KÜME BİRLEŞİMLİ SIRT ÇANTASI PROBLEMİNİN ADAPTİF YAPAY ARI KOLONİSİ YÖNTEMİ İLE ÇÖZÜMÜ

İlim Betül YAVUZ

2021
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Tez Danışmanı
Dr.Öğr.Üyesi Rafet DURGUT

KÜME BİRLEŞİMLİ SIRT ÇANTASI PROBLEMİNİN ADAPTİF YAPAY ARI KOLONİSİ YÖNTEMİ İLE ÇÖZÜMÜ

İlim Betül YAVUZ

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

Tez Danışmanı
Dr.Öğr.Üyesi Rafet DURGUT

KARABÜK
Ocak 2021

İlim Betül YAVUZ tarafından hazırlanan "KÜME BİRLEŞİMLİ SIRT ÇANTASI PROBLEMİNİN ADAPTİF YAPAY ARI KOLONİSİ YÖNTEMİ İLE ÇÖZÜMÜ" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr.Öğr.Üyesi Rafet DURGUT .....
Tez Danışmanı, Bilgisayar Mühedisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından Oy Birliği ile Bilgisayar Mühendsiliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 29/01/2021

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Dr. Öğr. Üyesi Yasemin GÜLTEPE (KÜ)	.....
Üye : Dr. Öğr. Üyesi Rafet DURGUT (KBÜ)	.....
Üye : Dr. Öğr. Üyesi Emrullah SONUÇ (KBÜ)	.....

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Hasan SOLMAZ .....
Lisansüstü Eğitim Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

İlim Betül YAVUZ

ÖZET

Yüksek Lisans Tezi

KÜME BİRLEŞİMLİ SIRT ÇANTASI PROBLEMİNİN ADAPTİF YAPAY ARI KOLONİSİ ALGORİTMASI İLE ÇÖZÜMÜ

İlim Betül YAVUZ

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr.Öğr.Üyesi Rafet DURGUT
Ocak 2021, 42 sayfa

Bu çalışmada, 0-1 yapılı sırt çantası probleminin özel bir hali olan Küme Birleşimli Sırt Çantası problemine, doğadan esinlenilmiş metasezgisel bir algoritma olan Yapay Arı Kolonisi algoritmasına adaptif bir yapı eklenmesi vasıtasıyla optimum çözüm sağlanmıştır.

Yapay Arı Kolonisi algoritması sürekli optimizasyon problemlerinin çözümü için önerilmiştir ve ikili yapıda olan bu probleme adapte olabilmesi için algoritmanın da ikili yapıdaki forma dönüştürülmesi gerekmektedir. Bu dönüştürme iki şekilde gerçekleştirilmektedir. Birincisi, eğer sürekli karar değişkeni kullanılırsa bu değişkenler haritalama fonksiyonu vasıtasıyla ikili karar değişkenine dönüştürülebilir. Diğer bir çözüm ise komşu çözümlerin ikili operatörler sayesinde üretilmesidir. İkinci yaklaşımda yöntemin başarısı operatörlerin başarısı ile belirlenmektedir. Çalışmada arama uzayının genişletilmesi maksadıyla birden fazla operatörün kullanımı

hedeflendiği için operatörlerin adaptif bir mekanizma içerisinde çalışması sağlanmıştır. Adaptif operatör seçimini benimseyen bu yöntem Adaptif Yapay Arı Kolonisi (AYAK) olarak isimlendirilmiş olup, tüm arama sürecinde tek bir operatör kullanılmasının yerine operatör havuzundan uygun operatörün seçilerek komşu çözüm üretilmesini önermektedir. Oluşturulan bu seçim şemalarının özellikleri kapsamlı olarak 30 kıyaslama problemi üzerinde incelenmiştir. Bu problem kümeleri için en iyi performans gösteren algoritma önerilmiştir. Elde edilen sonuçlar güncel literatürdeki yöntemler ile karşılaştırılmış ve oldukça rekabetçi sonuçlar elde edilmiştir. Çalışma, başarılı bir seçim şemasına sahip adaptif ve ikili yapıda bir yapay arı kolonisi algoritması sunmaktadır.

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

Bilim Kodu : 92402

ABSTRACT

M. Sc. Thesis

SOLVING SET UNION KNAPSACK PROBLEM USING ADAPTIVE ARTIFICIAL BEE COLONY ALGORITHM

İlim Betül YAVUZ

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Asst.Prof.Dr. Rafet DURGUT
January 2021, 42 pages

In this study, the optimum solution was provided to the Set Union Knapsack Problem which is a special form of the 0-1 backpack problem, by adding an adaptive structure to the Artificial Bee Colony algorithm which is a metaheuristic algorithm inspired by nature.

The Artificial Bee Colony algorithm has been proposed for the solution of continuous optimization problems, and in order to adapt to this binary problem, the algorithm must be transformed into a binary form. This conversion is carried out in two ways. First, if a continuous decision variable is used, these variables can be converted into a binary decision variable through the mapping function. Another solution is to produce neighboring solutions through binary operators. In the second approach, the success of the method is determined by the success of the operators. Since it is aimed to use more than one operator in order to expand the search space in the study, it is ensured that

the operators work in an adaptive mechanism. This method which adopts the selection of adaptive operators, is named Adaptive Artificial Bee Colony and instead of using a single operator during the entire search, it suggests selecting the appropriate operator from the operator pool and producing a neighboring solution. The features of these selection schemes have been extensively examined 30 comparison problems. The best performing algorithm has been proposed for these problem clusters. The results obtained were compared with the methods in the current literature and extremely competitive results were obtained. The study presents an adaptive binary artificial bee colony algorithm with a successful selection scheme.

Key Word **:** Artificial Bee Colony, Adaptive Operator Selection, Set Union Knapsack Problem

Science Code : 92402

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr.Öğr.Üyesi Rafet DURGUT'a sonsuz teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

x : problem tarafında: nesne, algoritma tarafında: kaynak

v : nesnenin değeri

w : nesnenin ağırlığı

W : çantanın kapasitesi

U : elemanlar kümesi

S : nesneler kümesi

A : küme

V : aday çözüm

φ : algoritmanın hangi ölçüde ilerleyeceğini belirten katsayı

p : olasılık değeri

$\oplus$: özel veya mantıksal operatörü

m : bit sayısı

M : aday çözüm için bit sayısı

p : olasılık değeri

৳ : pozitif ölçek faktörü

KISALTMALAR

YAK : Yapay Arı Kolonisi

AYAK : Adaptif Yapay Arı Kolonisi

SUKP : Set Union Knapsack Problem (Küme Birleşimli Sırt Çantası Problemi)

KBSÇ : Küme Birleşimli Sırt Çantası

AK : Adaptif Kovalama

OE : Olasılık Eşleme

ÜGS : Üst Güven Sınırı

NP-hard : Non-Deterministic Polinominal-Time Hardness (Deterministik Olmayan Polinomsal Zaman Zorluğu)

BÖLÜM 1

GİRİŞ

Optimizasyon, belirli amaç ya da amaçlar doğrultusunda bir problemin çözümü için en uygun (optimum) çözümün bulunmasıdır. Bu çözüm belirlenen amaca göre minimizasyon ya da maksimizasyon olarak değişmektedir. Algoritmalar ise optimizasyon işleminde kullanılan yöntemlerdir. Hangi problemin hangi algoritma ile çözüme gidileceği problemin ve algoritmanın yapısına göre değişmektedir.

Metasezgisel Algoritmalar, doğrusal zamanda çözümü zor problemlerler için kısa sürede en uygun çözüme yaklaşan çözümler üretmesi sayesinde son zamanlarda güçlü bir seçim haline gelmiştir. Doğadan esinlenilerek geliştirilen bu tip algoritmalar içerisinde; Karınca Kolonisi [1], Yapay Arı Kolonisi [2], Genetik Algoritma [3] ve Parçacık Sürüsü Optimizasyonu [4] gibi oldukça başarılı yöntemler bulunmaktadır. Popülasyon (sürü) tabanlı olan bu yaklaşımlarda her bir çözüm ilgili bireye atanır ve komşu çözümlerden yararlanarak arama uzayındaki en iyi çözümü sunan noktaya ulaşılmaya çalışılır.

Çalışmada kullanılan Yapay arı kolonisi algoritması Karaboğa vd. (2005) tarafından bal arılarının yiyecek bulma davranışından esinlenilerek sürekli optimizasyon problemlerine çözüm bulmak için geliştirilmiştir [2]. Yöntem temel halde sürekli optimizayon problemlerine uygulandığı için yöntemin eşitlikleri de bu tip problemlere göre düzenlenmiştir. Bu sebeple ikili optimizasyon problemlerine doğrudan uygulanamamakta ve düzenlemelere ihitiyaç duymaktadır. Bu düzenlemeler komşu çözüm üretmeye yönelik yapılmaktadır. Kiran vd. (2013) tarafından sunulan binABC algoritması [5], Kashan vd. (2012) tarafından sunulan disABC [6] algoritması gibi çeşitli komşu çözüm üretme mekanizmaları bulunmaktadır. Durgut (2020) ise, ibinABC ile binABC algoritmasının yakınsama hızını arttırmak için bazı düzenlemeler sunmuştur [7].

Metasezgisel algoritmalarda tüm arama süreci genellikle tek bir komşuluk operatörü ile yürütülür. Seçilen çözüm üzerine komşuluk operatörü uygulanarak en iyi olmaya aday yeni bir çözüm oluşturulur. Yapay arı kolonisi algoritması da, arama sürecinde ulaştığı çözümleri geliştirerek daha iyi çözümlere ulaşmaya çalışmakta (sömürü fazı) veya arama bölgesinin farklı noktalarına erişerek yeni ve başarılı çözümlere ulaşmaya (keşif fazı) çalışmaktadır. Bu iki faz (keşif ve sömürü) arasında denge sağlanmazsa ya yerel minimum takılma problemi ya da yakınsamanın yavaş olması problemi ortaya çıkmaktadır [8]. Bu problemleri en aza indirmek, yerel minimuma takılmadan en iyi çözüme hızlı bir şekilde ulaşabilmek amacıyla tüm arama sürecinde tek bir arama operatörünün kullanılmasının yerine adaptif operatör seçimi yaklaşımı önerilmektedir [9]. Bu sayede arama sürecinin farklı noktalarında daha başarılı olabilecek operatörlere imkân verilmektedir. Bu yaklaşımda komşu çözüm üretimi aşamasında bir operatör yerine operatör havuzu kullanılması ile farklı operatörlerin kullanılması ön plana çıkarılmıştır. Bu yöntemde operatörlerin birbirlerini tamamlayıcı davranış göstermesi beklentisi bulunmaktadır. Başarılı operatörlerin seçim şansının arttırılması sayesinde, operatörlerin birbirlerini tamamlayıcı davranabilecekleri ve en iyi çözüme hızlı şekilde ulaşılması mümkün hale gelmiştir

Çalışmada çözülmesi hedeflenen Sırt Çantası problemi, maksimum faydayı sağlamak amacıyla çözülmesi gereken ve doğrusal zamanda çözülmesi mümkün olmayan, başka bir deyişle NP-hard tip bir optimizasyon problemidir. Problemin mantalitesi, çantayı kapasitesini aşmayacak şekilde en değerli olan nesnelerle doldurmak için nesnelerden hangilerinin çantaya koyulması gerektiğinin cevabını bulmaya çalışmaktır.

Sırt çantası probleminin yapısına göre literatürde farklı tipleri bulunmaktadır. Küme Birleşimli Sırt Çantası (KBSÇ) problemi 0-1 tipli sırt çantası probleminin özel bir halidir. KBSÇ probleminin şimdiye kadar çeşitli uygulama ve alanlarda değerli oluşu bilinmektedir. Bunlardan bazıları; finansal kararlar [10, 11], esnek makine üretimi [12, 13], veritabanı bölütleme [14, 15], akıllı şehir uygulamaları [16] gibi alanlardır.

Çalışmada, küme birleşimli sırt çantası problemine uygun çözümler sunulabilmesi için Adaptif Yapay Arı Kolonisi (AYAK) yaklaşımı önerilmiştir. Çalışmada operatörlere atanacak olan, başarımı gösteren kredi değerinin belirlenmesi ve operatörlerin nasıl

seçileceği AYAK yaklaşımının temelini oluşturmaktadır. Kullanılan operatörün kredilendirilmesinin ödül değeri üzerinden geliştirilmesi hedeflenmiştir. Ödül değeri olarak başarılı olma sayıları veya amaç fonksiyonundaki geliştirilmeleri kullanılmaktadır. Bu ödül değerleri aynı zamanda, anlık veya bir pencere boyunca alınan ödüllerin ortalama ya da en yüksek değerleri olarak ele alınabilir. Operatör seçimi için ise Olasılık Eşleme, Adaptif Kovalama ve Üst Güven Sınırı yöntemleri denenmiştir

Önerilen yaklaşım üç farklı operatör ve üç farklı operatör seçimi için çalıştırılmış sonucunda en iyi konfigürasyon elde edilmiştir. Elde edilen sonuçlar literatürdeki diğer başarılı yöntemler ile karşılaştırılmış olup, var olan yöntemlerden daha uygun çözümler üretilmiştir. Bu sayede geliştirilen yöntemin KBSÇ probleminin uygulama alanlarında daha yüksek fayda ve düşük maliyet sunması öngörülmüştür.

Hazırlanan bu çalışma, ana hatlarıyla literatür taraması içerikli başlıklar ve deneysel çalışmalar olmak üzere iki kısımdan oluşmaktadır. Literatür taramasını içeren başlıklar kendi içinde beş adet konuyu barındırmaktadır. Birinci bölüm "Giriş" olup burada çalışmanın kısa özeti mevcuttur. İkinci bölümde Optimizasyon kavramına genel bir bakış sağlanmasının ardından alt başlıklar olarak Sırt Çantası Problemi, 0-1 Sırt Çantası ve Küme Birleşimli Sırt Çantası problemi formulasyonlarıyla birlikte iki kısım halinde anlatılmaktadır. Üçüncü bölümde Metasezgisel Algoritmalar ve alt başlığı olarak Yapay Arı Kolonisi Algoritması, metadolojisi ve çalışma adımlarıyla beraber alt başlıklar halinde anlatılmaktadır. Dördüncü bölümde algoritmayı ikili forma dönüştürme maksatlı literatürde şimdiye kadar kullanılmış üç teknikten üç alt başlık halinde söz edilmektedir. Literatür taramasının yapıldığı son bölüm olan beşinci bölüm, problemi çözmede kullanılacak olan Adaptif Yapay Arı Kolonisi algoritmasının çalışma mantalitesini içeren başlıklara sahiptir.

Deneysel çalışmaların ve nihai sonuçlarının açıklandığı altıncı ve en son bölümde, deneysel çalışmalar sonucunda elde edilen bulgular, deneysel çalışmanın amacına uygun bir şekilde yorumlanarak sonuçlandırılmıştır.

BÖLÜM 2

OPTİMİZASYON

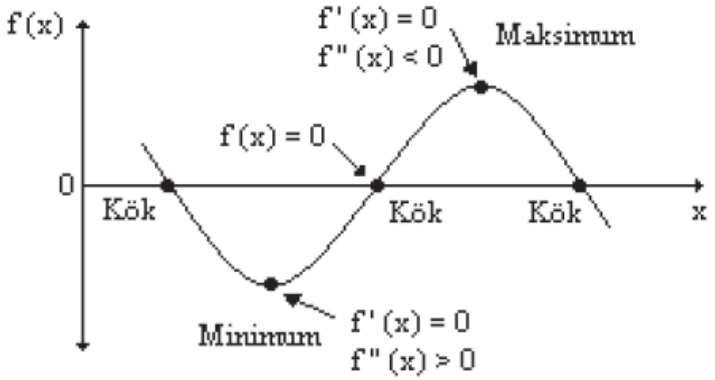
2.1. GENEL BAKIŞ

En iyileme işlemi olarak da adlandırılan optimizasyon kavramı verilen şartlar altında en iyi sonucu elde etmeyi amaçlar. İnsanlık tarihi boyunca var olan bu kavram, gerçek hayat problemlerini çözme maksadıyla yönetim, mühendislik vb. birçok alanda önemli kararlar alınmasında rol oynamıştır. Optimizasyonun amacı, maksimum kar veya minimum maliyeti sağlayacak durumları tespit etmektir. Bu doğrultuda zamanla kaynakların sınırlı hale gelmesi, gelişen teknoloji ile rekabetin ve çözülmesi amaçlanan problemlerin zorluğunun artması insanları optimizasyonda yeni teknikler geliştirmeye itmiştir.

Optimizasyon tarihte 17.yy'dan günümüze birçok önemli gelişme yaşamıştır. İkinci Dünya Savaşı'ndan sonra ise hızlı bilgisayarların gelişmesi optimizasyon dünyasına yeni teori ve teknikler kazandırmıştır. Her geçen gün geliştirilen optimizasyon teknikleri maddi, iş gücü ve zaman gibi kaynaklar için önemli derecede fayda sağlamaktadır.

2.1.1 Tarihçe, Amaç ve Genel Kavramlar

Optimizasyon kavramı Newton'un bir fonksiyonu minimize etme (minimum bulma) yöntemi önerisi ile başlamıştır [17]. Bu yöntem Newton'un kök belirleme algoritmasını temel alarak oluşturulmuştur. Kök belirlemede fonksiyonu sıfır yapan noktalar aranmaktadır, optimizasyonda ise fonksiyonu minimum ya da maksimum yapan noktaların bulunması amaçlanır. Kök belirleme ve optimizasyon işlemlerinin mantalitesinin ikisinde de "tahmin etme" ve "arama" ortak kavramlardır. Şekil 2.1'de kök ile optimum değerler arasındaki fark görsel olarak bir fonksiyonda belirtilmişti



Şekil 2.1: Optimum değerler ve kökleri içeren fonksiyon [18].

Şekil 2.1'de belirtilen fonksiyona göre, fonksiyonun türevinin yani $f'(x)$'in sıfır olduğu nokta (x değeri) optimum değeri ifade eder ve. Newton'a göre fonksiyonun optimum minimum ya da maksimum olduğunu anlamak için ise fonksiyonun ikinci türev yani $f''(x)$ değerlerine bakılır; eğer $f''(x) < 0$ ise bu nokta maksimum, $f''(x) > 0$ ise minimum değer olduğunu gösterir.

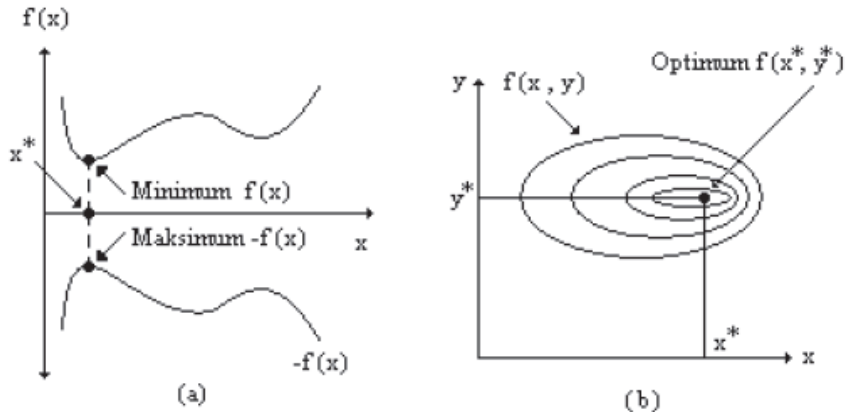
Optimizasyonda amaç, istenen çıktının elde edilebilmesi için girdilerin ne olacağının belirlenmesidir. Burada girdiye uygulanan sistem (model) ve çıktı bilinmekte fakat girdiler bilinmemektedir.

Bir optimizasyon probleminin modelinde üç temel kavram bulunmaktadır. Bunlar; amaç fonksiyonu (objective function), karar değişkenleri ve kısıtlardır. Belirli kısıtlar dahilinde amaç fonksiyonun değerinin en iyilenmesi için karar değişkenlerinin alacağı değerler belirlenmeye çalışılmaktadır ve bu da problemi çözme sürecidir. Burada amaç fonksiyonunun en iyilenmesi, problemin yapısına göre minimizasyon ya da maksimizasyon olarak değişmektedir.

Kısıtlamalar incelendiğinde ise, problem kısıtlamalı ya da kısıtlamasız olabilmektedir. Kısıtlamalı bir modelde iki tip kısıt mevcuttur. Bunlar; değişken kısıtları ve ana kısıtlardır.

Karar değişkenlerinin türüne göre problem sürekli ya da kombinatoryal (ayrık, kesikli) olmaktadır. Değişkenlerin sürekli olduğu problemde değişkenler reel sayılar üzerinde tanımlıdır. Başka bir deyişle problem sonsuz giriş ve çıkış değerlerine sahiptir. Kombinatoryal problemlerde ise değişkenler sonlu bir kümeden bir nesne ya da sayılabilir sonsuz bir küme (tamsayılar vb.) üzerinde tanımlıdır. Değişkenler sonlu sayıda belirli değeri alabilmektedir. Ayrık bir problemde gerçek hayat ile bağdaştırıldığında nesneler (değişkenler) bir amacı gerçekleştirecek biçimde seçilme, gruplandırma veya sıraya konma olarak problemler oluşturulabilir.

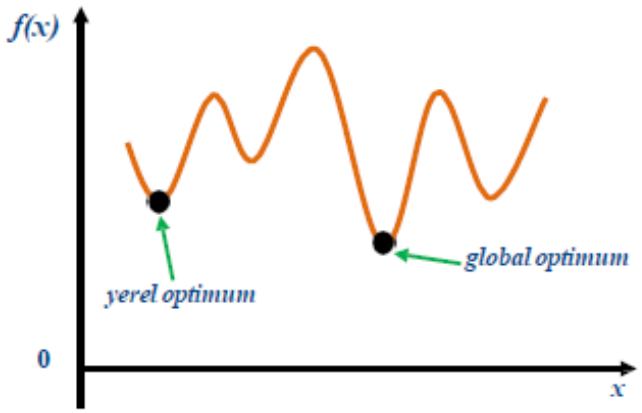
Karar değişkenlerin boyutu da problemin yapısını tek boyutlu veya çok boyutlu olarak değiştirmektedir. Tek boyutlu karar değişkeninde problem tek bir değişkene bağlıdır. Çok boyutluda ise sonucu etkileyen birden fazla karar değişkeni söz konusudur. Şekil 2.2 'de tek boyutlu ve iki boyutlu optimizasyon fonksiyonu görselleri vardır. Aynı zamanda Şekil 2.2.a'da x^* değerinin *f(x)*'i minimize ettiği ve *-f(x)*'i maksimize ettiği görülmektedir. Bu durum bir fonksiyonun maksimizasyonunun ya da minimizasyonunun eşdeğer olduğunu göstermektedir [19].



Şekil 2.2. Bir boyutlu ve iki boyutlu optimizasyon. a) Bir boyutlu optimizasyon. b) İki boyutlu optimizasyon [18].

Çözüm uzayında problemin kısıtlarını sağlayan herhangi bir çözüm uygun (feasible) çözüm olarak adlandırılır. Optimum çözüm ise uygun çözümler arasından belirlenmiş olan amaca göre en iyi çözümü ifade eder. Problemin amacına göre bulunmuş uygun çözümlere yerel optimum nokta, optimum çözüme global (evrensel) optimum nokta

denilmektedir. Yerel optimum ile global optimum arasındaki ilişki Şekil 2.3'te belirtilmiştir.



Şekil 2.3. Yerel ve global optimum.

Amaç fonksiyonunun, karar değişkenlerin ve kısıtlamaların yapısı problemi çözmede kullanılacak algoritmaları (yöntemleri) belirlemede önemli rol oynamaktadır. Bunların yanında problemin karmaşıklığı (complexity) da bu etmenlerden biridir. Problem karmaşıklığı kavramı, problemin polinomsal zamanda çözümünün olup olamaması ve modelin doğrulanıp doğrulanmaması ile ilgilidir.

Optimizasyon problemleri tümden gelindiğinde karar değişkenlerinin tipine göre sürekli ve ayrık problemler olmak üzere ikiye ayrılır. Bu karar değişkenleri üzerinde sınırlama durumuna göre kısıtlamalı ve kısıtlamasız olarak ayrılmaktadır. Problem karmaşıklığına göre ise tek modlu, çok modlu, polinomsal (P) ve belirleyici olmayan polinomsal (NP) problemler olarak gruplandırılırlar.

Tek modlu kavramı, problemin tek bir yerel (bölgesel) minimum ya da maksimum noktaya sahip olması ve bu noktanın da aynı zamanda optimum (global) nokta olmasıdır. Bu şekilde olan problemlerin çözümü kolay olmaktadır. Çok modlu problemlerde ise yerel minimum ya da maksimum noktası çok sayıda olmaktadır. Bu durum problemin çözümünü zorlaştırmaktadır.

Bir fonksiyonun grafiği görsel olarak incelendiğinde; eğer doğrulardan oluşuyorsa fonksiyon doğrusal eğrilerden oluşuyorsa doğrusal olmayan fonksiyondur denilebilir. Kısıtlamalı sürekli problemler, kısıtlama fonksiyonlarına bağlı olarak doğrusal ya da doğrusal olmayan formda olabilirler. Problem karmaşıklığı ise problemin ne kadar zor olduğunun bir ölçüsüdür. Polinomiyal (P) işlem karmaşıklığına sahip bir problem polinomsal zamanda kesin bir çözüme sahiptir. Fakat NP tip problemler polinomsal bir zamanda çözümü zor olan problemlerdir ve genellikle sezgisel algoritmalar ile optimum ya da optimuma yakın çözüm bulunur.

2.2. SIRT ÇANTASI PROBLEMİ

Sırt Çantası problemi yapı olarak bir yöneylem araştırması ve matematiksel olarak da kombinatoryal optimizasyon problemidir.

Bir yöneylem araştırmasında belirli kısıtların olduğu durumda, belirli bir amaca yönelik en uygun çözümün bulunması amaçlanır. En uygun çözüm ise problemin amacına göre maksimum ya da minimum olarak değişir. Kombinatoryal kavramı ise, problem karar değişkenlerinin yapısına göre sürekli ya da kesikli (kombinatoryal) olarak iki sınıfa ayrıldığı için karar değişkenlerinin yapısı ile doğrudan ilgilidir.
Problemin mantalitesi ise çantayı kapasitesini aşmayacak şekilde en değerli (faydalı) olan nesnelerle doldurmak için nesnelerden hangilerinin çantaya koyulması gerektiğinin cevabını bulmaya çalışmaktır. Başka bir deyişle amaç fonksiyonun maksimize edilmesini sağlayacak kaynakların elde edilmesiyle ve sınırlı kapasiteye sahip çantanın değerinin maksimum olması hedeflenmektedir.

2.2.1. 0-1 Sırt Çantası Problemi

Sırt çantası problemi, özel amaçlar doğrultusunda farklı yapıda olabilmektedir. 0-1 tipli Sırt Çantası probleminde 0-1 kavramı o nesnenin çantaya konulup konulamayacağını ifade eder. Bu da bu nesnelerin 0 ve 1 olarak kategorilere ayrılması anlamına gelir ve ayrıca bu durum sadece 0 ya da 1 değerine sahip başka bir karar değişkeninin de eklenmesi demektir. 0-1 Sırt Çantası probleminin formülasyonu, amaç fonksiyonu olarak Eşitlik (2.1)'de, kısıtlar olarak Eşitlik (2.2)'de belirtilmiştir.

Amaç Fonksiyonu: $\sum_{i=1}^{n} v_i x_i$ (2.1)

Kısıtlar: $\sum_{i=1}^{n} w_i x_i \leq W, \quad x_i \in \{0,1\}, \quad \forall_i \in \{1, \ldots, n\}$ (2.2)

Eşitlik (2.1) ve Eşitlik (2.2)'de x_i değişkeninin değeri, modelin çözümünde *i* nesnesinin alınıp alınmadığını gösterir. v_i ve w_i değerleri modelin parametreleri olup, sırasıyla *n* tane nesne arasından *i* numaralı nesnesinin değerini ve ağırlığını, *W* parametresi ise çantanın toplam kapasitesini ifade eder.

2.2.2. Küme Birleşimli Sırt Çantası Problemi

0-1 sırt çantasının özel bir hali olan Küme Birleşimli Sırt Çantası (KBSÇ) problemi NP-hard tip problemdir. NP-hard tip problem karmaşıklığına sahip problemlerin polinomsal zamanda çözümü olduğu ispatlanamamıştır. Bu sebeple problemin çözümünde sezgisel algoritmalar vasıtasıyla en yakın çözüm bulunur.

KBSÇ probleminde nesneleri içinde bulundukları kümeye göre değerlendirerek en uygun küme seçimlerinin yapılmasını amaçlamaktadır.
Goldschmidt vd. (1994) KBSÇ kesin çözümü için hipergraf tabanlı dinamik programlama algoritmasının kullanılmasını önermiştir [20]. Fakat bu algoritma düşük boyutlu problemler için hızlı çözüm önermesine karşın, problem boyutu arttıkça doğrusal zamanda bir çözüm sunamamaktadır. Benzer şekilde, Arulselvan vd. (2014) aç gözlü bir yakınsama algoritması önermişlerdir [10].

Problemin formülasyonu Eşitlik (2.3) ve Eşitlik (2.4)'te belirtilmiş olup *n* tane eleman *U = {1, 2, ..., n}* kümesi ve *m* tane nesne *S (1, 2, ..., m)* kümesi vardır. Nesneler için; *i* $\in$ *S(i = 1, 2, ..., m)* şeklinde ifade edilen *S* kümesindeki her i nesnesi, değeri p_i>0 olan ve Ui$\subseteq$ U şeklinde bir alt kümedir. Elemanlar için; *j* $\in$ *U (j = 1, 2, . . ., n)* şeklinde ifade edilen her *j* elemanının ağırlığı $w_j > 0$ şeklindedir.

KBSÇ probleminin amacı, ağırlık kapasitesinin *W(S*)* $\leq$ *C* (C: pozitif tamsayı) şeklinde aşılmaması kısıtı ile *P(S*)* değerini maksimize eden bir *S**$\subseteq$*S* altküme seçimi

yapmaktır. Boş olmayan bir A kümesi için $A \subseteq S$ şeklindedir ve A'nın değeri ve ağırlığı Eşitlik (2.3) ve (2.4)'teki gibi bulunur.

Amaç Fonksiyonu (Maksimize): $P(A) = \sum_{i \in A} p_i$ (2.3)

Kısıtlar: $W(A) = \sum_{j \in \bigcup_{i \in A} U_i} w_j \leq C, A \subseteq S$ (2.4)

BÖLÜM 3

METASEZGİSEL ALGORİTMALAR

3.1. GENEL BAKIŞ

Optimizasyon algoritmaları (yöntemleri), makul süreler içerisinde problemin amacına uygun çözümü bulma amacıyla geliştirilmiş algoritmalardır. Başka bir deyişle problemi çözme adımları denilebilir. Optimizasyon algoritmaları optimum çözümün kesin ya da yaklaşık olarak bulunması olarak sınıflandırılır. Analitik algoritmalar kesin çözüme ulaştırırken, sezgisel ve meta-sezgisel algoritmalar yaklaşık çözüm sunmaktadır. Analitik algoritmalar için büyük boyutlu problemlerde çözüme ulaşmak ya çok uzun sürer ya da ulaşılamaz. Sezgisel algoritmalar, probleme özgü olarak tasarlanmış çözüm yöntemleridir. Optimum çözümü garanti etmezler fakat analitik yöntemlere göre daha hızlı çözüm üretirler. Metasezgisel algoritmalar ise, belirli algoritmaların çözülecek olan problem yapısına uyarlanması ile elde edilirler. Problemin karmaşıklığı arttıkça kısa sürede optimuma yakın çözüm üretmek, uzun sürede kesin çözüm üretmekten daha değerli olmaktadır. Sezgisel oldukları için kesin bir çözüm garanti etmemekle birlikte en yakın çözümü sağlarlar.

Metasezgisel algoritmalar; doğadan esinlenen ve doğadan esinlenmeyen yöntemler, dinamik ve statik amaç fonksiyonuna sahip yöntemler, bir komşuluk yapısına ve değişken komşuluk yapısına sahip yöntemler, hafıza kullanan ve kullanmayan yöntemler ve tek çözüme dayalı ya da popülasyon tabanlı metasezgisel algoritmalar olarak sınıflandırılabilmektedir [22].

Doğadan esinlenen metasezgisel algoritmalar, doğada gerçekleşen bir olayı modelleyerek kombinatoryal optimizasyon problemlerine uygun çözümler getirmeyi amaçlamaktadır. Karınca kolonisi algoritması, yapay arı kolonisi algoritması ve genetik algoritmalar doğadan esinlenen metasezgisel algoritmalar arasındadır.

Amaç fonksiyonunun yapısına göre dinamik olan metasezgisel algoritmalar arama esnasında amaç fonksiyonun dinamik olarak değiştirilmesi ile yerel optimuma takılması engellenir [22]. Bu algoritmaların çoğunluğu bir komşuluk yapısı kullanırken, bazıları değişken komşuluk yapısı ile aramanın çeşitliliğini artırmayı amaçlamaktadır. Algoritmanın hafıza kullanması ise, arama sonucunda elde edilmiş uygun çözümlerin bir sonraki aramalarda kullanılmasını sağlamaktadır.

3.2. YAPAY ARI KOLONİSİ ALGORİTMASI

3.2.1. Sürü Zekası

Sürü zekasına dayalı bir optimizasyon algoritması olan Yapay Arı Kolonisi Algoritması, bal arılarının besin arama davranışlarından esinlenilerek Karaboğa tarafından 2005 yılında geliştirilmiştir [2].

Sürü zekâsı kavramı; karınca, kuş, balık, böcek gibi topluluk halinde yaşayan hayvanların davranışlarının anlamlandırılması ile ortaya çıkmıştır. Kendi kendine organize olabilme ve iş bölümü yapabilmek sürü zekasının temelini oluşturmaktadır.

Arılardaki sürü zekası dört adım halinde gösterilebilir. Bunlar; pozitif geri besleme, negatif geri besleme, salınımlar ve çoklu etkileşimlerdir. Pozitif geri besleme, bir işin daha çok yapılması için gerekli koşullardır. Örneğin kaynağın nektar miktarı arttıkça bu kaynağı geçen arı sayısı da artmaktadır [23]. Negatif geri beslemeye yiyecek tükentiminde ya da yarışma süreçlerinde doyuma gitmeyi önleme maksatlı ihtiyaç duyulmaktadır. Tükenen kaynağın bırakılması negatif geri beslemeye örnektir. Sanımlarda, keşifler veya hatalar gibi davranışlar yaratıcılık ve yenilikler açısından önem taşımaktadır. Örneğin, arılar yiyecek kaynağı bulabilmek için rastgele arama yapmaktadır. Çoklu etkileşimlerde ise sürüdeki bazı bireylerin diğer bireylerden gelen bilgi paylaşımı sağlanmaktadır. Örneğin, arılar yiyecek kaynakları ile ilgili olarak dans alanında bilgi dağılımı gercekleştirirler [24].

İşlerini dinamik olarak dağıtabilen ve çevresel değişimlere karşı sürü zekalarıyla uyarlanabilir cevaplar verebilen bal arıları fotoğrafik hafızaya, uzay çağı sensorlerine,

navigasyon sistemine, sezgisel kavrama yeteneğine ve yeni yuva yeri secerken sürü olarak karar verme ozelliğine sahiptirler [25].

3.2.2. Arıların Besin Arama Davranışı

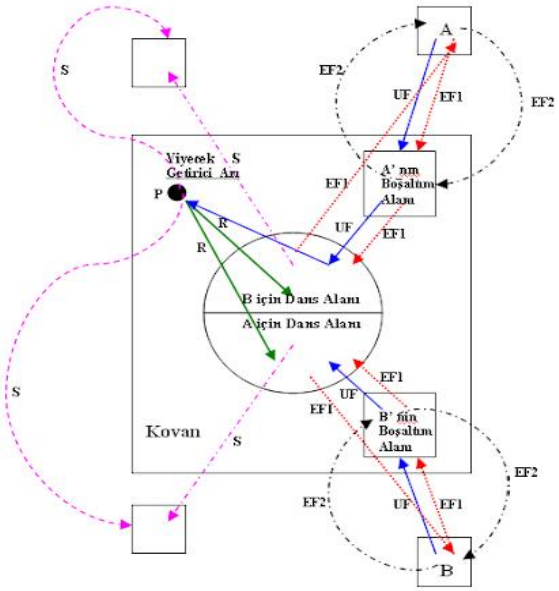
Arı kolonisinin yaşamının devamının sağlanması için en önemli işlerden birisi besin aramadır. Kovan içinde biriktirilen kaynaklar ve ortamdaki bulunabilecek besin kaynakları ve arıların etkileşimleri bu süreçteki önemli etkenlerdir. Arının kovandan ayrılmasıyla başlayan arama süreci, başlangıçta rastgele yapılan besin araştırmaları ile devam etmektedir. Bulunan kaynakta yiyecek miktarının azalması sonucunda arılar yeni besin aramaya ya da arılardan aldığı bilgiye göre başka kaynaklara yönelmeye başlarlar. Bulunan kaynakların bilgilerinin arılarca birbirine iletilmesi ve bulunan polen, su vb. kaynakların kovana getirilmesi bu süreç içinde yapılan faaliyetlerdir [26].

Tereshko'nun reaktif difüzyon denklemlerine temel olarak önerdiği kolektif zekayı sağlayan minimal besin arama modelinde üç temel bileşen vardır: Besin kaynakları, görevli işçi arılar ve görevsiz işçi arılardır. Minimal Model iki modda ele alınır. Bunlar; yiyecek kaynağına yönelme ve kaynağı bırakma durumlarıdır [27].

Bileşenlerden besin kaynağının değeri; arıların nektar (bal) elde etmek için gittikleri besinin türü, yuvaya yakınlığı, nektar konsantrasyonu veya nektarın elde edilmesinin kolaylığı gibi birçok faktöre bağlıdır. Görevli işçi arılar ise, daha önceden keşfedilen kaynakların kovana getirilmesinden sorumludur. İşçi arılar gittikleri kaynağın kalitesi ve yeriyle ilgili bilgiyi diğer arılara iletirler. Görevi belirsiz iki çeşit işçi arı bulunmaktadır: içsel bir duyguya veya bir dış faktöre bağlı olarak rastgele kaynak arayışında olan kaşif arılar ve kovanda izleyen ve görevli arılardan paylaşılan bilgiyi kullanarak yeni bir kaynağa yönelen gözcü arılardır [23].

Kaynağın kovana olan yakınlığına göre dairesel, kuyruk ve titreme dansı mevcuttur. Buradaki titreme dansı, nektar miktarı ve getirme yeteneği arasındaki dengeyi sağlamaktadır. Daire dansı, 50-100 metre uzaklıktaki kaynağın yön ve uzaklık bilgisi verilmeden sağlanan dans biçimidir. 100 metreden 10 kilometreye kadar geniş bir alan

içerisinde 8 rakamına benzeyen ve yineleme sayısına göre uzaklık bilgisi sunan kuyruk dansı ile güneş ve yiyecek arasındaki açının 45° olduğunu anlamaktadırlar [28].



Şekil 3.1. Yiyecek arama modeli [28].

Şekil 3.1'de yiyecek arayan arıların davranışları modellenmiştir. A ve B isminde iki keşfedilmiş kaynak olduğu varsayıldığında, bu durumdaki bir arı kaşif arı olabilir (Şekil 3.1'de S ile gösterilmektedir), gözcü arı olabilir (Şekil 3.1'de R ile gösterilmektedir) ya da bir işçi arı olabilir.

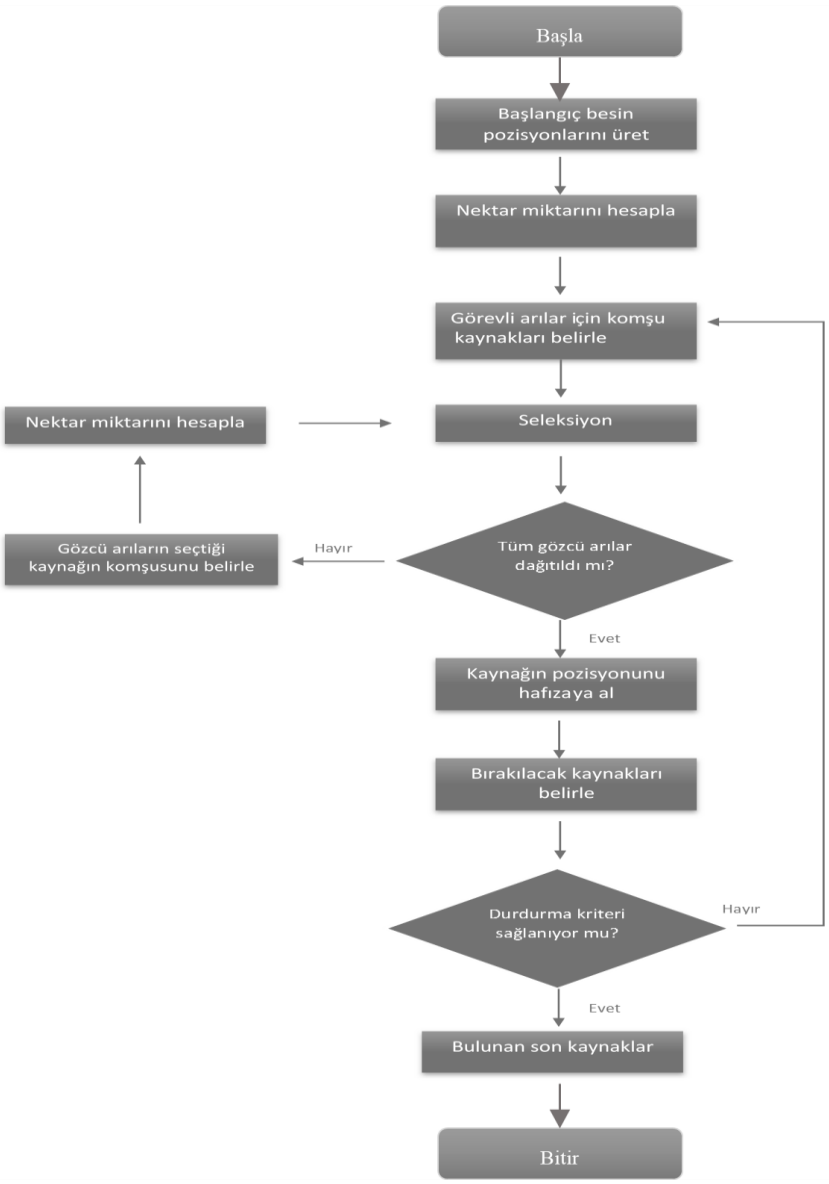
Arı nektarı aktardıktan sonra üç seçenek ortaya çıkar; gittiği kaynağa bırakarak bağımsız izleyici olabilir (Şekil 3.1'de UF ile gösterilmektedir), gittiği kaynağa dönmeden önce dans ile diğer arıları kaynağa yönlendirebilir (Şekil 3.1'de EF1 ile gösterilmektedir) ya da diğer arıları yönlendirmeden kaynağa gidebilir (Şekil 3.1'de EF2 ile gösterilmektedir) [29].

3.2.3. Akış Diyagramı ve Çalışma Adımları

Yapay arı kolonisi algoritmasında arama uzayındaki sonuclara karşılık gelen yiyecek kaynaklarının üretimi için gerekli olan parametrelerin alt ve üst sınırları arasındaki yerler gelişigüzel (rastgele) üretilmektedir. Oluşturulan her bir kaynağın

geliştirilememe sayaçları da aynı anda sıfırlanmaktadır. Bu süreçten sonra kaynağı bırakma kriteri sağlanıncaya kadar yiyecek kaynakları işçi arı, kâşif arı ve gözcü arı fazlarından gecirilerek en uygun değer bulunmaya calışılmaktadır. Yiyecek kaynağının komşuluğunda yeni bir kaynak belirlenir ve bu kaynak daha iyi ise işçi arı veya gözcü arı bu kaynağı hafızasına almakta ve geliştirilememe sayacı sıfırlanmaktadır. Tersi durumda ise geliştirilememe sayacı bir arttırılmaktadır [25].

Yapay arı kolonisi algoritması akış diyagramı Şekil 3.2' de belirtilmiştir.



Şekil 3.2. YAK akış diyagramı.

Arıların besin arama davranışlarının geçtiği aşamalar eşitlikler ile beraber şu şekildedir:

- Rastgele başlangıç kaynakları üretimi, $x_{ij}, \mathrm{i} = 1 \ldots \mathrm{SN}, \mathrm{j} = 1 \ldots \mathrm{D}$
- Her bir kaynağın uygunluk (fitness/amaç) fonksiyonunu hesapla, fi
- iterasyon = 1
- tekrarla
    - Eşitlik (3.2)'yi kullanarak V_i komşu kaynağını üret, *fi* değerini hesapla
    - V_i ile x_i arasında açgözlü seçim işlemi uygula
    - Kaynakların olasılık dağılımlarını belirle, p_i
    - p_i olasılığına göre x_i kaynağını seç
    - x_i komşuluğunda yeni bir V_i kaynağını üret ve fi değerini hesapla
    - V_i ile x_i arasında açgözlü seçim işlemi uygula
    - Tükenen kaynak var ise o kaynak için Eşitlik (3.1) ile rastgele yeni bir kaynak üret
    - Şimdiye kadar bulunan en iyi çözümü sakla
- iterasyon = iterasyon + 1
- olana kadar; iterasyon = maksimum çevrim sayısı

3.2.3.1 Başlangıç Kaynaklarının Üretilmesi

Başlangıç olarak yiyecek kaynaklarının üretilmesi için kâşif arılar yiyecek kaynaklarını rastgele aramaya başlarlar. Yiyecek kaynağı bulunduktan sonra arı kovana döner ve "sallanma dansı" ile kaynağın yerini işçi arılara iletir.

$$x_{ij} = x_{ij}^{min} + rand(0,1)(x_{ij}^{max} - x_{ij}^{min}) \quad (3.1)$$

Eşitlik (3.1)'de rastgele başlangıç kaynaklarının üretilmesi gösterilmetedir. Eşitlikte *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.

3.2.3.2. İşçi Arı Fazı

İşçi arı fazında, arılar mevcut yiyecek kaynağının komşuluğunda yeni bir yiyecek kaynağı (aday çözüm) arar. Yeni oluşturulan çözümün kalitesi eskisinden daha iyi ise aç gözlü seçim uygulanaraak nu çözüm güncellenir. Böylelikle işçi arı fazında her arı üzerinde çalıştığı besin kaynağını geliştirmektedir.

$$V_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{3.2}$$

Eşitlik (3.2)'de her bir arı rastgele 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 (parametresi) için önerilmiş yeni değerdir. Bu değer, mevcut kaynağın komşuluğunda rastgele başka bir kaynağın yine aynı boyut bilgisine göre (x_{kj}) güncellenmektedir. φ değeri hangi ölçüde ilerleneceğini ifade eden katsayıdır ve [-1,1] aralığında rastgele üretilmektedir.

$$fitness_i = \begin{cases} -\frac{1}{1+fi}, & fi \geq 0 \\ 1 + abs(fi), & fi < 0 \end{cases} \tag{3.3}$$

Eşitlik (3.3)'te bulunan kaynağın kalitesi, uygunluk fonksiyonu sonucunda hesaplanır. Kalite değerinin sıfırdan büyük ve eşit ya da küçük olmasına göre bu kaynağa denklemde hesaplanan uygunluk değeri atanmış olur.

3.2.3.3. Gözcü Arı Fazı

Bu aşamadaki 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 bir olasılık değeri belirlenir. Bu sayede gözcü arı aşamasında geliştirilmesi hedeflenen besin kaynakları belirlenmiş olur.

$$p_i = \frac{fitnees_i}{\sum_{j=i}^{SN} fitness_i} \tag{3.4}$$

Kaynakların kalitesine göre bulunmuş olan uygunluk değerlerinin ardından bu kaynakların hangisinin geliştirileceği tüm kaynaklar baz alınarak hesaplanan olasılıksal değere bağlıdır. Her bir çözümün olasılık değeri Eşitlik (3.4)'e göre belirlenmektedir. Bu olasılık değerlerine göre çözümler üzerinde güncelleme test edilir.

3.2.3.4. Kaşif Arı Fazı

Kaşif arı fazında, limit değerine ulaşan güncellemeye ragmen 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.

BÖLÜM 4

İKİLİ YAPAY ARI KOLONİSİ ALGORİTMASI

Yapay arı kolonisi algoritması sürekli optimizasyon problemlerini 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 [23], diğeri ise karar değişkenlerinin ikili formda [12, 14] 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, disABC ve yeni önerilmiş olan ibinABC operatörleri adaptif bir mekanizma içerisinde kullanılmaktadır.

4.1. KULLANILAN İKİLİ OPERATÖRLER

4.1.1. BinABC

Çözüm uzayı ikili (binary) yapıda olan optimizasyon problemleri için geliştirilmiş "Özel Veya" mantıksal işlevi tabanlı arı kolonisi algoritmasıdır. Bu algoritma Kiran vd. (2013) tarafından "Kapasite tesis yerleştirme problemi" 'nin çözümü için geliştirilmiştir [5]. YAK ikili yapıya uygun hale gelmesi için Eşitlik (3.1) ve Eşitlik (3.2) 'nin uygun forma dönüştürülmesi gerekir.

$$X_{i,j} = \begin{cases} 0, & r_{i,j} < p \\ 1, & r_{i,j} \geq p \end{cases} \tag{4.1}$$

Başlangıçta kullanılacak arı popülasyonu için rastgele [0,1] aralığında sayılar üretiliyordu. Eşitlik (4.1)'de, üretilen bu sayı ($r_{i,j}$) belirtilen p (var olma) olasılığından düşükse $X_{i,j}$=0 olur, büyük ve eşitse 1 değerini alır.

İlk arı popülasyonu oluşturulduktan sonra, işçi ve gözcü arı pozisyonlarının hesaplandığı Eşitlik (3.2), Eşitlik (4.2)'deki gibi güncellenir.

$$V_i^j = X_i^j \oplus [\varphi_{ij}(X_i^j \oplus X_k^j)] \quad (4.2)$$

i, k∈{1, 2, ..., N}, j∈{1, 2, …, D} ve $i \neq k$

Eşitlik (4.2)'de $V_i^j (V_{i,j})$ değişkeni *i.* aday çözümün *j*. boyutunu temsil eder. X_i^j *i.* işçi arının *j*. boyutu, X_i^j *k.* işçi arının *j*. boyutu, $\oplus$ Özel Veya mantıksal operatörü, φ_{ij} ise %50 olasılıkla mantıksal NOT 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. İşçi arıların durumuna göre aday çözümün bulunduğu Özel Veya tablosu Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Girişe göre Özel Veya mantıksal operatörü ile aday çözüm hesabı.

			1	2	Özel Veya (1 ile)	Özel Veya (2 ile)
X_i^j	X_k^j	$(X_i^j \oplus X_k^j)$	$\varphi < 0{,}5$	$\varphi \geq 0{,}5$	$X_i^j \oplus [\varphi(X_i^j \oplus X_k^j)]$	$X_i^j \oplus [\varphi(X_i^j \oplus X_k^j)]$
0	0	0	1	0	1	0
0	1	1	0	1	0	1
1	0	1	0	1	1	0
1	1	0	1	0	0	1

4.1.2. DisABC

ABC nin diğer ikili versiyonu olan DisABC algoritması Kashan vd. (2012) tarafından önerilmiştir [6]. Bu algoritmada ABC nin yapısal durumu baz alınarak işçi arı (X_i) diğer bir deyişle seçili çözüm ve komşu arı (X_k) arasındaki farklılığın ölçülmesi

önemlidir. Eşitlik (4.3)'te Jaccard benzerlik katsayısı bulunarak 1'den çıkarılır, böylelikle farklılık bulunur.

$$\text{Farklılık}\,(X_i, X_k) = 1 - \text{Benzerlik}\,(X_i, X_k) \tag{4.3}$$

$$\text{Farklılık}\,(X_i, X_k) = 1 - \frac{m_{11}}{m_{01}+m_{10}+m_{11}} \tag{4.4}$$

D: boyut, j = 1, 2, …, D

Eşitlik (4.4) için;

- m_{01} değeri $X_{i,j}$=0 ve $X_{k,j}$ =1 olduğu durumdaki bitlerin sayısını,

$$m_{01} = \sum_{j=1}^{D} F(X_{i,j} = 0\,, X_{k,j} = 1\,)$$

- m_{10} değeri $X_{i,j}$=1 ve $X_{k,j}$ = 0 olduğu durumdaki bitlerin sayısını,

$$m_{10} = \sum_{j=1}^{D} F(X_{i,j} = 1\,, X_{k,j} = 0\,)$$

- m_{11} değeri $X_{i,j}$=1 ve $X_{k,j}$ =1 olduğu durumdaki bitlerin sayısını temsil eder.

$$m_{11} = \sum_{j=1}^{D} F(X_{i,j} = 1\,, X_{k,j} = 1\,)$$

Eşitlik (4.5) için, Aday çözüm: V_i , pozitif ölçek faktörü: ৬

$$\text{Farklılık}\,(V_i, X_i) \approx \text{ ৬ x Farklılık}\,(X_i, X_k) \tag{4.5}$$

Aday çözüm değerleri için;

- M_{01} değeri $V_{i,j}$=0 ve $X_{k,j}$ =1 olduğu durumdaki bitlerin sayısını,

$$M_{01} = \sum_{j=1}^{D} F(V_{i,j} = 0\,, X_{k,j} = 1\,)$$

- M_{10} değeri $V_{i,j}$=1 ve $X_{k,j}$ = 0 olduğu durumdaki bitlerin sayısını,

$$M_{10} = \sum_{j=1}^{D} F(V_{i,j} = 1 , X_{k,j} = 0)$$

- M_{11} değeri $V_{i,j}$=1 ve $X_{k,j}$ =1 olduğu durumdaki bitlerin sayısını temsil eder.

$$M = \sum_{j=1}^{D} F(V_{i,j} = 1 , X_{k,j} = 1)$$

Eşitlik (4.6) için n_1 ve n_0 katsayıları sırasıyla X_i 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 haneler sıfırlanır.

$$min|(1 - \frac{M_{11}}{M_{01}+M_{10}+M_{11}}) - ৬(1 - \frac{m_{11}}{m_{01}+m_{10}+m_{11}})| \quad (4.6)$$

$$M_{11} + M_{01} = n_1 \quad , \quad M_{10} \leq n_0 \quad , \; M_{01}, M_{10}, M_{11} \geq 0 \; ve \; tam \; sayı$$

4.1.3. IbinABC

Durgut (2020), binABC operatörünü iyileştirerek ibinABC algoritmasını önermiştir [7]. ibinABC algoritması iki yeni düzenleme içermektedir. Bunlardan ilki komşuluk operatörünün birden fazla hane üzerinde uygulanmasıdır. Operatörün uygulanacağı hane sayısı Eşitlik (4.7)'ye göre belirlenmektedir.

$$d_t = rast[0, a] + 0.1D * e^{-(t/tmax)} + 1 \quad (4.7)$$

Eşitlik (4.7)'de a değeri yöntem parametresi olup, ölçeklendirme için kullanılmaktadır. *D*, problem boyutunu, *t* o anki iterasyon değerini, *tmax* 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 işleme daha fazla etkili olacak şekilde belirlenmektedir. Aksi durumda ise, φ iterasyon değerine göre Eşitlik (4.7)'deki gibi belirlenmektedir.

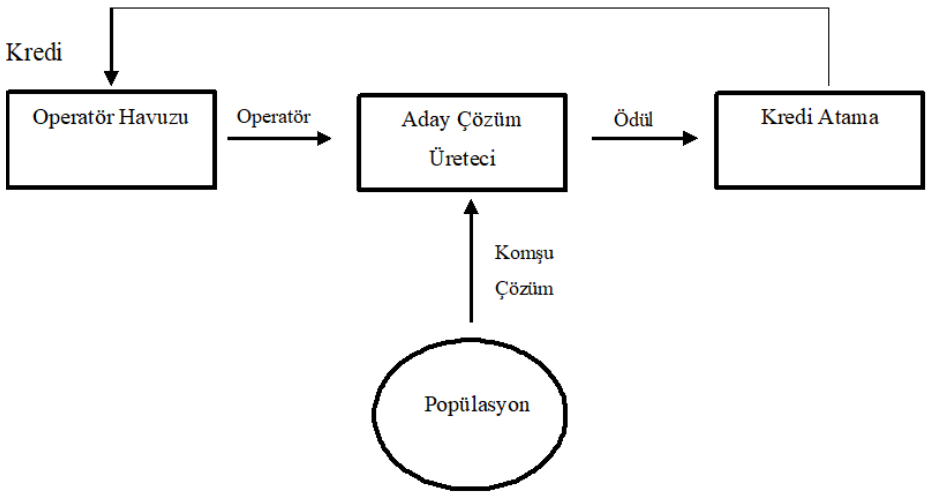
$$\varphi = \begin{cases} \varphi_{max} - \frac{(\varphi_{max} - \varphi_{min})}{\varphi_{max}} t, & fitness(X_i) < fitness(X_k) \\ 0, & otherwise \end{cases} \quad (4.7)$$

BÖLÜM 5

ADAPTİF İKİLİ YAPAY ARI KOLONİSİ ALGORİTMASI

Arama uzayının çeşitlendirilmesi maksadıyla tek operatör kullanımının yerine çok operatörlü bir çözüme gidilmiştir. Bu ikili operatörlerin (BinABC, DisABC ve IbinABC) birarada kullanılması için operatörlerin olasılıksal başarımını ifade eden bir seçim mekanizmasına ihtiyaç vardır. Bu evrimsel mekanizma adaptif mekanizma olarak adlandırılır.

Adaptif yapının çalışma mekanizması Şekil 5.1'de verilmiştir. Bu şekle göre operatör havuzundan seçilen ikili operatörler ile popülasyondan alınan komşu çözümler vasıtasıyla aday çözümler üretilir. Ardından her operatör için başarı değerlendirmesi için bir ödül değeri atanır. Sonrasında bu ödül değerlerine göre kredilendirme yapılır. Ardından operatörler bu kredilere göre bazı yaklaşımlar vasıtası ile seçilir. Bu operatör seçimi yaklaşımları Olasılık Eşleme (OE), Adaptif Kovalama (AK) ve Üst Güven Sınırı (ÜGS) yöntemidir.



Şekil 5.1. Adaptif mekanizma.

İkili yapay arı kolonisi içerisine adaptif bir yapıya eklemek için iki temel problemi yanıtlamak gerekmektedir.

Bunlardan ilki operatörlerin başarısının nasıl değerlendirileceği ve atanacak kredinin nasıl belirleneceğidir. İkinci problem ise, yeni çözüm üretilmesi için operatörlerin mevcut kredilere göre nasıl seçileceğidir. Her iki problemde kendi içerisinde farklı problemler barındırmaktadır.

5.1. OPERATÖR BAŞARISI

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 (miktarı) Eşitlik (5.1)'e göre belirlenmektedir.

$$\text{ödül} = \frac{PD}{GB}(f(x') - f(x)) \tag{5.1}$$

Eşitlik (5.1)'de; 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. Eşitlik (5.2) kullanılarak kredi değeri hesaplanmaktadır.

$$kredi_{i,t+1} = (1-a)kredi_{i,t} + a * \text{ödül}_{i,t} \quad , \; i = 1,2,\dots,K \tag{5.2}$$

Eşitlik (5.2)'ye göre, $kredi_{i,t+1}$ve $kredi_{i,t}$, değeri sırayla *t + 1* ve *t* zamanında güncellenmiş ve mevcut kredi seviyeleri iken, K havuzdaki operatör sayısı ve α∈ [0,1] adaptasyon faktörüdür. Durgut vd. (2020) çalışması incelendiğinde, anlık ödül

kullanımının yerine ortalama ve en yüksek ödül değerlerinin kredi atamasında kullanılması daha faydalı sonuçlar üretmektedir [7].

5.2. OPERATÖR SEÇİMİ YAKLAŞIMLARI

5.2.1. Olasılık Eşleme

Her bir operatöre ait kredi değerleri belirlendikten sonra operatör seçimi için kullanılabilecek farklı yaklaşımlar mevcuttur. Bunlardan ilki Olasılık Eşleme yaklaşımıdır. Bu yaklaşımda operatörlerin seçilme olasılığı kredi değerlerine göre dağıtılır. Eşitlik (5.2) kullanılarak her bir olasılığın o iterasyondaki seçilme olasılığı belirlenir.

$$p_{i,t} = p_{min} + (1 - (K-1) * p_{min}) \frac{Kredi_{i,t}}{\sum_{j=1}^{K} Kredi_{j,t}} \tag{5.2}$$

Eşitlik (5.2)'deki K değişkeni kullanılacak operatör sayısını, p_{min} en kötü operatöre atanacak minimum olasılık değerini, $kredi_i$ ise i. operatörün kredi değerini ifade etmektedir.

5.2.2. Adaptif Kovalama

Bir diğer operatör seçimi yöntemi olan Adaptif Kovalama 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 (5.3)'te verilmiştir.

$$p_{i,t+1} = \begin{cases} p_{i,t} + \beta(p_{max} - p_{i,t}), & eğer\ i_t = i_t^* \\ p_{i,t} + \beta(p_{min} - p_{i,t}), & aksi\ durumda \end{cases} \tag{5.3}$$

Eşitlik (5.3)'te β 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.

5.2.3. Üst Güven Sınırı

Üst Güven Sınırı yaklaşımında ise az fırsat bulan operatörlere daha fazla fırsat sunmak amacıyla kullanılmaktadır. Yöntem Eşitlik (5.4)'e göre operatörlere seçim olasılığı atamaktadır.

$$p_{i,t+1} = \begin{cases} 1-(K-1)p_{min}, & eğer\ i_t = i_t^* \\ p_{min}, & aksi\ durumda \end{cases} \tag{5.4}$$

i_t^* ise Eşitlik (5.5)'e göre belirlenmektedir. Burada *C* ayarlama katsayısı, *n* ise seçilme sayılarıdır.

$$i_t^* = \underset{i=1,..,K}{\text{argmax}}\{kredi_{i,t} + C\sqrt{\frac{2log\sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}\} \tag{5.5}$$

Operatörlere olasılık değeri atanmasından sonra rulet tekerine göre seçim yöntemi uygulanarak, operatör seçim işlemi gerçekleştirilmektedir.

5.3. ÇALIŞMA ŞEKLİ VE ADIMLARI

Algoritma, baslangıç parametrelerinin (limit, popülasyon boyutu vs.) belirlenmesiyle başlar. Geçerli ve rastgele çözüm içeren bireyler ile ilk popülasyon olusturulur 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 çalısma süresi, maksimum iterasyon sayısı veya maksimum fonksiyon değerlendirme sayısı olabilir. Adil bir karsılastırma için yöntemden bağımsız olarak maksimum fonksiyon değerlendirme sayısı sıklıkla kullanılmaktadır [30].

Üç temel fazdan ilki olan işçi arı fazında ilk olarak her birey 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 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 gerçeklestikten sonra, bilgi güncellemeleri gerçeklesir. İlk olarak elde edilen ödül bilgilerine göre her operatör için kredi atamaları yapılır. Yöntem içerisindeki en iyi çözüm saklanır. Kâşif arı fazında ise, limit değerini asan ilk bireye ait çözüm yeni, rastgele ve geçerli bir çözüm ile değistirilir. Adaptif YAK çalışma adımları maddeler halinde aşağıda verilmiştir.

- Başlangıç popülasyonunu oluştur.
- Durdurma kriteri sağlanana kadar,
    - Kullanılacak operatörü 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 pozitif ise toplam ödüle ekle.
    - Her besin kaynağı için olasılık değerini hesapla.
    - Olasılık değerine gör besin kaynağı seç.
    - Kullanılacak operatörleri 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.

BÖLÜM 6

DENEYSEL ÇALIŞMALAR

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 [31]. 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 ilgili çalışmalar incelenebilir [31, 32, 33].

Yöntemin geliştirilmesinde ve deneysel çalışmaların yapılmasında kullanılan programlama dili Python olup her işlev (YAK kodu, operatör seçimi, her ikili operatörün olduğu sınıflar, problem tanımlanması, parametre ayarlama, sonuçların çıkarımı) için sınıf içerikli kütüphaneler yazılmıştır. Matematiksel hesaplamalar için ise Numpy kütüphanesi kullanılmıştır.

6.1. VERİ SETİ

Yöntemlerin başarısının karşılaştırılabilmesi için He vd. (2018) 30 problem örneği içeren veri kümesi oluşturmuşlardır [31]. 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 ayırmışlardır ve sayıları 100'den 500'e kadar artmaktadır. Kullanılan örnek veri kümesi Çizelge 6.1'de verilmiştir.

Çizelge 6.1. Kullanılan problem örnekleri.

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

6.2. ADAPTİF OPERATÖR SEÇİMİ

Yöntem 3 farklı adaptif operator seçimi ile test edilmiştir. Bunlar; Olasılık Eşleme (OE), Adaptif Kovalama (AK), Üst Güven Sınırı (ÜGS) olarak belirlenmiştir.

Ödül mekanizması olarak pencere boyunca ortalama ve en yüksek ödül miktarları karşılaştırılmıştır. Pencere boyutu olarak 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 (Pmin) ise 0.1 ve 0.2 değerleri için test edilmiştir. Yöntem parametresi için 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.

Çizelge 6.2'de üç yöntem üzerinden parametre ayarlama için elde edilen sonuçlar görülmektedir.

Çizelge 6.2'ye göre; AK için en iyi konfigürasyonların; ortalama ödül, pencere uzunluğu için 5, minimum olasılık değeri (*Pmin*) için 0.2 ve *Alpha* katsayısı için 0.9 değerleri olduğu görülmektedir. OE için ise AK'dan farklı olarak pencere uzunluğunun 5 olduğu görülmektedir. ÜGS ise diğer yöntemlerden farklı olarak daha küçük *Alpha* değeri için en başarılı sonuçları üretmiştir.

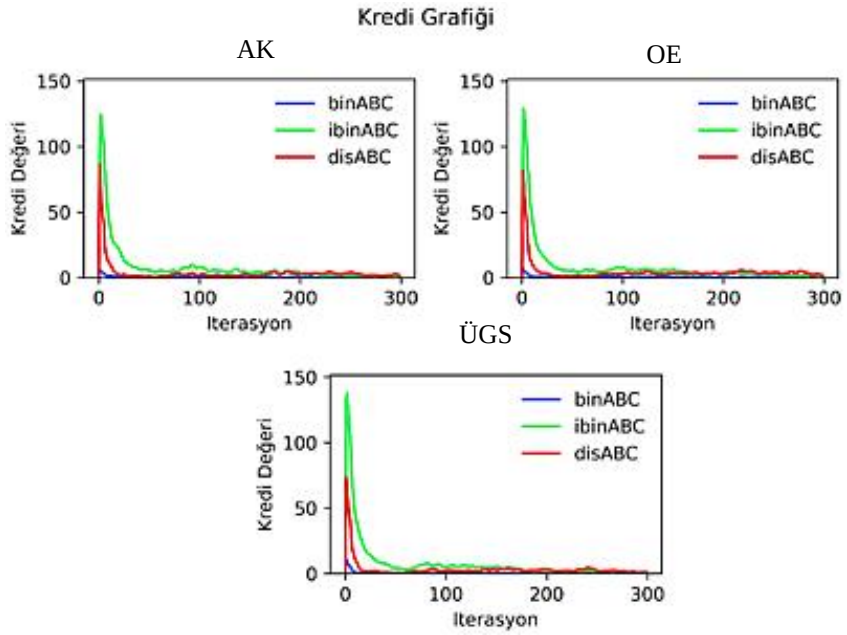
Bu üç yöntem arasında en iyi ortalama değeri OE ile üretildiği için sonraki deneylerde bu yöntem kullanılmıştır.

Çizelge 6.2. Parametre ayarlama için elde edilen sonuçlar.

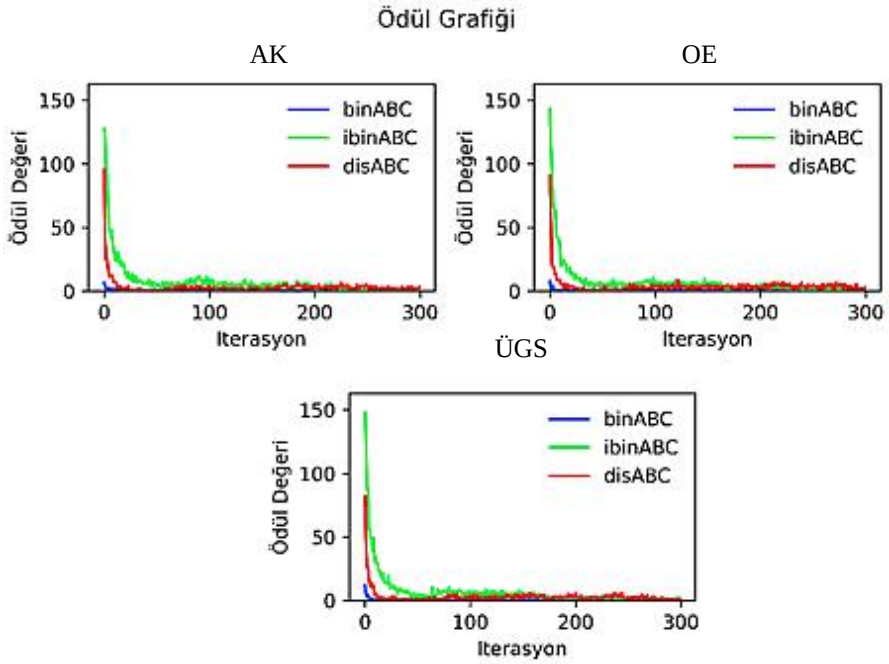
				AK			**OE**			**ÜGS**	
Ödül	Pencere	Pmin	Alpha	Yüksek	Ortalama		Yüksek	Ortalama		Yüksek	Ortalama
Ortalama	5	0.1	0,1	10998	10658,63		11113	10719,8		10948	10650,2
			0,5	11012	10676,30		10961	10692,13		10892	10686,57
			0,9	10735	10638,30		11023	10693,8		11054	10693,07
		0.2	0,1	11039	10670,40		10889	10700,8		**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 6.1'de yöntemlerin (OE için elde edilmiş en iyi konfigürasyon için) zamana bağlı olarak verilmiş kredi ortalama değerleri ve Şekil 6.2'de ödül değerleri verilmiştir. Başlangıçta daha çok başarılı çözüm üretilebildiğinden 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

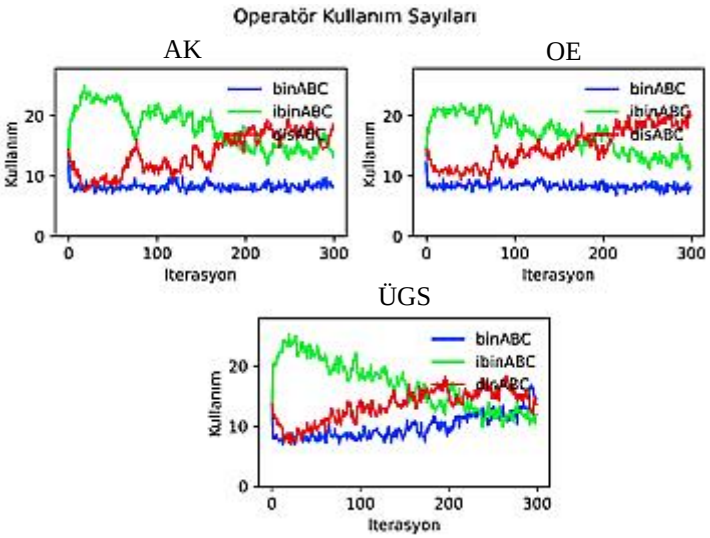


Şekil 6.1. Operatör seçim yöntemlerinin zamana bağlı kredi değerleri.

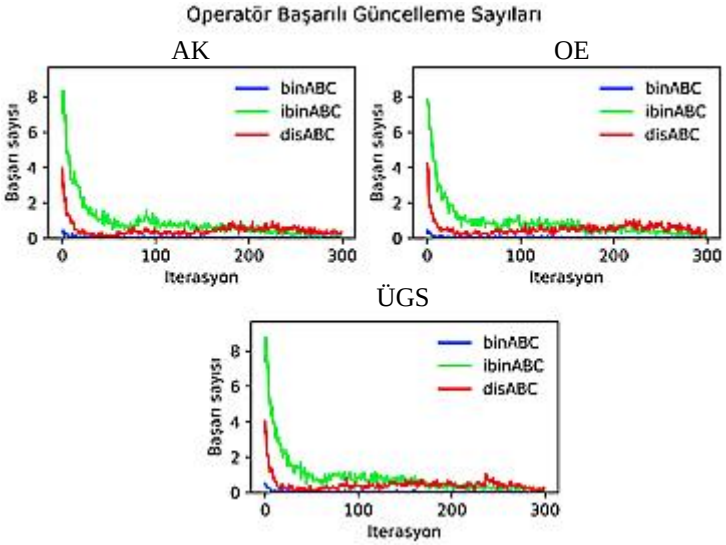


Şekil 6.2. Operatör seçim yöntemlerinin zamana bağlı ödül değerleri.

Şekil 6.3 ve Şekil 6.4'te 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. AK yaklaşımı kredisine göre bu operatöre diğer yaklaşımlarda daha çok fırsat vermiştir. ÜGS algoritmasında geçişler daha yumuşak olmasına karşın, en sert geçişler AK yaklaşımındadır. OE yaklaşımında 100. İterasyondan sonra disABC daha çok öne çıkmaya başlamıştır.



Şekil 6.3. Operatör kullanım sayılarının zamana göre değişimi.



Şekil 6.4. Operatör başarı sayılarının zamana göre değişimi.

Önerilen yaklaşım (AYAK) ile elde edilmiş sonuçlar literatürde var olan diğer yöntemler ile karşılaştırmalı olarak ek açıklamalar bölümünde Çizelge Ek A.1. ‘de verilmiş olup, karşılaştırılan yöntemlere ait değerler [31] ve [32] çalışmalarından doğrudan alınmıştır. Bu çizelgede 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 anlamlılığı kolaylaştırmak adına Çizelge 6.3’te ilk problem grubu için algoritmaların başarı sıralamaları verilmiştir. Önerilen adaptif operatör seçimi yaklaşımı hem diğer YAK (ABC) düzenlemelerinden hem de literatürdeki güncel çalışmalardan ortalama başarı sırasına göre birinci sıradadır. 10 farklı örnek için 7 kez en iyi çözümleri sunmuş 3 kez ise ikinci sırada kalmıştır. En yakın rakibi GPSO olmuştur.

Çizelge 6.3. İlk grup problem örnekleri (G1_1 – G1_10) için algoritmaların performansları.

P. No	A-SUKP	GA	BABC	ABCBin	binDE	GPSO*	GPSO	AYAK
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

Çizelge 6.4’te, ikinci grup problem örnekleri için algoritmaların performansları karşılaştırılmıştır. Önerilen çalışma bu problem örnekleri içinde en başarılı algoritma olmuştur. 10 farklı örnek için 6 kez en iyi çözümleri sunmuş, 4 kez ise ikinci sırada kalmıştır. En yakın rakibi GPSO olmuştur.

Çizelge 6.4. İkinci grup problem örnekleri (G2_1 – G2_10) için algoritmaların performansları.

P. No	A-SUKP	GA	BABC	ABCBin	binDE	GPSO*	GPSO	AYAK
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

Çizelge 6.5'te ise, üçüncü grup problem örnekleri için algoritmaların performansları karşılaştırılmıştır. Önerilen çalışma bu problem örnekleri içinde 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.

Çizelge 6.5. Üçüncü grup problem örnekleri (G3_1 – G3_10) için algoritmaların performansları.

P. No	A-SUKP	GA	BABC	ABCBin	binDE	GPSO*	GPSO	AYAK
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

BÖLÜM 7

SONUÇ

Metasezgisel optimizasyon algoritmaları, NP-hard problemlerin çözümü için sıklıkla kullanılmaktadır. Bu algoritmaların tüm arama sürecinde tek bir komşu çözüm üretme operatörü kullanması keşif ve sömürü fazlarındaki dengeyi baskılamaktadır. Ayrıca, algoritmanın başarısı tek bir operatörün başarısı ile sınırlandırılmaktadır. Tek bir operatör kullanımının yerine birden çok operatörün birlikte kullanılması adaptif operatör seçim mekanizmaları ile daha zeki olarak kontrol edilebilmektedir.

Bu çalışmada üç farklı operatör seçiminin küme birleşim sırt çantası problemi üzerindeki etkisi incelenmiş ve en iyi operatör olarak Olasılık Eşleme belirlenmiştir. En iyi konfigürasyonda son 5 iterasyon boyunca elde edilmiş ödüllerin ortalama değerleri kullanılması halinde en iyi çözümler elde edilmiştir. Geliştirilen yöntem literatürdeki güncel algoritmalar ile karşılaştırılmıştır. Karşılaştırma üç grup ve toplam otuz problem örneği üzerinde gerçekleştirilmiştir. Karşılaştırma sonuçlarına göre adaptif operatör seçimi barındıran Adaptif Yapay Arı Kolonisi algoritması (AYAK), aynı probleme uygulanmış metasezgisel yöntemlerden daha başarılı sonuçlar üretmektedir.

Sonraki çalışmalarda; operatör havuzu arttırılarak seçim işlemi gerçekleştirilmesi planlanmaktadır.

KAYNAKLAR

1. Dorigo, M., Maniezzo, V. and Colorni, A., “Ant system: Optimization by a colony of cooperating agents.”, ***IEEE Transactions on Systems, Man, and Cybernetics,*** 26(1): 29– 41 (1996).

2. 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).

3. Holland, J., "Adaptation in natural and artificial systems: an introductory analysis with application to biology.", ***Control and artificial intelligence*** (1975).

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

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

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

7. Durgut, R., "Improved binary artificial bee colony algorithm.", ***arXiv preprint arXiv***:2003.11641 (2020).

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

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

10. Arulselvan, A., “A note on the set union knapsack problem.”, ***Discrete Appl. Math.,*** 169: 214–218 (2014).

11. Kellerer, H., Pferschy, U., Pisinger, D., “Knapsack Problems.”, ***Springer***, Berlin (2004).

12. Khuler, S., Moss, A., Naor, J., "The budgeted maximum coverage problem", ***Inform. Process. Lett.,*** 70: 39–45 (1999).

13. Tang, C.S. and Denardo, E.V., "Models arising from a flexible manufacturing machine: Minimizing the number of switching instants", ***OR,*** 36: 778–784 (1988).

14. Goldschmidt, O., Nehme, D. and Yu, G., "On a Generalization of the Knapsack Problem with Applications to Flexible Manufacturing Systems and Database Partitioning.", ***Graduate School of Business, University of Texas at Austin***, 92193: 3-7 (1992).

15. S. Navathe, S. Ceri, G. Wiederhold, J. Dou, "Vertical partitioning algorithms for database design", ***ACM Trans. Database Syst.,*** 9: 680–710 (1984).

16. Tu, M., Xiao, L., "System resilience enhancement through modularization for large scale cyber systems", ***IEEE/CIC International Conference on Communications,*** China (2016)**.**

17. Gass, S.I., Assad, A. A., "An Annotated Timeline of Operations Research: An Informal History", ***International Series in Operations Research & Management Science, Springer*** (2004).

18. Berberler, M.Ş., "Sırt Çantası Problem Türleri ve Uygulamaları", ***Ege Üniversitesi, Fen Bilimleri Enstitüsü,*** Doktora tezi (2009).

19. Chapra, S.C., Canale, R. P., "Yazılım ve Programlama Uygulamalarıyla Mühendisler için Sayısal Yöntemler", ***Literatür Yayıncılık,*** İstanbul (2008).

20. Goldschmidt, O., Nehme, D. and Yu, G., "Note: On the set-union knapsack problem.", ***Naval Research Logistics (NRL)***, 41(6): 833-842 (1994).

21. Blum, C. and Roli, A., "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", ***ACM Computing Surveys,*** 35(3): 268-308 (2003).

22. Kaya, B. ve Eke, İ., "Yapay Arı Kolonisi Algoritması ile Yapılan Geliştirmeler ve Sonuçları", ***Verimlilik Dergisi T. C. Sanayi ve Teknoloji Bakanlığı Yayını,*** 1 (2020).

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

24. Waibel, M., "Divison of Labour and Colony Efficiency in Social Insects.", ***Proceedings of the Royal Society B***., 273: 1815-23 (2006).

25. Akay, B.,” Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi, Ek-5 ABC Algoritması Kodları.”, Doktora Tezi, ***Erciyes Universitesi Fen Bilimleri Enstitüsü,*** 10-25 (2009).

26. Küçüksille, E.U. ve Tokmak, M., " Yapay Arı Kolonisi Algoritması Kullanarak Otomatik Ders Çizelgeleme" ***Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü Dergisi***, 15(3): 203-210 (2011).

27. Tereshko, V., “Reaction-Diffusion Model of a Honeybee Colony’s Foraging Behaviour, in: Parallel Problem Solving from Nature PPSN VI, Lecture Notes in Computer Science”, ***Springer-Verlag,*** Berlin, 1917:807–816 (2000).

28. Tereshko, V., A. Loengarov, “Collective Decision-Making in Honeybee Foraging Dynamics”, ***Computing and Information Systems Journal,*** 9:(3) (2005).

29. Karaboğa, D., “Yapay Zekâ Optimizasyon Algoritmaları”, ***Nobel Akademik Yayıncılık***, 202-221 (2011).

30. Ezugwu, A.E., Adeleke, O.J., Akinyelu, A.A., and Viriri, S., “A conceptual comparison of several 473 metaheuristic algorithms on continuous optimisation problems. Neural Computing and Applications”, 474, 32:(10), 6207-6251 (2020).

31. He, Y., "A novel binary artificial bee colony algorithm for the set-union knapsack problem.", ***Future Generation Computer Systems***, 78: 77-86 (2018).

32. Ozsoydan, F. B. and Baykasoglu, A., "A swarm intelligence-based algorithm for the set-union knapsack problem.", ***Future Generation Computer Systems***, 93: 560-569 (2019).

33. Ozsoydan, F. B., "Artificial search agents with cognitive intelligence for binary optimization problems.", ***Computers & Industrial Engineering***, 136: 18-30 (2019).

EK AÇIKLAMALAR A.

AYAK İLE LİTERATÜRÜN KIYASLANMASI

Çizelge Ek A.1. Tüm problem gruplarının algoritmalara göre standart sapması, ortalama ve en iyi sonuçları.

	A-SUKP			GA			BABC			ABCBin			binDE			GPSO*			GPSO			AYAK		
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	23,.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

ÖZGEÇMİŞ

İlim Betül YAVUZ 1994 yılında Kastamonu'da doğdu. İlk ve orta öğrenimini Kırıkkale'de tamamladı. Kırıkkale Anadolu Öğretmen Lisesi'nden 2012 yılında mezun oldu. 2012 yılında Karabük Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Lisans öğrenimine başlayıp 2018 yılında mezun oldu. 2019 yılında Karabük Üniversitesi Bilgisayar Mühendisliği dalında başladığı Yüksek Lisans öğrenimini 2021 yılında tamamladı.

ADRES BİLGİLERİ

Adres : Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Balıklarkayası Mevkii / KARABÜK
Tel : (553) 467 0748
E-posta : ilimbyvz@gmail.com