E-06	3	1,70E-03	4	9,40E-04	5	2,42E-06	1
34	2	1,46E-03	5	6,29E-06	4	1,17E-05	3	1,90E-06	1
35	2	3,04E-04	4	1,72E-06	5	2,10E-06	6	1,71E-06	1
36	2	3,86E-02	6	7,90E-06	5	2,83E-03	3	1,71E-06	1
37	1	1,69E-06	4	1,73E-06	5	1,72E-06	6	1,73E-06	2
38	1	1,64E-06	4	1,73E-06	5	1,73E-06	6	1,73E-06	2
39	1	1,67E-06	4	1,73E-06	6	1,73E-06	4	1,73E-06	3
40	1	1,72E-06	5	1,73E-06	6	1,73E-06	4	1,73E-06	3
41	1	1,73E-06	5	1,73E-06	6	1,73E-06	4	1,73E-06	3
	1,93		4,76		4,22		5,51		1,20

İkinci problem kümesi için istatiksel analiz gerçekleştirildiğinde Tablo 6 elde edilmiştir. Tabloya göre AİYAK yöntemin daha iyi çözüm sunması istatistiksel olarak kanıtlanmaktadır. AİYAK 1,2 başarı sıralamasıyla en yakın rakibi olan BGWO'dan çok daha iyidir. Diğer yöntemlerin başarı sıralaması ise BGWO, HHS, BFOA, QPSO olarak sıralanır.

4. SONUÇLAR (CONCLUSIONS)

Bu çalışmada çok boyutlu sırt çantası probleminin, yapay arı kolonisi algoritması ile birden çok ikili operatörü adaptif kullanarak, çözümü önerilmiştir. Algoritma son zamanlarda geliştirilmiş ve başarısı ispat edilmiş üç adet ikili operatörü adaptif olarak kullanmaktadır. Ayrıca literatürde itibarı yüksek üç farklı adaptif operatör seçme yöntemi ön denemeler aracılığı ile yarıştırılmış ve en uygun adaptif yöntem olarak "adaptif takip" yöntemi seçilmiştir. Önerilen algoritmaya ait en iyi parametre yapılandırmasının belirlenmesi için ilk olarak parametre ayarlama deneysel çalışmaları gerçekleştirilmiştir. Bu parametrik çalışmalar sonucunda; kredi atama probleminde 25 pencere boyutuna ait ortalama ödül değerini tercih edilmiştir. Önerilen yöntemin başarısı ve literatürdeki dört farklı çalışma ile üç farklı problem kümesi üzerinden karşılaştırılmış ve başarının istatistiksel olarak anlamlılığı gösterilmiştir. Önerilen AİYAK algoritmasının literatürdeki diğer yöntemlerden daha başarılı ve sağlam sonuçlar verdiği gösterilmiştir. Algoritmanın farklı çalıştırmalar sonucunda sağlam/gürbüz olduğu da gösterilmiştir.

Sonraki çalışmalarda, probleme daha kaliteli çözümler üretilebilmesi için farklı adaptif yaklaşımlar ve farklı operatör sayıları denenecektir. Alternatif ve daha özgünce

geliştirilmiş makine öğrenmesi temelli adaptif operatör seçme yöntemleri üzerinde durulacaktır. Ayrıca önerilen çok operatörlü adaptif yaklaşım yapay ari algoritması dışındaki yapısal olarak daha sıkı geliştirilmiş sürü zekası algoritmalarına da (daha zor olması beklense de) denenebilir. Bir diğer alanda ise, onarma operatörlerinin de adaptif olarak kullanılması durumunda performans etkileri test edilebilir.

KAYNAKLAR (REFERENCES)

1. Fausto, F., Reyna-Orta, A., Cuevas, E., Andrade, Á. G., Perez-Cisneros, M., From ants to whales: metaheuristics for all tastes, Artificial Intelligence Review, 53 (1), 753-810, 2020.
2. Mirjalili, S., Lewis, A., The whale optimization algorithm, Advances in engineering software, 95, 51-67, 2016.
3. Whitley, D., A genetic algorithm tutorial, Statistics and computing, 4 (2), 65-85, 1994.
4. Kennedy, J., Eberhart, R., Particle swarm optimization. In Proceedings of ICNN'95-International Conference on Neural Networks, 4, 1942-1948,1995.
5. Karaboga, D., Basturk, B., On the performance of artificial bee colony (ABC) algorithm. Applied soft computing, 8 (1), 687-697, 2008.
6. Price, K., Storn, R. M., Lampinen, J. A., Differential evolution: a practical approach to global optimization, Springer Science & Business Media, 2006.
7. Hussain, K., Salleh, M. N. M., Cheng, S., Shi, Y., Metaheuristic research: a comprehensive survey. Artificial Intelligence Review, 52 (4), 2191-2233, 2019.
8. Garip Z , Çimen M., Boz A., Comparative performance analysis on parameter extraction of solar cell models using meta-heuristic algorithms, Journal of the Faculty of Engineering and Architecture of Gazi University, 36 (2), 1133-1144, 2021.
9. Siarry, P., Metaheuristics, Springer International Publishing, 2016.
10. Sergeyev, Y. D., Kvasov, D. E., Mukhametzhanov, M. S., On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Scientific reports, 8 (1), 1-9, 2018.
11. Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F., Rodríguez, A., A better balance in metaheuristic algorithms: Does it exist?. Swarm and Evolutionary Computation, 54, 100671. 2020
12. Karaboga, D., An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 200, 1-10, 2005.
13. Öztürk C., Hançer E., Karaboğa D., Automatic Clustering with Global Best Artificial Bee Colony Algorithm, Journal of the Faculty of Engineering and Architecture of Gazi University, 29 (4), 677-687, 2014.
14. Wang, H., Wang, W., Xiao, S., Cui, Z., Xu, M., Zhou, X. . Improving Artificial Bee Colony Algorithm Using a New Neighborhood Selection Mechanism. Information Sciences, 527, 227-240 2020.
15. Xue, Y., Jiang, J., Zhao, B., Ma, T., A self-adaptive artificial bee colony algorithm based on global best for global optimization. Soft Computing, 22 (9), 2935-2952, 2018.
16. Yurtkuran, A., Emel, E., An adaptive artificial bee colony algorithm for global optimization. Applied Mathematics and Computation, 271, 1004-1023, 2015.
17. Babaoglu, I., Artificial bee colony algorithm with distribution-based update rule. Applied Soft Computing, 34, 851-861, 2015.
18. Cengiz M., Çomak E., A solution of some commonly used optimization functions by a hybrid BFGS-PSO algorithm, Journal of the Faculty of Engineering and Architecture of Gazi University, 36 (2) , 925-938, 2021.
19. Gao, W., Liu, S., Huang, L., A global best artificial bee colony algorithm for global optimization. Journal of Computational and Applied Mathematics, 236 (11), 2741-2753, 2012.
20. Bansal, J. C., Joshi, S. K., Sharma, H., Modified global best artificial bee colony for constrained optimization problems. Computers & Electrical Engineering, 67, 365-382, 2018.
21. Gao, W., Liu, S., Improved artificial bee colony algorithm for global optimization. Information Processing Letters, 111 (17), 871-882, 2011.
22. Gao, W. F., Liu, S. Y., A modified artificial bee colony algorithm. Computers & Operations Research, 39 (3), 687-697, 2012.
23. Kiran, M. S., Hakli, H., Gunduz, M., Uguz, H., Artificial bee colony algorithm with variable search strategy for continuous optimization. Information Sciences, 300, 140-157, 2015.
24. Durgut, R., Yavuz, İ. B., Aydın, M. E., Küme Birleşimli Sırt Çantası Probleminin Adaptif Yapay Arı Kolonisi Algoritması ile Çözümü, Journal of Intelligent Systems: Theory and Applications, 4 (1), 43-54, 2021.
25. Cui, L., Li, G., Wang, X., Lin, Q., Chen, J., Lu, N., Lu, J., A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. Information Sciences, 417, 169-185, 2017.
26. Li, K., Fialho, A., Kwong, S., Zhang, Q., Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation, 18 (1), 114-130, 2013.
27. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M., Extreme value based adaptive operator selection. In International Conference on Parallel Problem Solving from Nature, Springer, Berlin, Heidelberg, 175-184, 2008.
28. Durgut, R., Aydin, M. E., Adaptive binary artificial bee colony algorithm, Applied Soft Computing, 101, 107054, 2021.
29. Pirkul, H., An integer programming model for the allocation of databases in a distributed computer system. European Journal of Operational Research, 26 (3), 401-411, 1986.

30. Martello, S., Knapsack problems: algorithms and computer implementations. Wiley-Interscience series in discrete mathematics and optimization, 1990.

31. Kellerer, H., Pferschy, U., Pisinger, D., Multidimensional knapsack problems, Knapsack problems, 235-283, Springer, Berlin, Heidelberg, 2004.

32. Engwall, M., Jerbrant, A., The resource allocation syndrome: the prime challenge of multi-project management?. International journal of project management, 21 (6), 403-409, 2003.

33. Shih, W., A branch and bound method for the multiconstraint zero-one knapsack problem. Journal of the Operational Research Society, 30 (4), 369-378, 1979.

34. Puchinger, J., Raidl, G. R., Pferschy, U., The multidimensional knapsack problem: Structure and algorithms. INFORMS Journal on Computing, 22 (2), 250-265, 2010.

35. Balev, S., Yanev, N., Fréville, A., Andonov, R., A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem. European Journal of Operational Research, 186 (1), 63-76, 2008.

36. Vimont, Y., Boussier, S., Vasquez, M., Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem. Journal of Combinatorial Optimization, 15 (2), 165-178, 2008.

37. Ozturk, C., Hancer, E., Karaboga, D., Dynamic clustering with improved binary artificial bee colony algorithm. Applied Soft Computing, 28, 69-80, 2015.

38. Jia, D., Duan, X., Khan, M. K., Binary Artificial Bee Colony optimization using bitwise operation. Computers & Industrial Engineering, 76, 360-365, 2014.

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

40. 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.

41. He, Y., Xie, H., Wong, T. L., Wang, X., A novel binary artificial bee colony algorithm for the set-union knapsack problem. Future Generation Computer Systems, 78, 77-86, 2018.

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

43. Xiang, W. L., Li, Y. Z., He, R. C., Gao, M. X., An, M. Q., A novel artificial bee colony algorithm based on the cosine similarity. Computers & Industrial Engineering, 115, 54-68, 2018.

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

45. Durgut, R., Improved binary artificial bee colony algorithm. arXiv preprint arXiv:2003.11641, 2020.

46. Thierens, D. Adaptive strategies for operator allocation, Parameter setting in evolutionary algorithms,77-90, Springer, Berlin, Heidelberg, 2007.

47. Hitomi, N., Selva, D., A classification and comparison of credit assignment strategies in multiobjective adaptive operator selection. IEEE Transactions on Evolutionary Computation, 21 (2), 294-314, 2016.

48. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M., Analyzing bandit-based adaptive operator selection mechanisms. Annals of Mathematics and Artificial Intelligence, 60 (1-2), 25-64, 2010.

49. Ezugwu, A. E., Adeleke, O. J., Akinyelu, A. A., Viriri, S., A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems. Neural Computing and Applications, 32 (10), 6207-6251, 2020.

50. Chu, P. C., Beasley, J. E., A genetic algorithm for the multidimensional knapsack problem. Journal of heuristics, 4 (1), 63-86, 1998.

51. Drake, J. H., Ozcan, E. and Burke, E. K., A Case Study of Controlling Crossover in a Selection Hyper-heuristic Framework using the Multidimensional Knapsack Problem. Evolutionary Computation, 24 (1), 113–141, 2016.

52. Queen Mary University of London, MKP Problem Instances, http://www.eecs.qmul.ac.uk/~jdrake/mkp.html, Erişim tarihi Haziran 10, 2020

53. Lai, X., Hao, J. K., Fu, Z. H., Yue, D., Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem. Expert Systems with Applications, 149, 113310, 2020.

54. Laabadi, S., Naimi, M., El Amri, H., Achchab, B. The 0/1 multidimensional knapsack problem and its variants: a survey of practical models and heuristic approaches. American Journal of Operations Research, 8 (05), 395, 2018

55. Glover, F., Kochenberger, G. A. Critical event tabu search for multidimensional knapsack problems. In Meta-heuristics, Springer, Boston, MA, 407-427, 1996.

56. Khemakhem, M., Haddar, B., Chebil, K., Hanafi, S., A filter-and-fan metaheuristic for the 0-1 multidimensional knapsack problem. International Journal of Applied Metaheuristic Computing (IJAMC), 3 (4), 43-63, 2012.

57. Drexl, A., A simulated annealing approach to the multiconstraint zero-one knapsack problem. Computing, 40 (1), 1-8, 1988.

58. Angelelli, E., Mansini, R., Speranza, M.G., Kernel search: A general heuristic for the multi-dimensional knapsack problem, Computers & Operations Research, 37 (11), 2017-2026, 2010.

59. Chih, M., Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem, Applied Soft Computing, 26, 378-389, 2015.

60. Lin, C. J., Chern, M. S., Chih, M., A binary particle swarm optimization based on the surrogate information with proportional acceleration coefficients for the 0-1 multidimensional knapsack problem, Journal of Industrial and Production Engineering, 33 (2), 77-102, 2016.

61. Drake, J. H., Özcan, E., Burke, E. K., A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional knapsack problem. Evolutionary computation, 24 (1), 113-141, 2016.

62. Lai, X., Hao, J. K., Glover, F., Lü, Z., A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. Information Sciences, 436, 282-301, 2018.

63. Al-Shihabi, S., Ólafsson, S., A hybrid of nested partition, binary ant system, and linear programming for the multidimensional knapsack problem. Computers & Operations Research, 37 (2), 247-255, 2010.

64. Zhang, B., Pan, Q. K., Zhang, X. L., Duan, P. Y., An effective hybrid harmony search-based algorithm for solving multidimensional knapsack problems. Applied Soft Computing, 29, 288-297, 2015.

65. Wang, L., Zheng, X. L., Wang, S. Y., A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. Knowledge-Based Systems, 48, 17-23, 2013.

66. García, J., Maureira, C., A KNN quantum cuckoo search algorithm applied to the multidimensional knapsack problem. Applied Soft Computing, 102, 107077, 2021.

67. García, J., Lalla-Ruiz, E., Voss, S., Droguett, E. L., Enhancing a machine learning binarization framework by perturbation operators: analysis on the multidimensional knapsack problem. International Journal of Machine Learning and Cybernetics, 1-20, 2020.

68. Garcia, J., Moraga, P., Valenzuela, M., Pinto, H., A db-scan hybrid algorithm: an application to the multidimensional knapsack problem. Mathematics, 8 (4), 507, 2020.

69. Luo, K., Zhao, Q., A binary grey wolf optimizer for the multidimensional knapsack problem. Applied Soft Computing, 83, 105645, 2019.