Araştırma Makalesi
Zeki Sistemler Teori ve Uygulamaları Dergisi 4(1) (2021) 43-54
DOI: 10.38016/jista.854584

# Küme Birleşimli Sırt Çantası Probleminin Adaptif Yapay Arı Kolonisi Algoritması ile Çözümü

Rafet DURGUT[1]*, İlim Betül YAVUZ[2], Mehmet Emin AYDIN[3]

[1]Karabük Üniversitesi, Bilgisayar Mühendisliği Bölümü, Karabük, Türkiye

[2]Karabük Üniversitesi, Bilgisayar Mühendisliği Bölümü, Karabük, Türkiye

[3]UWE Bristol, Dept. of Computer Science and Creative Technologies, Bristol, İngiltere

rafetdurgut@karabuk.edu.tr, 1828126016 @karabuk.edu.tr, Mehmet.aydin@uwe.ac.uk

**Öz**

Meta-sezgisel ve sürü zekâsı algoritmaları, NP-Zor optimizasyon problemlerine yaklaşık çözümler sunmak için uzun süredir kullanılmaktadır. Özellikle kombinatoryal ve ikili problemler söz konusu olduğunda, algoritmalar içerisine gömülü komşu çözüm üretmek için kullanılan operatör fonksiyonları, aramanın çeşitliliğine sınırlamalar getirirken algoritmaların başarısında önemli bir rol oynar. Bu tür sınırlamalardan kaçmak ve çeşitliliği iyileştirmek için, birden fazla operatörün tek bir operatör yerine bir seçim şeması yoluyla kullanılması tercih edilir. Daha önce farklı sürü zekâsı ve meta-sezgisel algoritmalarla çeşitli kombinatoryal problemleri çözmek için bir dizi operatör seçim şeması kullanılması daha yüksek etkinlik elde etmek için kullanılmıştır. Bu makalede, küme birleşimli sırt çantası problemleri, ilk kez, alternatif operatör seçim şemaları aracılığıyla seçilen birden fazla operatör içeren ikili bir yapay arı kolonisi algoritması ile çözülmüştür. Önerilen yöntem için farklı kredi atama yaklaşımları, farklı kayan pencere boyutları ve parametre konfigürasyonları test edilmiştir. Seçim şemalarının özellikleri kapsamlı olarak 30 kıyaslama problemi üzerinde incelenmiştir. Bu problem kümeleri için en iyi performans gösteren algoritma konfigürasyonu önerilmiştir. Çalışma, başarılı bir seçim şemasına sahip adaptif ikili yapay arı kolonisi algoritmasını sunmaktadır.

**Anahtar kelimeler:** Yapay Arı Kolonisi, Adaptif Operatör Seçimi, Küme Birleşimli Sırt Çantası Problemi

# Solving Set Union Knapsack Problems with Adaptive Binary Artificial Bee Colony

**Abstract**

Metaheuristic and swarm intelligence algorithms have been utilised to solve optimization problems with NP-Hard nature providing approximate solutions for a long time. Especially in the case of combinatorial and binary problems, operator functions embedded in the algorithms to generate neighboring solutions play a crucial role in the success of the algorithms while each operator imposes limitations upon the diversity of the search. In order to escape of such limitations and improve the diversity, multiple operators are preferred to use through a selection scheme instead of a single operator. However, the nature of selection scheme whereby operators are opted out also matters for higher efficiency, where a number of operator selection schemes have been used to solve various combinatorial problems with different swarm intelligence and metaheuristic algorithms before. In this paper, set union knapsack problems are, first time, solved with a binary artificial bee colony algorithm embedded with multiple operators selected through alternative operator selection schemes. Different credit assignment approaches, different sliding window lengths and parameter configurations are tested for the proposed method. The characteristics of the selection schemes are studied and the best performing one is suggested using a comprehensive experimentation over 30 benchmark problems. The study concludes a particular variant of binary artificial bee colony algorithm with a successful selection scheme.

**Keywords:** Artificial Bee Colony, Adaptive Operator Selection, Set Union Knapsack Problem.

* Sorumlu yazar. Rafet Durgut
E-posta adresi: rafetdurgut@karabuk.edu.tr

Alındı : 05 Ocak 2021
Revizyon : 08 Şubat 2021
Kabul : 15 Şubat 2021

## 1. Giriş (Introduction)

Sırt çantası problemi eldeki imkanlardan en yüksek faydayı sağlamak amacıyla çözülmesi gereken ve doğrusal zamanda çözülmesi mümkün olmayan bir optimizasyon problemdir. Günlük yaşamda, kaynak planlama (Bitran vd., 1981), kriptografi (Odlyzko vd., 1990), lojistik (Klamroth vd., 2000), üretim sistemleri (Bretthauter vd., 2002), gibi birçok alanda uygulama alanına sahiptir.

Sırt çantası uygulanacak probleminin yapısına göre 0/1 Sırt çantası (Moradi vd., 2021), çok boyutlu sırt çantası (García vd., 2020) veya küme birleşimli sırt çantası (Wei ve Hao, 2021) gibi farklı tipleri bulunmaktadır; Küme birleşimli sırt çantası (KBSÇ) problemi 0-1 sırt çantası probleminin özel bir halidir. Bu problemde mevcut elemanlar bulundukları kümeye göre ele alınmaktadır. KBSÇ probleminin şimdiye kadar çeşitli uygulama ve alanlarda değerli oluşu bilinmektedir (Goldschmidt vd., 1994).

Problemde $n$ tane eleman için $U = \{1,2,\dots,n\}$ elemanlar kümesini ve $m$ tane nesne için $S = \{1,2,..,m\}$ nesne kümesini ifade etmektedir. Öyle ki S kümesindeki her nesne $i \in S\ (i = 1,2,\dots,m)$ $U_i \subseteq U$ karşılık gelmektedir ve bir fayda değerine ($p_i > 0$) sahiptir.

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

$$W(A) = \sum_{u_j \in \bigcup_{i \in A} U_i} w_j \leq C, A \subseteq S \tag{2}$$

Amaç fonksiyonu Eşitlik (1)'de belirtilmiş olup, bu fonksiyonun amacı maksimum faydayı sağlayacak en iyi $A \subseteq S$ alt kümesini ($S^*$) belirlemektir. Eşitlik (2)'deki $w_j$ değişkeni elemanın ağırlığını, $C$ ise sırt çantasının kapasite sınırını ifade etmektedir.

Goldschmidt vd. (1994) KBSÇ probleminin kesin çözümü için hipergraf tabanlı dinamik programlama algoritmasının kullanılmasını önermiştir. Fakat bu algoritma düşük boyutlu problemler için hızlı çözüm önermesine karşın, problem boyutu arttıkça doğrusal zamanda bir çözüm sunamamaktadır. Benzer şekilde, Arulselvan (2014) aç gözlü bir yakınsama algoritması önermiştir.

Metasezgisel algoritmalar, doğrusal zamanda çözülemeyen problemlere uygulanabilecek alternatif ve güçlü bir seçimdir. Doğadan esinlenilerek geliştirilen bu algoritmalar içerisinde; Genetik Algoritma (Holland, 1975, Lin vd., 2020), Parçacık Sürüsü Optimizasyonu (Russel ve Eberhart, 1995, Bansal, 2019) veya Yapay Arı Kolonisi (Karaboğa ve Baştürk, 2008, Xiang vd., 2021) gibi oldukça başarılı ve popüler yöntemler bulunmaktadır. Popülasyon tabanlı olan bu yaklaşımlarda her bir çözüm ilgili bireye atanır ve komşu çözümlerden faydalanılarak arama uzayındaki en iyi çözümü sunan noktaya ulaşılmaya çalışılır.

Yapay arı kolonisi algoritması Karaboğa (2005) tarafından bal arılarının yiyecek bulma davranışından esinlenilerek sürekli optimizasyon problemlerine çözüm bulmak için geliştirilmiştir (Wang vd, 2020). Yöntem kullandığı eşitlikler gereği ikili optimizasyon problemlerine doğrudan uygulanamamaktadır. Bu sebeple, ikili optimizasyon problemlerine uygulanabilmesi için çeşitli düzenlemeler gerekmektedir (Lin vd., 2020). Bu düzenlemeler komşu çözüm üretilmesi için yapılmaktadır. Kiran vd. (2013) tarafından sunulan mantıksal özel veya kapısı kullanan binABC algoritması, Kashan vd. (2012) tarafından sunulan disABC algoritması gibi çeşitli komşu çözüm üretme mekanizmaları bulunmaktadır. Durgut (2020) ise, binABC algoritmasının yakınsama hızını arttırmak için bazı düzenlemeler sunmuştur.

Yapay arı kolonisi yaklaşımı ve benzeri sürü zekâsına dayalı metasezgisel optimizasyon algoritmaları, arama sürecinde ulaştıkları çözümleri geliştirerek (sömürü fazı) daha iyi çözümlere ulaşmaya çalışmakta veya arama bölgesinin farklı noktalarına ulaşarak (keşif fazı) yeni başarılı çözümlere ulaşmaya çalışmaktadır. Bahsi geçen iki faz (sömürü, keşif) arasında denge sağlanması oldukça zor olmakla beraber, bu denge yerel minimuma takılma ya da hızlı yakınsayamama probleminin oluşmaması için önemlidir (Arani vd., 2013). Bu tip problemlerin oluşma durumunu en aza indirmek, yerel minimuma takılmadan en iyi çözüme hızlı bir şekilde ulaşabilmek amacıyla adaptif operatör seçimi yaklaşımı önerilmektedir (Fialho, 2010).

Bu yaklaşımda komşu çözüm üretimi aşamasında bir operatör yerine birden çok operatör içeren operatör havuzu kullanılması öne çıkarılmıştır. Temel fikir operatörlerin birbirlerini tamamlayıcı davranış gösterebilmeleri beklentisidir. Burada tamamlayıcılık bir kritere dayandırılarak tarif edilmektedir. Başarılı operatörlerin seçim şansının arttırılması sayesinde operatörlerin birbirlerini tamamlayıcı davranabilecekleri ve böylece en iyi çözüme hızlı şekilde ulaşılması mümkün olabilecektir. Bu amaçla bu çalışma yapılmış ve tatmin edici sonuçlara ulaşılmıştır.

Bu çalışmada, küme birleşimli sırt çantası problemine kaliteli çözümler sunulabilmesi için adaptif yapay arı kolonisi yaklaşımı önerilmiştir. Literatür incelendiğinde, KBSÇ probleminin çözümü için kullanılmış olan Metasezgisel algoritmaların tüm süreci tek bir operatör ile yürüttüğü görülmüştür (He, 2018, Özsoydan vd., 2019). Önerilen yaklaşım ise, üç farklı komşuluk operatörü ve üç farklı operatör seçim yöntemi için çalıştırılmış ve en iyi konfigürasyon elde edilmiştir. Elde edilen sonuçlar literatürdeki diğer başarılı yöntemler ile karşılaştırılmış ve var olan yöntemlerden daha kaliteli çözümler üretilmiştir. Sonuç olarak bu çalışmanın katkısı, KBSÇ probleminin çözümü için literatürdeki çalışmalardan daha kaliteli çözümler üretmesi olarak görülmektedir.

## 2. YAPAY ARI KOLONİSİ (Artificial Bee Colony)

Sürü zekasına dayalı bir optimizasyon algoritması olan Yapay Arı Kolonisi (YAK) Algoritması, bal arılarının yiyecek arama/bulma davranışlarından esinlenilerek Karaboğa (2005) tarafından geliştirilmiştir.

Yapay Arı Kolonisi algoritmasının esinlendiği doğada arılar besin arama davranışlarında işçi, gözcü ve kâşif olmak üzere farklı görevlere sahiptirler. Arıların besin arama davranışların geçtiği aşamalar, başka bir deyişle YAK çalışma adımları Şekil 1'de verilmiştir.

**Algorithm 1** Yapay Arı Kolonisi

1: Başlangıç popülasyonunu oluştur.
2: Durdurma kriteri sağlanana kadar
3: Eşitlik 4'ü kullanarak komşu çözüm üret.
4: Seçili çözüm ile aday çözüm arasında aç gözlü seçim işlemi yap.
5: Her besin kaynağı için olasılık değerini hesapla.
6: Olasılık değerine göre besin kaynağı seç.
7: Eşitlik 4'ü kullanarak komşu çözüm üret.
8: Seçili çözüm ile aday çözüm arasında aç gözlü seçim işlemi yap.
9: Tükenen besin kaynağı var ise yeni geçerli çözüm ile değiştir.
10: Şimdiye kadar bulunan en iyi çözümü sakla

**Şekil 1.** YAK çalışma adımları (ABC Working Steps)

Başlangıç olarak yiyecek kaynaklarının üretilmesi için Kâşif arılar yiyecek kaynaklarını rastgele aramaya başlarlar. Yöntem içerisinde ise, rastgele geçerli çözüm atanması Eşitlik (3)'e göre yapılmaktadır.

$$x_{ij} = x_{ij}^{min} + rand(0,1)(x_{ij}^{max} - x_{ij}^{min}) \quad (3)$$

Eşitlik (3)'te, $i.$ besin kaynağının $j.$ boyutuna (parametresine) ait rastgele değer ($x_{ij}$) verilen sınır ($x_{ij}^{max} - x_{ij}^{min}$) aralığında rastgele olarak belirlenir. Bütün besinlerin tüm boyutlarına denklemden elde edilmiş değerler atanarak ilk çözümler (kaynaklar) oluşturulur.

İşçi arı fazında, arılar mevcut yiyecek kaynağının komşuluğunda yeni aday çözüm oluşturulur. Yeni oluşturulan çözümün kalitesi eskisinden daha iyi ise aç gözlü seçim uygulanarak bu çözüm ile güncellenir. Böylelikle işçi arı fazında her arı üzerinde çalıştığı besin kaynağını geliştirmektedir.

$$V_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \quad (4)$$

Eşitlik (4)'te her bir arı komşu bir arıdan aldığı bilgi dahilinde bu arının komşuluğunda denklem ile yeni bir çözüm üretir ve bu sayede besin kaynağını geliştirir. $V_{ij}$ değeri, i. arının j. boyutu için üretilmiş yeni değerdir. Bu değer, mevcut kaynağın komşuluğunda rastgele başka bir kaynağın yine aynı boyut bilgisine göre ($x_{kj}$) güncellenmektedir. $\varphi$ değeri hangi ölçüde ilerleneceğini ifade eden katsayıdır ve [-1,1] aralığında rastgele üretilmektedir.

$$F_i = \begin{cases} -\frac{1}{1+fi}, & fi \geq 0 \\ 1 + abs(fi), & fi < 0 \end{cases} \quad (5)$$

Bulunan kaynağın kalitesi, uygunluk fonksiyonu (fi) sonucunda hesaplanır. Kalite değerinin sıfırdan büyük ve eşit ya da küçük olmasına göre bu kaynağa Eşitlik (5)'te bulunan denklemde hesaplanan uygunluk değeri atanmış olur.

Gözcü arı fazında, arılar tüm görevli arıların elde ettikleri kaynak bilgilerini toplarlar. Bu bilgiler ışığında gözcü arının en iyi besin kaynağını seçebilmesi için kaynaktaki nektar miktarı ile orantılı olarak Eşitlik (6)'da belirtildiği gibi bir olasılık değeri belirlenir. Bu sayede gözcü arı aşamasında geliştirilmesi hedeflenen besin kaynakları belirlenmiş olur.

$$p_i = \frac{F_i}{\sum_{j=i}^{SN} F_i} \quad (6)$$

Kaynakların kalitesine göre bulunmuş olan uygunluk değerlerinin ardından bu kaynakların hangisinin geliştirileceği tüm kaynaklar baz alınarak hesaplanan olasılık değerine bağlıdır. Her bir çözümün olasılık değeri Eşitlik 6'da elde edilen değere göre belirlenmektedir. Bu olasılık değerlerine göre çözümler üzerinde güncelleme test edilir. Kâşif arı fazında, limit değerine ulaşan güncellemeye rağmen iyileşmeyen ilk çözüm popülasyondan atılarak, yerine rastgele ve geçerli bir çözüm dahil edilir. Bu şekilde algoritma ilk iterasyonunu tamamlayıp tekrar işçi arı aşamasına dönmektedir. Algoritma durdurma kriteri sağlanana kadar devam etmektedir.

## 3. İKİLİ YAPAY ARI KOLONİSİ (Binary Artificial Bee Colony)

Yapay arı kolonisi algoritması sürekli optimizasyon problemleri çözme amacıyla geliştirildiği için, ikili optimizasyon problemlerine doğrudan uygulanamamaktadır. Uygulanabilmesi için bazı düzenlemeler yapılmalıdır. Bu düzenlemeler iki sınıfa ayrılabilir; bunlardan biri probleme uygulama aşamasında sürekli karar değişkenlerinin ikili uzaya yerleştirilmesi (Kiran vd., 2015), diğeri ise karar değişkenlerinin ikili formda (Kiran vd., 2013) olmasıdır. İlk sınıftaki düzenleme kullanıldığında yöntemin sürekli uzayda aramaya devam eder, fakat amaç fonksiyonu içerisinde sürekli uzaydan ikili uzaya haritalama fonksiyonu gereklidir. İkinci sınıftaki düzenlemeler de ise, genellikle mantıksal karşılaştırma ve ifadeler kullanılmaktadır.

Bu çalışmada literatürde sıklıkla kullanılan binABC (Kirav vd., 2013), disABC (Kashan vd., 2013) ve yeni önerilmiş olan ibinABC (Durgut, 2020) operatörleri adaptif bir mekanizma içerisinde birlikte kullanılmaktadır.

### *3.1. BinABC Algoritması (BinABC Algorithm).*

Çözüm uzayı ikili yapıda olan optimizasyon problemleri için geliştirilmiş XOR tabanlı arı kolonisi algoritmasıdır. Bu algoritma Kiran vd. tarafından "Kapasite tesis yerleştirme problemi" çözümü için geliştirilmiştir (Kirav vd., 2013). YAK algoritmasının ikili yapıya uygun hale gelmesi için Eşitlik (3) ve Eşitlik (4)'ün uygun formata dönüştürülmesi gerekmektedir.

$$X_{i,j} = \begin{cases} 0, & r_{i,j} < p \\ 1, & r_{i,j} \geq p \end{cases} \quad (7)$$

Diğer yaklaşımlarda olduğu gibi bu yöntemde de başlangıçta kullanılacak arı popülasyonu için üretilen rastgele çözümler, Eşitlik (3)'ün Eşitlik (7) olarak güncellenmesiyle oluşturulmaktadır.

$$V_i^j = X_i^j \oplus [\varphi(X_i^j \oplus X_k^j)] \quad (8)$$

İlk arı popülasyonu oluşturulduktan sonra, işçi ve gözcü arı pozisyonlarının hesaplandığı aday çözümü ifade eden Eşitlik (4), Eşitlik (8)'deki gibi güncellenir. Eşitlik (8)'de $V_i^j$ , i. aday çözümün j. boyutunu temsil eder. $X_i^j$ i. işçi arının j. boyutu, $X_i^j$ k. işçi arının j. boyutu, $\oplus$ XOR mantıksal operatörü, φ ise %50 olasılıkla mantıksal değil kapısını ifade eder. Eğer φ 0,5'ten küçükse sonuç $(X_i^j \oplus X_k^j)$ ifadesinin terslenmiş halidir, büyük ve eşitse sonuç terslenmemiştir.

### *3.2. DisABC Algoritması (DisABC Algorithm)*

ABC nin diğer ikili versiyonu olan DisABC algoritması Kashan vd. tarafından önerilmiştir (Kashan vd., 2012). Bu algoritmada YAK algoritmasının yapısal durumu baz alınarak işçi arı $(X_i)$ diğer bir deyişle seçili çözüm ve komşu arı $(X_k)$ arasındaki farklılığın ölçülmesi önemlidir. Bu noktada, Eşitlik (9)'da Jaccard benzerlik katsayısı bulunarak 1'den çıkarılır, böylelikle farklılık bulunur.

$$\text{Dissimilarity}\,(X_i, X_k) = 1 - \text{Similarity}\,(X_i, X_k) \quad (9)$$

$$\text{Dissimilarity}\,(X_i, X_k) = 1 - \frac{m_{11}}{m_{01} + m_{10} + m_{11}} \quad (10)$$

Eşitlik (10)'da, $m_{01}$ değeri $X_{i,j}$=0 ve $X_{k,j}$ =1 olduğu durumdaki bitlerin sayısını, $m_{10}$ değeri $X_{i,j}$=1 ve $X_{k,j}$ = 0 olduğu durumdaki bitlerin sayısını,$m_{11}$ değeri $X_{i,j}$=1 ve $X_{k,j}$ =1 olduğu durumdaki bitlerin sayısını temsil eder.

Eşitlik (11) için, Aday çözüm: $V_i$ , pozitif ölçek faktörü: φ

$$\text{Dissimilarity}\,(V_i, X_i) \approx \varphi \text{ x Dissimilarity}\,(X_i, X_k) \quad (11)$$

Eşitlik (12) için $n_1$ ve $n_0$ katsayıları sırasıyla $X_i$ binary (ikili) vektördeki 1 ve 0' ların sayısını temsil eder. Model sonucunda aday çözüm ve seçili çözüm arasındaki farklılık en aza indirgenmeye çalışılmaktadır. Seçili çözümde 1 olan hanelerden (bitlerden) seçilir, aday çözüme aktarılır ve diğer bitler sıfırlanır.

$$M_{11} + M_{01} = n_1,\ M_{10} \leq n_0,$$

$M_{01}, M_{10}, M_{11} \geq 0$ *ve tam sayı* olmak üzere;

$$min|(1 - \frac{M_{11}}{M_{01}+M_{10}+M_{11}}) - \varphi(1 - \frac{m_{11}}{m_{01}+m_{10}+m_{11}})| \quad (12)$$

### *3.3. IbinABC Algoritması (IbinABC Algorithm)*

Durgut (2020), binABC operatörünü iyileştirerek ibinABC algoritmasını önermiştir. ibinABC algoritması iki yeni düzenleme içermektedir. Bunlardan ilki komşuluk operatörünün birden fazla bit üzerinde uygulanmasıdır. Operatörün uygulanacağı bit sayısı Eşitlik (13)'e göre belirlenmektedir.

$$d_t = rast[0, a] + 0.1D * e^{-(t/t_{max})} + 1 \quad (13)$$

Eşitlik 10'da a 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 φ değişkenin seçili $(X_i)$ ve komsu $(X_k)$ çözümün uygunluk değerine göre belirlenmesidir. binABC algoritmasında φ değeri rastgele olarak belirlenmektedir. Böyle bir durumda, iki çözümün uygunluk değerlerinin bir önemi yoktur. ibinABC'de ise eğer komsu çözüm daha iyi ise komşuluk operatöründe komsu çözümün ilgili hanesi (biti) işleme daha fazla etkili olacak şekilde belirlenmektedir. Aksi durumda ise, φ iterasyon değerine göre Eşitlik (14) 'teki gibi belirlenmektedir.

$$\varphi = \begin{cases} \varphi_{max} - \frac{(\varphi_{max}-\varphi_{min})}{\varphi_{max}} t, & f(X_i) < f(X_k) \\ 0, & aksi\ halde \end{cases} \quad (14)$$

## 4. Adaptif İkili Yapay Arı Kolonisi (Adaptive Binary Artificial Bee Colony)

İkili yapay arı kolonisi içerisine adaptif bir yapı eklemek için iki temel problemi yanıtlamak gerekmektedir. Bunlardan ilki operatörlerin başarısının nasıl değerlendirileceği ve atanacak kredinin nasıl belirleneceğidir. İkinci problem ise, operatörlerin mevcut kredilere göre nasıl seçileceğidir. Her iki problemde kendi içerisinde farklı sorunlar barındırmaktadır.

Bir operatöre kredi değeri atanması için ilk olarak operatörün önceki uygulamalardaki başarısı dikkate alınmalıdır. Üretilen komşu çözümlerin amaç değerine göre veya iyileşme durumuna göre ödül atanmalıdır. Bu çalışmada ödül değeri Eşitlik (15)'e göre belirlenmektedir.

$$ödül = \frac{PD}{GB}(f(x') - f(x)) \quad (15)$$

Burada, PD problem boyutunu, GB o ana kadar elde edilmiş en iyi çözümün amaç fonksiyon değerini, x' aday çözümü, x ise mevcut çözümü ifade etmektedir. Her bir komşu çözüm üretilmesi sonucunda ödül değeri hesaplanmakta ve eğer pozitif bir değer elde edilmiş ise dikkate alınmaktadır. Her iterasyon sonrasında operatörlerin toplam ödül değerleri belirlenmektedir. Ardından bu ödül değerlerine göre operatörlere kredi değeri ataması yapılmaktadır. Bu noktada, o iterasyondaki elde edilen ödül değerleri kullanılabileceği gibi belirli sayıda iterasyondan elde edilecek ödüller de dikkate alınabilir. Önceki çalışmalarımızda (Durgut ve Aydın, 2020) görüldüğü üzere anlık ödül kullanımının yerine ortalama ve en yüksek ödül değerlerinin kredi atamasında kullanılması daha faydalı sonuçlar üretmektedir.

Her bir operatöre ait kredi değerleri belirlendikten sonra operatör seçimi için kullanılabilecek farklı yaklaşımlar mevcuttur. Bunlardan ilki Probability Matching (PM) yaklaşımıdır. Bu yaklaşımda operatörlerin seçilme olasılığı kredi değerlerine göre dağıtılır. Eşitlik (16) kullanılarak her bir olasılığın o iterasyondaki seçilme olasılığı belirlenir.

$$p_{i,t} = p_{min} + (1 - (K-1) * p_{min}) \frac{Kredi_{i,t}}{\sum_{j=1}^{K} Kredi_{j,t}} \quad (16)$$

Eşitlik (16)'daki K değişkeni kullanılacak operatör sayısını, $p_{min}$ en kötü operatöre atanacak minimum olasılık değerini, $Kredi_i$ ise i. operatörün kredi değerini ifade etmektedir. Bir diğer operatör seçimi yöntemi olan Adaptive Pursuit (AP) ise "kazanan tümünü alır" prensibini kullanır. En iyi operatör en yüksek olasılık değeri ile güncellenirken diğer operatörler minimum olasılığa sahip olur. Yöntemin olasılık atama eşitliği Eşitlik (17)'de verilmiştir.

$$p_{i,t+1} = \begin{cases} p_{i,t} + \beta(p_{max} - p_{i,t}), & eğer\ i_t = i_t^* \\ p_{i,t} + \beta(p_{min} - p_{i,t}), & aksi\ durumda \end{cases} \quad (17)$$

Eşitlik (17)'de $\beta$ uyarlama katsayısı, $i_t^*$ ise en yüksek krediye sahip operatördür. Bu iki yaklaşımda operatörlerin kullanım sayıları ile ilgilenmemektedir. Bu sebeple az kullanılan operatörlere öncelik tanımamaktadır. Upper Confidence Bound (UCB) yaklaşımında ise az fırsat bulan operatörlere daha fazla fırsat sunmak amacıyla kullanılmaktadır. Yöntem Eşitlik (18)'e göre operatörlere seçim olasılığı atamaktadır.

$$p_{i,t+1} = \begin{cases} 1 - (K-1)p_{min}, & eğer\ i_t = i_t^* \\ p_{min}, & aksi\ durumda \end{cases} \quad (18)$$

$i_t^*$ ise Eşitlik (19)'a göre belirlenmektedir. Burada C ayarlama katsayı n ise seçilme sayılarıdır.

$$i_t^* = \underset{i=1,..,K}{\operatorname{argmax}} \{kredi_{i,t} + C \sqrt{\frac{2log\sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}\} \quad (19)$$

Operatörlere olasılık değeri atanmasından sonra rulet tekerine seçim yöntemi uygulanarak, operatör seçim işlemi gerçekleştirilmektedir. Çalışmada kullanılan algoritmanın çalışma adımları Şekil 2'de verilmiştir.

**Algorithm 2** Adaptif Yapay Arı Kolonisi

```
Başlangıç popülasyonunu oluştur.
Durdurma kriteri sağlanana kadar
    Kullanılacak operatörleri seç ve Toplam ödülleri sıfırla.
    Operatörü kullanarak komşu çözüm üret.
    Seçili çözüm ile aday çözüm arasında aç gözlü seçim işlemi yap.
    Ödül hesapla. Eğer ödül değeri pozitif ise toplam ödüle ekle.
    Her besin kaynağı için olasılık değerini hesapla.
    Olasılık değerine göre besin kaynağı seç.
    Kullanılacak operatörleri seç ve Toplam ödülleri sıfırla.
    Operatörü kullanarak komşu çözüm üret.
    Seçili çözüm ile aday çözüm arasında aç gözlü seçim işlemi yap.
    Ödül hesapla. Eğer ödül değeri pozitif ise toplam ödüle ekle.
    Tükenen besin kaynağı var ise yeni geçerli çözüm ile değiştir.
    Operatör kredilerini güncelle.
    Şimdiye kadar bulunan en iyi çözümü sakla
```

**Şekil 2.** Adaptif YAK çalışma adımları (Adaptive ABC working steps)

## 5. Deneysel Çalışmalar (Experimental Studies)

KBSÇ problemi kapasite kısıtlı optimizasyon problemi olduğu için geçersiz çözümler mevcuttur. YAK içerisinde geçersiz çözümlere izin verilmediğinden dolayı, çözümler üzerinde onarma işlemi yapılması gerekmektedir. He vd. (2018) önerdikleri çalışmada aç gözlü bir onarma yaklaşımı

uygulamaktadır. Bu yaklaşımda elemanlar küme içerisindeki frekanslarına ve fayda değerlerine göre sıralanır. Ardından kapasite sınırını aşan en değersiz elemanlar çantadan çıkarılır. Eğer mevcut kapasitede hala eleman eklenebilecek yer mevcut ise, en yüksek fayda sağlayacak elemanlar çanta içerisine eklenir. Daha detaylı bilgi için (He vd., 2018), (Özsoydan, Bakyasoğlu, 2019) ve (Özsoydan, 2019) çalışmaları incelenebilir.

Yöntemlerin başarılarının karşılaştırılabilmesi için He vd. (2018) 30 problem örneği içeren veri kümesi oluşturmuşlardır. Tüm problem örnekleri *m_n_x_y* formatına göre oluşturulmuştur. *m* ve *n* nesne ve eleman sayılarını ifade ederken *x* ve *y* elemanların yoğunluğunu ve toplam ağırlığın çanta kapasitesine oranını ifade etmektedir. Veri kümesini, nesne ve eleman sayısına göre üç farklı gruba ayrılmış olup sayıları 100'den 500'e kadar artmaktadır. Kullanılan örnek veri kümesi Tablo 1'de verilmiştir.

**Tablo 1.** Kullanılan problem örnekleri. (The problem instances)

| Grup 1 ($m > n$) | | Grup 2 ($m = n$) | | Grup 3 ($m < n$) | |
|---|---|---|---|---|---|
| Örnek No | Örnek Adı | Örnek No | Örnek Adı | Örnek No | Örnek Adı |
| G1_1 | sukp 100_85_0.1_0.75 | G2_1 | sukp 100_100_0.1_0.75 | G3_1 | sukp 85_100_0.1_0.75 |
| G1_2 | sukp 100_85_0.15_0.85 | G2_2 | sukp 100_100_0.15_0.85 | G3_2 | sukp 85_100_0.15_0.85 |
| G1_3 | sukp 200_185_0.1_0.75 | G2_3 | sukp 200_200_0.1_0.75 | G3_3 | sukp 185_200_0.1_0.75 |
| G1_4 | sukp 200_185_0.15_0.85 | G2_4 | sukp 200_200_0.15_0.85 | G3_4 | sukp 185_200_0.15_0.85 |
| G1_5 | sukp 300_285_0.1_0.75 | G2_5 | sukp 300_300_0.1_0.75 | G3_5 | sukp 285_300_0.1_0.75 |
| G1_6 | sukp 300_285_0.15_0.85 | G2_6 | sukp 300_300_0.15_0.85 | G3_6 | sukp 285_300_0.15_0.85 |
| G1_7 | sukp 400_385_0.1_0.75 | G2_7 | sukp 400_400_0.1_0.75 | G3_7 | sukp 385_400_0.1_0.75 |
| G1_8 | sukp 400_385_0.15_0.85 | G2_8 | sukp 400_400_0.15_0.85 | G3_8 | sukp 385_400_0.15_0.85 |
| G1_9 | sukp 500_485_0.1_0.75 | G2_9 | sukp 500_500_0.1_0.75 | G3_9 | sukp 485_500_0.1_0.75 |
| G1_10 | sukp 500_485_0.15_0.85 | G2_10 | sukp 500_500_0.15_0.85 | G3_10 | sukp 485_500_0.15_0.85 |

Yöntem üç farklı adaptif operatör seçimi (Probability Matching (PM), Adaptive Pursuit (AP) ve Upper Confidence Bound (UCB)) ile test edilmiştir. Ödül mekanizması olarak pencere boyunca ortalama ve en yüksek ödül miktarları karşılaştırılmıştır. Pencere boyutu olarak ise küçük (5) ve orta (25) değerleri test edilmiş olup, bu kavram yöntem içerisinde son 5 ve 25 iterasyon boyunca elde edilen ödül değerlerini temsil etmektedir. Minimum seçilme olasılığı ($P_{min}$) ise 0.1 ve 0.2 değerleri için test edilmiştir. Yöntem parametresi olan *Alpha* için ise 0.1, 0.5 ve 0.9 değerleri kullanılmıştır.

Parametre ayarlama için literatürdeki çalışmalarda olduğu gibi G2_6 problem örneği kullanılmıştır. Bu fazda 30 ve diğer çalıştırmalarda için algoritma 100 farklı kez çalıştırılmış ve sonuçlar tablo halinde görselleştirilmiştir. Popülasyon boyutu 20 olarak seçilmiştir. Maksimum iterasyon sayısının belirlenmesi için ise önceki çalışmalar referans alınarak eleman veya nesne sayısına göre büyük olan belirlenmiştir.

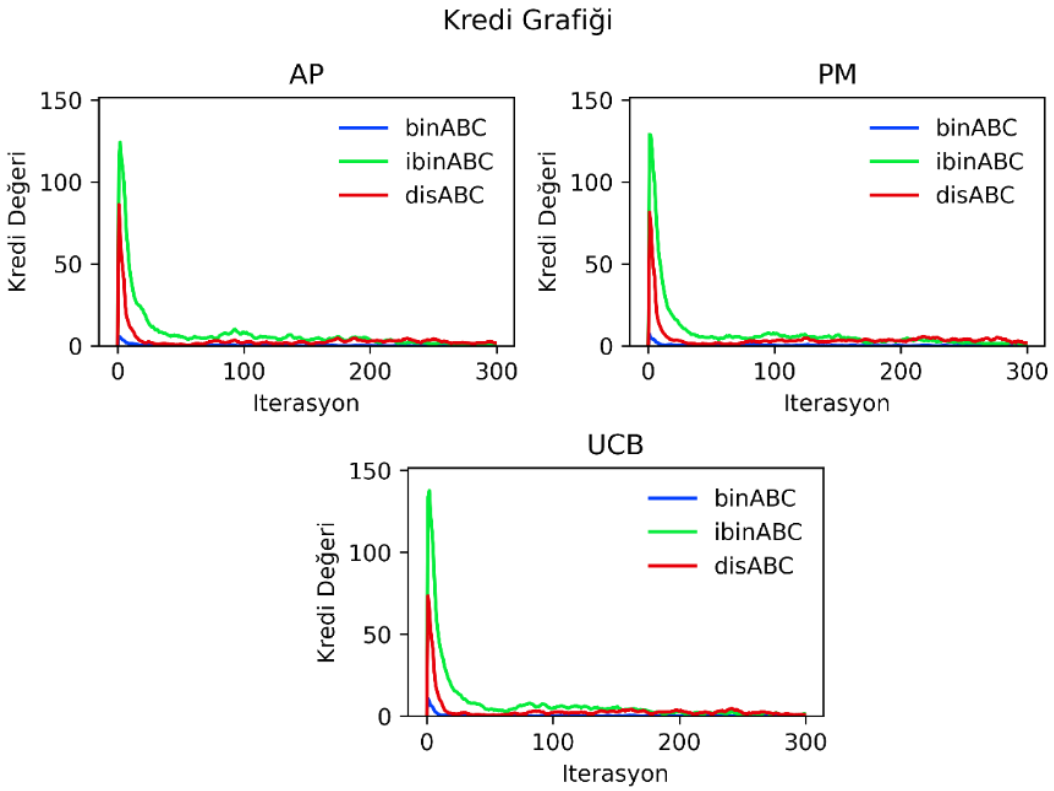
Tablo 2'de üç yöntem üzerinden parametre ayarlama için elde edilen sonuçlar görülmektedir. AP için en iyi konfigürasyonların; ortalama ödül, pencere uzunluğu için 5, minimum olasılık değeri için 0.2 ve *Alpha* katsayısı için 0.9 değerleri olduğu görülmektedir. PM için ise AP'den farklı olarak pencere uzunluğunun 5 olduğu görülmektedir. UCB ise diğer yöntemlerden farklı olarak daha küçük *Alpha* değeri için en başarılı sonuçlarını üretmiştir. Bu üç yöntem arasında en iyi ortalama değer PM ile üretildiği için sonraki deneylerde bu yöntem kullanılmıştır.

Şekil 3'te yöntemlerin (PM için elde edilmiş en iyi konfigürasyon için) zamana bağlı olarak verilmiş kredilerin ortalama değerleri ve Şekil 4'te ödül değerleri verilmiştir. Başlangıçta daha çok başarılı çözüm üretilebildiği için, ödül değerleri ve buna bağlı olarak kredi değerleri de yüksektir. Fakat yeni üretilen çözümlerin iyileşme oranları ve sayıları azalmaya başladığında ödül değeri ve dolayısıyla kredi değerleri de azalmaktadır. Bu noktada operatör seçiminin farklılıkları oldukça azdır.

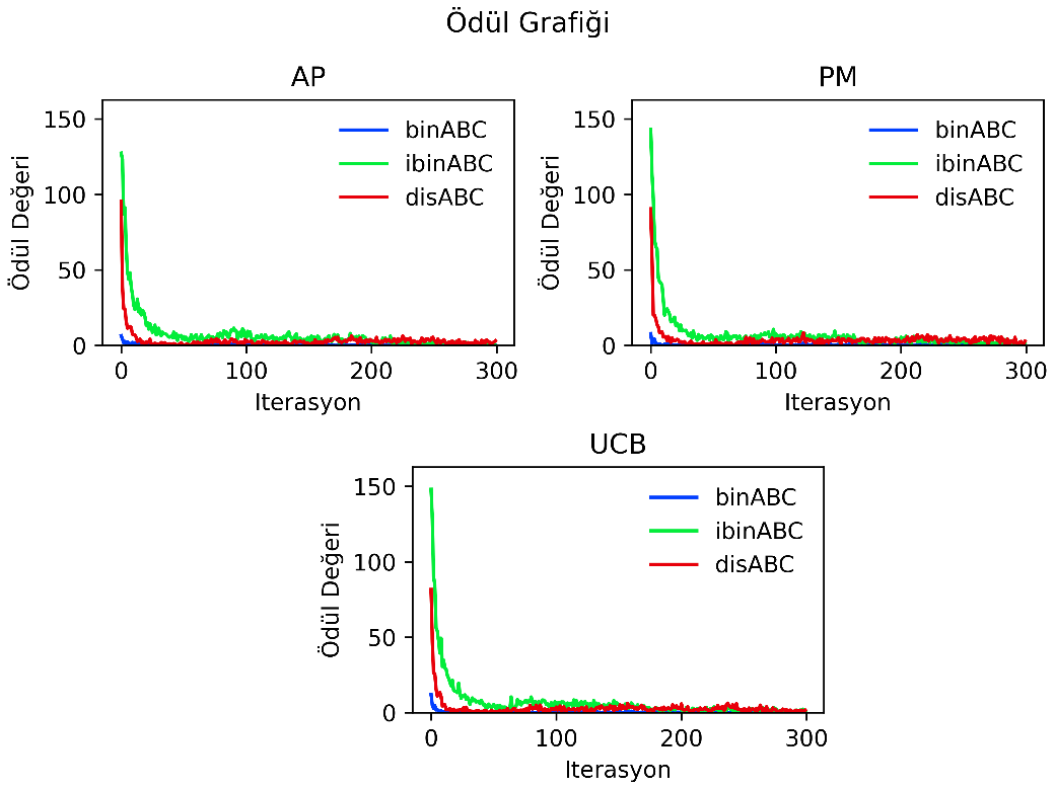
Şekil 5 ve Şekil 6'da ise, sırasıyla operatör kullanım sayıları ve başarılı çözüm üretme sayıları verilmiştir. Kullanım grafikleri de 30 çalıştırmanın ortalaması olarak verilmiştir. Her üç yaklaşımda da başlangıç iterasyonlarında ibinABC daha fazla kullanılmaktadır. Farklılık, seçilme sayılarıdır. AP yaklaşımı kredisine göre bu operatöre diğer yaklaşımlarda daha çok fırsat vermiştir. UCB algoritmasında geçişler daha yumuşak olmasına karşın, en sert geçişler AP yaklaşımındadır. PM yaklaşımında 100. İterasyondan sonra disABC daha çok öne çıkmaya başlamıştır.

**Tablo 2.** Parametre ayarlama için elde edilen sonuçlar (The results of parameter tuning)

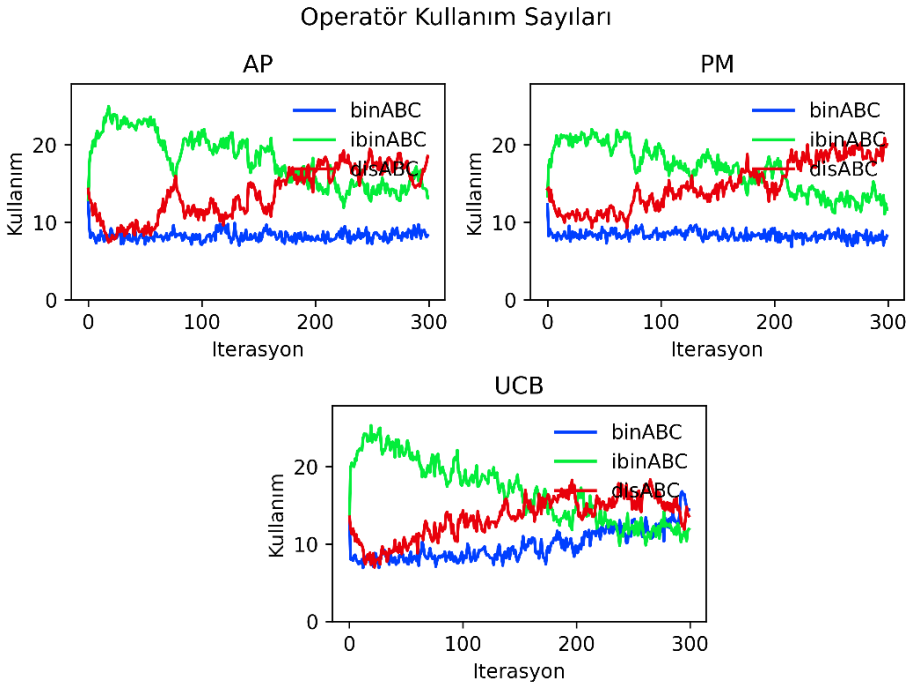
| | | | | AP | | PM | | UCB | |
|---|---|---|---|---|---|---|---|---|---|
| Ödül | Pencere | $P_{min}$ | *Alpha* | En iyi | Ortalama | En iyi | Ortalama | En iyi | Ortalama |
| Ortalama | 5 | 0.1 | 0,1 | 10998 | 10658,63 | 11113 | 10719,8 | 10948 | 10650,2 |
| | | | 0,5 | 11012 | 10676,30 | 10961 | 10692,13 | 10892 | 10686,57 |
| | | | 0,9 | 10735 | 10638,30 | 11023 | 10693,8 | 11054 | 10693,07 |
| | | 0.2 | 0,1 | 11039 | 10670,40 | 10889 | 10700,8 | **11410** | **10697,23** |
| | | | 0,5 | 11113 | 10667,50 | 11000 | 10697,07 | 11081 | 10712,23 |
| | | | 0,9 | 10793 | 10651,83 | **11425** | **10713,03** | 11057 | 10697,7 |
| | 25 | 0.1 | 0,1 | 11178 | 10709,37 | 11093 | 10679,13 | 10793 | 10667,37 |
| | | | 0,5 | 11106 | 10704,50 | 11081 | 10670,27 | 11113 | 10657,9 |
| | | | 0,9 | 11025 | 10672,17 | 11051 | 10652,43 | 11106 | 10663,37 |
| | | 0.2 | 0,1 | 11093 | 10662,50 | 11305 | 10687,93 | 10889 | 10677,67 |
| | | | 0,5 | 11082 | 10682,70 | 10851 | 10667,5 | 11025 | 10692,53 |
| | | | 0,9 | **11425** | **10708,63** | 11139 | 10700,73 | 10804 | 10669,37 |
| En Yüksek | 5 | 0.1 | 0,1 | 11007 | 10657,13 | 11222 | 10704,13 | 11039 | 10684,9 |
| | | | 0,5 | 11046 | 10683,53 | 11064 | 10724,37 | 11113 | 10689,3 |
| | | | 0,9 | 10949 | 10690,63 | 11178 | 10692,93 | 10835 | 10666,57 |
| | | 0.2 | 0,1 | 11093 | 10722,43 | 11093 | 10684,7 | 11093 | 10656,2 |
| | | | 0,5 | 11081 | 10691,70 | 11139 | 10666,73 | 10931 | 10660,97 |
| | | | 0,9 | 10949 | 10634,17 | 11057 | 10671,23 | 10953 | 10657,57 |
| | 25 | 0.1 | 0,1 | 11078 | 10683,90 | 11132 | 10691,23 | 11251 | 10693,33 |
| | | | 0,5 | 11425 | 10691,93 | 11251 | 10697,4 | 11113 | 10708,83 |
| | | | 0,9 | 10990 | 10707,43 | 11410 | 10723,53 | 10953 | 10688,03 |
| | | 0.2 | 0,1 | 10968 | 10681,27 | 10860 | 10654,13 | 11093 | 10715,7 |
| | | | 0,5 | 11093 | 10695,57 | 10990 | 10700,33 | 11178 | 10724,33 |
| | | | 0,9 | 10966 | 10662,50 | 11037 | 10704,23 | 11132 | 10713,1 |



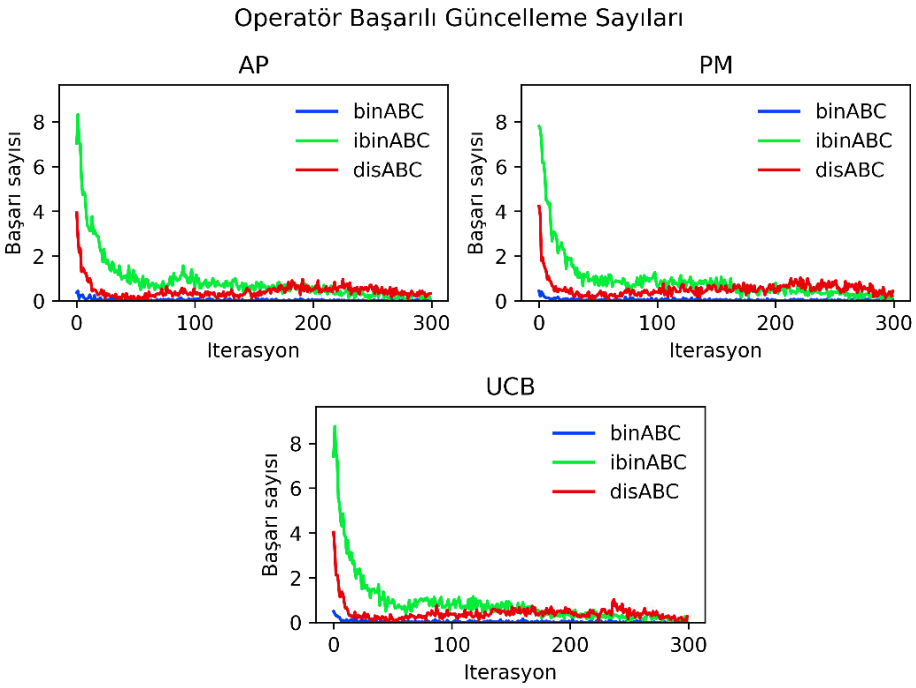

**Şekil 3.** Operatör seçim yöntemlerinin iterasyona bağlı kredi değerleri (The credit values of operatör selection methods through iterations)




**Şekil 4**. Operatör seçim yöntemlerinin iterasyona bağlı ödül değerleri (The reward values of operator selection methods through iterations)




**Şekil 5.** Operatör kullanım sayılarının iterasyona göre değişimi (The usage counters of operator selection methods through iterations)




**Şekil 6.** Operatör başarı sayılarının iterasyona göre değişimi (The succes counters of operator selection methods through iterations)

Önerilen yaklaşım ile elde edilmiş sonuçlar literatürde var olan diğer yöntemler ile karşılaştırmalı olarak Ek Tablo' da verilmiş olup, karşılaştırılan yöntemlere ait değerler (He, 2018) ve (Özsoydan vd., 2019) çalışmalarından doğrudan alınmıştır. Bu tabloda 100 farklı çalıştırma sonucunda elde edilmiş en iyi sonuç, ortalama, standart sapma değerleri verilmiştir. Geliştirilen yaklaşım ile karşılaştırılan diğer yöntemler; A-SUKP, BABC, ABCBin, BinDE, GPSO*, GPSO algoritmalarıdır. A-SUKP, KBŞÇ problemi için önerilmiş yakınsama algoritmasıdır. GA, Genetik Algoritma, BABC ve ABCBin İkili YAK algoritmalarıdır. BinDE ikili diferansiyel evrim algoritması, GPSO* ve GPSO ise açgözlü ikili parçacık sürüsü algoritmasıdır.

Elde edilen sonuçların anlamlılığını kolaylaştırmak adına, Tablo 3'te ilk problem grubu için algoritmaların başarı sıralamaları verilmiştir. Önerilen adaptif operatör seçimi yaklaşımı hem diğer YAK düzenlemelerinden hem de güncel literatürdeki çalışmalar içerisinden ortalama başarı sırasına göre birinci sıradadır. Burada 10 farklı örnek için 7 kez en iyi çözümleri sunmuş 3 kez ise ikinci sırada kalmıştır. En yakın rakibi GPSO olmuştur.

Tablo 4'te ise, ikinci grup problem örnekleri için algoritmaların performansları karşılaştırılmıştır. Önerilen çalışma bu problem örnekleri içerisinde en

başarılı algoritma olmuştur. Burada 10 farklı örnek için 6 kez en iyi çözümleri sunmuş, 4 kez ise ikinci sırada kalmıştır. En yakın rakibi GPSO olmuştur.

Tablo 5'te ise, üçüncü grup problem örnekleri için algoritmaların performansları karşılaştırılmıştır. Önerilen çalışma bu problem örnekleri içerisinde en başarılı algoritma olmuştur. 10 farklı örnek için 7 kez en iyi çözümleri sunmuş 3 kez ise ikinci sırada kalmıştır. En yakın rakibi GPSO olmuştur.

**Tablo 3.** İlk grup problem örnekleri (G1_1 – G1_10) için algoritmaların performansları (The performance of algorithms on first group problem instances)

| P. No | A-SUKP | GA | BABC | ABCBin | binDE | GPSO* | GPSO | PMABC |
|---|---|---|---|---|---|---|---|---|
| G1_1 | 7 | 3 | 5 | 8 | 6 | 4 | 2 | 1 |
| G1_2 | 8 | 7 | 2 | 5 | 4 | 6 | 3 | 1 |
| G1_3 | 8 | 6 | 4 | 7 | 5 | 3 | 1 | 2 |
| G1_4 | 8 | 5 | 4 | 7 | 6 | 3 | 2 | 1 |
| G1_5 | 8 | 6 | 3 | 7 | 5 | 4 | 2 | 1 |
| G1_6 | 8 | 3 | 5 | 7 | 6 | 4 | 2 | 1 |
| G1_7 | 7 | 4 | 5 | 8 | 6 | 3 | 1 | 2 |
| G1_8 | 7 | 5 | 4 | 8 | 6 | 3 | 2 | 1 |
| G1_9 | 8 | 6 | 3 | 7 | 4 | 5 | 1 | 2 |
| G1_10 | 8 | 7 | 4 | 3 | 6 | 5 | 2 | 1 |
| Ortalama: | 7,7 | 5,2 | 3,9 | 6,7 | 5,4 | 4 | 1,8 | 1,3 |

**Tablo 4.** İkinci grup problem örnekleri (G2_1 – G2_10) için algoritmaların performansları (The performance of algorithms on second group problem instances)

| P. No | A-SUKP | GA | BABC | ABCBin | binDE | GPSO* | GPSO | PMABC |
|---|---|---|---|---|---|---|---|---|
| G2_1 | 8 | 5 | 4 | 7 | 6 | 3 | 2 | 1 |
| G2_2 | 8 | 7 | 2 | 6 | 4 | 5 | 3 | 1 |
| G2_3 | 8 | 6 | 3 | 7 | 5 | 4 | 2 | 1 |
| G2_4 | 8 | 3 | 5 | 7 | 6 | 4 | 2 | 1 |
| G2_5 | 7 | 6 | 3 | 8 | 5 | 4 | 1 | 2 |
| G2_6 | 7 | 4 | 5 | 8 | 6 | 3 | 2 | 1 |
| G2_7 | 8 | 5 | 4 | 7 | 6 | 3 | 1 | 2 |
| G2_8 | 8 | 6 | 3 | 7 | 5 | 4 | 1 | 2 |
| G2_9 | 8 | 6 | 3 | 7 | 4 | 5 | 1 | 2 |
| G2_10 | 8 | 6 | 3 | 7 | 4 | 5 | 2 | 1 |
| Ortalama: | 7,8 | 5,4 | 3,5 | 7,1 | 5,1 | 4 | 1,7 | 1,4 |

**Tablo 5.** Üçüncü grup problem örnekleri (G3_1 – G3_10) için algoritmaların performansları (The performance of algorithms on third group problem instances)

| P. No | A-SUKP | GA | BABC | ABCBin | binDE | GPSO* | GPSO | PMABC |
|---|---|---|---|---|---|---|---|---|
| G3_1 | 8 | 5 | 3 | 7 | 4 | 6 | 2 | 1 |
| G3_2 | 8 | 7 | 1 | 5 | 3 | 6 | 4 | 2 |
| G3_3 | 8 | 6 | 3 | 7 | 4 | 5 | 1 | 2 |
| G3_4 | 8 | 6 | 3 | 7 | 4 | 5 | 2 | 1 |
| G3_5 | 8 | 5 | 4 | 7 | 6 | 3 | 2 | 1 |
| G3_6 | 8 | 6 | 3 | 7 | 5 | 4 | 2 | 1 |
| G3_7 | 8 | 6 | 3 | 7 | 5 | 4 | 1 | 2 |
| G3_8 | 8 | 4 | 5 | 7 | 6 | 3 | 2 | 1 |
| G3_9 | 8 | 3 | 5 | 7 | 6 | 4 | 2 | 1 |
| G3_10 | 8 | 6 | 3 | 7 | 4 | 5 | 2 | 1 |
| Ortalama: | 8 | 5,4 | 3,3 | 6,8 | 4,7 | 4,5 | 2 | 1,3 |

## 6. Sonuçlar (Conclusions)

Temel metasezgisel optimizasyon algoritmalarının başarısı kullandıkları komşu çözüm üretme operatörleri ile doğrudan ilişkilidir. Bu operatörün yerel minimumdan kurtulabilme ve hızlı yakınsama kabiliyetlerine aynı anda sahip olması oldukça zordur. Bu kabiliyetleri kazandırabilmek için yöntemlerin arama uzayının koşullarına göre adaptif olması gerekmektedir. Bu adaptasyon sayesinde zor problemlerin çözümü hızlandırılabilmektedir. Bu çalışmada yapay arı koloni algoritmasına adaptif operatör seçimi eklenerek, ilk kez, küme birleşimli sırt çantası problemleri üzerindeki etkisi ortaya konulmuştur. Ele alınan problem için, üç farklı operatör seçim yöntemi ile yapay arı kolonisinin birleştirilmesi ve detaylı parametre analizi yapılmış olup, en iyi konfigürasyon belirlenmiştir. En iyi konfigürasyonda son 5 iterasyon boyunca elde edilmiş ödüllerin ortalama değerleri kullanılması halinde en iyi çözümler elde edilmiştir. Ardından, 30 farklı problem örneği için üretilmiş çözümler literatürdeki çalışmalar ile karşılaştırılmıştır. Karşılaştırma sonucunda önerilen yöntemin literatürdeki çalışmalardan daha gürbüz çözümler ürettiği görülmüştür.

Sonraki çalışmalarda, Operatör sayısının arttırılarak KBSÇ problemi üzerindeki etkisi incelenebilir. Bu sayede operatörlerin problem üzerindeki etkinliği ortaya konulabilir.

## Kaynaklar (References)


Arani, B.O., Mirzabeygi, P., Panahi, M.S., 2013, "An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration–exploitation balance.", Swarm and Evolutionary Computation, 11,1-15.

Arulselvan, A., 2014, "A note on the set union knapsack problem.", Discrete Applied Mathematics, 169, 214-218.

Bansal, J.C., 2019, "Particle swarm optimization. In Evolutionary and swarm intelligence algorithms", Springer, Cham, 11-23.

Bitran, G.R., Hax, A.C., 1981, "Disaggregation and resource allocation using convex knapsack problems with bounded variables.", Management Science, 27(4), 431-441.

Bretthauer, K.M., Shetty, B., 2002, "The nonlinear knapsack problem–algorithms and applications." European Journal of Operational Research 138(3), 459-472.

Durgut, R., Aydin, M.E., 2020, "Adaptive binary artificial bee colony algorithm.", Applied Soft Computing, 107054.

Durgut, R., 2020, "Improved binary artificial bee colony algorithm.", arXiv preprint, arXiv, 2003(11641).

Eberhart, R., Kennedy, J., 1995, "A new optimizer using particle swarm theory.", MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Ieee.

Fialho, Á., 2010, "Analyzing bandit-based adaptive operator selection mechanisms.", Annals of Mathematics and Artificial Intelligence, 60(1-2), 25-64.

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

Goldschmidt, O., Nehme, D., Yu., G., 1994, "Note: On the set-union knapsack problem." Naval Research Logistics (NRL), 41(6),833-842.
He, Y., 2018, "A novel binary artificial bee colony algorithm for the set-union knapsack problem.", Future Generation Computer Systems, 78, 77-86.
Holland, J., 1975, "Adaptation in natural and artificial systems: an introductory analysis with application to biology.", Control and artificial intelligence.
Karaboga, D., Basturk.B., 2008, "On the performance of artificial bee colony (ABC) algorithm.", Applied soft computing, 8(1),687-697.
Karaboga D.,2005 "An Idea Based on Bee Swarm for Numerical Optimization", Technical Report-TR06.
Kashan, M.H., Nahavandi, N. and Kashan, A. H., 2012, "DisABC: A new artificial bee colony algorithm for binary optimization.", Applied Soft Computing, 12(1),342-352.
Kiran, M.S., Gündüz, M., 2003, "XOR-based artificial bee colony algorithm for binary optimization.", Turkish Journal of Electrical Engineering & Computer Sciences, 21(2), 2307-2328.
Klamroth, K., Margaret, M.W., 2000, "Dynamic programming approaches to the multiple criteria knapsack problem.", Naval Research Logistics (NRL), 47(1), 57-76.
Kiran, M.S., 2015, "The continuous artificial bee colony algorithm for binary optimization.", Applied Soft Computing, 33, 15-23.
Lin, G., Xu, H., Chen, X., Guan, J., 2020, "An effective binary artificial bee colony algorithm for maximum set k-covering problem.", Expert Systems with Applications, 161, 113717.
Mirjalili, S., 2019, "Genetic algorithm. In Evolutionary algorithms and neural networks ", Springer, Cham, 43-55.
Moradi, N., Kayvanfar, V., Rafiee, M., 2021, "An efficient population-based simulated annealing algorithm for 0–1 knapsack problem", Engineering with Computers, 1-20.
Odlyzko, A.M., 1990, "The rise and fall of knapsack cryptosystems.", Cryptology and computational number theory, 42, 75-88.
Ozsoydan, F.B., Baykasoglu, A., 2019, "A swarm intelligence-based algorithm for the set-union knapsack problem.", Future Generation Computer Systems, 93, 560-569.
Ozsoydan, F. B., 2019, "Artificial search agents with cognitive intelligence for binary optimization problems.", Computers & Industrial Engineering, 136, 18-30.
Wei, Z., Hao, J. K., 2021, "Kernel based tabu search for the Set-union Knapsack Problem", Expert Systems with Applications, 165, 113802.
Wang, H., Wang, W., Xiao, S., Cui, Z., Xu, M., Zhou, X., 2020, "Improving artificial bee colony algorithm using a new neighborhood selection mechanism.", Information Sciences, 527, 227-240.
Xiang, W.L., Li, Y.Z., He, R.C., An, M.Q., 2021, "Artificial bee colony algorithm with a pure crossover operation for binary optimization", Computers & Industrial Engineering, 152, 107011.

## Ek (Appendix)

**Ek Tablo:** Problem örnekleri üzerinde yöntemlerden elde edilen sonuçlar (Results obtained from methods on problem instances).

| | A-SUKP | | | GA | | | BABC | | | ABCBin | | | binDE | | | GPSO* | | | GPSO | | | PMABC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P. No | En iyi | Ort | Std | En iyi | Ort | Std | En iyi | Ort | Std | En iyi | Ort | Std | En iyi | Ort | Std | En iyi | Ort | Std | En iyi | Ort | Std | En iyi | Ort | Std |
| G1_1 | 12459 | 12459 | 0 | 13044 | 12956 | 130.66 | 13251 | 13029 | 92.63 | 13044 | 12819 | 153.06 | 13044 | 12991 | 75.95 | 13167 | 12937 | 190 | 13283 | 13051 | 37.41 | 13251 | 13056,31 | 44 |
| G1_2 | 11119 | 11119 | 0 | 12066 | 11546 | 214.94 | 12238 | 12155 | 53.29 | 12238 | 12049 | 96.11 | 12274 | 12124 | 67.61 | 12210 | 11778 | 277 | 12274 | 12085 | 95.38 | 12274 | 12137,04 | 60 |
| G1_3 | 11292 | 11292 | 0 | 13064 | 12493 | 320.03 | 13241 | 13064 | 99.57 | 12946 | 11862 | 324.65 | 13241 | 12941 | 205.7 | 13302 | 12766 | 305 | 13405 | 13287 | 93.18 | 13405 | 13266,85 | 110 |
| G1_4 | 12262 | 12262 | 0 | 13671 | 12803 | 291.66 | 13829 | 13359 | 234.99 | 13671 | 12537 | 289.53 | 13671 | 13110 | 269.69 | 13993 | 12949 | 326 | 14044 | 13493 | 328.72 | 14215 | 13640,36 | 223 |
| G1_5 | 8941 | 8941 | 0 | 10553 | 9981 | 142.97 | 10428 | 9995 | 154.03 | 9751 | 9339 | 158.15 | 10420 | 9899 | 153.18 | 10600 | 10090 | 236 | 11335 | 10670 | 227.85 | 11411 | 10702,97 | 163 |
| G1_6 | 9432 | 9432 | 0 | 11016 | 10350 | 215.13 | 12012 | 10903 | 449.45 | 10913 | 9958 | 276.9 | 11661 | 10499 | 403.95 | 11935 | 10750 | 525 | 12245 | 11607 | 477.8 | 12245 | 11711,15 | 293 |
| G1_7 | 9076 | 9076 | 0 | 10083 | 9642 | 168.94 | 10766 | 10065 | 241.45 | 9674 | 9188 | 167.08 | 10576 | 9681 | 275.05 | 10698 | 9947 | 295 | 11484 | 10916 | 367.75 | 11244 | 10733,08 | 229 |
| G1_8 | 8514 | 8514 | 0 | 9831 | 9327 | 192.2 | 9649 | 9136 | 151.9 | 8978 | 8540 | 161.83 | 9649 | 9021 | 150.99 | 10168 | 9417 | 360 | 10710 | 9865 | 315.38 | 10328 | 10071,65 | 120 |
| G1_9 | 9864 | 9864 | 0 | 11031 | 10568 | 123.15 | 10784 | 10452 | 114.35 | 10340 | 9910 | 120.82 | 10586 | 10364 | 93.39 | 11258 | 10566 | 260 | 11722 | 11185 | 322.98 | 11546 | 11195,34 | 142 |
| G1_10 | 8299 | 8299 | 0 | 9472 | 8693 | 180.12 | 9090 | 8858 | 94.55 | 8789 | 8364 | 114.1 | 9191 | 8784 | 131.05 | 9759 | 8779 | 300 | 10022 | 9300 | 277.62 | 10194 | 9361,02 | 182 |
| G2_1 | 13634 | 13634 | 0 | 14044 | 13806 | 144.91 | 13860 | 13735 | 70.76 | 13860 | 13547 | 199.11 | 13814 | 13676 | 119.53 | 13963 | 13740 | 120 | 14044 | 13855 | 96.23 | 14044 | 13920,20 | 91 |
| G2_2 | 11325 | 11325 | 0 | 13145 | 12235 | 388.66 | 13508 | 13352 | 155.14 | 13498 | 13103 | 343.46 | 13407 | 13212 | 287.45 | 13498 | 12937 | 418 | 13508 | 13347 | 194.34 | 13508 | 13434,01 | 66 |
| G2_3 | 10328 | 10328 | 0 | 11656 | 10889 | 237.85 | 11846 | 11194 | 249.58 | 11191 | 10424 | 197.88 | 11535 | 10969 | 302.52 | 11972 | 11233 | 369 | 12522 | 11899 | 391.83 | 12350 | 11890,76 | 211 |
| G2_4 | 9784 | 9784 | 0 | 11792 | 10828 | 334.43 | 11521 | 10945 | 255.14 | 11287 | 10346 | 273.47 | 11469 | 10717 | 341.08 | 12167 | 11027 | 21 | 12317 | 11585 | 275.32 | 12317 | 11691,87 | 187 |
| G2_5 | 10208 | 10208 | 0 | 12055 | 11755 | 144.45 | 12186 | 11946 | 127.8 | 11494 | 10922 | 182.63 | 12304 | 11865 | 160.42 | 12736 | 11934 | 294 | 12695 | 12411 | 225.8 | 12713 | 12583,80 | 127 |
| G2_6 | 9183 | 9183 | 0 | 10666 | 10099 | 337.42 | 10382 | 9860 | 177.02 | 9633 | 9187 | 147.78 | 10382 | 9710 | 208.48 | 10724 | 9907 | 399 | 11425 | 10568 | 327.48 | 11093 | 10688,38 | 118 |
| G2_7 | 9751 | 9751 | 0 | 10570 | 10112 | 157.89 | 10626 | 10101 | 196.99 | 10160 | 9549 | 141.27 | 10462 | 9976 | 185.57 | 11048 | 10400 | 282 | 11531 | 10959 | 274.9 | 11310 | 10861,11 | 140 |
| G2_8 | 8497 | 8497 | 0 | 9235 | 8794 | 169.52 | 9541 | 9033 | 194.18 | 9033 | 8366 | 153.4 | 9388 | 8768 | 212.24 | 10264 | 9195 | 312 | 10927 | 9845 | 358.91 | 10725 | 9880,29 | 283 |
| G2_9 | 9615 | 9615 | 0 | 10460 | 10185 | 114.19 | 10755 | 10328 | 91.61 | 10071 | 9738 | 111.64 | 10546 | 10228 | 103.32 | 10647 | 10205 | 190 | 10888 | 10681 | 125.36 | 10885 | 10637,93 | 82 |
| G2_10 | 7883 | 7883 | 0 | 9496 | 8883 | 158.21 | 9318 | 9181 | 84.91 | 9262 | 9618 | 141.32 | 9312 | 9096 | 145.45 | 9839 | 9107 | 258 | 10194 | 9704 | 252.84 | 10176 | 9819,00 | 160 |
| G3_1 | 10231 | 10231 | 0 | 11454 | 11093 | 171 | 11664 | 11183 | 184 | 11206 | 10880 | 164 | 11352 | 11075 | 119 | 11710 | 11237 | 169 | 12045 | 11487 | 138 | 12020 | 11590,33 | 171 |
| G3_2 | 10483 | 10483 | 0 | 12124 | 11326 | 417 | 12369 | 12082 | 194 | 12006 | 11485 | 248 | 12369 | 11876 | 337 | 12369 | 11684 | 354 | 12369 | 11994 | 437 | 12369 | 12156,68 | 184 |
| G3_3 | 11508 | 11508 | 0 | 12841 | 12237 | 198 | 13047 | 12523 | 201 | 12308 | 11668 | 177 | 13024 | 12278 | 234 | 13298 | 12514 | 356 | 13696 | 13204 | 367 | 13609 | 13307,07 | 136 |
| G3_4 | 8621 | 8621 | 0 | 10920 | 10352 | 208 | 10602 | 10151 | 153 | 10376 | 9684 | 185 | 10547 | 10085 | 161 | 10856 | 10208 | 264 | 11298 | 10801 | 206 | 11298 | 10817,49 | 140 |
| G3_5 | 9961 | 9961 | 0 | 10994 | 10640 | 127 | 11158 | 10776 | 117 | 10269 | 9957 | 141 | 11152 | 10661 | 150 | 11310 | 10762 | 199 | 11568 | 11318 | 183 | 11538 | 11217,25 | 191 |
| G3_6 | 9618 | 9618 | 0 | 11093 | 10190 | 250 | 10528 | 9898 | 187 | 10051 | 9424 | 197 | 10528 | 9832 | 233 | 11226 | 10309 | 389 | 11517 | 10899 | 300 | 11590 | 11042,90 | 217 |
| G3_7 | 8672 | 8672 | 0 | 9799 | 9433 | 164 | 10085 | 9538 | 185 | 9235 | 8905 | 112 | 9883 | 9315 | 192 | 8971 | 9552 | 234 | 10483 | 10013 | 202 | 10397 | 9955,62 | 105 |
| G3_8 | 8064 | 8064 | 0 | 9173 | 8704 | 154 | 9456 | 9090 | 157 | 8932 | 8407 | 149 | 9352 | 8847 | 211 | 9389 | 8881 | 283 | 10338 | 9525 | 286 | 9865 | 9439,48 | 141 |
| G3_9 | 9559 | 9559 | 0 | 10311 | 9993 | 118 | 10823 | 10483 | 228 | 10537 | 9615 | 151 | 10728 | 10159 | 198 | 10595 | 10145 | 200 | 11094 | 10688 | 168 | 11018 | 10603,61 | 126 |
| G3_10 | 8157 | 8157 | 0 | 9329 | 8849 | 142 | 9333 | 9086 | 116 | 8799 | 8348 | 123 | 99218 | 8920 | 169 | 9807 | 8917 | 267 | 10104 | 9383 | 241 | 9686 | 9389,84 | 125 |