AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

Ahsen KÜÇÜK

ÜÇ BOYUTLU KUTU PAKETLEME PROBLEMLERİNİN ÇÖZÜMÜNDE METASEZGİSEL ALGORİTMALARIN KIYASLANMASI

Ekonometri Ana Bilim Dalı
Doktora Tezi

Antalya, 2023

AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

Ahsen KÜÇÜK

ÜÇ BOYUTLU KUTU PAKETLEME PROBLEMLERİNİN ÇÖZÜMÜNDE METASEZGİSEL ALGORİTMALARIN KIYASLANMASI

Danışman
Prof. Dr. Emre İPEKÇİ ÇETİN

Ekonometri Ana Bilim Dalı
Doktora Tezi

Antalya, 2023

Akdeniz Üniversitesi

Sosyal Bilimler Enstitüsü Müdürlüğüne,

Ahsen KÜÇÜK'ün bu çalışması jürimiz tarafından Ekonometri Ana Bilim Dalı Doktora Programı tezi olarak kabul edilmiştir.

Başkan	: Prof.Dr.İpek Deveci Kocakoç	(İmza)
Üye (Danışmanı)	: Prof. Dr. Emre İPEKÇİ ÇETİN	(İmza)
Üye	: Doç.Dr. Pınar KAYA SAMUT	(İmza)
Üye	: Doç.Dr. Osman PALA	(İmza)
Üye	: Doç.Dr. Ömür TOSUN	(İmza)

Tez Başlığı: Üç Boyutlu Kutu Paketleme Problemlerinin Çözümünde Metasezgisel Algoritmaların Kıyaslanması

Onay : Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Tez Savunma Tarihi : 25/08/2023

Mezuniyet Tarihi : 24/08/2023

(İmza)

Prof. Dr. Engin KARADAĞ

Müdür

AKADEMİK BEYAN

Doktora Tezi olarak sunduğum "Üç Boyutlu Kutu Paketleme Problemlerinin Çözümünde Metasezgisel Algoritmaların Kıyaslanması" adlı bu çalışmanın, akademik kural ve etik değerlere uygun bir biçimde tarafımca yazıldığını, yararlandığım bütün eserlerin kaynakçada gösterildiğini ve çalışma içerisinde bu eserlere atıf yapıldığını belirtir; bunu şerefimle doğrularım.

Ahsen KÜÇÜK

T.C.
AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

18/08/2023

TEZ ÇALIŞMASI ORİJİNALLİK RAPORU BEYAN BELGESİ

Öğrenci Bilgileri	
Adı-Soyadı	Ahsen KÜÇÜK
Öğrenci Numarası	20165245001
Anabilim Dalı	Ekonometri
Programı	Doktora
Danışman Öğretim Üyesi Bilgileri	
Unvanı, Adı-Soyadı	Prof. Dr. Emre İPEKÇİ ÇETİN
Doktora Tez Başlığı	Üç Boyutlu Kutu Paketleme Problemlerinin Çözümünde Metasezgisel Algoritmaların Kıyaslanması
Turnitin Bilgileri	
Ödev Numarası	2147470096
Rapor Tarihi	18/08/2023
Benzerlik Oranı	Alıntılar hariç: %8 Alıntılar dahil: %8

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE,

Yukarıda bilgileri bulunan öğrenciye ait tez çalışmasının a) Kapak sayfası, b) Giriş, c) Ana Bölümler ve d) Sonuç kısımlarından oluşan toplam 112 sayfalık kısmına ilişkin olarak Turnitin adlı intihal tespit programından Sosyal Bilimler Enstitüsü Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Uygulama Esaslarında belirlenen filtrelemeler uygulanarak yukarıdaki detayları verilen ve ekte sunulan rapor alınmıştır.
Danışman tarafından uygun olan seçenek işaretlenmelidir:
(X) Benzerlik oranları belirlenen limitleri aşmıyor ise:
Yukarıda yer alan beyanın ve ekte sunulan Tez Çalışması Orijinallik Raporunun doğruluğunu onaylarım.
( ) Benzerlik oranları belirlenen limitleri aşıyor, ancak tez/dönem projesi danışmanı intihal yapılmadığı kanısında ise:
Yukarıda yer alan beyanın ve ekte sunulan Tez Çalışması Orijinallik Raporunun doğruluğunu onaylar ve Uygulama Esaslarında öngörülen yüzdelik sınırlarının aşılmasına karşın, aşağıda belirtilen gerekçe ile intihal yapılmadığı kanısında olduğumu beyan ederim.

Gerekçe:

Benzerlik taraması yukarıda verilen ölçütlere uygun olarak tarafımca yapılmıştır. İlgili tezin orijinallik raporunun uygun olduğunu beyan ederim.

Danışman Öğretim Üyesi
Unvanı, Adı-Soyadı
Prof. Dr. Emre İPEKÇİ ÇETİN

İmza

İ Ç İ N D E K İ L E R

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

KISALTMALAR LİSTESİ

1-D BPP	Bir Boyutlu Kutu Paketleme Problemi (1 Dimensional Bin Packing Problem)
2-D BPP	İki Boyutlu Kutu Paketleme Problemi (2 Dimensional Bin Packing Problem)
3-D BPP	Üç Boyutlu Kutu Paketleme Problemi (3 Dimensional Bin Packing Problem)
ABC	Yapay Arı Kolonisi (Artificial Bee Colony)
ACO	Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
BBC	Büyük Patlama – Büyük Çöküş (Big-Bang Big-Crunch)
BBO	Biyocoğrafya Tabanlı Optimizasyon (Biogeography-Based Optimizer)
BPP	Kutu Paketleme Problemi (Bin Packing Problem)
C&P	Kesme ve Paketleme (Cutting and Packing)
CFO	Merkezi Kuvvet Optimizasyonu (Central Force Optimization)
CLP	Konteyner Yükleme Problemi (Container Loading Problem)
CS	Guguk Kuşu Arama Algoritması (Cuckoo Search Algorithm)
CSS	Yüklü Sistem Arama Algoritması (Charged System Search)
CPP	Konteyner Paketleme Problemi (Container Packing Problen)
DE	Diferansiyel Evrim (Differential Evolution)
FA	Ateş böceği Algoritması (Firefly Algorithm)
FOA	Meyve Sineği Optimizasyon Algoritması (Fruit fly Optimization Algorithm)
GA	Genetik Algoritma (Genetic Algortihm)
GP	Genetik Programlama (Genetic Programming)
GSA	Yerçekimsel Arama Algoritması (Gravitational Search Algorithm)
GSO	Grup Arama Optimize Edici (Group Search Optimizer)
HBA	Bal Porsuğu Algoritması (Honey Badger Algorithm)
HS	Harmoni Arama (Harmony Search)
KP	Sırt Çantası Problemi (Knapsack Problem-KP)
NFL	Ücretsiz Öğle Yemeği Yoktur Teoremi (No Free Lunch)
PSO	Parçacık Sürüsü Optimizasyonu (Particle Swarm Optimization)
TLBO	Öğretme Ve Öğrenme Tabanlı Optimizasyon (Teaching Learning Based Optimization)
TS	Tabu Arama (Tabu Search)
TSO	Tuna Balık Sürüsü Optimizasyonu (Tuna Swarm Optimization)

ÖZET

Kutu paketleme problemleri, maliyetleri azaltmak ve tesislerle ekipmanların daha iyi kullanılmasını sağlamak için yük taşımacılığı ve tedarik zinciri sistemlerinin planlanmasında merkezi bir rol oynamaktadır. Yönetim süreçlerinde, operasyonel kararları desteklemek veya daha karmaşık stratejik karar süreçlerinin bir bileşeni olarak ortaya çıkarlar. Bu nedenle, büyük örneklerle başa çıkabilecek çözüm yöntemlerine ihtiyaç bulunmaktadır.

Bu doktora tezi, 3 boyutlu kutu paketleme problemlerinin çözümünde kullanılan metasezgisel algoritmaların kıyaslanmasını amaçlamaktadır. Problemin çözümü için 8 sınıf, 32 alt grup ve 320 örnekten oluşan bir veri seti kullanılmıştır.

Tezde, dört farklı metasezgisel algoritma kullanılarak 3 boyutlu kutu paketleme problemlerinin çözümüne odaklanılmaktadır. Bu algoritmalar Ateş Böceği Algoritması, Guguk Kuşu Arama Algoritması, Tuna Balık Sürüsü Optimizasyonu ve Bal Porsuğu Algoritması olarak adlandırılmaktadır. Her bir algoritma, doğal davranışı taklit eden sürü zekâsı tabanlı optimizasyon yaklaşımları sunmaktadır. Tuna Balık Sürüsü Optimizasyonu ve Bal Porsuğu Algoritmasının daha önce kutu paketleme problemlerinde kullanılmamış olması sebebiyle literatüre katkı sağlanması hedeflenmektedir. Ateş Böceği Algoritması, Guguk Kuşu Arama Algoritması ise yaygın olarak bir ve iki boyutlu kutu paketleme problemlerine uygulandığından zorluk derecesi daha yüksek olan üç boyutlu problemin çözüm değerlerinin de katkı niteliği sağlayacağı düşünülmektedir.

Anahtar Kelimeler: Kutu paketleme problemleri, Metasezgisel algoritmalar, Ateş Böceği Algoritması, Guguk Kuşu Arama Algoritması, Tuna Balık Sürüsü Optimizasyonu, Bal Porsuğu Algoritması

ABSTRACT

A COMPARISON OF METAHEURISTIC ALGORITHMS FOR SOLVING THREE-DIMENSIONAL BIN PACKING PROBLEMS

Multi-dimensional bin packing problems play a central role in the planning of freight transportation and supply chain systems to reduce costs and improve the utilization of facilities and equipment. They arise as a component of operational decision support or as part of more complex strategic decision processes. Therefore, there is a need for solution methods capable of handling large instances.

This doctoral thesis aims to compare metaheuristic algorithms used in solving three-dimensional bin packing problems. A data set consisting of 8 classes, 32 subgroups and 320 samples was used to solve the problem.

The thesis focuses on solving three-dimensional bin packing problems using four different metaheuristic algorithms: Firefly Algorithm, Cuckoo Search Algorithm, Tuna Swarm Optimization, and Honey Badger Algorithm. Each algorithm offers swarm intelligence-based optimization approaches that mimic natural behavior. Tuna Swarm Optimization and Honey Badger Algorithm contribute to the literature as they have not been previously applied to bin packing problems. The Firefly Algorithm and Cuckoo Search Algorithm, being commonly used for one and two-dimensional bin packing problems, are expected to provide valuable insights into the solution quality of the more challenging three-dimensional problem.

Keywords: Bin packing problems, Metaheuristic algorithms, Firefly Algorithm, Cuckoo Search Algorithm, Tuna Swarm Optimization, Honey Badger Algorithm

GİRİŞ

Günümüz endüstrisinde maliyetleri azaltmak, lojistik operasyonları optimize etmek ve müşteri memnuniyetini artırmak önemli hedefler arasında yer almaktadır. Ürünlerin hasar riskini minimize etmek, depolama alanını etkin bir şekilde kullanmak, taşıma kapasitesini en üst düzeye çıkarmak ve teslimat sürelerini kısaltmak gibi başarı faktörleri ortaya çıkmaktadır. Bu nedenle, kutu paketleme problemleri endüstri tarafından büyük bir ilgi görmektedir.

Üç boyutlu kutu paketleme problemleri, konteyner gemi yüklemesi, palet yüklemesi, uçak kargo yönetimi, depo yönetimi gibi çeşitli alanlarda pratik uygulamalara sahiptir. Malzemelerin ve taşıma kapasitesinin etkin kullanımı, işletmeler için rekabet avantajı sağlamaktadır. Modern ekonomi, büyük miktarlarda malzeme ve ürünün kıtalar arası nakliyesini gerçekleştiren çok uluslu holdingler tarafından yönlendirilmektedir. Bu durum, maliyetleri düşürmenin bir yolu olarak tedarik zincirinin verimli yönetimine odaklanmaya neden olmuştur. Bu tez çalışmasında ise 3 boyutlu kutu paketleme problemlerine metasezgisel algoritmalarla çözümler geliştirilmesi amaçlanmaktadır.

Bu problem, sınırlı bir alanda optimum alan kullanımına odaklanan kombinatoryal bir optimizasyon problemidir. Bir konteynerin boşa giden hacmini en aza indirmek, daha düşük konteyner kiralama-satın alma ve nakliye maliyetleri sağlayabilir. Yöneticiler, tüm öğeleri minimum sayıda konteynere paketlemenin en iyi yolunu aramaktadır. Tez çalışması üç bölümden oluşmaktadır.

Birinci bölümünde kutu paketleme problemlerinin tanımına yer verilmiştir. Bu kapsamda, problemin kesme ve paketleme problemlerinin bir alt dalı olarak sınıflandırılması ele alınmış, problemin amaç fonksiyonu ve girdi özellikleri tanıtılmıştır. Ardından problemin bir, iki ve üç boyutlu olmak üzere türlerine değinilmiştir. Bu kısımda türlerin formülasyonu, örnekleri ve literatürü ele alınmıştır. İlgili problemler kısmında, 3 boyutlu kutu paketleme problemlerinin diğer kesme ve paketleme uygulamalarıyla benzerlikleri ve farklılıklarının vurgulanması hedeflenmiştir. Problemin bazı kaynaklarda sırt çantası problemlerinin veya konteyner yükleme problemlerinin alt başlığı olarak ele alınması sebebiyle aralarındaki temel ayrım incelenmiştir. Kutu paketleme problemleri, doğası gereği kombinatoryel optimizasyon problemleri grubuna girmektedir. Bu durum, problemin zorluk derecesinin (NP-hard) yüksek olduğunu göstermektedir ve daha iyi sonuçlar elde etmek için farklı çözüm yöntemlerinin geliştirilmesi ihtiyacını ortaya çıkarmaktadır.

İkinci bölümde, kutu paketleme problemlerinin çözümü için geliştirilmiş olan yöntemler ele alınmıştır. Bu yöntemler kesin, sezgisel ve metasezgisel yöntemler olmak üzere

3 başlık altında incelenmiş ve her yöntemin literatürü sunulmuştur. Sonraki kısımlarda da bu tez çalışmasında 3 boyutlu kutu paketleme problemlerinin çözümü için kullanılmış olan Ateş Böceği Algoritması, Guguk Kuşu Arama Algoritması, Tuna Balık Sürüsü Optimizasyonu ve Bal Porsuğu Algoritmaları açıklanmıştır. Yöntemlerin kronolojik sırayla tanıtılması hedeflenmiştir.

Üçüncü bölümde, 3 boyutlu kutu paketleme problemlerinin çözümünde metasezgisel yöntemlerin performanslarının araştırılması amaçlanmıştır. İlk olarak, uygulamada kullanılan veri seti tanıtılmıştır. 320 örneğin bulunduğu veri seti, 8 sınıf ve 32 alt gruptan oluşmaktadır. Veri setinin tanıtımı yapıldıktan sonra, ikinci bölümde tanıtılan 4 metasezgisel algoritmada kullanılan parametre değerlerine odaklanılmıştır. Parametre değerleri açıklandıktan sonra, nasıl bir yerleştirme stratejisinin izleneceği tanıtılmıştır. Yerleştirme stratejisi tanıtıldıktan sonra ise kesikli form üzerine yoğunlaşılmıştır. Kutu paketleme problemleri kesikli bir yapıya sahip optimizasyon problemleridir. Bu neden sürekli çözüm yapısını kullanan algoritmaların probleme uygulanması için bir dönüşüm yapılması gerekmetedir. Çalışmada kullanılan metasezgisel yöntemlerin performansı dokuz farklı durum altında değerlendirilerek çeşitli testler gerçekleştirilmiştir. Her bir durum farklı iterasyon ve popülasyon sayılarına bağlı farklı parametre kombinasyonlarına dayanmaktadır. Bu durumlar altında, çözüm sürelerinin minimum ve ortalama değerleri, doluluk oranlarının minimum ve ortalama değerleri, konteyner sayılarının minimum ve ortalama değerleri hesaplanmıştır. Üçüncü bölümün son kısmında, elde edilen sonuçlar tablolar aracılığıyla raporlanmış ve yorumlanmıştır. Her bir durumun performansı, kullanılan iterasyon ve popülasyon değerleriyle ilişkilendirilmiş ve sonuçlar üzerindeki etkileri ayrıntılı bir şekilde tartışılmıştır. Algoritmalar Python dilinde kodlanmıştır. Kullanılan yöntemler arasında istatistiksel açıdan anlamlı farklılıkların varlığı, Friedman testi ile değerlendirilmiştir. Bunun yanı sıra, bu anlamlı farklılıkların hangi yöntemler arasında olduğu, Wilcoxon testi kullanılarak detaylı bir şekilde incelenmiştir. Analizler IBM SPSS Statistics 23 programı kullanılarak yapılmıştır.

Bu çalışmada, problemin çözümünde kullanılan tuna balık sürüsü optimizasyonu ve bal porsuğu algoritmasının daha önce kutu paketleme problemlerinde kullanımına rastlanmaması sebebiyle, yeni yöntemlerin literatüre katkı sağlanması hedeflenmektedir. Bir diğer hedef; kutu paketleme problemlerinde daha önce kullanılmış olan ateş böceği algoritması ve guguk kuşu arama algoritmasının yeni algoritmalarla kıyaslanması ve elde edilen sonuçların sonraki çalışmalar için bir referans noktası olmasıdır.

BİRİNCİ BÖLÜM
KUTU PAKETLEME PROBLEMLERİ

Kutu paketleme problemi, endüstri ve bilgisayar bilimlerinde ortaya çıkan bir dizi problem için geliştirilmiş bir modeldir. Bu modelde farklı hacimlerdeki küçük kutular, belirli ölçülerdeki büyük kutulara en iyi biçimde yerleştirilmeye çalışılır. Amaç, kutularda kalan boşluğun en aza indirilmesi ve yerleştirmenin mümkün olan en az sayıda kutuya yapılmasıdır (Johnson, 1974).

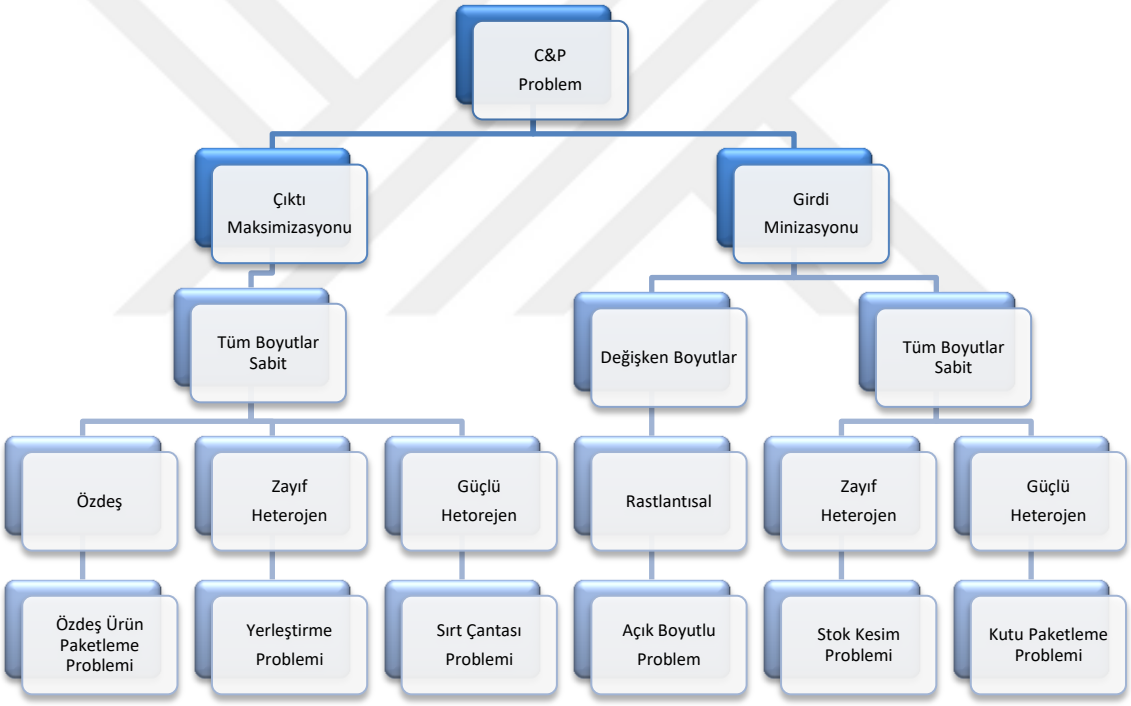
Konuyla ilgili ilk çalışmalar, paketleme ve ambalajlama (packing - packaging) olarak ele alınmış ve 1960'lardan bu yana birçok araştırmanın konusu olmuştur. Wilson'ın (1965) çalışmasında, tüketim malları depolama sistemleri incelenmiş ve maliyetlerin optimize edilmesi hedeflenmiştir. Elde edilen sonuçlara göre, mevcut ambalaj ve alan maliyetlerinden yılda %25'in üzerinde tasarruf sağlanabileceği görülmüştür. Moon ve Moser (1967), paketleme ve kaplama teoremleri üzerine çalışmışlardır. Meir ve Moser (1968), karelerin ve küplerin paketlenmesi konusunda çözüm önerilerinde bulunmuştur. Blaisdell ve Solomon (1970), uçakta rastgele sıralama ile paketleme ve Palásti varsayımı üzerine bir araştırma yapmıştır. Baranovskii (1971), sabit kavisli uzaylarda; paketlemeler, kaplamalar, bölmeler ve diğer dağılımlar hakkında çalışmıştır. Graham (1971), çok işlemcili anormalliklere ilişkin sınırlar ve ilgili paketleme algoritmaları üzerinde durmuştur.

Kutu paketleme (bin packing) ifadesi ise ilk olarak Johnson (1973)'ın doktora tezinde yer almıştır. Johnson'ın kavramsal çerçeveyi çizmesinin ardından, Coffman vd. (2004) ile hazırladıkları "Kutu Paketlemeye Giriş" başlıklı yayında 1960'lardan 2004'e kadar paketleme ile ilgili yapılmış çalışmaların detaylı bir derlemesi sunulmuştur. Yakın dönemde yapılan derleme çalışmalarında ise kutu boyutlarının ayrımı gözetilmiştir. Delorme vd. (2016), tek boyutlu kutu paketleme ve stok kesme problemlerinin kesin çözümü için geliştirilen önemli matematiksel modelleri ve algoritmaları ele almıştır. Prituja (2020) ise üç boyutlu kutu paketleme problemleriyle ilgili literatürün kapsamlı bir incelemesini sunmaktadır.

Bu bölümde ilk olarak kutu paketleme probleminin hangi sınıfta incelendiği ele alınmış, daha sonra türlerine değinilmiştir. Boyutlarına göre bir, iki ve üç boyutlu olmak üzere üç farklı türü olan problemin, bu tez çalışmasında 3 boyutlu hali üzerinde durulmuştur. Son olarak ilgili problemler kısmında 3 boyutlu kutu paketleme problemlerinin diğer kesme ve paketleme uygulamalarıyla benzerlikleri ve farklılıkları vurgulanmıştır.

1.1. Problemin Sınıflandırılması

Kutu paketleme problemlerinin (Bin Packing Problem-BPP) içinde bulunduğu, Kesme ve Paketleme (Cutting and Packing-C&P) Problemlerine ilişkin genel kabul görmüş olan sınıflandırma Wäscher vd. (2007) tarafından yapılmıştır. Bu sınıflandırma; Dyckhoff (1990)'da ifade edilen tipolojinin geliştirilmiş, farklı kategoriler ve yeni kriterler eklenmiş halidir. C&P, temelde iki farklı problem türüne dallanmaktadır. Bunlardan ilki maksimizasyon problemidir ve her bir konteyner içerisine yerleştirilen nesnelerin boşluk kullanma oranları ve faydaları gibi değerlerin maksimize edilmesi bu türe örnek teşkil eder. İkincisi ise minimizasyon problemidir, birden fazla konteynerin doldurulduğu durumlarda konteyner sayısının veya konteynerde kalan boşluğun minimize edilmesi bu gruba örnek gösterilebilir (Zhao vd., 2016). Şekil 1.1'de C&P problemlerine ilişkin temel problem türleri görülmektedir:



Şekil 1.1 Temel Problem Türleri

Kaynak: Wäscher vd., 2007

Bu sınıflamada bahsi geçen problem tipleri aşağıda özetlenmektedir (Wäscher vd., 2007):

- *Özdeş Ürün Paketleme Problemi*: Bu problem kategorisi, mümkün olan en fazla sayıda özdeş küçük öğenin, sınırları belirli büyük nesneler kümesine atanmasından oluşur. Tüm ürünlerin aynı olması nedeniyle, ürünler arasında bir seçim veya gruplama yapılmamaktadır. Başka bir deyişle, C&P problemlerinin genel yapısı, bir yerleşim problemine indirgenmiştir.

- *Yerleştirme Problemi*: Büyük nesneler kümesine atanan küçük öğelerin zayıf heterojen bir ürün yelpazesine sahip olduğu durumları tanımlar. Amaç, atanan küçük öğelerin toplam değerinin maksimize edilmesi veya alternatif olarak toplam firenin minimize edilmesidir (Bortfeldt ve Wäscher, 2013).

- *Sırt Çantası Problemi*: Güçlü heterojen çeşitliliğe sahip küçük parçaların, sınırlı bir büyük parçalar kümesine atanmasıdır. Küçük parçaların tümümün atanması mümkün olmadığı için atanan öğelerin değeri en üst düzeye çıkarılmalıdır.

- *Açık Boyutlu Problem*: Büyük nesnelerin en az bir boyutundaki uzantıları değişkendir. Amaç küçük parçaların yerleşiminde sınırlanmayan boyuta doğru genişlemenin minimize edilmesidir. Kâğıt veya çelik rulolarından yapılan kesimler bu tür problemlerin grubuna girmektedir (Wäscher vd., 2007).

- *Stok Kesim Problemi*: Zayıf heterojen yapıdaki küçük öğeler kümesinin tamamının, boyutları belirli büyük nesneler kümesine atanmasıdır. Açık boyut problemindeki gibi büyük nesnelerin bir boyutta genişlemesi söz konusu değildir. Tüm boyutlar sabittir. Ancak büyük nesnelerin çeşitliliği konusunda herhangi bir kısıtlama bulunmamaktadır.

- *Kutu Paketleme Problemi*: Güçlü heterojen çeşitlilikteki küçük parçalar kümesinin tümünün boyutları belirli büyük nesneler kümesine atanmasıdır (Bortfeldt ve Wäscher, 2013). Boyutlar ve amaç konusunda stok kesme problemi ile benzer özelliktedir. Ayrıldıkları nokta; BPP'de zayıf heterojen değil, güçlü heterojen küçük parçaların yer almasıdır.

1.2. Kutu Paketleme Probleminin Türleri

BPP'yi, paketlenecek nesne boyutuna, nesne çeşitliliğine, nesne şekillerine, kullanılan amaç türüne ve kutu çeşitliliğine göre sınıflandırmak mümkündür. Kutu çeşitliliğine göre problemin üç farklı boyutu bulunmaktadır. Bu kısımda her üç boyut için; problemin tanımı, matematiksel modeli, örnekleri ve ilgili literatürü sunulmuştur.

Bu tez çalışmasında, birbirlerinden farklı ölçülerde sınırlı sayıda kutunun, birbiriyle aynı ölçülerdeki sınırsız sayıda konteynere doldurulduğu üç boyutlu kutu paketleme problemleri ele alınmaktadır. Bu problem; sabit konteyner ölçülü üç boyutlu kutu paketleme problemi olarak da ifade edilmektedir.

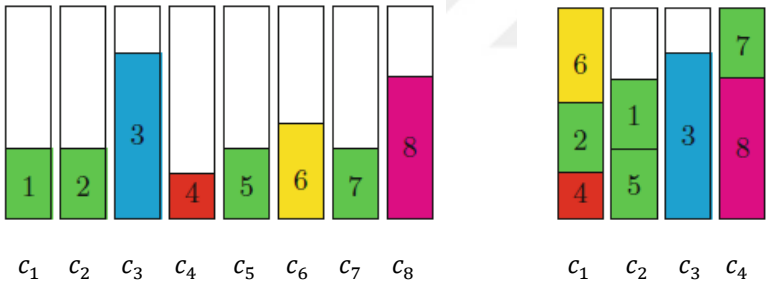
1.2.1. Bir Boyutlu Kutu Paketleme Problemleri

BPP'ın en sade hali bir boyutlu kutu paketleme problemidir (1 Dimensional Bin Packing Problem- 1-D BPP). Bu problemde nesnelerin sadece bir boyutu dikkate alınır. Bu boyut ağırlık veya yükseklik olabilir. Bu problemde önemli olan yerleştirme yapılırken, kapasite sınırının aşılmamasıdır. Kutu yerleştirme söz konusu olduğunda amaç; farklı hacimlerdeki nesneleri, kullanılan kutu sayısını veya kullanılmayan kutu hacmini en aza indirecek şekilde C kapasiteli kutulara yerleştirmektir. Aşağıdaki örnekte; kapasiteleri $C = 6$ olan bir kutu seti ve ağırlıkları g_i olan $i = 1, 2, \dots, 8$ öğeler kullanılmıştır (Hadj Salem ve Kieffer, 2020).

Tablo 1.1 Sekiz Öğeli Bir Paketleme Örneği

i	1	2	3	4	5	6	7	8
g_i	2	2	5	1	2	3	2	4

İlk durumda 8 öğe için 8 kutu kullanılarak uygun bir çözüm, ikinci durumda öğelerin sıralanması ile 4 kutu kullanılarak optimal çözüm elde edilmiştir. Çözüm Şekil 1.2'de verilmiştir.



Şekil 1.2 Bir Boyutlu Kutu Paketleme

Klasik 1-D BPP, tamsayılı doğrusal programlama olarak modellenebilir. İlk olarak, tüm öğeleri paketlemek için gereken minimum kutu sayısında bir üst sınır kabul edilir ve bu kutuların $1, 2, \dots, u$ olarak numaralandırıldığı varsayılır. Paketlenecek ürün sayısı n için, ikili karar değişkenleri ve matematiksel model aşağıdaki gibidir (Munien ve Ezugwu, 2021):

$$y_i = \begin{cases} 1 & konteyner\ i\ kullanıldıysa \\ 0 & değilse \end{cases} \qquad (1 \le i \le u) \qquad (1.1)$$

$$x_{ij} = \begin{cases} 1 & kutu\ j\ konteyner\ i'ye\ paketlenmiş\ ise \\ 0 & değilse \end{cases} \qquad (1 \le i \le u\ ;\ 1 \le j \le n) \qquad (1.2)$$

$$Min = \sum_{i=1}^{u} y_i \tag{1.3}$$

$$\sum_{j=1}^{n} w_i x_{ij} \leq c y_i \qquad (1 \leq \mathrm{i} \leq \mathrm{u}) \tag{1.4}$$

$$\sum_{i=1}^{u} x_{ij} = 1 \qquad (1 \leq \mathrm{j} \leq \mathrm{n}) \tag{1.5}$$

$$y_i \epsilon \{0,1\} \qquad (1 \leq \mathrm{i} \leq \mathrm{u}) \tag{1.6}$$

$$x_i \epsilon \{0,1\} \qquad (1 \leq \mathrm{i} \leq \mathrm{u}\ ;\ 1 \leq \mathrm{j} \leq \mathrm{n}) \tag{1.7}$$

Kısıtlamalar, kutunun maksimum kapasitesinin aşılmamasını ve her öğenin yalnızca bir kutuya paketlenmesini zorunlu kılar. 1D-BPP'nin amacı; tüm n öğeyi paketlemek için kullanılan toplam kutu sayısını *(N)* en aza indirmektir, bu aşağıdaki gibi tahmin edilebilir:

$$N \geq \left[\left(\sum_{i=1}^{n} S_i \right) / C \right] \tag{1.8}$$

Burada S_i, *n* öğelik listedeki her bir öğenin boyutunu veya ağırlığını temsil eder ve *C*, kutuların sabit kapasitesini temsil eder.

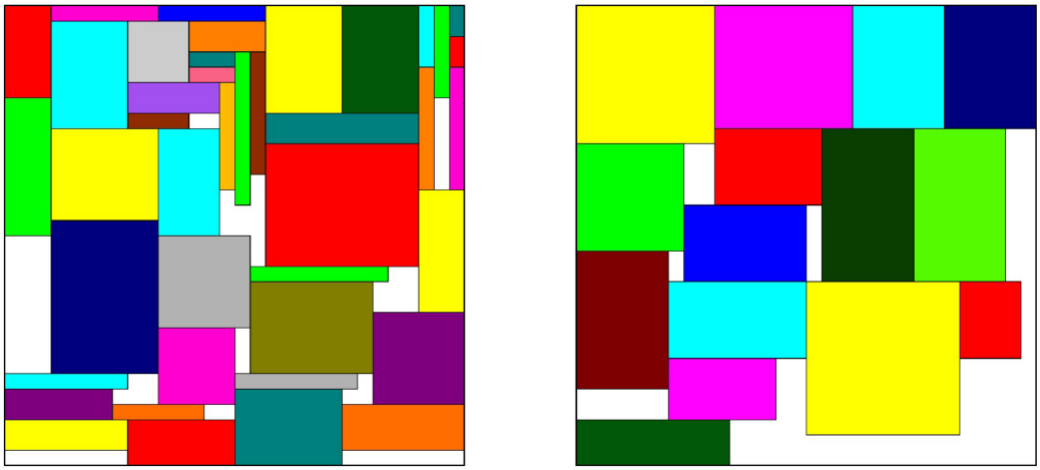
Bir boyutlu kutulama problemlerinin pek çok farklı uygulama alanı bulunmaktadır. Örneğin; her biri farklı zaman gerektiren işlerin planlanması, zaman çizelgeleme olarak ele alınabilir. Ders programına derslerin verimli bir şekilde atanması ve sınırlı çalışma süresinin etkin bir biçimde kullanılması buna örnektir. Bilgisayar bilimlerinde, programların hafızaya yerleştirilmesi için iyi bir bellek yönetiminin yapılması yine 1-D BPP örneğidir.

Problemin karmaşıklığı ve gerçek dünya uygulamalarının çeşitliliği nedeniyle, BPP'yi çözmek için birçok yaklaşım kullanılmıştır. Bu yaklaşımlar, yaklaşık yöntemleri, metasezgiselleri ve hibrit metasezgiselleri içerir. Yakın dönemde kullanılan birçok metasezgisel yaklaşımın, benzerlerine göre çok daha hızlı ve iyi bir performans sergilediği görülmüştür. Ozcan vd. (2016), 1-D BPP için yeni bir gruplama genetik algoritması önermişlerdir. 1238 adet 1-D BPP örneği üzerinde elde ettikleri deneysel sonuçlar, önerdikleri algoritmanın yüksek hesaplama hızı ile arama uzayını benzerlerinden daha etkin bir şekilde keşfetmeyi sağladığını göstermiştir. Abdel-Basset vd. (2018), çalışmalarında geliştirilmiş Lévy tabanlı balina optimizasyon algoritmasını önermektedir. Deneysel sonuçlar, önerilen algoritmanın en uygun çözümü ve yakınsama hızını bulma yeterliliğini doğrulamaktadır. Kucukyilmaz ve Kiziloz (2018), 1-D BPP için işbirlikçi paralel gruplama genetik algoritması

geliştirmiştir. Yaptıkları çalışma sonucunda genetik algoritmalardan daha iyi performans elde ederek, problem örneklerinde önemli iyileştirmeler rapor etmişlerdir. Gherboudj (2019), araştırmasında sürü zekâsına dayanan ve en yeni biyolojik ilhamlı metasezgisel yöntemlerden biri olan afrika mandası optimizasyonunu kullanmıştır. Elde edilen sonuçlar, literatürdeki diğer değerlerle karşılaştırılmış, algoritmanın umut verici çözümlere ulaşma yeteneği olduğu gözlenmiştir. Abdul-Minaam vd. (2020), 1-D BPP'yi çözmek için yakın zamanda optimize edilmiş bir sürü algoritması kullanan uyarlanabilir bir prosedür önermektedir. Uyarlanabilir algoritma, 30 örnek üzerinde test edilmiş ve değerlendirme sonuçları, parçacık sürüsü optimizasyonu algoritması, karga arama algoritması ve jaya algoritması gibi diğer popüler algoritmaların sonuçlarıyla karşılaştırılmıştır. Uyarlanabilir algoritma, uygunluk değerleri ve süreleri konusunda diğerlerine göre üstünlük göstermiştir.

1.2.2. İki Boyutlu Kutu Paketleme Problemleri

İki boyutlu kutu paketleme problemlerinde (2 Dimensional Bin Packing Problem- 2-D BPP), 1-D BPP'den farklı olarak nesnelerin iki boyutuyla ilgilenilir. Bu boyutlar; nesnelerin genişliği (w_i) ile yüksekliği (h_i) veya genişliği (w_i) ile uzunluğu (l_i) veya yüksekliği (h_i) ile uzunluğu (l_i) şeklinde ikililer olarak ele alınabilir. Amaç; nesnelerin üst üste gelmeyeceği biçimde konteynerlere yerleştirilmesi ve tümünün paketlenmesi için gerekli olan konteyner sayısının en az yapılmasıdır. Şekil 1.3'de Martello ve Vigo'nun 1998'de yaptığı bir çalışmada yer alan, paketlenecek 50'den fazla öğeye sahip örnek için en uygun çözüm gösterilmektedir. Beyaz boşluklar artık alanları temsil etmektedir (Cid-Garcia ve Rios-Solis, 2020).



Şekil 1.3 İki Boyutlu Kutu Paketleme

Klasik 2-D BPP aşağıdaki gibi formüle edilebilir. $R = \{1, \dots, n\}$ dikdörtgenler kümesinde yer alan, i nesnesinin genişliği w_i ve yüksekliği h_i ölçüsündedir. Nesneler, W genişliğine ve H yüksekliğine sahip kutulara yerleştirilir. 2-D BPP probleminin matematiksel modelinde; l_{ij}, i nesnesinin j nesnesinin soluna yerleştirilmesi durumunda 1 değerini, aksi halde 0 değerini alan ikili değişkeni; b_{ij}, i nesnesinin j nesnesinin altına yerleştirilmesi durumunda 1 değerini, aksi halde 0 değerini alan ikili değişkeni göstermektedir. Modelde, $\left(x_i, y_j\right)$, i nesnesinin yerleştirildiği m_i kutusunun sol-alt koordinatlarını, v kullanılan kutu sayısını göstermektedir ve p_{ij} ise $m_i < m_j$ durumunda 1 değerini alan ikili değişkendir. Bu modele göre, iki nesnenin üst üste gelmemesi için aşağıdaki eşitsizliklerin sağlanması gerekmektedir (Pisinger ve Sigurd, 2007).

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} + p_{ij} + p_{ji} \geq 1 \quad (1.9)$$

$$l_{ij} = 1 \;\Rightarrow\; x_i + w_i \leq x_j \quad (1.10)$$

$$b_{ij} = 1 \;\Rightarrow\; y_i + h_i \leq y_j \quad (1.11)$$

$$p_{ij} = 1 \;\Rightarrow\; m_i + 1 \leq m_j \quad (1.12)$$

Dikdörtgenlerin hiçbir kısmı kutuyu aşmamalıdır, bu nedenle $0 \leq x_i \leq W - w_i$ ve $0 \leq y_i \leq H - h_i$ kullanılır. v, kullanılan kutu sayısıdır, bu nedenle $1 \leq m_i \leq v$ olarak ifade edilir. 2D-BPP aşağıdaki gibi tamsayılı programlama olarak formüle edilebilir:

$$Min\ v$$

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} + p_{ij} + p_{ji} \geq 1 \qquad i, j \in R,\ i < j \quad (1.13)$$

$$x_i - x_j + W l_{ij} \leq W - w_i \qquad i, j \in R \quad (1.14)$$

$$y_i - y_j + H b_{ij} \leq H - h_i \qquad i, j \in R \quad (1.15)$$

$$m_i - m_j + n p_{ij} \leq n - 1 \qquad i, j \in R \quad (1.16)$$

$$0 \leq x_i \leq W - w_i \qquad i \in R \quad (1.17)$$

$$0 \leq y_i \leq H - h_i \qquad i \in R \quad (1.18)$$

$$1 \leq m_i \leq v \qquad i \in R \quad (1.19)$$

$$l_{ij}, b_{ij}, p_{ij} \in \{0,1\} \qquad i, j \in R \quad (1.20)$$

$$x_i, y_j \in R \qquad i \in R \quad (1.21)$$

$$m_i, v \in N \qquad i \in R \quad (1.22)$$

Bu problem; tekstil, cam, çelik, ahşap veya kâğıt gibi daha büyük dikdörtgen levhalardan küçük dikdörtgen parçaların kesilmesi gereken birçok endüstriyel uygulama için büyük öneme sahiptir. Seri üretim yapılan bu sektörlerde hammadde kullanımında yapılabilecek iyileştirmeler, maliyetlerde önemli azalmalar sağlar. Depolama alanından tasarruf edilmesini amaçlayan raf yerleştirme işlemi de problemin başka bir örneğidir. Benzer şekilde; dergi ve gazetelerde yer alan sütunların haber, makale vb. için uygun bir şekilde bölümlenmesi ve yayının sayfa sınırı aşılmadan yerleştirme yapılması da iki boyutlu kutulamaya birer örnektir (Loh, 2006).

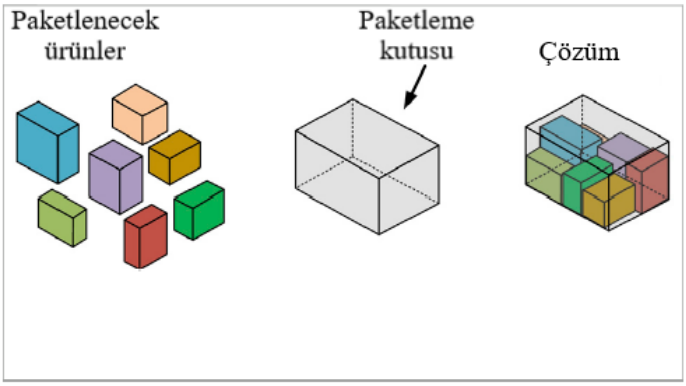
2-D BPP'nin çözümünde metasezgisel algoritmalar sıklıkla kullanılmıştır. Nitekim bu algoritmalar, çok sayıda örneğin bulunduğu durumlarda kabul edilebilir çözümlere makul sürelerde ulaşmaktadır. Virk ve Singh (2018), makalelerinde 2-D BPP'yi çözmek için guguk kuşu arama ve yarasa metasezgisel algoritmalarını uygulamıştır. Çalışmalarının amacı, bu yeni metasezgisel yöntemlerin potansiyelini keşfetmek ve bu problemin performansını artırmaya katkıda bulunup bulunamayacaklarını kontrol etmektir. Sorunu çözmek için standart kıyaslama testi verileri kullanılmıştır. Performanslar literatürde yaygın olarak kullanıldığı görülen genetik algoritma ve tabu arama teknikleri ile karşılaştırılmıştır. Yeni metasezgisel algoritmaların, genetik algoritma ve tabu aramaya göre daha iyi performans gösterdiği görülmüştür. Laabadi vd. (2020), eserinde 2-D BPP'yi çözmek için sürekli optimizasyon problemlerine başarıyla uygulanan karga arama algoritmasını kullanmıştır. Elde edilen sonuçlar, çözüm kalitesi ve hesaplama süresi açısından algoritmanın etkinliğini kanıtlamaktadır. Rakotonirainy ve Vuuren (2020), iki geliştirilmiş şerit paketleme metasezgiseli önermiştir. İlk algoritma, benzetilmiş tavlama yönteminin, aşamalı bir sezgisel algoritma ile birleştirildiği hibrit yaklaşımdır. İkinci algoritma ise, benzetilmiş tavlama yönteminin, tanımlanmış paketleme düzenleri alanında doğrudan uygulanmasını içerir. Bu iki algoritma, çözüm kalitesi açısından literatürdeki temsili bir metasezgisel örnekle karşılaştırılır. Yeni algoritmaların bazı durumlarda literatürdeki mevcut şerit paketleme metasezgisel yöntemlerinden daha iyi performans gösterdiği bulunmuştur. Grandcolas ve Pain-Barre (2022), ilerleme ve doğrulama stratejisi adı verilen hibrit metasezgisel bir yaklaşım sunmuştur. Yaklaşım iki prosedüre dayanır: öğelerin yatay eksende tatmin edici yerleşimlerini sağlayan bir yerel arama algoritması ve dikey eksende konumlarını arayan kesin bir prosedür. Yöntem, kanıtlanabilir şekilde optimal olana veya belirli bir zaman sınırına ulaşılana kadar mevcut çözümü sürekli olarak iyileştiren bir strateji izler. Deneysel sonuçlar, yöntemin en iyi bilinen yaklaşımlara kıyasla orta ölçekli örneklerde rekabetçi olduğunu göstermektedir. Gezici ve Livatyalı (2022)'nın, çalışmalarında yeni bir melez

metasezgisel algoritma önerilmektedir. Önerilen algoritma çiçek tozlaşma ve genetik algoritmanın hibritlenmesiyle oluşturulmuştur. Çiçek tozlaşma algoritmasının global arama kabiliyetini geliştirmek için yerel arama operatöründe değişiklik yapılmış ve önerilen algoritma, son yıllarda yayınlanan altı farklı metasezgisel algoritma ile karşılaştırılmıştır. Veri setinin 10 sınıfının 6'sında ve 50 alt grubunun 33'ünde en başarılı sonuçlar elde edilmiştir.

2-D BPP için geliştirilmiş ve en iyiye yaklaşık çözümler veren algoritmaların bazıları, üç boyutlu problemler için de kullanılabilmektedir.

1.2.3. Üç Boyutlu Kutu Paketleme Problemleri

Üç boyutlu kutu paketleme problemlerinde (3 Dimensional Bin Packing Problem- 3-D BPP) nesnelerin genişlik, uzunluk ve yükseklik olmak üzere üç boyutu ile de ilgilenilmektedir. Yerleştirilen paketler homojen ve dış bükeydir. Guo vd. (2022), üç boyutlu problemleri aşağıdaki şekilde ifade etmiştir:



Şekil 1.4 Üç Boyutlu Kutu Paketleme

3-D BPP'de, boyutları bilinen n adet kutu (w_i, l_i, h_i), boyutları bilinen konteynerlerin içerisine yerleştirilir (W, L, H). Yerleştirme işlemi yapılırken kutuların üst üste gelmemesi ve konteynerlerin boyutlarının aşılmaması gerekmektedir. İlaveten kutuların ölçüleri, konteynerlerin ölçülerini geçmemelidir ($w_i \leq W$, $l_i \leq L$, $h_i \leq H$). Buradaki amaç, tüm kutuları konteynerlere yerleştirdikten sonra kullanılan konteynerlerin sayısını ve boşa harcanan alanını en aza indirmektir. Problemin modeli aşağıdaki gibi formüle edilmiştir (Li ve Zhang, 2015):

İkili değişkenler:

$$p_{ij}, \forall\, i \in \{1, \dots, m\}, \forall\, j \in \{1, \dots, N\},\ p_{ij} = \begin{cases} 1 & i\ kutusu, j\ konteynerına\ paketlenmiş\ ise \\ 0 & değilse \end{cases} \quad (1.23)$$

$$u_j, \forall\, j \in \{1, \dots, N\}, \qquad u_j = \begin{cases} 1 & j\ konteyneri\ kullanıldı\ ise \\ 0 & değilse \end{cases} \quad (1.24)$$

$x_{ik}^{+}, \forall\, i\,,\ k \in \{1, \dots, m\},\ i < k,\ x_{ik}^{+} = 1 \begin{cases} Eğer\ i\ ve\ k\ kutuları\ aynı\ konteynerde\ paketlenir \\ ve\ i\ kutusu\ k\ kutusunun\ soluna\ yerleştirilirse. \end{cases}$

(1.25)

$x_{ik}^{-}, \forall\, i\,,\ k \in \{1, \dots, m\},\ i < k,\ x_{ik}^{-} = 1 \begin{cases} Eğer\ i\ ve\ k\ kutuları\ aynı\ konteynerde\ paketlenir \\ ve\ i\ kutusu\ k\ kutusunun\ sağına\ yerleştirilirse. \end{cases}$

(1.26)

$y_{ik}^{+}, \forall\, i\,,\ k \in \{1, \dots, m\},\ i < k,\ y_{ik}^{+} = 1 \begin{cases} Eğer\ i\ ve\ k\ kutuları\ aynı\ konteynerde\ paketlenir \\ ve\ i\ kutusu\ k\ kutusunun\ arkasına\ yerleştirilirse. \end{cases}$

(1.27)

$y_{ik}^{-}, \forall\, i\,,\ k \in \{1, \dots, m\},\ i < k,\ y_{ik}^{-} = 1 \begin{cases} Eğer\ i\ ve\ k\ kutuları\ aynı\ konteynerde\ paketlenir \\ ve\ i\ kutusu\ k\ kutusunun\ önüne\ yerleştirilirse. \end{cases}$

(1.28)

$z_{ik}^{+}, \forall\, i\,,\ k \in \{1, \dots, m\},\ i < k,\ z_{ik}^{+} = 1 \begin{cases} Eğer\ i\ ve\ k\ kutuları\ aynı\ konteynerde\ paketlenir \\ ve\ i\ kutusu\ k\ kutusunun\ altına\ yerleştirilirse. \end{cases}$

(1.29)

$z_{ik}^{-}, \forall\, i\,,\ k \in \{1, \dots, m\},\ i < k,\ z_{ik}^{-} = 1 \begin{cases} Eğer\ i\ ve\ k\ kutuları\ aynı\ konteynerde\ paketlenir \\ ve\ i\ kutusu\ k\ kutusunun\ üstüne\ yerleştirilirse. \end{cases}$

(1.30)

$l_i^x,\ l_i^y,\ l_i^z,\quad l_i^x = 1,\quad l_i^y = 1,\quad l_i^z = 1,\quad i$ kutusunun uzunluk kenarının sırasıyla X eksenine, Y eksenine veya Z eksenine paralel yerleştirildiğini gösterir. (1.31)

$w_i^x, w_i^y, w_i^z,\ w_i^x = 1,\ w_i^y = 1,\ w_i^z = 1,\ i$ kutusunun genişlik kenarının sırasıyla X eksenine, Y eksenine veya Z eksenine paralel yerleştirildiğini gösterir. (1.32)

$h_i^x, h_i^y, h_i^z,\quad h_i^x = 1,\quad h_i^y = 1,\ h_i^z = 1,\quad i$ kutusunun yükseklik kenarının sırasıyla X eksenine, Y eksenine veya Z eksenine paralel yerleştirildiğini gösterir. (1.33)

Sürekli değişkenler:

$(x_i, y_i, z_i),\qquad \forall\, i \in \{1, \dots, m\}$, üç değişken birlikte bir kaptaki i kutusunun sol alt-arka koordinatını belirtir.

Amaç fonksiyonu:

$$Min \sum_{j=1}^{N} u_j\,.\,(L_j\,.\,W_j\,.\,H_j) - \sum_{i=1}^{m} l_i\,.\,w_i\,.\,h_i \qquad (1.34)$$

Kısıtlar;

$$p_{ij} \leq u_j \qquad \forall\, i \in \{1,\dots,m\}, \qquad \forall\, j \in \{1,\dots,N\} \tag{1.35}$$

$$\sum_{j}^{N} p_{ij} = 1, \qquad \forall\, i \in \{1,\dots,m\} \tag{1.36}$$

- $$x_i + l_i \,.\, l_i^x + w_i \,.\, w_i^x + h_i \,.\, h_i^x \leq L_j + \left(1 - p_{ij}\right).M \tag{1.37}$$
$$\forall\, i \in \{1,\dots,m\}, \qquad \forall\, j \in \{1,\dots,N\}$$

- $$y_i + l_i \,.\, l_i^y + w_i \,.\, w_i^y + h_i \,.\, h_i^y \leq W_j + \left(1 - p_{ij}\right).M \tag{1.38}$$
$$\forall\, i \in \{1,\dots,m\}, \qquad \forall\, j \in \{1,\dots,N\}$$

- $$z_i + l_i \,.\, l_i^z + w_i \,.\, w_i^z + h_i \,.\, h_i^z \leq H_j + \left(1 - p_{ij}\right).M \tag{1.39}$$
$$\forall\, i \in \{1,\dots,m\}, \qquad \forall\, j \in \{1,\dots,N\}$$

- $$x_i + l_i \,.\, l_i^x + w_i \,.\, w_i^x + h_i \,.\, h_i^x \leq x_k + (1 - x_{ik}^{+}).M \tag{1.40}$$
$$\forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$

- $$x_k + l_k \,.\, l_k^x + w_k \,.\, w_k^x + h_k \,.\, h_k^x \leq x_i + (1 - x_{ik}^{-}).M \tag{1.41}$$
$$\forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$

- $$y_i + l_i \,.\, l_i^y + w_i \,.\, w_i^y + h_i \,.\, h_i^y \leq y_k + (1 - y_{ik}^{+}).M \tag{1.42}$$
$$\forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$

- $$y_k + l_k \,.\, l_k^y + w_k \,.\, w_k^y + h_k \,.\, h_k^y \leq y_i + (1 - y_{ik}^{-}).M \tag{1.43}$$
$$\forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$

- $$z_i + l_i \,.\, l_i^z + w_i \,.\, w_i^z + h_i \,.\, h_i^z \leq z_k + (1 - z_{ik}^{+}).M \tag{1.44}$$
$$\forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$

- $$z_k + l_k \,.\, l_k^z + w_k \,.\, w_k^z + h_k \,.\, h_k^z \leq z_i + (1 - z_{ik}^{-}).M \tag{1.45}$$
$$\forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$

- $$x_{ik}^{+} + x_{ik}^{-} + y_{ik}^{+} + y_{ik}^{-} + z_{ik}^{+} + z_{ik}^{-} \geq p_{ij} + p_{kj} - 1 \qquad \forall\, i\,,\ k \in \{1,\dots,m\}, \qquad i < k$$
$$i < k,\ \forall j \in \{1,\dots,N\} \tag{1.46}$$

$$l_i^x + l_i^y + l_i^z = 1 \qquad \forall\, i \in \{1, \dots, m\} \tag{1.47}$$

$$w_i^x + w_i^y + w_i^z = 1 \qquad \forall\, i \in \{1, \dots, m\} \tag{1.48}$$

$$h_i^x + h_i^y + h_i^z = 1 \qquad \forall\, i \in \{1, \dots, m\} \tag{1.49}$$

$$l_i^x + w_i^x + h_i^x = 1 \qquad \forall\, i \in \{1, \dots, m\} \tag{1.50}$$

$$l_i^y + w_i^y + h_i^y = 1 \qquad \forall\, i \in \{1, \dots, m\} \tag{1.51}$$

$$p_{ij}, u_j, x_{ik}^+, x_{ik}^-, y_{ik}^+, y_{ik}^-, z_{ik}^+, z_{ik}^-, l_i^x, l_i^y, l_i^z, w_i^x, w_i^y, w_i^z, h_i^x, h_i^y, h_i^z \in \{0,1\} \;\; \forall\, i, k \in \{1, \dots, m\}, \;\; \forall j \in \{1, \dots, N\} \tag{1.52}$$

$$x_i, y_i, z_i \geq 0 \qquad \forall\, i \in \{1, \dots, m\} \tag{1.53}$$

Kısıt (1.35) ve (1.36), tüm kutuların paketlenmesini ve bir kutunun bir konteynere ancak bu konteyner kullanılıyorsa atanabilmesini sağlar. Kısıt (1.37)–(1.39), tüm kutuların konteynerin boyutuna yerleştirilmesini sağlar. Kısıt (1.40)–(1.45), herhangi iki kutu i ve k'nin birbiriyle üst üste gelmediğini belirtir. Kısıt (1.46), iki kutu aynı konteynerde paketlenirse, aşağıdaki göreli konumlardan en az birinin (sol, üst, arka) doğru olacağını belirtir. Bu, üst üste binmemelerini garanti eder. Kısıt (1.47)–(1.51) birlikte bir kutunun uygun yönünü tanımlar (Li ve Zhang, 2015:37). Benzer bir modelin daha ayrıntılı açıklaması Chen vd. (1995)'nin çalışmasında bulunabilir.

Üç boyutlu paketleme problemlerine çözümü için önerilen kısıtlardan bazıları aşağıdaki gibi sıralanmıştır:

- Kutuların paketlenmesi, paketlemenin yapılacağı büyük kabın (kutu, konteyner, koli, palet vb.) sınırlarını aşmayacak şekilde yapılmalıdır.
- Paketlemenin yapılacağı kap ve paketlenecek cisimler (kutular) aynı boyutta veya farklı boyutlarda olabilirler.
- Paketleme esnasında nesneler döndürülerek veya döndürülmeden paketlenebilirler.
- Paketlenecek nesneler farklı ağırlıklarda olabilirler.
- Bir kaba bir sipariş veya birden fazla sipariş yüklenebilir.

3-D BPP; kutulama problemlerinin, teoride ve pratikte en çok uygulaması bulunan çeşididir. Taşımacılıktaki artan maliyetler ve ihtiyaçlar nedeniyle, kapasitelerin etkin planlanması gereken durumlarda fazlasıyla önem kazanmaktadır. Kamyonların, yük trenlerinin, gemi ve uçak kargolarının verimli şekilde yerleştirilmesi ve depo yönetimi gibi çeşitli uygulamalar üç boyutlu paketleme problemleri kapsamında incelenmektedir. 3-D BPP, BPP'nin en zor türüdür. Bu nedenle problemin çözümü için yeni yöntemler geliştirilmeye devam edilmektedir.

Lodi vd. (2002) çalışmalarında, 3-D BPP için tabu arama yöntemini kullanarak yeni bir çözüm önermektedir. Yeni bir yapılandırıcı sezgisel kullanarak, problem için bir komşuluk değerlendirmesi yapmışlardır. Yapılan hesaplamalı sonuçlar, literatürdeki diğer kesin ve sezgisel algoritmalara kıyasla önerilen yaklaşımın etkili olduğunu göstermektedir. Faroe vd. (2003), 3-D BPP için yönlendirilmiş yerel arama temelli bir sezgisel algoritma sunmaktadır. Algoritma, açgözlü sezgisel ile elde edilen bir üst sınıra dayanarak başlar ve ardışık olarak kutuların uygun bir şekilde paketlenmesini araştırarak kutu sayısını azaltır. Süre sınırına ulaşıldığında veya üst sınırdan önceden hesaplanmış bir alt sınıra ulaşıldığında işlem sona erer. 200 kutuya kadar olan iki ve üç boyutlu örnekler için yapılan deneyler, algoritmanın ortalama olarak literatürdeki sezgisellere göre daha iyi çözümler bulduğunu göstermektedir. Boschetti (2004), probleme yönelik yeni alt sınırlar önermektedir ve daha sonra bu alt sınırları, her bir nesne için 90 derece dönüşlere izin verilen alt kümelerin belirlendiği daha genel bir problem için genişletmektedir. Önerilen alt sınırlar, literatürden elde edilen farklı test problemleri üzerinde değerlendirilmiştir. Sonuçlar, yeni alt sınırların etkinliğini göstermektedir. Crainic vd. (2009), üç boyutlu dikdörtgen kutuların minimum sayıda üç boyutlu kutuya ortogonal olarak paketlenmesi problemi üzerinde çalışmışlardır. Makalelerinde, bu problem için iki seviyeli bir tabu arama algoritması sunulmaktadır. İlk seviye, kutu sayısını azaltmayı hedeflemektedir. İkinci seviye ise kutuların paketlenmesini optimize etmektedir. Yapılan kapsamlı hesaplama sonuçları, önerilen yaklaşımın mevcut yöntemlere kıyasla daha iyi sonuçlar elde ettiğini göstermektedir. Fueller vd. (2010), üç boyutlu yükleme ve araç rotalama kombinasyonu olan üç boyutlu yükleme kapasiteli araç rotalama problemini ele almaktadır. Çalışmalarında, yükün araçlara yüklenmesi ve araçların yol ağı üzerinde rotalanması için birleşik bir optimizasyon hedeflenir. Makalede, paketleme sezgiselleri kullanarak karınca kolonisi optimizasyonu algoritmasıyla problem çözülmüştür. Algoritma, rota belirleme ve paketleme için iki farklı sezgisel bilgiyi birleştirir. Sayısal testlerde, mevcut tüm test örnekleri çözülmüş ve çoğu örnekte yeni en iyi çözümler bulunmuştur. Allen vd (2011), üç boyutlu şerit paketleme problemi için hibrit bir yerleştirme stratejisi sunmaktadır. Bu problem, stok kesme (ahşap, polistiren vb. - israfı en aza indirgeme), kargo yükleme ve çok boyutlu kaynak planlaması gibi alanlarda potansiyel endüstri uygulamalarına sahiptir. Algoritmanın deneysel test sonuçları, hız ve çözüm kalitesi açısından sürekli olarak literatürdeki diğer yöntemleri geride bıraktığını göstermektedir. Jiang ve Cao (2012), 3-D BPP'yi etkin bir şekilde çözmek için yeni bir hibrit benzetilmiş tavlama algoritması sunmaktadır. Sonuçlar, yeni algoritmanın oldukça verimli olduğunu ve çok kısa süreler içinde neredeyse optimal çözümler elde edebildiğini göstermektedir. Silveira vd.

(2013), giyotin kesme kısıtlamasına sahip 3-D BPP için yeni bir karınca kolonisi optimizasyonu algoritması önermektedir. Bu problem, değişken boyutlara sahip bir dizi kutuyu bir dizi kaba paketlemeyi içermektedir. Algoritma, çelik endüstrisindeki gerçek bir problem üzerine uygulanmaktadır. Amaç, hurda metal miktarını en aza indirgemek ve dolayısıyla çelik blok stoğunu azaltmaktır. Önerilen algoritma ile hurda metal miktarı %90 oranında ve hammadde kullanımı %25 oranında azaltılmıştır. Viegas vd. (2014) çalışmalarında, bir perakende çelik distribütörünün gerçek dünya çelik kesme problemini ele almak için sezgisel ve metasezgisel yaklaşımlar sunmaktadır. Bu problem, müşteriler tarafından sipariş edilen daha küçük parçaları elde etmek için büyük çelik blokların kesilmesini içermektedir. Tabu arama ve en uygun sıralama yaklaşımlarının performansları, sezgisel ve karınca kolonisi optimizasyonu algoritmalarına göre karşılaştırılmıştır. Farklı yaklaşımların sonuçlarının analizi, problemin hurda minimizasyonunda önemli faktörlere ışık tuttuğunu göstermektedir. Chandu (2014), farklı kaplarla 3-D BPP için paralel bir genetik algoritma sunmaktadır. Hesaplama açısından, kutuların farklı boyutlara sahip olduğu ve döndürülmesine izin verildiği 3-D BPP, genel 3-D BPP'den daha zordur. Önerilen uygulama, farklı makinelerde 3-D BPP için genetik algoritmayı paralel olarak çalıştırmaya ve çözümü nispeten kısa sürede hesaplamaya yardımcı olmuştur. Wu vd. (2017), üç boyutlu paketleme problemine ayrıştırılabilen yeni bir pratik problemi incelemektedir. Bunlar; değişken boyutlu kartonlarla üç boyutlu düzensiz paketleme problemi, üç boyutlu değişken boyutlu BPP ve tek konteyner yükleme problemidir. Makalede, her alt problemin matematiksel modelleri geliştirilmiş ve bu yeni problemi çözmek için üç aşamalı sezgisel algoritmalar önerilmiştir. Gerçek yaşam durumunda oluşturulan rastgele örneklerle deneyler yapılmıştır. Sonuçlar, önerilen algoritmanın etkili olduğunu ve tatmin edici sonuçlar verebileceğini göstermektedir. Zudio vd. (2018), 3-D BPP için eğilimli rasgele anahtarlı genetik algoritmayı geliştiren değişken bir komşu inişten ilham alan bir algoritma sunmuştur. Kutuları paketlemek için yapısal açgözlü sezgisel yöntem, kutuların paketlenme sırasını belirlemek için bir tamsayı dizisini kullanır. Geliştirilen hibrit yöntem, nesiller boyunca önemli ölçüde üstün ortalama uygunluk gösterir, bu nedenle yüksek kaliteli çözümler daha hızlı bulunur. Yeni yaklaşım, standart bir sette 320 örnek üzerinde test edilmiştir. Hesaplamalı deneyler, yeni yöntemin literatürde önerilen diğer en iyi algoritmalara göre eşit veya daha iyi sonuçlar ürettiğini göstermektedir. Leon vd. (2019), hassas ürünlerin konteynerlere paketlenmesi hakkında bir araştırma yapmıştır. Yaptıkları yayında, önceki algoritmalarla karşılaştırıldığında 3-D BPP varyantı için yaklaşık olarak %28 atık azaltma elde edilen bir tabu arama algoritması önerilmektedir. Araştırmada Perulu seramik endüstrisinden gerçek veriler kullanılmıştır. Ali

vd. (2022), çalışmalarında, derinlemesine ve güncel bir literatür taraması yaparak, araştırmacıların çevrimdışı ve çevrimiçi 3-D BPP'lerde kullandığı kısıtlamaları ve çözüm yöntemlerini belirlemiştir. Ayrıca, gelecekte bu araştırma alanını geliştirmeye yardımcı olabilecek ilgili araştırma boşluklarını ve bunları kapatma yollarını ele almıştır. Moura vd. (2023), Portekiz otomotiv endüstrisinde bir şirketin dağıtım sorununu çözmek için bir hibrit yaklaşım sunmaktadır. Yaklaşımda iki matematiksel model, bir matematiksel-sezgisel yaklaşım içinde entegre edilmiştir. Tüm yaklaşımlar, şirket tarafından sağlanan gerçek örnekler kullanılarak test edilmiştir. Guo vd. (2022), kutulama problemlerinin üç boyutunu da ele alarak örnek ve uygulamaları Tablo 1.2'deki gibi derlemişlerdir:

Tablo 1.2 Kutulama Problemlerinin Uygulamaları

Sınıflandırma		Uygulamalar	Örnekler	Kalite değerlendirme indeksi
1-D BPP		Mekanik imalat, İnşaat sektörü vb.	Bar malzemesi, tel stoğu ve profil çelik kesimi	Malzeme kullanımı veya hurda oranı
2-D BPP	Düzenli Paketleme	Kağıt sektörü, Cam sektörü, Matbaa sektörü, Mikroelektronik sektörü, Yapı sektörü vb.	Kağıt, cam ve taş kesimi; Sac kesme, mikroelektronik birim düzenlemesi;	Doluluk oranı veya doldurma yüksekliği
	Düzensiz Paketleme	Mekanik imalat, Gemi inşa sanayi, Havacılık, Otomotiv sanayi, Giyim sanayi, Deri işleme, Mobilya imalatı, İnşaat sanayi vb.	CNC işleme; Levha, ahşap, kayrak, kumaş ve deri kesimi	Doluluk oranı veya paketleme yüksekliği değeri
3-D BPP	Düzenli Paketleme	Taşımacılık sektörü, Lojistik sektörü vb.	Ekspres gönderi; Gemi kabin düzeni	Dolum yoğunluğu
	Düzensiz Paketleme	3D baskı, Havacılık vb.	3D baskı alanı düzeni; Uzay aracı kabin düzeni	Dolum yoğunluğu

Kaynak: Guo vd., 2022

1.3. Üç Boyutlu İlgili Problemler

Genel olarak, 3-D BPP ile ilgili akademik araştırmalar, bir paketleme stratejisi geliştirmeye ve iyileştirmeye odaklanmıştır. 3-D BPP ile yakından ilgili olan başka kombinatoryal optimizasyon problemleri de vardır. Bu kısımda tezde ilgilendiğimiz problemin hangi kapsamda incelendiğini net bir şekilde ifade edebilmek adına bu problemler arasındaki benzerlikler ve farklılıklar vurgulanmaya çalışılacaktır:

❖ Sırt Çantası Problemi (Knapsack Problem - KP):

Belirli bir toplam ağırlığı aşmadan mümkün olan en yüksek toplam değeri elde etmek için tek bir sırt çantasına hangi öğelerin seçilebileceğini araştırır (Raidl ve Kodydek, 1998). Her öğenin ilişkili bir kârı vardır ve amaç, tek bir konteynere paketlenmiş öğelerin toplam kârını maksimize etmektir (Lin ve Yu, 2006). Bu nedenle tüm nesneleri yerleştirmek mümkün olmadığından nesneler arasında seçim yapmak gereklidir.

❖ Kutu Paketleme Problemi(Bin Packing Problem- BPP):

Sınırsız sayıda konteyner ve sınırlı sayıda eşyanın olduğu bir durumda, tüm eşyanın eksiksiz olarak yüklenmesi ve aynı zamanda minimum sayıda konteyner kullanılması ile ilgilenen problem sınıfıdır (He vd., 2009). Bu problemde amaç, n öğeyi paketlemek için gerekli kap sayısını en aza indirmektir. Öğelerin değerleri bir rol oynamaz (Raidl ve Kodydek, 1998).

❖ Konteyner Paketleme Problemi (Container Packing Problem- CPP):

Dikdörtgen bir kap içine, bir dizi öğeyi dikey yönde olarak yerleştirmeyi amaçlar. Böylece kap alanının israfı en aza indirilir veya yerleştirilen öğelerin toplam değeri en yüksek seviyeye çıkarılır (Jin vd., 2004). Sınırlı sayıda konteyner, sınırsız sayıda mal olan bu durumda tüm kaplar kullanılmalı, aynı zamanda konteyner alanından maksimum düzeyde faydalanılmalıdır. Konteyner paketleme problemi, genellikle sıralama ve konumlandırma kuralları ile çözülen bir yerleştirme problemidir (He vd., 2009). Konteyner paketleme problemi Raidl ve Kodydek (1998) tarafından geleneksel sırt çantası probleminin bir uzantısı olarak ortaya atılmıştır. Aralarındaki fark, KP'de yalnızca bir sırt çantası varken, CPP'de birden fazla sırt çantası (konteyner) olmasıdır (Soak ve Lee, 2012). Bu nedenle çoklu sırt çantası problemi olarak da ifade edilir. CPP aynı anda çözülmesi gereken iki güçlü bağımlı parçaya bölünebileceğinden, KP ve BPP'nin karmaşık bir kombinasyonu gibi görülebilir;

(a) Paketlenecek öğelerin seçimi ve (b) seçilen öğelerin mevcut kaplara dağıtımı.

❖ Konteyner Yükleme Problemi (Container Loading Problem- CLP):

BPP üzerindeki ek kısıtlamalar, konteyner yükleme adlı bir problem sınıfına yol açar. CLP'de tüm konteynerlerin boyutları sabittir ve tüm kutular minimum sayıda konteynere paketlenir (Kang vd., 2012). Bu kısıtlar; öğelerin yük taşıma gücü, işlem kısıtlamaları, yük kararlılığı, öğelerin gruplandırılması, belirli öğelerin sevkiyat öncelikleri, yükleme düzenlemesinin karmaşıklığı, konteyner ağırlık limiti, bir kap içindeki ağırlık dağılımı vb. olabilir. Kutuların ağırlığı stabiliteyi ve dengeyi etkiler. Her bir kutunun ne kadar yüke dayanabileceği ve stabiliteyi sağlamak için paketlemenin nasıl yapılacağı konusunda ek pek çok kısıtlama olabilir (Lim vd., 2005). BPP'de bu kısıtlar göz ardı edilir ve yalnızca boyutlar dikkate alınarak en uygun sığdırma gerçekleştirilmeye çalışılır.

Özetlenecek olursa; 3-D BPP iki ana başlık altında incelenebilir: Bunlar KP ve BPP olarak ele alınabilir. KP; nesne sayısının sınırsız, konteyner sayısının sınırlı olduğu durumlardır. BPP ise nesne sayısının sınırlı, konteyner sayısının sınırsız olduğu problem tipidir. CPP, ikisinin birleşimi niteliğindedir ve hem sınırsız nesneler arasından seçim yapmayı hem de bu nesneleri sınırlı sayıda konteynere yerleştirmeyi hedefler. Bu problemi çok sayıda sırt çantası olan durum olarak ifade etmek de mümkündür. CLP ise; yükün döndürülmesi, ağırlığı, sabitlik (stabilite), denge, yükleme önceliği, yük dayanımı, vb ek kısıtlamalar getirilen genişletilmiş paketleme problemleridir.

BPP, kombinatoriyel optimizasyon problemleri arasında yer alır ve NP-zor problemler olarak sınıflandırılır. Bu tür problemlerde en iyi çözümü elde etmek genellikle çok fazla zaman ve kaynak gerektirir. Bu nedenle, BPP'nin çözümünde zamanla farklı çözüm tekniklerine ihtiyaç duyulmuştur. Geleneksel matematiksel programlama teknikleri, kesin çözümler sunabilmesine rağmen, büyük ölçekli problemlerde pratikte zaman ve kaynak açısından etkili olmayabilir. Zira problem, çözülmesi son derece zor ve hesaplama açısından karmaşıklığı yüksek olan yapıdadır. Bu zorluğun üstesinden gelmek için, sezgisel ve metasezgisel yöntemler BPP'de kullanılmaktadır. Sezgisel yöntemler, sorunun özelliklerini ve yapılarını kullanarak yaklaşık çözümler üretmeyi amaçlar. Örneğin, bir kutuyu paketlemek için gelen ilk ürünü seçmek ve uygun bir kutuya yerleştirmek gibi basit bir strateji uygulayabilir. Bu tür sezgisel yaklaşımlar, pratikte hızlı bir şekilde çalışabilir ve kabul edilebilir çözümler sunabilir.

Metasezgisel yöntemler ise daha karmaşık ve optimize edilmiş algoritmalar kullanır. Bunlar, birden çok sezgisel yöntemi birleştirerek veya çözüm alanını genişleterek daha iyi sonuçlar elde etmeyi hedefler.

Kısacası, BPP, zamanla farklı çözüm tekniklerine ihtiyaç duymuştur. Kesin yöntemlerle tam çözümler elde etmek genellikle mümkün değildir ve bu nedenle sezgisel ve metasezgisel yöntemler kullanılarak yaklaşık çözümler bulunur. Bu teknikler, BPP'nin pratikte çözülmesini daha etkili ve verimli hale getirirken, zorluk derecesi yüksek olan bu problemler için daha iyi çözümler elde edilmesini sağlamaktadır.

İKİNCİ BÖLÜM
KUTU PAKETLEME PROBLEMLERİNE ÇÖZÜM YAKLAŞIMLARI

BPP, birçok endüstriyel sektörde, üretim, lojistik, dağıtım, taşımacılık ve benzeri alanlarda karşılaşılan önemli optimizasyon problemlerinden biridir. Bu problemlerin çözümünde çeşitli yaklaşımlar kullanılmaktadır. Bu yaklaşımlar; kesin yöntemler, sezgisel yöntemler ve metasezgisel yöntemler olarak sıralanabilir. Kesin yöntemler, problemin tüm olası çözüm alanını araştırarak en iyi çözümü bulmaya çalışır ve genellikle matematiksel programlama tekniklerini veya ağaç tabanlı arama algoritmalarını içerir. Lineer programlama, tamsayılı programlama ve karışık tamsayılı programlama gibi teknikler bu kategoriye girer. Kesin çözümler elde etmek için kullanılan bir diğer yaklaşım, ağaç tabanlı arama algoritmalarıdır. Bu algoritmalar, arama ağacı yapısını kullanarak olası çözümleri sistemli bir şekilde keşfeder ve en iyi çözümü bulmaya çalışır. Örnek olarak, dal-sınır ve dal-kesim algoritmaları verilebilir.

Ancak, kesin yöntemlerin kullanımı bazı durumlarda pratik olmayabilir, özellikle büyük ve karmaşık problemler için bu durum söz konusudur. Bu yöntemler genellikle küçük ölçekli problemler için uygundur, çünkü büyük ölçekli problemlerde çözüm süresi artabilir. Bu nedenle, sezgisel ve metasezgisel yaklaşımları kullanmak daha yaygın hale gelmiştir. Metasezgisel yöntemler, sezgisel yöntemleri daha da geliştiren ve optimizasyon problemlerini daha etkili bir şekilde çözmek için kullanılan tekniklerdir. Bu yöntemler, çeşitli arama stratejileri ve optimizasyon tekniklerini birleştirerek daha iyi çözümler elde etmeyi hedefler. Genellikle hızlılardır ve büyük ölçekli problemlerde pratik uygulamalarda tercih edilirler.

Bu bölümde; BPP'nin çözümünde kullanılabilecek yöntemlere değinilmekte, bu tez çalışmasında incelenen 3-D BPP'nin çözüm değerlerini elde etmek için kullanılan metasezgisel algoritmaların tanıtılması hedeflenmektedir.

2.1. Çözüm Yöntemleri

BPP'nin çözümünde kesin, sezgisel ve metasezgisel yöntemler kullanılabilir. Küçük ölçekli problemler için kesin yöntemler tercih edilirken, büyük ölçekli problemler için sezgisel ve metasezgisel yöntemler daha uygundur. Bu yöntemlerin kombinasyonu, daha karmaşık problemlerin çözümünde daha iyi performans ve optimal çözümlere daha yakın değerler elde etmek için kullanılabilir.

2.1.1. Kesin Yöntemler

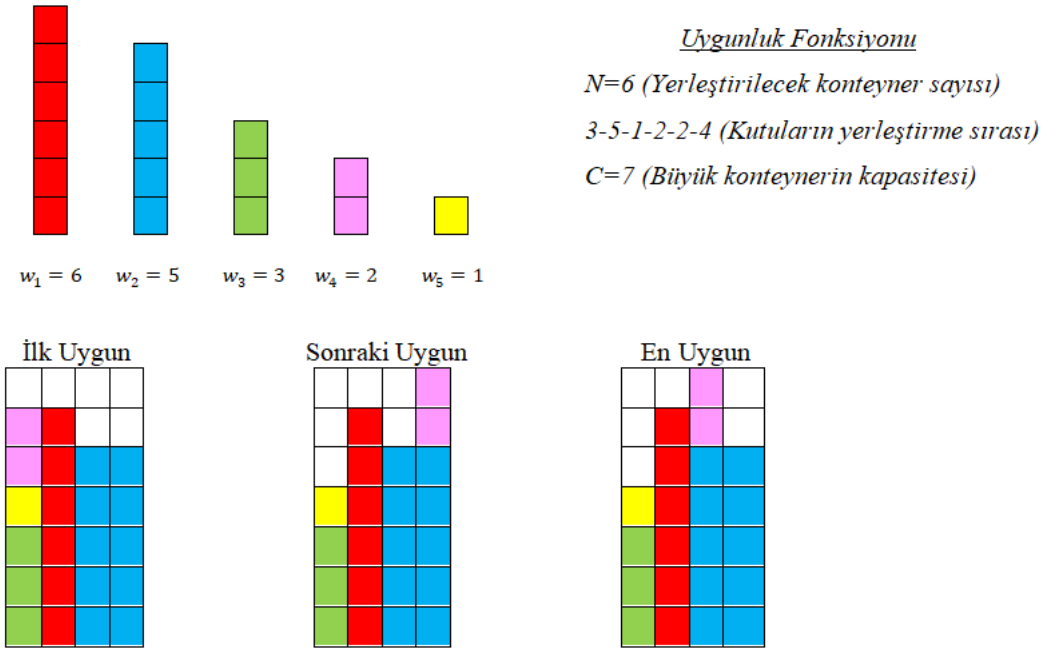
Kesin yöntemler, optimizasyon problemlerinin çözümünde kullanılan çözüm yöntemlerinden birisidir. İlk olarak, matematiksel yöntemler önerilmiştir (Kantorovich, 1960). Daha sonra Gilmore ve Gomory (1961), sütun oluşturma kavramını geliştirmiştir. İlerleyen dönemde, klasik yaklaşımlardan dal-sınır yöntemi ve dinamik programlama kullanılmaya başlanmıştır (Lawler vd., 1966; Bellman, 1966). Bir diğer yöntem Martello ve Toth Prosedürü'dür. Martello ve Vigo (1998), sınırsız sayıda standart stok parçasından kesilecek bir dizi dikdörtgen parça verildiğinde, minimum stok parçası sayısını belirlemeyi hedefleyen, sonlu 2-D BPP çalışmışlardır. İyi bilinen bir alt sınırı analiz edip ve en kötü durum performansını belirlemişlerdir. Problemin kesin çözümü için bir dal ve sınır algoritması içinde kullanılan yeni alt sınırlar önerilmekte, gerçekleştirilen kapsamlı testler, önerilen yaklaşımın etkinliğini göstermektedir. Valério de Carvalho (1999), 1-D BPP problemi için yan kısıtlamalara sahip bir ark akış formülasyonunu araştırmaktadır. Modelde, akış koruma kısıtları ve uygun sayıda öğenin pakete dâhil edilmesini sağlayan kısıtlar bulunmaktadır. Model, ertelemeli değişken üretimi ve dal-sınır tekniklerini birleştiren bir dal ve fiyat prosedürü kullanılarak tam olarak çözülmektedir. Her yinelemede, alt problem, toplu olarak tek bir kutu için arzu edilen ve geçerli bir paketleme düzenlemesini temsil eden bir dizi sütun üretir. OR kütüphanesinden (OR-Library) alınan test veri setleri kullanılmıştır. Bu kütüphanede çeşitli Yöneylem Araştırması (Operations Research- OR) problemleri için test veri setleri koleksiyonu mevcuttur. Elde edilen hesaplama sürelerine göre, bu modelin lineer gevşetmesi, BPP için güçlü bir alt sınıra işaret eder. Haouari ve Serairi (2011), eşit olmayan kutu boyutlarına ve maliyetlere sahip klasik 1-D BPP'nin genelleştirilmesini ele almıştır. Çalışmalarının temel katkısı, sıkı bir ağ akışı temelli alt sınıra, üstünlük kurallarına ve etkili bir sırt çantası tabanlı sezgisel yönteme yer vererek bir dal-ve-sınır algoritmasında çok iyi bir performans elde edilebileceğini göstermektir. Sonuçlar büyük ölçüde rastgele oluşturulan örneklerin makul süreler içinde optimal olarak çözüldüğünü kanıtlamıştır. Festa (2014), zor kombinatoriyel optimizasyon problemlerinin çözümünde kesin, yaklaşık ve sezgisel algoritmalar için kısa bir giriş sunmuştur. Iori vd. (2021), hem öğelerin hem de kutuların dikdörtgen olduğu iki boyutlu ortogonal kesme ve paketleme problemlerinin ana formüllerini ve çözüm yöntemlerini incelemiştir. Literatürdeki dört ana problem: minimum yüksekliğe sahip bir paketleme bulma, öğeleri minimum sayıda kutuya paketleme, maksimum değerde bir paketleme bulma ve uygun bir paketlemenin varlığını belirleme için kesin yöntemlere ve gevşetmelere odaklanmışlardır. Melega vd. (2022), kesme işlemi içerisindeki karmaşık kurulum işlemlerinin modellemesi hakkında çalışmış ve bunun için bir BPP formülasyonu

kullanmıştır. Daha spesifik olarak, kesme işlemi sırasında bıçakların yerleştirilmesi veya çıkarılmasıyla ilgili operasyonu ele almışlardır. Bu işlem, mevcut kesme işlemi ve önceki kesme işlemiyle kesilen öğe sayısına bağlı olduğundan, yerleştirme ve çıkarma sayısı sıra bağımlıdır. Bu tür bir problemle başa çıkmak için iki kompakt formülasyon önerilmektedir. BPP'deki sıralama bağımlı kurulumlar iki farklı şekilde modellenir: literatürdeki bilinen kısıtlamalara dayalı olarak ve mikro dönemlere dayalı olarak. Kurulum kararlarındaki bağımlılık nedeniyle, elde edilen formülasyonlar karışık tamsayılı, doğrusal olmayan matematiksel modellerdir. Sıra bağımlı kesme ve üretim kurulumlarıyla başa çıkmak için, birleşik problemde farklı polinom büyüklüğünde dolaşım kısıtlamaları kümesi kullanılır. Sıra bağımlı kurulumları modellemek için önerilen yaklaşımların ve birleşik kutulama ve parti boyutlandırma problemi çözmek için farklı dolaşım eliminasyon stratejilerinin etkisini analiz etmek amacıyla bir hesaplama çalışması yapılmıştır. Bu analiz, otomatik-Benders ayrışma algoritması aracılığıyla gerçekleştirilmiştir.

2.1.2. Sezgisel Yöntemler

Kesin yöntemlerin sınırlı başarılarından dolayı araştırmacılar sezgisel yöntemler geliştirmeye başlamışlardır. Sezgisel yöntemler en doğru sonucu garanti etmezler. Ancak kabul edilebilir sürelerde çözümler sağlayabilirler. Aşağıda bazı sezgisel yöntemler açıklanmıştır (Sensarma ve Sarma, 2014):

- İlk uygun (First Fit): Bu algoritma, kutuların geldiği sıraya göre yerleştirildiği bir yöntemdir. Mevcut konteynerler sıra ile taranır ve kutu sığdığı ilk konteynere yerleştirilir. Eğer mevcut konteynerde yer yoksa yeni bir konteyner açılır ve kutunun yerleştirilmesi sağlanır.
- Sonraki Uygun (Next Fit): Yerleştirilecek kutunun mevcut konteynere sığıp sığmadığı kontrol edilir. Eğer yer varsa mevcut konteyner içine yerleştirilir. Yer yoksa mevcut konteyner kapatılır ve yeni bir konteyner açılarak kutu yerleştirilir. Ancak kapatılan konteynerler yeni nesne geldiğinde tekrar açılmazlar.
- En uygun (Best fit): Bu algoritmada da yerleştirilecek kutu geldiğinde, mevcut konteynerler sıra ile taranır. Konteynerler taranırken, hangi konteynerde daha az artık alan kalıyorsa kutu ona yerleştirilir. Eğer mevcut konteynerlerden hiçbiri o nesneye uygun değilse yeni bir konteynere geçilir ve nesne bu konteynere atanır. Algoritmanın yapısı ilk uygun yer algoritması ile benzerdir ancak daha yavaştır, çünkü her seferinde tüm listeyi tarar ve en küçük boşluğu bulmaya çalışır. Şekil 2.1'de verilen değerlerdeki kutuların konteynerlere yerleştirilmesi gösterilmiştir.



Şekil 2.1 Sezgisel Yöntemlere Göre Kutu Peketleme Örneği

- En Kötü Uygun (Worst Fit): Bu algoritma, en uygun yer algoritmasının tam tersi prensiple çalışır. Yerleşecek kutu geldiğinde, konteynerler sıra ile taranırken hangi konteynerde daha fazla artık alan kalıyorsa ona yerleştirilir. Eğer mevcut konteynerlerde yer yoksa yeni bir konteyner açılır ve kutu yerleştirilir.
- İlk uygun azalan (First Fit Decreasing): Bu algoritma ilk uygun algoritmasının çevrimdışı halidir. Kutular yerleştirilmeye başlanmadan önce, boyutları azalan sırasına göre sıralanırlar ve ardından ilk uygun algoritması uygulanır.
- En uygun azalan (Best Fit Decreasing): Bu algoritma en uygun yer algoritmasının çevrimdışı halidir. Kutular boyutlarına göre azalan sıra ile sıralanır ve ardından en uygun algoritması uygulanır (Baldi vd., 2012).

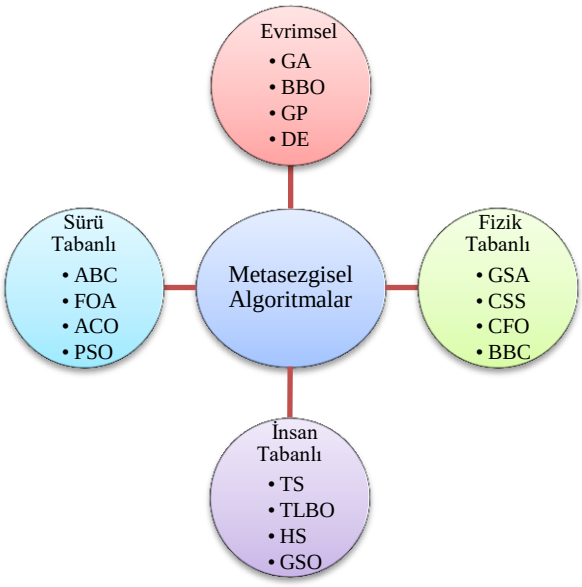
2.1.3. Metasezgisel Yöntemler

Bu yöntemler, basitlikleri ve kolay uygulama süreçleri nedeniyle rekabetçi alternatif çözücüler olarak birçok problemle başa çıkmak için tasarlanmış ve kullanılmıştır. Metasezgiseller ilham kaynağına göre dört ana grupta incelenebilirler: evrimsel algoritmalar, fizik temelli, insan temelli ve sürü zekâsı temelli algoritmalar (Heidari vd., 2019).

En popüler evrimsel algoritma, evrim teorisini taklit eden genetik algoritmadır (Genetic Algortihm- GA). Diğer popüler evrimsel algortimalara; diferansiyel evrim (Differential Evolution- DE), genetik programlama (Genetic Programming- GP) ve biyocoğrafya tabanlı optimizasyon (Biogeography Based Optimizer- BBO) örnek

gösterilebilir (Koza, 1994; Storn ve Price, 1997; Simon, 2008). Fizik temelli algoritmalar, fiziksel yasalardan esinlenir. Bu algoritmaların bazı örnekleri; büyük patlama - büyük çöküş (Big Bang Big Crunch- BBBC), merkezi kuvvet optimizasyonu (Central Force Optimization- CFO), yüklü sistem arama algoritması (Charged System Search- CSS), yerçekimsel arama algoritması (Gravitational Search Algorithm- GSA) ve benzetilmiş tavlamadır (Erol ve Eksin, 2006; Formato, 2007; Kaveh ve Talatahari, 2010; Rashedi vd., 2009).

Metasezgisellerin üçüncü kategorisi, bazı insan davranışlarını taklit eden algoritma setini içerir. İnsan temelli algoritmaların bazı örnekleri; tabu arama (Tabu Search- TS), öğretme ve öğrenme temelli optimizasyon (Teaching Learning Based Optimization- TLBO), grup arama optimize edici (Group Search Optimizer- GSO) ve harmoni aramadır (Harmony Search- HS). Bu algoritmalar, insanların deneyimlerinden, toplumsal etkileşimlerden ve öğrenme mekanizmalarından ilham alarak, etkili ve esnek bir şekilde optimize edilmesi gereken problemleri çözmek için kullanılır (Glover, 1998; Rao vd, 2012; Geem vd, 2001; He vd., 2009). Metasezgisellerin son sınıfı olan sürü zekâsı algoritmaları; sürü, sürüngen veya sürülerde yaşayan organizmaların sosyal davranışlarını taklit eder. Örneğin, kuşların sürü oluşturma davranışı, Eberhart ve Kennedy (1995) tarafından önerilen, parçacık sürü optimizasyonunun (Particle Swarm Optimization- PSO) ana ilham kaynağıdır. Karınca kolonisi optimizasyonu (Ant Colony Optimization- ACO), meyve sineği optimizasyon algoritması (Fruit fly Optimization Algorithm- FOA) ve yapay arı kolonisi (Artificial Bee Colony- ABC) diğer örnekleri arasındadır (Dorigo ve Stutzle, 2006; Gao ve Huang, 2012; Pan, 2012). Metasezgisel algoritmaların örnekleri Şekil 2.2'de özetlenmiştir.



Şekil 2.2 Metasezgisel Algoritmaların Sınıflandırılması

Kaynak: Mirjalili ve Lewis, 2016

Bu algoritmaların çeşitliliğinin yanı sıra, ortak bir özellik vardır. Arama adımları iki aşamaya ayrılır: keşif ve sömürü. İyi tasarlanmış bir algoritmanın keşif davranışları, arama sürecinin ilk aşamalarında zengin bir rastgele doğaya sahip olmalıdır. Sömürü aşaması ise genellikle keşif aşamasından sonra gerçekleştirilir ve arama sürecini tüm kapsamlı bölgeleri yerine yerel bir bölgeye yoğunlaşılır. İyi organize edilmiş bir algoritma, keşif ve sömürü aşamaları arasında denge kurabilme yetisine sahip olmalıdır. Aksi halde, yerel optimumlara yakalanma olasılığı artar (Mirjalili ve Lewis, 2016).

BPP, karmaşık optimizasyon problemleri olarak kabul edilir ve bu alanda farklı çözüm yöntemleri kullanılmaktadır. Teorik olarak bir algoritmanın evrensel açıdan en iyi optimize edici olduğu kabul edilemez. Her bir algoritma, farklı bir yaklaşım ve optimizasyon stratejisi benimsemektedir.

Bu konudaki esin noktası: ücretsiz öğle yemeği yoktur (NFL: No Free Lunch) teoremi, olmuştur. Bu teorem, farklı optimizasyon problemlerinin farklı yaklaşımlar gerektirdiğini ve herhangi bir algoritmanın tüm problemlere en iyi çözümü sunamayacağını ifade etmektedir. Başka bir deyişle, bir algoritma bazı problemlerde başarılı olsa bile, diğer problemlerde aynı performansı gösteremeyebilir.

Bu nedenle, optimizasyon algoritmalarının genel bir çözüm sağlamaktan ziyade, belirli bir problem için optimize edilmiş çözümler sunma amacıyla tasarlanması gerektiği vurgulanır. Her bir problem, farklı kısıtlar, hedefler ve özellikler içerir ve bu nedenle farklı optimizasyon stratejileri gerektirebilir.

NFL teoremi daha verimli optimize edicilerin geliştirilmesi için arayışta bulunmaya teşvik etmektedir. Bunun sonucunda, geleneksel algoritmaların etkinlik, performans yönleri ve sonuçları üzerine yaygın çalışmaların yanı sıra, son yıllarda özellikle global ve lokal arama stratejilerine sahip yeni optimize ediciler farklı alanlardaki araştırmacılar ve uzmanlar için daha fazla seçenek çeşitliliği sunmaktadır.

Bu tez çalışmasında, BPP'yi çözmek için Ateş Böceği Algoritması (Firefly Algorithm- FA), Guguk Kuşu Arama Algoritması (Cuckoo Search Algorithm- CS), Tuna Balık Sürüsü Optimizasyonu (Tuna Swarm Optimization- TSO) ve Bal Porsuğu Algoritması (Honey Badger Algorithm- HBA) olmak üzere dört farklı algoritma kullanılmıştır. Bu algoritmalar, doğadan ilham alarak geliştirilmiş sürü zekâsı temelli metasezgisel algoritmalardır.

Farklı yöntemler kullanma ihtiyacı, BPP'nin karmaşıklığından kaynaklanmaktadır. Bu adım, problemin farklı boyutlarına ve gereksinimlerine uyum sağlayarak daha iyi sonuçlar elde etme potansiyeli sunar. Aynı zamanda, bu algoritmaların farklı avantajlara ve sınırlamalara sahip olması, çeşitli durumlarda çözüm esnekliği sağlar.

Sonuç olarak, BPP'de farklı algoritmaların kullanımı optimale daha yakın çözümler elde etme potansiyeli nedeniyle büyük önem taşımaktadır. TSO gibi algoritmalar, kutuların yerleştirilmesinde balık sürüsünün davranışından ilham alırken, HBA depolama alanını etkin bir şekilde kullanmayı hedefler. FA, kutu yerleştirme işlemine ışık kaynaklarıyla benzetme yaparak yaklaşırken, CS doğal bir seçilim sürecini taklit eder. Bu farklı algoritmaların kullanımıyla, BPP'de daha iyi çözümler elde edilmesi hedeflenmektedir.

2.2. Ateş Böceği Algoritması

2007'de Cambridge Üniversitesi'nde Xin-She Yang tarafından geliştirilen FA, doğal ateş böceklerinin davranışından esinlenen metasezgisel bir algoritmadır (Yang, 2010:81). Bu algoritma, ışık yayma, çekme prensiplerine dayanır ve yüksek kaliteli çözümleri aramak için kullanılır. Ateş böcekleri, belirli bir ışık yoğunluğuna sahip olurlar ve bu ışığı yayarak diğer böcekleri çekerler. Her bir ateş böceği, çözümün bir temsilidir. Işık yoğunluğu, böceğin cazibesini ve çözümün kalitesini temsil eder. Böcekler, mevcut çözümleri gözlemleyerek daha iyi çözümleri ararlar ve ışık yoğunluklarına göre hareket ederler. Yaklaşık iki bin ateş böceği türü vardır ve çoğu ateş böceği kısa, ritmik ışıklar üretir. Bu tür parlamaların iki temel işlevi; eşleri ve potansiyel avları çekmektir. Ek olarak, yanıp sönme, potansiyel yırtıcılara ateşböceklerinin acı tadını hatırlatmak için koruyucu bir uyarı mekanizması görevi görmektedir. Ritmik ışıldama, her iki cinsiyeti bir araya getiren sinyal sisteminin bir parçasını oluşturur.

Işık kaynağından belirli bir r mesafesindeki ışık yoğunluğunun, ters kare yasasına uyduğu bilinmektedir. Yani, $I \propto 1/r^2$ cinsinden, r mesafesi arttıkça ışık yoğunluğu I azalır. Ayrıca mesafe arttıkça; hava, daha da zayıflayan ışığı emer. Bu iki birleşik faktör, ateşböceklerinin çoğunun, sınırlı bir mesafeyi görmesini sebep olur. Yanıp sönen ışık, optimize edilecek amaç fonksiyonu ile ilişkilendirilecek şekilde formüle edilebilir (Yang, 2010):

Amaç fonksiyonu $f(x), \quad x = (x_1, \dots, x_d)^T$
Ateşböceklerinin ilk popülasyonu, $x_i \ (i = 1, 2, \dots, n)$
x_i 'deki ışık yoğunluğu I_i , $f(x_i)$ ile belirlenir. (2.1)

FA'da aşağıdaki gibi idealleştirilmiş üç kural kullanılmaktadır (Fister vd., 2014):

1. Tüm ateşböcekleri cinsiyetsizdir, bu nedenle cinsiyete bakılmaksızın bir ateşböceği diğer ateşböceğini çekebilir.

2. Çekicilikleri, parlaklıkları ile doğru orantılıdır, bu nedenle, yanıp sönen herhangi iki ateşböceği için, daha az parlak olan daha parlak olana doğru hareket edecektir. Mesafe arttıkça çekicilik de parlaklık da azalır. Belirli bir ateş böceğinden daha parlağı yoksa yani parlaklık seviyeleri eşitse rastgele hareket gerçekleşir.
3. Bir ateşböceğinin parlaklığı, amaç fonksiyonundan etkilenerek belirlenir. Bir maksimizasyon problemi için, parlaklık amaç fonksiyonunun değeri ile orantılı olabilir.

- Işık Yoğunluğu Ve Çekicilik

FA'da iki önemli konu vardır: ışık yoğunluğunun değişimi ve çekiciliğin formülasyonu. Maksimum optimizasyon problemleri için en basit durumda, bir ateşböceğinin belirli bir x konumundaki parlaklığı $I, I(x) \propto f(x)$ olarak seçilebilir. Bununla birlikte, çekicilik β görecelidir, bakanın gözünde görülmeli veya diğer ateşböcekleri tarafından değerlendirilmelidir. Böylece, ateş böceği i ve ateş böceği j arasındaki r_{ij} mesafesi ile değişecektir. Ek olarak, ışık yoğunluğu kaynağından uzaklaştıkça azalır ve ışık da ortamda emilir, bu nedenle çekiciliğin soğurulma derecesine göre değişmesi gözlenmektedir.
En basit haliyle ışık yoğunluğu $I(r)$ ters kare yasasına göre değişir (Yang, 2010):

$$I(r) = \frac{I_s}{r^2} \tag{2.2}$$

I_s , kaynaktaki yoğunluktur. Sabit bir ışık soğurma katsayısı γ olan belirli bir ortam için, ışık yoğunluğu I, mesafe r ile değişir.

$$I = I_0\, e^{-\gamma r} \tag{2.3}$$

Yani burada I_0, orijinal ışık yoğunluğudur. I_s/r_s ifadesinde $r = 0$' daki tekillikten kaçınmak (hesaplama zorluklarına neden olan durumları engellemek) için, hem ters kare yasasının hem de ışığın emilmesinin birleşik etkisi aşağıdaki Gauss formu olarak tahmin edilebilir:

$$I(r) = I_0\, e^{-\gamma r^2} \tag{2.4}$$

Bir ateşböceğinin çekiciliği, komşu ateşböceklerinin gördüğü ışık yoğunluğuyla orantılı olduğundan, artık bir ateşböceğinin çekiciliği β şu şekilde tanımlanabilir:

$$\beta = \beta_0 \, e^{-\gamma r^2} \tag{2.5}$$

burada β_0, $r = 0$' daki çekiciliktir. $1/(1 + r^2)$'yi hesaplamak genellikle üstel bir fonksiyondan daha hızlı olduğundan, yukarıdaki fonksiyon şu şekilde tahmin edilebilir:

$$\beta = \frac{\beta_0}{1 + \gamma r^2} \tag{2.6}$$

Hem (2.5) hem de (2.6), çekiciliğin denklem (2.5) için β_0'dan $\beta_0 \, e^{-1}$ 'e veya denklem (2.6) için $\beta_0/2$ ' ye önemli ölçüde değiştiği karakteristik bir mesafe $\Gamma = 1/\sqrt{\gamma}$ tanımlar.

Gerçek uygulamada, çekicilik fonksiyonu $\beta(r)$, aşağıdaki genelleştirilmiş fonksiyonlar gibi uniform azalan fonksiyonlar şeklinde olabilir (Yang, 2010):

$$\beta(r) = \beta_0 \, e^{-\gamma r^m}, \qquad (m \geq 1) \tag{2.7}$$

Sabit bir γ için, karakteristik uzunluk şu hale gelir:

$$\Gamma = \gamma^{-1/m} \to 1, \qquad m \to \infty \tag{2.8}$$

Tersine, bir optimizasyon probleminde belirli bir uzunluk ölçeği Γ için, γ parametresi tipik bir başlangıç değeri olarak kullanılabilir:

$$\gamma = \frac{1}{\Gamma^m} \tag{2.9}$$

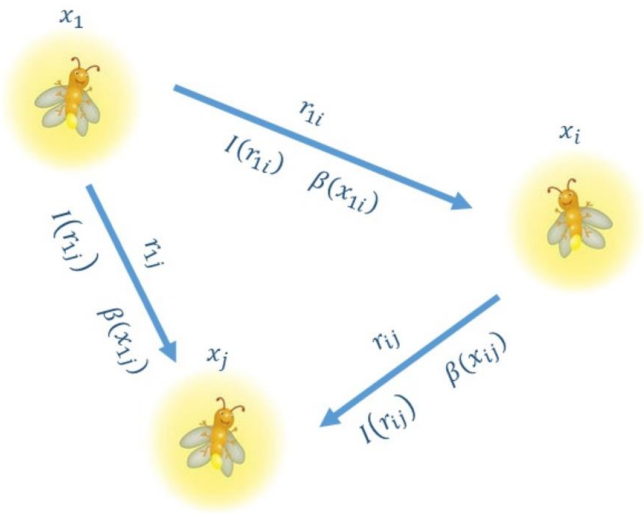
Sırasıyla x_i ve x_j'deki herhangi iki i ve j arasındaki mesafe, Kartezyen mesafedir (Yang, 2010):

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}, \tag{2.10}$$

burada $x_{i,k}$, i. ateşböceğinin x_i uzamsal koordinatının k. bileşenidir. 2 boyutlu durumda,

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \qquad \text{olur.} \tag{2.11}$$

Şekil 2.3'de bir ateşböceğinin komşu ateşböceği arasındaki mesafe gösterilmiştir.



Şekil 2.3 Ateş Böceklerinin Çekim Mesafesi

Kaynak: Louzazni vd., 2018

Ateş böceği *i*'nin hareketi, daha çekici (daha parlak) başka bir ateş böceği *j* tarafından çekilmesi durumunda aşağıdaki gibi belirlenir (Senthilnath vd. 2011):

$$x_i(t+1) = x_i(t) + \beta_0\, e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha\, \epsilon_i \qquad (2.12)$$

burada ikinci terim çekimden kaynaklanmaktadır. Üçüncü terim ve α, rasgeleleştirme parametresidir. ϵ_i, bir Gauss dağılımından veya tek biçimli dağılımdan alınan rasgele sayıların bir vektörüdür. Örneğin, ϵ_i' nin en basit şekli $rand - 1/2$ ile değiştirilebilir, burada $rand$ $[0,1]$' de düzgün bir şekilde dağıtılmış bir rasgele sayı üretecidir. Çoğu uygulama için $\beta_0 = 1$ ve $\alpha \in [0,1]$ alınmaktadır.

(2.12), daha parlak ateşböceklerine doğru eğimli rastgele bir yürüyüş olduğunu belirtmektedir. $\beta_0 = 0$ ise, basit bir rasgele yürüyüş olur.

Böylece γ parametresi çekiciliğin varyasyonunu karakterize eder. Bu değerin, yakınsama hızının ve FA algoritmasının nasıl davrandığının belirlenmesinde oldukça önemli bir etkisi vardır. Teorik olarak $\gamma \in [0,\infty)$, ancak pratikte $\gamma = O(1)$, optimize edilecek sistemin karakteristik uzunluğu Γ tarafından belirlenir. Bu nedenle, çoğu uygulama için tipik olarak 0,1 ila 10 arasında değişir. Bu algoritmada kullanılan parametre değerlerine bölüm 3.2'de yer verilmiştir.

FA birçok uygulamada dikkat çekmiş ve kullanılmıştır. Horng vd. (2012), ateş böceği temelli algoritmanın dijital görüntü sıkıştırmada en az hesaplama zamanını kullandığını göstermiştir. Banati ve Bajaj (2011) ise FA'yı özellik seçimi için kullanmış ve FA'nın diğer algoritmalara göre zaman ve optimallik açısından tutarlı ve daha iyi performans sergilediğini

göstermişlerdir. Mühendislik tasarım problemlerinde, Gandomi vd. (2011) ile Azad ve Azad (2011), FA'nın yüksek dereceli, çok modlu tasarım problemlerini verimli bir şekilde çözebileceğini doğrulamışlardır. Sayadi vd. (2010), NP-zorluğundaki zamanlama problemlerini etkin bir şekilde çözebilen bir tür kesikli FA geliştirmiştir, ayrıntılı bir analiz ise FA'nın çok çeşitli test problemlerinde, çok amaçlı yük dağıtım problemleri dâhil olmak üzere etkinliğini göstermiştir. Basu ve Mahanti (2011) ile Chatterjee vd. (2010), FA'yı anten tasarım optimizasyonunda uygulamış ve FA'nın yapay arı kolonisi algoritmasından daha iyi performans sergilediğini göstermişlerdir. FA'nın bir diğer uygulama alanı global optimizasyon problemlerinden biri olan BPP'dir.

Yesodha ve Amudha (2013), BPP'yi çözmek için; FA ile geleneksel algoritmalardan ikisi olan ilk uygun ve en uygun yöntemlerini uygulamıştır. Bu algoritmaların performansı, OR Kütüphanesi'nden alınan üç farklı sınıf referans veri kümesi üzerinde test edilerek değerlendirilmiştir. Sonuçlar, doğadan esinlenen metasezgisellerin çoğu problem durumu için iyi bir performans sergilediğini göstermiştir. Chandra ve Singh (2014), FA'yı, 2-D BPP'yi çözmek için kullanmakta ve ilk uygunla, en uygun algoritmalarıyla karşılaştırmaktadır. Bu algoritmaların performansı, OR Kütüphanesinden alınan üç sınıf kalite testi veri setinde test edilmiştir. Sonuçlar, ateşböceği ve en uygunun çoğu problem durumu için yaklaşık olarak aynı olduğunu ve ilk uyguna göre daha iyi olduğunu göstermektedir. Rajesh Kanna ve Udaiyakumar (2017), çalışmasında standart boyutlu bir konteynere, rastgele boyutlu farklı tipte heterojen kutuların üç boyutlu paketlenmesini çözmek için FA'yı kullanılmaktadır. Araştırmanın temel amacı; küp, dikdörtgen prizma, silindir ve küre şeklindeki dört farklı tip kutuyu farklı boyutlarda bir konteynere optimal bir şekilde yerleştirmektir. Performans, OR Kütüphanesinden alınan üç sınıf veri setinde test edilerek değerlendirilmiştir. Sonuçlar, ateşböceği algoritmasının çoğu problem durumu için karşılaştırmalı olarak daha iyi performans elde ettiğini göstermektedir. Perdana (2017), çalışmasında genetik algoritma ve FA'nın 1-D BPP'nin çözüm sonuçları karşılaştırılmaktadır. Makalede özellikle NP-problem gibi kombinatoryal problemlerin çözümünde sezgisel yöntemlerin kullanımı hakkında bilgi sağlanması amaçlanmaktadır. Zhao vd. (2020), üç boyutlu düzensiz paketleme problemlerini çözmek için yeni bir yöntem geliştirmiştir. İlk olarak, düzensiz nesneleri yaklaşık olarak temsil etmek için üçlü bir ızgara yaklaşım tekniği kullanılmıştır. Daha sonra, her nesneyi yerleştirmek ve sıkıştırmak için karma bir sezgisel yöntem geliştirilmiştir. Paketleme sırasını optimize etmek için geliştirilen bu yöntemde; kaos arama, ateşböceği algoritmasına entegre edilerek algoritmanın çeşitliliği artırmak hedeflenmiştir. Deneyin sonuçları, karma algoritmanın etkinliğini göstermektedir.

2.3. Guguk Kuşu Arama Algoritması

Doğadan ilham alan ve son yıllarda önemli bir popülarite kazanan metasezgisel algoritmalar arasında yer alan CS Algoritması, optimizasyon problemlerinin etkili bir şekilde çözümünde kullanılan yenilikçi bir yaklaşımdır. Yang ve Deb (2009), tarafından geliştirilen algoritma, özellikle sayısal optimizasyon problemlerinde diğer popüler algoritmalara kıyasla üstün performans sergilemektedir. Guguk kuşlarının yuvalama davranışından esinlenen bu algoritma, doğal hayatta gözlemlenen karmaşık üreme stratejilerini taklit ederek optimize edilecek problemlerin çözümünde başarılı sonuçlar elde etmektedir.

Guguk kuşları, sadece güzel sesleriyle değil, saldırgan üreme stratejileriyle de ilgi çekici kuşlardır. Ani ve Guira gugukları gibi bazı türler, yumurtalarını ortak yuvalara bırakırken başkalarının yumurtalarını çıkararak kendi yumurtalarının gelişim olasılığını artırırlar. Birçok tür ise zorunlu kuluçka asalaklığıyla, genellikle başka türlerin yuvalarına yumurtalarını bırakarak başka kuşların yavrularını büyütme stratejisini benimserler (Yang, 2010:105).

Kuluçka asalaklığının üç temel türü vardır: aynı tür içinde kuluçka asalaklığı, işbirlikçi ebeveynlik ve yuva ele geçirme. Bazı ev sahibi kuşlar, izinsiz giren guguk kuşlarıyla doğrudan çatışmaya girebilir. Ev sahibi (konukçu) bir kuş, yumurtaların kendisine ait olmadığını anlarsa, ya bu yabancı yumurtalardan kurtulacak ya da yuvasını terk edip başka bir yerde yeni bir yuva yapacaktır. Yeni Dünya kuluçka asalaklığı yapan Tapera gibi bazı guguk türleri, dişi asalak gugukların seçtikleri birkaç konukçu türün yumurtalarının renk ve desen taklidinde çok uzmanlaşmışlardır. Bu, yumurtalarının terk edilme olasılığını azaltır ve üreme başarısını artırır (Mareli ve Twala, 2018).

Ayrıca, bazı türlerin yumurtlama zamanlaması da şaşırtıcıdır. Asalak guguklar genellikle, konukçu kuşun kendi yumurtalarını yeni bıraktığı bir yuvayı seçerler. Genel olarak, guguk kuşu yumurtaları, konukçu yumurtalarından biraz daha erken açılır. İlk guguk yavrusu yumurtadan çıktıktan sonra, yapacağı ilk içgüdüsel hareket, konukçu yumurtalarını körü körüne yuvadan dışarı atmak olur, bu da guguk yavrusunun konukçu kuş tarafından sağlanan yiyecek payını artırır. Ayrıca çalışmalar bir guguk yavrusunun daha fazla beslenme fırsatı elde etmek için konukçu yavru sesini taklit edebildiğini göstermektedir. Aşağıdaki şekilde diğer yavrulardan farklı olan guguğun konukçu yuvasındaki temsili görülmektedir. Konukçu kuş, guguk yavrusunun varlığını fark etmemiş ve beslemeye devam etmiştir (Yang, 2010:105).

Şekil 2.4 Serçe Yuvasındaki Guguk Kuşu

CS algoritmasını basit bir şekilde açıklamak için aşağıdaki üç idealize edilmiş kural kullanılmaktadır (Mareli ve Twala, 2018):

1. Her guguk kuşu, her defasında bir yumurta bırakır ve yumurtayı rastgele seçilen bir yuvaya bırakır;
2. Yüksek kaliteli yumurtalara sahip en iyi yuvalar, bir sonraki nesle aktarılır;
3. Mevcut konukçu yuva sayısı sabittir ve guguk kuşunun bıraktığı yumurta, ev sahibi kuş tarafından $p_a \in [0, 1]$ olasılıkla keşfedilir. Bu durumda, ev sahibi kuş ya yumurtadan kurtulabilir ya da yuvayı terk edip tamamen yeni bir yuva inşa edebilir.

Daha ileri bir yaklaşım olarak, bu son varsayım n ana konak yuvasının p_a kesrinin yeni yuvalarla (yeni rasgele çözümlerle) değiştirilmesiyle yaklaşık olarak tahmin edilebilir.

Bir maksimizasyon problemi için, bir çözümün kalitesi veya uygunluğu basitçe amaç fonksiyonunun değeri ile orantılı olabilir. Uygulama açısından, bir yuvadaki her yumurtanın bir çözümü temsil ettiği ve her guguk kuşunun yalnızca bir yumurta bırakabildiği (dolayısıyla bir çözümü temsil ettiği) basit temsiller kullanabilir. Amaç; yuvadaki biraz daha zayıf bir çözümü değiştirmek için, yeni ve potansiyel olarak daha iyi çözümleri (gugukları) kullanmaktır. Bu algoritma her yuvada birden fazla çözümü temsil eden yumurtaların olduğu daha karmaşık duruma genişletilebilir. Bu tez çalışmasında, her yuvanın yalnızca tek bir yumurtaya sahip olduğu en basit yaklaşım ile devam edilmektedir. Bu durumda yumurta, yuva veya guguk arasında ayrım yoktur, çünkü her yuva bir yumurtaya karşılık gelir ve aynı zamanda bir guguğu temsil eder (Yang, 2010:106). Bu algoritmaya özgü parametre değerlerine bölüm 3.2'de yer verilmiştir.

Bir guguk kuşu i için $x^{(t+1)}$ yeni çözümler üretirken, bir Lévy uçuşu gerçekleştirilir:

- *Lévy Uçuşları*

Lévy uçuşları, yönleri rastgele olan ve adım uzunlukları Lévy dağılımından türetilen rastgele yürüyüşlerdir. Bu uçuşlar, hayvanlar ve böcekler tarafından gerçekleştirilir ve düz uçuşların ardından ani 90 derecelik dönüşlerle karakterizedir. Normal rastgele yürüyüşlere kıyasla, Lévy uçuşları büyük ölçekli arama alanlarını keşfetmede daha verimlidir. Bu durum, Lévy uçuşlarının varyansının normal rastgele yürüyüşün varyansından çok daha hızlı bir şekilde artması nedeniyle başarılır (Mareli ve Twala, 2018). Sonuç olarak, bu tür davranışlar optimizasyon ve optimal arama alanında uygulanmış ve ön sonuçlar umut verici yeteneklerini göstermektedir (Naik ve Panda, 2016).

$$x_i^{(t+1)} = x_i^{(t)} + \alpha s \otimes \mathrm{H}(p_a - \varepsilon) \otimes (x_j^t - x_k^t) \tag{2.13}$$

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \mathrm{L}(\mathrm{s}, \lambda) \tag{2.14}$$

burada $\alpha > 0$, ilgilenilen problemin ölçekleriyle ilişkili olması gereken adım boyutunu temsil eder. Çoğu durumda, $\alpha = O(L/10)$ kullanabilir, burada L, ilgilenilen problemin karakteristik ölçeğini temsil eder. Yukarıdaki denklem, temelde rastgele bir yürüyüş için stokastik denklemdir. Genel olarak, rastgele yürüyüş, sonraki durumu/konumun yalnızca mevcut konuma (yukarıdaki denklemdeki ilk terim) ve geçiş olasılığına (ikinci terim) bağlı olan bir Markov zinciridir. Bu girdi bazlı çarpım, parçacık sürü optimizasyonunda kullanılanlara benzer, ancak burada Lévy uçuşuyla gerçekleştirilen rastgele yürüyüş, adım uzunluğunun uzun vadede çok daha uzun olması nedeniyle arama uzayını daha verimli bir şekilde keşfeder. Lévy uçuşu, temel olarak, rastgele bir adım uzunluğunun, Lévy dağılımından seçildiği rastgele bir yürüyüş sağlar (Yang, 2010).

$$\text{Lévy} \sim \mathrm{u} = t^{-\lambda}, \qquad (1 < \lambda \leq 3), \tag{2.15}$$

Burada, adımların sonsuz bir varyansa ve sonsuz bir ortalamaya sahip olan bir Lévy dağılımıyla temsil edildiği belirtilmektedir. Esas olarak, ağır kuyruklu bir adım uzunluğu dağılımına sahip rastgele bir yürüyüş süreci oluşturur. Şimdiye kadar elde edilen en iyi çözüm etrafında Lévy uçuşu ile yeni çözümler oluşturulmalıdır, bu yerel aramayı hızlandıracaktır. Ancak, yeni çözümlerin önemli bir kısmı, şimdiki en iyi çözümden yeterince uzak bir yerden rastgeleleştirilmeli ve yerlerinin şu anki en iyi çözümle yeterince uzak olması sağlanmalıdır, böylece sistem yerel bir optimumda sıkışmaz.

Hızlı bir bakışta, CS ve tepe tırmanışı arasında bazı benzerlikler olduğu görülmektedir, ancak önemli farklılıklar bulunmaktadır. İlk olarak, CS, genetik algoritma ve parçacık sürü optimizasyonuna benzer bir şekilde popülasyon tabanlı bir algoritmadır, ancak bir çeşit elitizm veya seçim kullanır. İkinci olarak, CS'deki rasgeleleştirme, adım uzunluğu ağır kuyruklu olduğundan daha verimlidir ve herhangi bir büyük adım mümkündür. Üçüncü olarak, CS'de ayarlanması gereken parametre sayısı genetik algoritma ve parçacık sürü optimizasyonuna göre daha azdır ve bu nedenle daha geniş bir optimizasyon problemleri sınıfına uyarlanabilecek potansiyele sahiptir. Ayrıca, her yuva bir çözüm kümesini temsil edebilir, bu nedenle CS, meta-popülasyon algoritmalarının türüne genişletilebilir.

CS, algoritması, birçok optimizasyon ve yapay zeka alanında umut verici bir etkinlikle uygulanmıştır. Örneğin, mühendislik tasarımı ve problemlerin çözümünde, diğer algoritmalara kıyasla daha iyi performans göstermiştir (Yang ve Deb, 2010). Ayrıca, Walton vd. (2011), tarafından yapılan modifiye edilmiş CS, örneğin ağ oluşturma gibi doğrusal olmayan problemlerin çözümünde oldukça etkili olduğunu göstermiştir. Ayrıca, Kumar ve Chakarverty (2011), güvenilir bir yerleşik sistem için optimal tasarım elde etmek amacıyla CS yöntemini kullanmışlardır. Kaveh ve Bakhshpoori (2011), çelik çerçevelerin başarılı bir şekilde tasarlanması için CS'ye başvurmuşlardır. Yıldız (2013), işleme operasyonunda geliştirilmiş sonuçlarla birlikte optimal makine parametrelerini seçmek için CS'yi kullanmıştır. Zheng ve Zhou (2012) ise Gauss sürecini temel alan bir CS varyasyonunu sunmuşlardır. Algoritma analizi açısından, Civicioglu ve Besdok (2013) tarafından CS'i, parçacık sürü optimizasyonu, diferansiyel evrim, yapay arı kolonisi ile kavramsal olarak karşılaştıran bir çalışma, CS ve diferansiyel evrim algoritmalarının daha sağlam sonuçlar sağladığını öne sürmüştür. Tein ve Ramli (2010), tarafından hemşire çizelgeleme problemlerini çözmek için kesikli bir CS algoritması önerilmiştir.

Layeb ve Boussalia (2012), makalelerinde, 1-D BPP problemi ile ilgili yeni bir yaklaşım sunmaktadır ve bu yaklaşım kuantum esinli CS'ye dayanmaktadır. Elde edilen sonuçlar oldukça umut vericidir ve önerilen yaklaşımın uygulanabilirliğini ve etkinliğini göstermektedir. Zendaoui ve Layeb (2016) çalışmasında, BPP'nin çözümü için CS algoritmasının uyarlanabilir bir versiyonu kullanılmaktadır. Bu algoritma, birçok optimizasyon problemini çözmede etkili olduğunu kanıtlamıştır. Adaptif CS fikri, tam sayı permutasyonlarına, Levy uçuşuna ve kesirli çözümler elde etmek için bir kod çözme mekanizmasına dayanmaktadır.Deneysel sonuçlar, adaptif algoritmanın birçok BPP örneği için bazı metasezgisellere göre üstün olabileceğini göstermektedir. Virk ve Singh (2016), sundukları çalışmada, dikdörtgen şekilli nesnelerin dikdörtgen bir stok levhasına

yerleştirildiği dikdörtgen paketleme problemini optimize etmektedirler. Amaç, dikdörtgen levhanın kullanımını maksimize etmek ve çeşitli parçaları kesmek için gereken kesik sayısını minimize etmektir. Bu kesikli problemi çözmek için ikili bir CS algoritması kullanılmıştır. Hesaplamalı analizden görüldüğü kadarıyla, CS algoritmasının iyi çözümler verme yeteneği vardır. Munien vd.(2020) çalışmasında, 1-D BPP, iki temel sezgisel yöntem olan en uygun ve daha uygun yöntemlerinin yanı sıra dört temsilci güncel metasezgisel algoritma olan FA, genetik algoritma, adaptif CS algoritması ve yapay arı kolonisi algoritması kullanılarak çözülmektedir. İki temel sezgisel yöntem, global metasezgisel algoritmalar tarafından yeniden sıralama sezgiseli olarak kullanılmakta ve iyi bir paketleme düzeni oluşturmak için yerel aramayı iyileştirme görevi üstlenmektedir. Ayrıca, bu iki yerel sezgisel yöntem, global metasezgisel algoritmalarla birleştirildiğinde yerel optimumlardan kaçınma ve sıkışma durumlarından kurtulma yeteneği sağlayan özel özelliklere sahiptir. Böylece algoritmaların iyi kalitede çözümler üretmelerine olanak tanır. Makalenin ana odak noktası, temsilci algoritmalar için sistemli bir performans değerlendirme çalışması sunmaktır. Burada gerçekleştirilen deneyler, toplamda 1.210'den fazla örnek içeren BPP veri seti kullanılarak gerçekleştirilmiştir. Her bir veri seti farklı kapasiteler, öğe sayıları ve öğe ağırlıkları arasında dağılımlara sahiptir. Temsilci algoritmaların sayısal sonuçları, "en uygun" ve "daha uygun" sezgisel yöntemleriyle elde edilen çözümlerle karşılaştırılmıştır. Benzer şekilde, elde edilen başlangıç hesaplama sonuçlarının analizi, bireysel algoritma uygulamalarının üstün performansını ortaya koymuştur. Performans, hem algoritmaların hesaplama süresini hem de çözüm kalitesini dikkate alarak değerlendirilmiştir. "En uygun" sezgisel yönteminin kullanılması, daha küçük kapasiteli kutular gerektiren kolay veri seti örnekleri için optimal çözümler elde etmeyi sağlamıştır. Ancak, durumların karmaşıklığı arttıkça, yüksek kalitede sonuç üretme yeteneği azalmıştır. Öte yandan, daha uygun sezgisel yöntemi kullanmak, kapasite ve öğe sayısından bağımsız olarak neredeyse her zaman optimal çözümler sağlamıştır. Munien ve Ezugwu (2021) eserinde ise, özellikle popülasyon tabanlı metasezgisel algoritmalar üzerine odaklanarak bir boyutlu BPP için elde edilen son gelişmelerin bir derlemesini sunulmaktadır. Bu makalenin, araştırmacılara BPP'nin yeni çıkan varyantlarını çözmek için daha güçlü ve güncel metasezgisel algoritmalar geliştirmelerinde bir referans noktası olarak hizmet edebileceği düşünülmüştür. Yachba vd. (2022) yaptıkları çalışmada, bir deniz limanında konteyner depolama problemine odaklanılmaktadır. Bu problemi çözmek için CS'ye dayalı bir teknik sunulmaktadır. Bu yöntem, her konteyner için depolama alanındaki en uygun konumu belirleyen bir atama sonucunu üretmektedir. Yazarlar, üç kısıt dikkate almaktadır: konteynerin kalkış tarihi, depolama maliyeti ve hareket sayısı.

2.4. Tuna Balık Sürüsü Optimizasyonu

Xie vd. (2021), tarafından geliştirilen TSO, doğadan esinlenen ve sürü davranışını temel alan bir optimizasyon algoritmasıdır. Bu yenilikçi algoritma, tuna balıklarının doğal beslenme ve avlanma stratejilerini taklit ederek karmaşık optimizasyon problemlerini çözmek için tasarlanmıştır.

Bilimsel adı Thunnini olan Tuna balığı, etçil bir deniz balığıdır. Çeşitli orta su ve yüzey balıklarıyla beslenen en iyi deniz yırtıcılarındandır. Tunalar sürekli yüzücülerdir ve uzun, ince kuyrukları hızla sallanırken vücudun sert kaldığı benzersiz ve verimli bir yüzme yöntemine sahiptirler. Ancak Tuna balığı çok hızlı yüzse de çevik küçük balık tepkisi kadar hızlı değildir. Bu nedenle, Tuna balığı avlanmak için "grup seyahati" yöntemini kullanmaktadır. Avlarını bulmak ve saldırmak için çeşitli yiyecek arama stratejileri geliştirmişlerdir.

- İlk strateji spiral yiyecek aramadır. Beslenirken avlarını daha kolay saldırıya uğrayabilecekleri sığ sulara sürmek için spiral bir düzen oluşturarak yüzerler.
- İkinci strateji parabolik yiyecek aramadır. Her Tuna balığı, avını çevrelemek için parabolik bir şekil oluşturarak önceki bireyin ardından yüzer.

Önerilen algoritmanın matematiksel modeli aşağıda ayrıntılı olarak açıklanmaktadır. Bu algoritmanın parametreleri için belirlenen değerlere bölüm 3.2'de yer verilmiştir. TSO, arama alanına eşit şekilde ve rastgele dağıtılan başlangıç popülasyonları üreterek optimizasyon sürecine başlar (Xie vd., 2021).

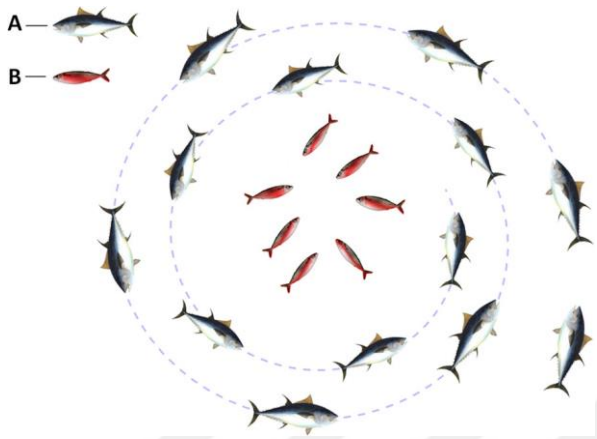
$$X_i^{int} = \text{rand} \cdot (\text{ub-lb}) + \text{lb},\ i = 1,2,\dots,NP, \tag{2.16}$$

burada X_i^{int}, i. ilk bireydir, ub ve lb arama uzayının üst ve alt sınırlarıdır. NP Tuna balığı popülasyonlarının sayısıdır ve rand 0 ile 1 arasında değişen düzgün dağılmış rastgele bir vektördür.

- *Spiral toplayıcılık*

Sardalya, ringa balığı ve diğer küçük sürü halindeki balıklar avcılarla karşılaştığında, sürekli olarak yüzme yönünü değiştirerek avcıların bir hedefe kilitlemesini zorlaştırır. Bu sırada Tuna balık sürüsü sıkı bir sarmal oluşturarak avın peşine düşer. Balıkların çoğunun yön duygusu çok az olmasına rağmen, küçük bir balık grubu belirli bir yöne sıkıca yüzdüğünde, yakındaki diğer balıklar birer birer yönlerini ayarlayarak, büyük bir grup oluşturup aynı amaç için yüzmeye başlarlar. Tuna sürüleri avlarının peşinden gitmenin yanı sıra birbirleriyle bilgi

alışverişinde de bulunurlar. Her Tuna balığı bir önceki balığı takip eder, böylece komşu Tuna balığı arasında bilgi paylaşımı sağlanır. Aşağıdaki şekilde bu avlanma biçimi aktarılmıştır, (A) avcı tuna balığı ve (B) av olan uskumru balığı olarak ifade edilmiştir



Şekil 2.5 Tuna Balıklarının Spiral Şekilde Avlanması
Kaynak: Tuerxun vd., 2022

Bu prensiplere dayanarak, sarmal yiyecek arama stratejisinin matematiksel formülü aşağıdaki gibidir (Xie vd., 2021):

$$X_i^{t+1} = \begin{cases} \alpha_1 . (X_{best}^t + \beta . |X_{best}^t - X_i^t|) + a_2 . X_i^t, & i = 1, \\ \alpha_1 . (X_{best}^t + \beta . |X_{best}^t - X_i^t|) + a_2 . X_{i-1}^t, & i = 2,3, \dots, NP, \end{cases} \quad (2.17)$$

$$\alpha_1 = \alpha + (1 - \alpha) . \frac{t}{t_{max}}, \quad (2.18)$$

$$\alpha_2 = (1 - \alpha) - (1 - \alpha) . \frac{t}{t_{max}}, \quad (2.19)$$

$$\beta = e^{bl} . \cos(2\pi b), \quad (2.20)$$

$$l = e^{3cos\left(((t_{max}+1/t)-1)\pi\right)}, \quad (2.21)$$

burada X_i^{t+1}, $t + 1$ yinelemesinin *i.* bireyidir. X_{best}^t, mevcut optimal bireydir. α_1 ve α_2, bireylerin optimal bireye ve önceki bireye doğru hareket etme eğilimini kontrol eden ağırlık katsayılarıdır. α, Tuna balığının optimal bireyi ne ölçüde takip ettiğini belirlemek için kullanılan bir sabittir. *t* mevcut yineleme sayısı, t_{max} maksimum yinelemeler ve *b*, *0* ile *1* arasında düzgün dağılmış rastgele bir sayıdır. Tüm tuna balıkları yiyeceğin etrafında spiral olarak yemlendiğinde, arama alanı için iyi bir kullanım kabiliyetine sahip olurlar.

Ancak, optimal birey yiyecek bulmakta başarısız olduğunda, en uygun bireyi körü körüne takip etmek grup halinde yiyecek aramak için elverişli değildir. Bu nedenle, sarmal arama için bir referans noktası olarak arama uzayında rastgele bir koordinat oluşturulur. Bu işlem, her bireyin daha geniş bir alanı aramasını kolaylaştırır ve TSO'ya küresel keşif yeteneği kazandırır. Spesifik matematiksel model şu şekilde açıklanmaktadır (Xie vd., 2021):

$$X_i^{t+1} = \begin{cases} \alpha_1 . (X_{rand}^t + \beta . |X_{rand}^t - X_i^t|) + a_2 . X_i^t, & i = 1, \\ \alpha_1 . (X_{rand}^t + \beta . |X_{rand}^t - X_i^t|) + a_2 . X_{i-1}^t, & i = 2,3, \dots, NP, \end{cases} \quad (2.22)$$

burada X_{rand}^t, arama uzayında rastgele oluşturulmuş bir referans noktasıdır. TSO, yineleme arttıkça sarmal yiyecek aramanın referans noktalarını rastgele bireylerden optimal bireylere değiştirir. Özetle, sarmal yiyecek arama stratejisinin nihai matematiksel modeli aşağıdaki gibidir (Xie vd., 2021):

$$X_i^{t+1} = \begin{cases} \alpha_1 . (X_{rand}^t + \beta . |X_{rand}^t - X_i^t|) + a_2 . X_i^t, & i = 1, \\ & , if\ rand < \frac{t}{t_{max}} \\ \alpha_1 . (X_{rand}^t + \beta . |X_{rand}^t - X_i^t|) + a_2 . X_{i-1}^t, & i = 2,3, \dots, NP, \\ \alpha_1 . (X_{best}^t + \beta . |X_{best}^t - X_i^t|) + a_2 . X_i^t, & i = 1, \\ & , if\ rand \geq \frac{t}{t_{max}} \\ \alpha_1 . (X_{best}^t + \beta . |X_{best}^t - X_i^t|) + a_2 . X_{i-1}^t, & i = 2,3, \dots, NP, \end{cases} \quad (2.23)$$

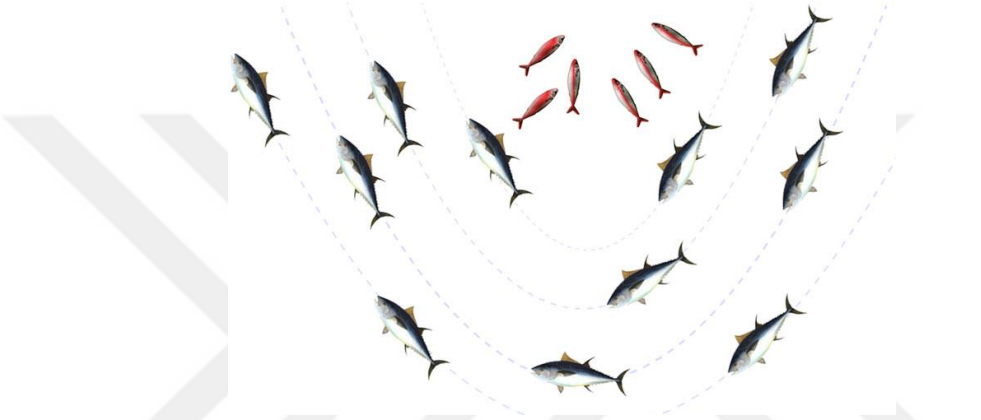
- *Parabolik Toplayıcılık*

Tunalar, sarmal bir oluşum oluşturarak beslenmenin yanı sıra parabolik bir iş birliği ile de beslenirler ve yiyeceği referans noktası şeklinde kullanarak parabolik bir düzen oluştururlar. Ayrıca, etraflarında yiyecek arayarak avlanırlar. Bu iki yaklaşım eşzamanlı olarak gerçekleştirilir ve her ikisi için seçim olasılığının %50 olduğu varsayılır. Belirli matematiksel model aşağıdaki gibi açıklanmaktadır (Xie vd., 2021):

$$X_i^{t+1} = \begin{cases} X_{best}^t + rand . (X_{best}^t - X_i^t) + TF . p^2 . (X_{best}^t - X_i^t), & if\ rand < 0.5, \\ TF . p^2 . X_i^t, & if\ rand \geq 0.5, \end{cases} \quad (2.24)$$

$$p = \left(1 - \frac{t}{t_{max}}\right)^{\left(\frac{t}{t_{max}}\right)}, \quad (2.25)$$

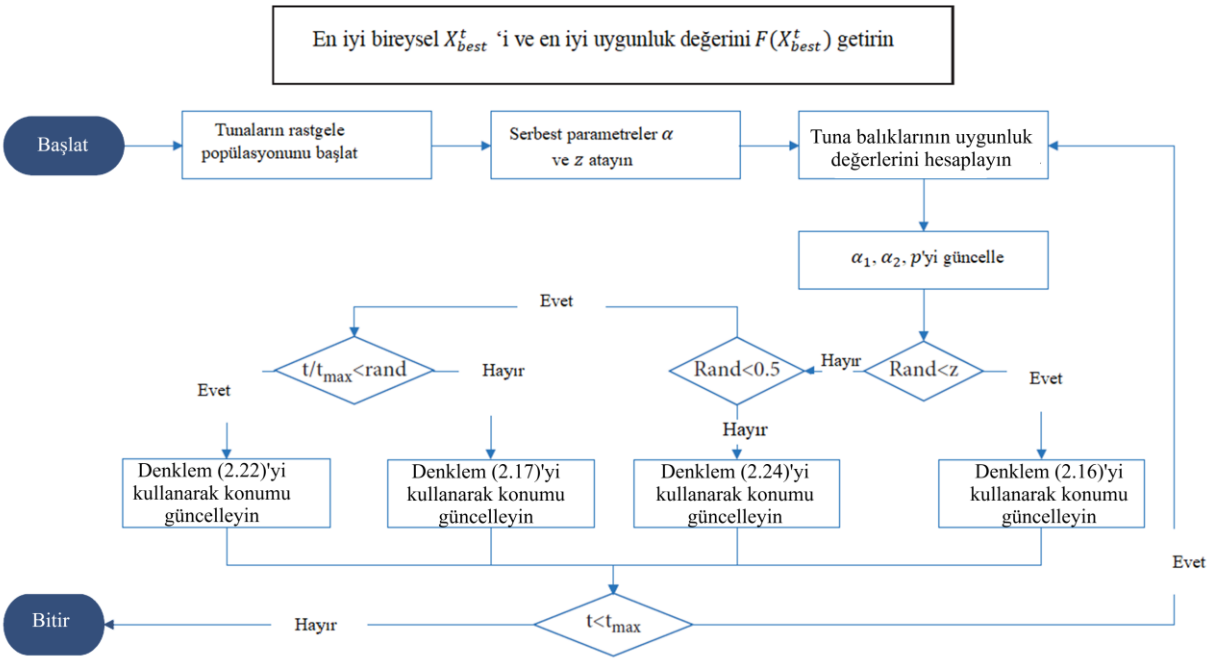
burada *TF*, 1 veya - 1 değerine sahip rastgele bir sayıdır. Tuna balığı, iki yiyecek arama stratejisi aracılığıyla işbirliği içinde avlanır ve ardından avlarını bulur. TSO'nun optimizasyon işlemi için, popülasyon ilk olarak arama uzayında rastgele oluşturulur. Her bir yinelemede, her bir birey, yürütmek için iki yiyecek arama stratejisinden birini rastgele seçer veya *z* olasılığına göre arama uzayındaki konumu yeniden oluşturmayı seçer. Tüm optimizasyon süreci boyunca, TSO'nun tüm bireyleri sürekli olarak güncellenir ve son koşul sağlanana kadar hesaplanır ve ardından optimal birey ve karşılık gelen uygunluk değeri döndürülür. Parabolik Toplayıcılık aşağıdaki şekilde ifade edilmiştir.



Şekil 2.6 Tuna Balıklarının Parabolik Şekilde Avlanması

Kaynak: Tuerxun vd., 2022

TSO'nun ayrıntılı süreci Şekil 2.7'da gösterilmektedir.



Şekil 2.7 TSO'nun Akış Şeması

Kaynak: Xie vd., 2021

TSO'nun sağladığı yenilikçi ve etkili optimizasyon stratejileri, gerçek dünya problemlerinde daha iyi çözümler bulma potansiyelini sunmaktadır. Çeşitli çalışmalarda, algoritmanın farklı problem ve alanlarda kullanımı incelenmiş ve performansını artırmak için farklı yöntemler önerilmiştir. Ashraf vd. (2022), yakıt hücrelerinin verimliliğini artırmak için bulanık yapay sinir ağı denetleyicisinin ve tuna sürü optimizasyon algoritmasının birlikte kullanılmasını araştırmaktadır. Önerilen yöntem yakıt hücrelerinin performansının artırabileceğini ve enerji verimliliğinin iyileştirebileceğini göstermektedir. Wang ve Tian (2022), TSO algoritmasının geliştirilmiş bir versiyonunu sunmaktadır. Bu yaklaşım, TSO'nun performansını iyileştirmek amacıyla kaotik haritaları ve Levy uçuş operatörünü kullanarak daha etkili bir optimizasyon süreci sağlamayı amaçlamaktadır. Sonuçlar, algoritmanın çözüm uzayında daha geniş bir arama yapabildiği ve potansiyel olarak daha iyi çözümler üretebildiğini göstermektedir. Tan vd. (2022), sayısal optimizasyon problemleri için JADE algoritmasının geliştirilmiş bir versiyonunu sunmaktadır. Makale, TSO algoritmasını JADE ile birleştirerek daha etkili bir optimizasyon yaklaşımı önermektedir. Elde edilen sonuçlar, önerilen JADE-TSO yönteminin diğer optimizasyon algoritmalarına kıyasla daha iyi çözümler üretebildiğini ve daha hızlı yakınsama sağladığını göstermektedir. Kumar ve Mary (2022), TSO algoritmasını Newton-Raphson yöntemiyle birleştirerek güneş enerjisi sistemlerinde doğru parametre tahminlerinin elde edilmesini hedeflemektedir. Elde edilen sonuçlar, önerilen kaotik TSO algoritmasının diğer yöntemlere kıyasla daha iyi performans gösterdiğini ve parametrelerin doğru bir şekilde tahmin edildiğini göstermektedir. Wang vd. (2022), TSO algoritmasının geliştirilmiş bir versiyonunu kullanarak ormanlık alanların kütüklerini doğru bir şekilde ayırmayı hedeflemektedir. Yöntem, görüntü segmentasyonunda TSO'nun kullanılmasıyla daha iyi sonuçlar elde edebilmeyi amaçlamaktadır. Tuerxun vd. (2022), çalışmalarında rüzgar türbinlerindeki hataların doğru ve etkili bir şekilde sınıflandırılmasını amaçlamaktadır. Makale, hata sınıflandırma problemini çözmek için modifiye edilmiş TSO algoritmasını optimize etmektedir. Sonuçlar, önerilen yöntemin rüzgâr türbinlerindeki hataların sınıflandırılmasında yüksek doğruluk sağladığını göstermektedir. Yan ve diğerleri (2023), otonom su altı araçları için yol planlaması problemine odaklanan yeni bir öğrenme tabanlı TSO algoritması sunmaktadır. Sonuçlar, bu yeni algoritmanın otonom su altı araçlarının etkili ve güvenli bir şekilde yol planlaması yapabilmesini sağladığını göstermektedir.

Sonuçta, TSO, farklı alanlarda birçok problemin çözümünde kullanılabilecek bir optimizasyon algoritmasıdır. Bu tez çalışması, TSO'nun kutulama problemlerindeki başarısını test etmeyi hedeflemektedir.

2.5. Bal Porsuğu Algoritması

Hashim vd. (2022), tarafından geliştirilen bal porsuğu algoritması, bal porsuğunun akıllı yiyecek arama davranışından esinlenmiştir. Bal porsuğu, korkusuz doğasıyla tanınan Afrika, Güneybatı Asya ve Hint Yarımadası'nın yarı çöllerinde ve yağmur ormanlarında sıklıkla bulunan siyah beyaz kabarık kürklü bir memelidir. Bu korkusuz toplayıcı, tehlikeli yılanlar da dâhil olmak üzere altmış farklı türü avlar. Alet kullanabilen zeki bir hayvandır ve balı sever. Cesur yapısı nedeniyle, kaçamadığı durumlarda kendisinden çok daha büyük yırtıcılara bile saldırmaktan çekinmez. HBA, bal porsuğunun yiyecek arama davranışını taklit eder. Bal porsuğu, besin kaynağını bulmak için ya koku alır ve eşeler ya da rehber kuşu takip eder. Birinci duruma kazma modu, ikinci duruma ise bal modu denir. Avın konumuna yaklaşmak için koku alma yeteneğini kullanır; oraya vardığında kazmak ve avını yakalamak için uygun yeri seçmek üzere avın etrafında hareket eder. İkinci modda, arı kovanını doğrudan bulmak için bal rehberi kuşunun rehberliğini alır. HBA, "kazma aşaması" ve "bal aşaması" olmak üzere iki aşamaya ayrılır. Matematiksel formülasyonu ve önerilen algoritmik adımları aşağıdaki gibi detaylandırılmıştır. Burada, HBA'daki aday çözüm popülasyonu şu şekilde temsil edilir (Hashim vd., 2022):

Aday çözümlerin popülasyonu = $\begin{bmatrix} x_{11}\ x_{12}\ x_{13} \dots\ x_{1D} \\ x_{21}\ x_{22}\ x_{23} \dots x_{2D} \\ \dots\dots\dots\dots \\ x_{n1}\ x_{n2}\ x_{n3} \dots\ x_{nD} \end{bmatrix}$

bal porsuğunun *i.* pozisyonu = $[x_i^1, x_i^2, \dots, x_i^D]$

Adım 1: Başlatma aşaması. Denklem (2.26)'ya göre bal porsuğu sayısı (popülasyon büyüklüğü N) ve ilgili konumları başlatılır.

$x_i = lb_i + r_1 \times (ub_i - lb_i), \quad r_1$ 0 ile 1 arasında rastgele bir sayıdır (2.26)

burada x_i, *N* popülasyonundaki bir aday çözüme atıfta bulunan *i.* bal porsuğu pozisyonu iken, lb_i ve ub_i sırasıyla arama alanının alt ve üst sınırlarıdır.

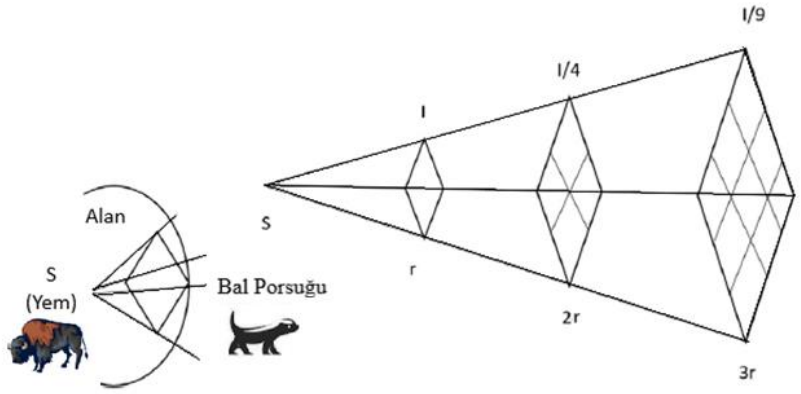
Adım 2: Yoğunluğun tanımlanması (I). Yoğunluk, avın konsantrasyon gücü ve av ile bal porsuğu arasındaki mesafe ile ilgilidir. I_i, avın koku yoğunluğu; koku yüksekse hareket hızlı olur ve tersi Şekil 2.8'de gösterildiği gibi Ters Kare Yasası ile verilir ve Denklem (2.27) ile tanımlanır. Bu algoritmada kullanılan parametre değerleri bölüm 3.2'de ifade edilmiştir.

$I_i = r_2 \times \frac{S}{4\pi d_i^2}$, r_2 *0* ile *1* arasında rastgele bir sayıdır (2.27)

$S = (x_i - x_{i+1})^2$ (2.28)

$d_i = x_{prey} - x_i$ (2.29)

Burada S , kaynak gücü veya konsantrasyon gücüdür. Denklem (2.27)'de d_i, av ile i'inci porsuk arasındaki mesafeyi belirtir.



Şekil 2.8 Ters Kare Kanunu

I koku yoğunluğu, S avın yeri ve r 0 ile 1 arasındaki rastgele sayıdır.

Adım 3: Yoğunluk faktörünün güncellenmesi. Yoğunluk faktörü (α), zamanla değişen rastgeleleştirme miktarını kontrol ederek keşiften (global arama) sömürüye (yerel arama) geçişi düzenler. Zamanla rastgeleleştirmeyi azaltmak için iterasyonlarla azalan bir faktör olan α 'yı güncellemek için (2.28) kullanılır.

$\alpha = C \times exp\left(\frac{-t}{t_{max}}\right), t_{max} =$ maksimum yineleme sayısı (2.30)

burada C sabit ≥ 1'dir (varsayılan = 2).

Adım 4: Yerel optimumdan kaçış. Bu adım ve sonraki iki adım, yerel optimum bölgelerinden kaçmak için kullanılır. Bu bağlamda, önerilen algoritma, ajanların arama alanını yoğun bir şekilde tarayabilmeleri için arama yönünü değiştiren F işaretini kullanır ve böylece yüksek fırsatlar elde edilir.

Adım 5: Aracıların pozisyonlarının güncellenmesi. Daha önce bahsedildiği gibi, HBA konum güncelleme süreci (x_{new}), "kazı aşaması" ve "bal aşaması" olmak üzere iki kısma ayrılır. Aşağıda daha iyi bir açıklama verilmiştir:

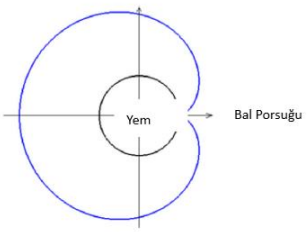
Adım 5-1: Kazma aşaması. Kazma aşamasında, bir bal porsuğu, Şekil 2.9'de gösterildiği gibi Kardioid şekline benzer bir eylem gerçekleştirir. Kardioid hareketi Denklem (2.29) ile simüle edilebilir.

$$x_{new} = x_{prey} + F \times \beta \times I \times x_{prey} + F \times r_3 + \alpha \times d_i \times |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]| \quad (2.31)$$

burada x_{prey}, avın şu ana kadar bulunan en iyi konumudur, başka bir deyişle, küresel en iyi konum. β ≥ 1 (varsayılan = 6), bal porsuğunun yiyecek elde etme yeteneğidir. d_i, av ile *i*'inci bal porsuğu arasındaki mesafeyi ifade eder, Denklem (2.27). r_3, r_4 ve r_5, 0 ile 1 arasında üç farklı rasgele sayıdır. F, arama yönünü değiştiren işaret olarak çalışır, Denklem (2.30) kullanılarak belirlenir:

$$F = \begin{cases} eğer\ r_6 \leq 0.5\ ise\ 1 \\ değil\ ise\ -1 \end{cases} \qquad r_6\ \ 0\ ile\ 1 \text{ arasında rastgele bir sayıdır} \quad (2.32)$$

Kazma aşamasında, bir bal porsuğu, av x_{prey}'in koku yoğunluğu I, porsuk ile av arasındaki mesafe d_i ve zamanla değişen arama etki faktörü α ile güçlü bir bağlantı kurar. Ayrıca, kazma faaliyeti sırasında bir porsuk, daha da iyi bir av yeri bulmasını sağlayan herhangi bir F etkisini alabilir. Mavi çerçeve koku yoğunluğunu, siyah dairesel çizgi avın yerini gösterir.



Şekil 2.9 Kazma Aşaması

Adım 5-2: Bal aşaması. Bir bal porsuğunun kovana ulaşmak için bal rehber kuşunu takip etmesi durumu Denklem (2.31) olarak simüle edilebilir.

$$x_{new} = x_{prey} + F \times r_7 \times \alpha \times d_i, \qquad r_7 \text{ 0 ile 1 arasında rastgele bir sayıdır} \quad (2.33)$$

burada x_{new} bal porsuğunun yeni konumunu ifade ederken, x_{prey} av yeridir, F ve α sırasıyla (2.30) ve (2.28) denklemleri kullanılarak belirlenir. Denklem (2.31)'den, bir bal porsuğunun uzaklık bilgisine (d_i) dayanarak o ana kadar bulunan x_{prey} av konumuna yakın arama yaptığı gözlemlenebilir. Bu aşamada arama, zamana (α) göre değişen arama davranışından etkilenebilir. Ayrıca, bir bal porsuğu F etkisi bulabilir.

HBA, araştırmacılar ve mühendisler için etkileyici bir araç haline gelmiştir. Hem basit hem de karmaşık optimizasyon problemlerini çözerken yüksek performans ve çözüm kalitesi sunar. Bu nedenle, problem çözümünde verimlilik ve etkinlik için önemli bir seçenek haline gelmiştir. Chen vd. (2022), bal porsuğunun davranışlarından esinlenerek elektrikli araçların şarj ihtiyaçlarını, etkili ve düzenli bir şekilde karşılamak için yeni bir optimizasyon yaklaşımı sunmaktadır. Makalede sunulan geliştirilmiş bal porsuğu algoritması, enerji kaynaklarını daha verimli bir şekilde kullanmak için önemli bir araç olarak öne çıkmaktadır. Zhou vd. (2023) dünyanın en sıcak ve en soğuk bölgelerinde atmosfer hava sıcaklığının tahmin edilmesinde HBO kullanımını ele almaktadır. Long vd. (2022) çok amaçlı reaktif güç optimizasyonunda geliştirilmiş HBA uygulanmasını ele almaktadır. Geleneksel yöntemlere alternatif olarak, geliştirilmiş algoritmanın daha iyi performans sağladığı ve birden fazla hedefi dikkate alabilen bir yaklaşım olduğu ortaya koyulmuştur. Fadhil ve Hassan (2023), güvenilirlik optimizasyonunda daha etkili bir yaklaşım sağlamak amacıyla Nelder-Mead yöntemi ile HBA birleştiren bir hibrit algoritma sunmaktadır. Yapılan deneyler, bu hibrit algoritmanın güvenilirlik optimizasyonunda daha iyi performans gösterdiğini ve daha hassas sonuçlar elde ettiğini göstermektedir. Elsayed (2022) çalışması, ekonomik yük dağıtım problemlerinde daha iyi bir performans elde etmek amacıyla Lévy uçuşlarıyla geliştirilmiş HBA sunmaktadır. Yapılan deneyler, bu geliştirilmiş algoritmanın ekonomik yük dağıtımı problemlerini daha etkili bir şekilde çözebildiğini ve daha iyi optimizasyon sonuçları sağladığını göstermektedir. Nikhil vd. (2022) HBA'yı kullanarak yenilenebilir enerji kaynakları ve enerji depolama sistemlerinin frekans kontrolünün daha etkili bir şekilde gerçekleştirilebileceğini göstermektedir.

HBA, 2022 yılında oldukça yaygınlaşsa da henüz BPP'de kullanımına rastlanmamıştır. Bu tez çalışmasında yöntemin etkinliğini test etmek için bir katkı sunulması amaçlanmaktadır.

ÜÇÜNCÜ BÖLÜM
ÜÇ BOYUTLU KUTU PAKETLEME PROBLEMLERİNDE METASEZGİSEL ALGORİTMALARIN KARŞILAŞTIRMALI ANALİZİ

Bu bölümde, 3-D BPP'nin çözümünde metasezgisel yöntemlerin performanslarının araştırılması amaçlanmaktadır. Bu amaç çerçevesinde dört farklı metasezgisel yöntem olan FA, CS, TSO ve HBA ile çözümler ele alınmaktadır.

Bu bölümün ilk kısmında, uygulamada kullanılan veri seti tanıtılmıştır. Çalışmada kullanılan veri seti, farklı boyutlarda ve sayılarda kutuları içermekte ve gerçekçi senaryoları temsil etmektedir. Bu veri seti üzerinde gerçekleştirilen testler, algoritmaların etkinliğini analiz etmek ve karşılaştırmak için kullanılmıştır. Veri setinin tanıtımı yapıldıktan sonra, kullanılan parametre değerlerine odaklanılmıştır.

Çalışmada, metasezgisel algoritmaların performansını değerlendirmek için kullanılan parametre değerleri, bu algoritmaları geliştiren yazarlar tarafından kabul edilen (varsayılan) değerler olarak tercih edilmiştir. Bu değerler, ilgili literatürde yaygın olarak kullanılan ve önerilen parametre değerleridir. Bu sayede, 3-D BPP çözümü için kullanılan algoritmaların test edilmesi ve elde edilen sonuçların hızlı bir şekilde analiz edilmesi için iyi bir referans noktası sağlanmıştır. Parametre değerleri açıklandıktan sonra, nasıl bir yerleştirme stratejisinin izlendiği gözden geçirilmiştir.

Yerleştirme stratejisi, kutuların nasıl yerleştirileceğini ve düzenleneceğini belirleyen kuralları içermektedir. Bu strateji, kutuların optimize edilmiş bir şekilde konteynerlere yerleştirilmesini hedeflemektedir. Yerleştirme stratejisi tanıtıldıktan sonra, kesikli form üzerine yoğunlaşılmıştır.

Kesikli form, BPP'de kullanılan bir optimizasyon yöntemi olarak tercih edilmekte ve algoritmaların daha etkin sonuçlar elde etmesine imkân sağlamaktadır. Ayrıca, optimizasyon problemlerinde kullanılan kesikli formülasyonlar, BPP'yi matematiksel olarak tanımlar ve algoritmaların çözüm sürecini yönlendirir.

Son olarak, elde edilen sonuçlar tablolar aracılığıyla raporlanmış ve yorumlanmıştır. Her bir durumun performansı, kullanılan iterasyon ve popülasyon değerleriyle ilişkilendirilmiş ve sonuçlar üzerindeki etkileri ayrıntılı bir şekilde tartışılmıştır. Ayrıca, tablolar optimizasyon sürecinin etkinliğini kapsamlı bir şekilde analiz etmeyi amaçlayarak verilerin görsel olarak sunulmasını sağlamış ve sonuçların daha anlaşılır bir şekilde değerlendirilmesine katkıda bulunmuştur.

Kullanılan yöntemler arasında istatistiksel olarak anlamlı farklılıkların varlığı, Friedman testi kullanılarak incelenmiştir. Ayrıca, hangi yöntemler arasında bu anlamlı farklılıkların bulunduğu, Wilcoxon testi ile analiz edilmiştir.

Çalışmada tüm testler, 64,0 GB RAM ve Intel Core i9 10850K CPU 3.60GHz işlemciye sahip 64 bit Windows işletim sistemli bir bilgisayarda gerçekleştirilmiş olup algoritmalar Python dilinde kodlanmıştır. İstatistik testleri ise IBM SPSS Statistics 23 programı kullanılarak yapılmıştır.

3.1. Veri Seti

3-D BPP analizlerinde 320 örneğin bulunduğu bir veri seti kullanılmıştır (Martello vd., 2000). Veri seti 8 sınıftan oluşmaktadır ve her bir sınıfın 4 alt grubu vardır. Bu alt gruplar farklı kutu tiplerinin bulunduğu 10 adet örneğe sahiptir. Kutu tipleriyle ilgili bilgiler Tablo 3.1'de ve sınıflarla ilgili bilgiler Tablo 3.2'de verilmiştir. Tablo 3.1 ve Tablo 3.2'de W (konteynerin genişliği), H (konteynerin yüksekliği), D (konteynerin derinliği), konteyner ölçülerini, w (kutunun genişliği), h (kutunun yüksekliği), d (kutunun derinliği), kutu ölçülerini göstermektedir.

Tablo 3.1 Kutu Tipleri

Tip_1	$w_j \in \left[1, \frac{1}{2}W\right]$	$h_j \in \left[\frac{2}{3}H, H\right]$	$d_j \in \left[\frac{2}{3}D, D\right]$
Tip_2	$w_j \in \left[\frac{2}{3}W, W\right]$	$h_j \in \left[1, \frac{1}{2}H\right]$	$d_j \in \left[\frac{2}{3}D, D\right]$
Tip_3	$w_j \in \left[\frac{2}{3}W, W\right]$	$h_j \in \left[\frac{2}{3}H, H\right]$	$d_j \in \left[1, \frac{1}{2}D\right]$
Tip_4	$w_j \in \left[\frac{1}{2}W, W\right]$	$h_j \in \left[\frac{1}{2}H, H\right]$	$d_j \in \left[\frac{1}{2}D, D\right]$
Tip_5	$w_j \in \left[W, \frac{1}{2}W\right]$	$h_j \in \left[1, \frac{1}{2}H\right]$	$d_j \in \left[1, \frac{1}{2}D\right]$

Deneylerde kullanılan veri setinde, 8 sınıf ve her sınıfın 4 alt grubu vardır. Bu alt gruplar kutu sayısına göre belirlenmiştir. Tablo 3.2'de n ile gösterilen sütun her bir alt grubun kaç adet kutu içerdiğinin göstermektedir. Her bir alt grup; 50, 100, 150 ve 200 kutuya sahip 40 adet örnek içermektedir ve veri seti toplamda 320 örnekten oluşmaktadır (Gonçalves ve Resende, 2013).

Tablo 3.2 Kutu Tiplerinin Ağırlıkları

Sınıf	Örnek	n	W,H,D	Tip_1	Tip_2	Tip_3	Tip_4	Tip_5
1	40	50,100,150,200	100×100×100	60%	10%	10%	10%	10%
2	40	50,100,150,200	100×100×100	10%	60%	10%	10%	10%
3	40	50,100,150,200	100×100×100	10%	10%	60%	10%	10%
4	40	50,100,150,200	100×100×100	10%	10%	10%	60%	10%
5	40	50,100,150,200	100×100×100	10%	10%	10%	10%	60%
6	40	50,100,150,200	10×10×10	$w, h, d \in [1,10]$				
7	40	50,100,150,200	40×40×40	$w, h, d \in [1,35]$				
8	40	50,100,150,200	100×100×100	$w, h, d \in [1,100]$				

3.2. Parametre Değerleri

Bu tez çalışmasında kullanılan parametrelerin temelini algoritmaların geliştiricileri tarafından kabul edilen varsayılan değerler oluşturmuştur. FA ve CS için Yang, X.S. (2010), TSO için Xie vd. (2021), HBA için ise Hashim vd. (2022)'nin, çalışmalarından yararlanılmıştır.

Varsayılan bu değerler, ilgili literatürde yaygın olarak kullanılan ve önerilen parametre değerleridir. Bu yaklaşım, çalışmanın geçerliliğini ve karşılaştırılabilirliğini sağlamak için literatürde kabul edilen normlara bağlı kalmayı hedeflemiştir. Kullanılan bu değerler, algoritmaların potansiyelini ortaya çıkarmak ve optimize edilmiş çözümler üretmek için etkili bir başlangıç noktası sağlamış, algoritmaların performansını objektif bir şekilde karşılaştırma ve değerlendirme imkânı sunmuştur.

Çalışmada kullanılan metasezgisel yöntemlerin performansı dokuz farklı durum altında değerlendirilerek çeşitli testler gerçekleştirilmiştir. Her bir durum aşağıda ifade edilen iterasyon ve popülasyon sayılarına bağlı farklı parametre kombinasyonlarına dayanmaktadır. Bu durumlar şu şekildedir:

1. Durum: 100 iterasyon ve n popülasyon
2. Durum: 100 iterasyon ve 2n popülasyon
3. Durum: 100 iterasyon ve 4n popülasyon
4. Durum: 500 iterasyon ve n popülasyon
5. Durum: 500 iterasyon ve 2n popülasyon
6. Durum: 500 iterasyon ve 4n popülasyon
7. Durum: 1000 iterasyon ve n popülasyon
8. Durum: 1000 iterasyon ve 2n popülasyon
9. Durum: 1000 iterasyon ve 4n popülasyon

Her bir durum altında, çözüm sürelerinin minimum ve ortalama değerleri, doluluk oranlarının minimum ve ortalama değerleri, konteyner sayılarının minimum ve ortalama değerleri hesaplanmıştır.

Tablo 3.3 Parametre Değerleri

Algoritma	Parametre	İterasyon	Popülasyoyn
FA	$\beta_0 = 1, \quad \gamma = 1$	100 – 500 – 1000	n – 2n – 4n
CS	$N = 25, \quad p_a = 0.25, \quad \alpha = 1$	100 – 500 – 1000	n – 2n – 4n
TSO	$Z = 0.05, \ \alpha = 0.7$	100 – 500 – 1000	n – 2n – 4n
HBA	$\beta = 6, \quad C = 2$	100 – 500 – 1000	n – 2n – 4n

Çalışmada Tablo 3.3'de ifade edilen parametre ve durumlara bağlı olarak çözüm sürelerinin minimum ve ortalama değerleri, doluluk oranlarının minimum ve ortalama değerleri, konteymer sayılarının minimum ve ortalama değerleri hesaplanarak yöntemler bazında karşılaştırmalar yapılmaktadır.

3.3. Kesikli Form

Metasezgisel algoritmalar, hem sürekli hem de kesikli optimizasyon problemleri için kullanılabilmektedir. Bazı algoritmalar özellikle sürekli optimizasyon problemlerinde daha etkiliyken, bazıları ise kesikli optimizasyon problemleri için daha uygundur. Bu çalışmada kullanılan, FA, CS ve TSO gibi algoritmalar sürekli optimizasyon problemlerinde başarılı bir şekilde kullanılmaktadır ve gerçel sayı çözümleri üzerinde çalışmayı hedeflemektedir. Ancak, BPP kesikli bir yapıya sahiptir. 3-D BPP gibi problemler için sürekli form doğrudan uygulanamamaktadır. Bu nedenle, kesikli formun kullanılması gerekmektedir.

Kesikli form, öğelerin tamsayılı değerlerle temsil edildiği ve öğe kombinasyonlarının optimize edildiği bir yaklaşımı benimsemiştir. Bu form, BPP'deki gibi öğelerin belirli kısıtlar altında konteynerlere atanması gereken durumlar için daha uygun bir çözüm sunmaktadır. Örneğin, bir sürekli çözümde her öğenin bir pozisyon değeri bulunur. Kesikli forma geçerken, bu pozisyon değerleri sıralanarak farklı bir öğe permütasyonu elde edilir.

Bu konuda farklı teknikler önerilmiştir; en büyük sıralı değer yöntemi LRV (Liang vd., 2011), en küçük pozisyon değeri SPV (Fatih vd, 2006), en büyük pozisyon değeri LOV (Qian vd., 2008) ve en küçük sıralı değer yöntemi ROV (Liu., 2008). LRV, bir öğenin değeri ne kadar büyükse, daha yüksek sıralamaya sahip olduğunu varsayar. SPV ise bir öğenin pozisyon değerine odaklanır ve en küçük pozisyona sahip öğeleri öncelikli olarak seçer. LOV, aynı şekilde öğenin pozisyon değerine odaklanır ancak en büyük pozisyona sahip öğeleri öncelikli olarak seçer. ROV ise değeri küçük olan sayıdan büyüğe doğru sıralama yapar.

Tablo 3.4'de, dört kuralın ele alındığı bir örnek gösterilmektedir. ROV yöntemine göre en küçük değer 1,00 olduğu için sıralamaya onunla başlanır ve en büyük değer 5,10 sonuncu yani 6. değer olur. LRV yöntemi ROV yönteminin tam tersidir ve 1.sıraya en büyük değer olan 5,10'u yerleştirerek sıralamaya başlar ve en küçük değer 1,00'in 6. değer olması ile sıralama sonlanır. SPV ve LOV'da ise pozisyon değerlerine göre sırala yapılır. Parantez içinde pozisyon değerleri ifade edilecek olursa 3,29 (1) – 1,00 (2) – 4,59 (3) – 5,00 (4) – 2,00 (5) – 5,10 (6) şeklinde gösterilirler. SPV'de küçükten büyüğe olan sıralama olduğundan, en küçük değer 1,00 sayısının pozisyon değeri 2 ile sıralamaya başlanır. 5,10 sayısının pozisyon değeri 6 ile sıralama sonlanır. LOV da SPV'nin tam tersidir ve büyükten küçüğe sıralama olduğundan 5,10 sayısının pozisyon değeri 6 ile sıralama başlarken 1,00 sayısının pozisyon değeri 2 ile sıralama sonlanır. Bu tez çalışmasında kullanılmış olan, Martello vd., (2000)'un veri seti üzerinde daha önce yapılmış olan testlerde 3-D BPP için ROV yönteminin daha iyi sonuçlar verdiği görülmüştür (Gezici ve Livatyalı, 2023).

Bu çalışmada da ROV yöntemi tercih edilmiştir. Tablo 3.4'de gösterildiği gibi, ROV yöntemi sürekli çözüme en yakın tam sayı çözümünü verir. ROV yöntemi BPP'deki en uygun çözüme daha yakın sonuçlar üretme potansiyeline sahiptir (Zendaoui vd., 2016).

Tablo 3.4 Kesikli Form

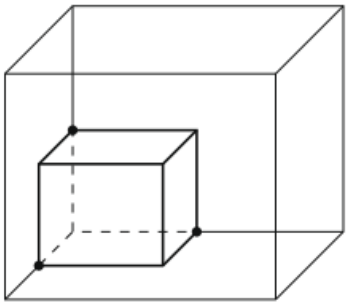
Sürekli Çözüm Kümesi	**3,29**	**1,00**	**4,59**	**5,00**	**2,00**	**5,10**
LRV	4	6	3	2	5	1
SPV	2	5	1	3	4	6
LOV	6	4	3	1	5	2
ROV	**3**	**1**	**4**	**5**	**2**	**6**

3.4. Yerleştirme Stratejisi

Kutuların konteynerlere yerleştirilmesinde kullanılan strateji, 3-D BPP çalışmalarında büyük öneme sahiptir. Yerleştirme stratejisi, kutuların boyutları, konteyner boyutları ve diğer ilgili kısıtlar göz önünde bulundurularak tasarlanır. Bu strateji, kutuların yerleştirilme sırasını ve pozisyonunu belirleyerek, konteynerlerin doluluk oranını optimize etmeyi hedefler. Bu tez çalışmasında, yerleştirme stratejisi olarak Köşe Noktalar Algoritması benimsenmiştir. Bu algoritma, her bir konteynerin köşe noktalarını belirleyerek, kutuların bu noktalara yerleştirilmesini sağlar.

Köşe noktalar algoritması (Corner Points): kutuların konteynerlere yerleştirilmesi sırasında kutunun köşe noktalarını dikkate alır. Her yerleştirilmiş kutu, takip eden kutular için yeni olası yerleştirme noktalarını temsil eden, üç köşe noktası (yukarı, sağa ve ön) oluşturur

ve yeni yerleşecek kutu bu noktalardan bir tanesine yerleştirilir. Uygulama sırasında, köşe noktalarını her düğümünde yeniden hesaplama yapmak, problemin zorluk derecesini artıran unsurlardan biridir. Şekil 3.1'de kutuya yerleştirilen nesnenin köşe noktaları gösterilmiştir (Silva ve Wauters 2019).



Şekil 3.1 Ekstrem Noktalar

Bu yöntem, kutuları hacimlerine göre sıralama ve uygun pozisyonlara yerleştirme sayesinde, işlemi basit ve anlaşılır bir şekilde ele alır. Aynı zamanda bir dallanma ve sınırlandırma algoritması olarak kullanıldığından, çözüm uzayını daha etkili bir şekilde inceleyebilir. Köşe noktalarını oluşturarak geçerli pozisyonları değerlendirmek, kutuların yerleştirme seçeneklerini daha iyi anlamak ve optimize etmek için kullanılabilir (Martello vd, 2000).

3.5. Uygunluk Fonksiyonu

3-D BPP'nin çözümünde, uygunluk fonksiyonunun çıktısı olarak konteyner sayısı kullanılır ise bu, algoritma durgunluğu adı verilen bir soruna yol açabilir. Bunun nedeni birden fazla dizilimin aynı konteyner sayısını verebilecek olmasıdır. Bu nedenle algoritmaların verimliliğini artırmak için konteyner doluluk oranının (DO) uygunluk fonksiyonunun çıktısı olarak kullanılması daha avantajlı olacaktır (Gezici ve Livatyalı, 2023).

$$DO_i = \frac{\sum_{j=1}^{n} w_j . d_j . h_j}{W . D . H} \tag{3.1}$$

$$minf = 1 - \left(\frac{\sum_{i=1}^{m} (DO_i)^2}{m} \right) \tag{3.2}$$

Burada n kutuların sayısını, w_j j. kutunun genişliğini,d_j j. kutunun derinliğini, h_j j. kutunun yüksekliğini $(j \in \{1, \dots, n\}), W$, konteynerin genişliğini, D, konteynerin derinliğini, H, konteynerin yüksekliğini, DO_i i. konteynerin doluluk oranını, m kullanılan toplam konteyner sayısını, $minf$ ise uygunluk fonksiyonunu göstermektedir.

3.6. Analizlerin Raporlanması

Bu bölümde, çalışmanın temel sonuçları paylaşılmış ve önemli performans ölçütleri olan çözüm süreleri, doluluk oranları ve konteyner sayıları analiz edilmiştir. Sonuçlar, önceki kısımda belirtilen 9 farklı durum üzerinde değerlendirilmiş ve her durum için minimum ve ortalama değerler elde edilmiştir. Bu sonuçlar, çalışmanın hedeflerine ulaşma derecesini ve algoritmaların performansını değerlendirmek için kullanılmıştır. Her durum 30 tekrarlı çözümlerle elde edilmiştir.

Çözüm süreleri, her bir durumda algoritmanın işleyiş hızını gösteren önemli bir ölçüt olmuştur. Minimum çözüm süreleri, en hızlı ve en verimli sonuçları temsil etmektedir. Ortalama çözüm süreleri ise genel bir bakış açısı sunarak, algoritmanın istikrarlı bir şekilde nasıl performans gösterdiğini göstermektedir.

Doluluk oranları, kutuların konteynerlere ne kadar verimli bir şekilde yerleştirildiğini gösteren bir ölçüttür. Yüksek doluluk oranları, kutuların mümkün olduğunca az boş alan bırakacak şekilde yerleştirildiğini gösterirken, düşük doluluk oranları daha fazla boş alanın olduğunu işaret etmektedir. Çalışmada doluluk oranı, diğer ölçütlere kıyasla daha fazla önemsenmektedir. Bunun nedeni; kutuların konteynerlere verimli bir şekilde yerleştirilmesini sağlayarak maliyetleri azaltması, nakliye verimliliğini artırması ve kaynakların etkin kullanımını sağlamasıdır. Sonuçta, yüksek doluluk oranı hedeflenmiştir.

Konteyner sayıları, kullanılan konteynerlerin toplam sayısını göstermektedir. Minimum konteyner sayıları, en verimli şekilde yerleştirme yapıldığını ve minimum miktarda konteynerin kullanıldığını gösterirken, ortalama konteyner sayıları genel bir perspektif sunarak, algoritmanın genel olarak konteyner kullanımını ne kadar etkin bir şekilde optimize ettiğini göstermektedir.

Sonuçlar, ilerleyen kısımdaki tablolarda ve eklerde özetlenmiştir. Analiz edilen veriler, algoritmaların performansını değerlendirmek ve gelecekteki çalışmalara temel oluşturmak için önemli bir kaynak sağlamaktadır.

Ek1-Ek9'da, ele alınan 9 durum için minimum (S_min) ve ortalama (S_ort) çözüm süreleri verilmektedir. Ek1'de 100 iterasyon n popülasyon durumu için çözüm süreleri paylaşılmıştır. Bu değerlere göre; TSO, hem minimum hem ortalama sürelerde 8 sınıfın 32 alt grubunun tamamında diğer algoritmalardan daha iyi değerler elde ederek öne geçmiştir. Ek2'de 100 iterasyon 2n popülasyon durumu için çözüm süreleri paylaşılmıştır. TSO 32 alt grubun 31'inde en iyi minimum çözüm süresi değerlerini elde etmiş, ortalama çözüm sürelerinin tamamında diğer algoritmalara üstünlüğünü sürdürmüştür. Ek3'de 100 iterasyon

4n popülasyon durumu için çözüm süreleri paylaşılmıştır. TSO aynı şekilde 32 durumun tamamında hem minimum hem de ortalama çözüm süreleri için en iyi değerleri elde etmiştir.

Ek4'de 500 iterasyon n popülasyon durumu için çözüm süreleri paylaşılmıştır. TSO'nun üstünlüğü yerini FA'ya bırakmıştır, minimum sürelere bakıldığında 32 alt grubun 24'ünde FA 7'sinde TSO, yalnızca 1'inde HBA en iyi çözüm sürelerini vermiştir. Ortalama sürelerde 29 alt grupta FA, 2 alt grupta HBA, 1 alt grupta TSO en iyi değerleri elde etmiştir. Ortalama değerlerde FA'nın daha yüksek bir başarı elde etmesi algoritmanın tutarlı biçimde iyi değerlere yakın çözümler elde ettiğini göstermektedir. Ek5'de 500 iterasyon 2n popülasyon durumu için çözüm süreleri paylaşılmıştır. Minimum çözüm sürelerinin 32 alt grubunun 13'ünde FA, 11'inde TSO, 7'sinde HBA, 1'inde CS en iyi değerleri elde etmiştir. Ortalama çözüm sürelerinin ise 32 alt grubunun 16'sında FA, 12'sinde TSO, 4'ünde HBA en iyi değerleri vermiştir. Ek6'da 500 iterasyon 4n popülasyon durumu için çözüm süreleri paylaşılmıştır. Minimum çözüm süreleri için alt grupların 17'sinde TSO, 13'ünde FA, 2'sinde HBA en iyi değerleri elde etmiştir. Ortalama çözüm sürelerinde ise 19 durumda TSO, 13 durumda FA, en iyi değerleri vermiştir. Bu sonuçlara göre iterasyon sayısı arttıkça FA öne geçerken, popülasyon büyüklüğündeki artış ise TSO'nun etkinliğini artırmaktadır.

Ek7'de 1000 iterasyon n popülasyon durumu için çözüm süreleri paylaşılmıştır. Minimum çözüm sürelerine göre 16 alt grupta HBA, 8 alt grupta FA, 7 alt grupta CS, TSO 1, öne geçmiştir. Ortalama çözüm sürelerinde ise 16 alt grupta HBA, 8 alt grupta FA, 8 alt grupta CS, en iyi değerleri elde etmiştir. İterasyon sayısının artması HBA'nın çözüm ivmesini artırmıştır. Ek8'de 1000 iterasyon 2n popülasyon durumu için çözüm süreleri paylaşılmıştır. Minimum çözüm sürelerinin alt gruplarının 12'sinde FA, 11'inde TSO, 8'inde CS, 1'inde HBA en iyi değerleri vermiştir. Ortalama çözüm sürelerine göre, 12 alt grupta FA, 9 alt grupta CS, 8 alt grupta TSO, 3 alt grupta ise HBA en iyi değerleri vermiştir. Ek9'da 1000 iterasyon 2n popülasyon durumu için çözüm süreleri paylaşılmıştır. Minimum çözüm sürelerine göre 24 alt grupta FA, 4 alt grupra TSO, 2 alt grupta HBA en iyi değerleri elde etmiştir. Ortalama çözüm sürelerine göre ise, 25 alt grupta FA, 2 alt grupta TSO, 1 alt grupta HBA en iyi değerleri vermiştir. İterasyon sayısının artışıyla en iyi değerleri vermeye başlayan HBA popülasyon büyüklüğünün artmasıyla diğer algoritmaların gerisinde kalmıştır. FA ve CS ise tutarlı biçimde farklı popülasyon büyüklerinde iyi değerler ortaya koymuştur. Ortalama süreler de dikkate alındığında FA 1000 iterasyon durumunun 2n ve 4n durumlarıyla diğer algoritmaların önüne geçmiştir, hemen ardında CS iyi değerlerle FA'ya yakın bir performans elde etmiştir.

Tablo 3.5'de algoritmaların tüm durumlardan elde ettiği toplam minimum ve ortalama çözüm süreleri ifade edilmiştir. Burada da görüleceği gibi başlangıçtaki durumlarda daha iyi sonuçlar veren TSO, iterasyon sayısı ve popülasyon büyüklüğü artışıyla yerini FA'ya bırakmıştır. Tüm durumlar için toplam çözüm sürelerine bakıldığında ise FA hem minimum hem ortalama değerlerde en iyi sürelere ulaşmıştır.

Tablo 3.5 Toplam Çözüm Süreleri (milisaniye)

	FA S_min	CS S_min	TSO S_min	HBA S_min	FA S_ort	CS S_ort	TSO S_ort	HBA S_ort
Durum 1	18.385	85.507	**7.553**	26.987	20.849	95.962	**8.213**	30.333
Durum 2	151.303	315.619	**99.016**	176.210	165.490	341.442	**113.856**	194.830
Durum 3	252.832	545.417	**190.249**	299.238	282.111	592.311	**212.778**	340.229
Durum 4	**176.551**	529.436	186.816	204.979	**197.416**	572.646	207.209	230.269
Durum 5	503.030	790.777	**443.515**	505.649	552.886	863.109	**504.864**	566.889
Durum 6	**437.003**	1317.927	441.270	554.815	**492.375**	1428.792	493.214	623.151
Durum 7	500.376	821.347	532.503	**478.880**	549.364	900.784	595.733	**526.967**
Durum 8	**887.906**	1146.125	970.637	1039.766	**975.513**	1259.710	1091.141	1154.106
Durum 9	**1289.569**	1990.604	1349.900	1407.649	**1438.134**	2189.938	1531.280	1578.291

Ek10-Ek18'de, ele alınan 9 durum için minimum (DO_min) ve ortalama (DO_ort) doluluk oranları verilmektedir. Ek10'de 100 iterasyon n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarının 32 alt grubunun 21'inde CS, 5'inde FA, 5'inde HBA, 1'inde TSO öne geçmiştir. Ortalama doluluk oranlarının değerlerine göre ise CS tamamından daha iyi sonuçlar elde etmiştir. Ek11'de 100 iterasyon 2n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarında 32 alt grubun 10'unda CS, 10'unda TSO, 8'inde FA, 4'ünde HBA en küçük değerleri elde etmiştir. Ortalama doluluk oranlarında ise alt grupların 27'sinde CS, 4'ünde HBA, 1'inde FA en iyi değeri elde etmiştir. Bu durum, CS'nin tutarlı biçimde en iyi değerlere yaklaştığını göstermektedir. Ek12'de 100 iterasyon 4n popülasyon durumu için doluluk oranları paylaşılmıştır. Bu değerlere göre, minimum doluluk oranlarının alt gruplarının 12'sinde CS, 9'unda FA, 6'sında HBA, 5'inde TSO öne geçmiştir. Ortalama doluluk oranlarında ise alt grupların 26'sında CS, 3'ünde FA, 2'sinde HBA, 1'inde TSO en iyi değerleri vermiştir.

Ek13'de 500 iterasyon n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarına bakıldığında 32 alt grubun 13'ünde CS, 12'sinde FA, 4'ünde TSO, 3'ünde HBA en iyi doluluk oranlarını vermiştir. Ortalama doluluk oranlarında ise 19 alt grupta CS, 12 alt grupta FA, 1 alt grupta HBA en iyi değerleri elde etmiştir. Ek14'de 500 iterasyon 2n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarına bakıldığında 32 alt grubun 15'inde FA, 12'sinde CS, 3'ünde HBA, 2'sinde TSO en

iyi doluluk oranlarını vermiştir. Ortalama doluluk oranlarında ise 17 alt grupta FA, 13 alt grupta CS, 1 alt grupta TSO, 1 alt grupta HBA en iyi değerleri elde etmiştir. Ek15'de 500 iterasyon 4n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarına bakıldığında 32 alt grubun 15'inde CS, 11'inde FA, 5'inde HBA, 1'inde TSO en iyi doluluk oranlarını vermiştir. Ortalama doluluk oranlarında ise 18 alt grupta CS, 9 alt grupta FA, 3 alt grupta HBA, 2 alt grupta TSO en iyi değerleri elde etmiştir.

Ek16'da 1000 iterasyon n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarına bakıldığında 32 alt grubun 14'ünde FA, 9'unda CS, 5'inde TSO, 4'ünde HBA en iyi doluluk oranlarını vermiştir. Ortalama doluluk oranlarında ise 23 alt grupta FA, 7 alt grupta CS, 2 alt grupta HBA en iyi değerleri elde etmiştir. Ek17'de 1000 iterasyon 2n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarına bakıldığında 32 alt grubun 22'sinde FA, 5'inde CS, 3'ünde HBA, 2'sinde TSO en iyi doluluk oranlarını vermiştir. Ortalama doluluk oranlarında ise 25 alt grupta FA, 6 alt grupta CS, 1 alt grupta TSO en iyi değerleri elde etmiştir. Ek18'de 1000 iterasyon 4n popülasyon durumu için doluluk oranları paylaşılmıştır. Minimum doluluk oranlarına bakıldığında 32 alt grubun 23 FA, 5 CS, 2 TSO, 2 HBA en iyi doluluk oranlarını vermiştir. Ortalama doluluk oranlarında ise 27 alt grupta FA, 4 alt grupta CS, 1 alt grupta TSO en iyi değerleri elde etmiştir.

Tablo 3.6'da algoritmaların tüm durumlardan elde edilen toplam minimum ve ortalama doluluk oranları ifade edilmiştir. Toplam değerlere bakıldığında CS'nin iterasyon sayısı 100 ve 500 olan ilk 6 durumda en iyi değerleri elde ettiği görülürken, iterasyon sayısı 1000'e çıktığında FA'nın hem minimum hem ortalama değerler için daha iyi sonuçlar elde ettiği görülmektedir.

Tablo 3.6 Toplam Doluluk Oranları

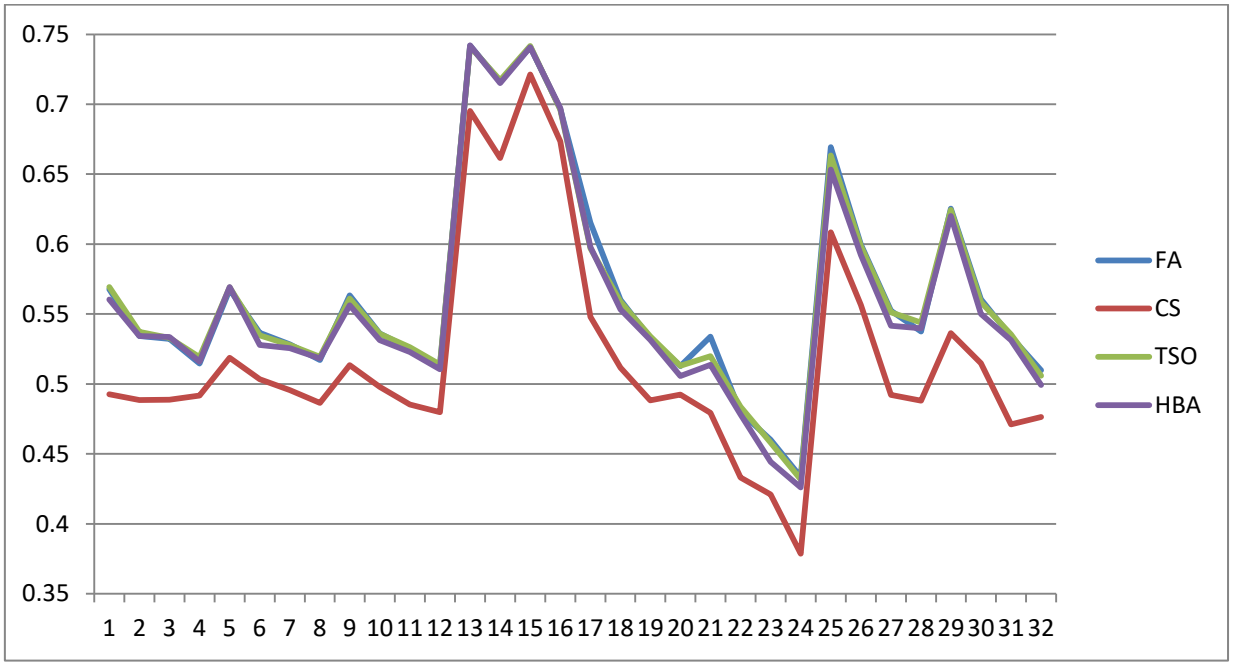
	FA DO_min	CS DO_min	TSO DO_min	HBA DO_min	FA DO_ort	CS DO_ort	TSO DO_ort	HBA DO_ort
Durum 1	17.017	**16.591**	16.940	16.785	18.049	**16.591**	18.021	17.880
Durum 2	16.521	**16.368**	16.486	16.513	17.599	**17.423**	17.602	17.541
Durum 3	16.395	**16.332**	16.433	16.351	17.454	**17.281**	17.514	17.412
Durum 4	16.218	**16.162**	16.482	16.425	17.329	**17.253**	17.526	17.438
Durum 5	16.111	**16.085**	16.337	16.279	**17.138**	17.144	17.371	17.292
Durum 6	16.207	**16.156**	16.283	16.277	17.251	**17.184**	17.342	17.280
Durum 7	**16.089**	16.144	16.283	16.175	**17.100**	17.165	17.395	17.286
Durum 8	**15.886**	15.961	16.183	16.113	**16.957**	17.048	17.246	17.178
Durum 9	**15.841**	16.002	16.229	16.106	**16.897**	17.034	17.286	17.188

Tezin kapsamında elde edilen sonuçların daha etkili bir şekilde sunulması amaçlanarak, çizgi grafik üzerinde çözüm yöntemlerinin doluluk oranları paylaşılmaktadır.

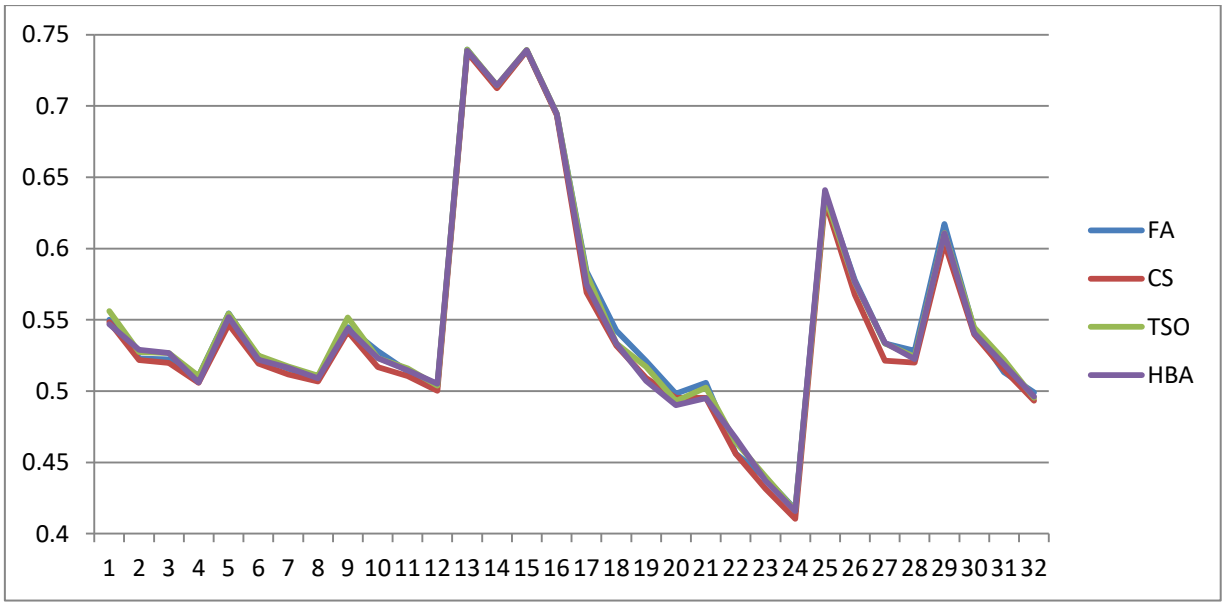
Bu grafikler, her bir yöntemin 32 alt grup için elde ettiği doluluk oranlarını görsel bir şekilde yansıtarak, çözüm yöntemlerinin konteyner dolulukları üzerindeki etkilerini ve performanslarını daha anlaşılır bir şekilde ortaya koymayı hedeflemektedir.

Değerler daha önce belirtilen 9 farklı durum için FA, CS, TSO ve HBA yöntemlerinin 32 alt grupta doluluk oranlarını temsil etmektedir. Tablo 3.7 - Tablo 3.15 bu durumları ele almaktadır.

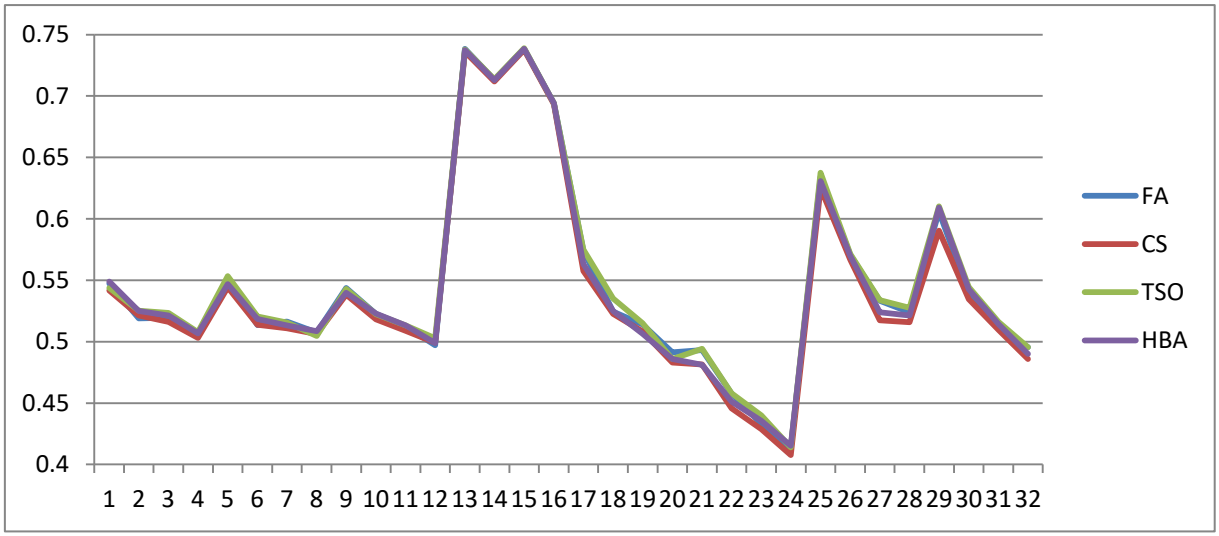
Tablo 3.7 Durum 1 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



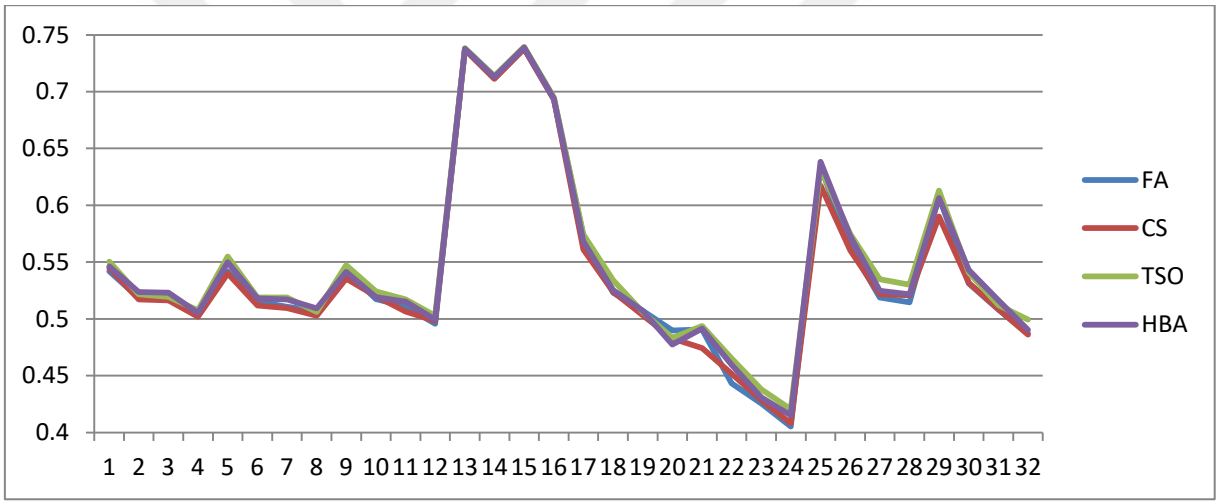
Tablo 3.8 Durum 2 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



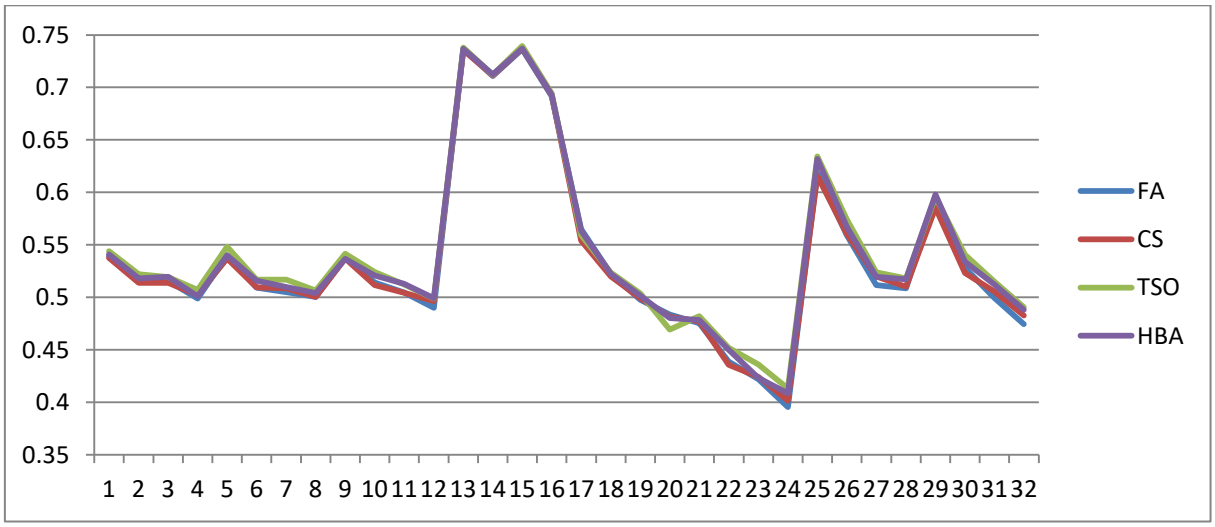
Tablo 3.9 Durum 3 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



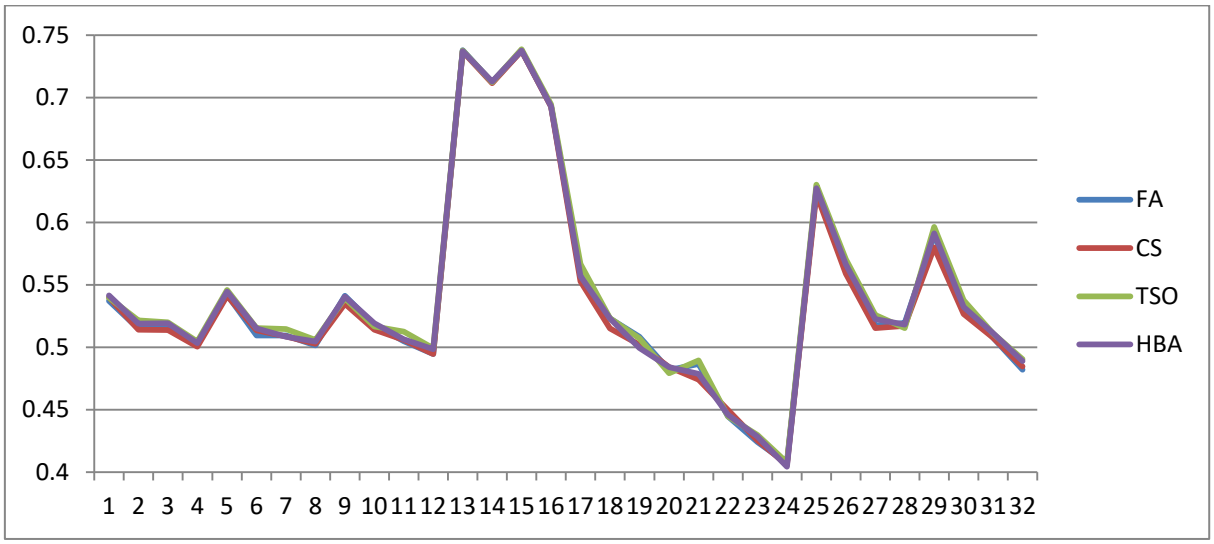
Tablo 3.10 Durum 4 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



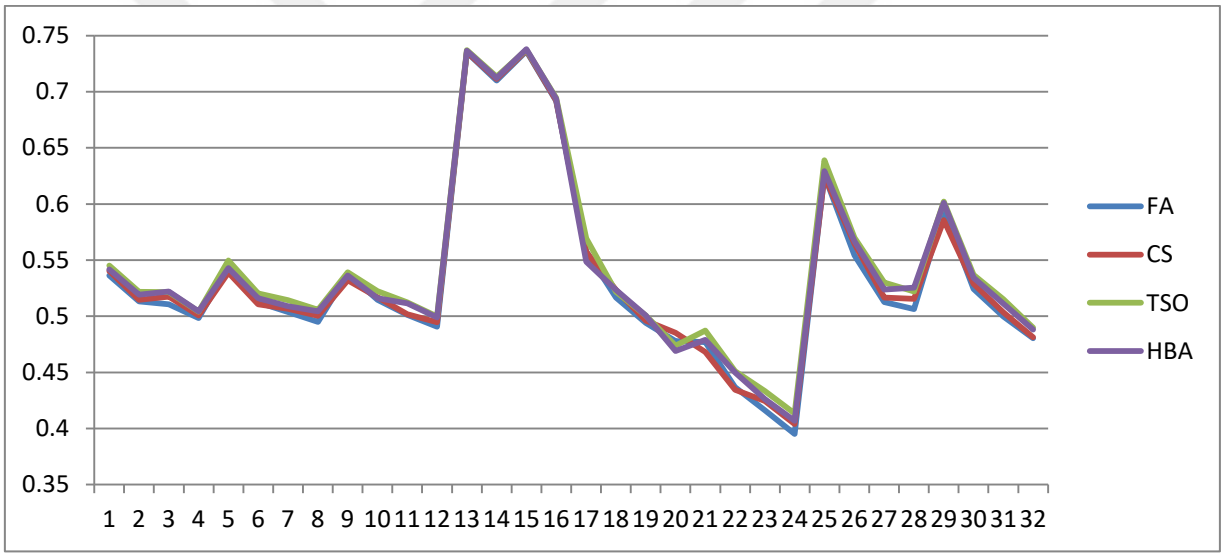
Tablo 3.11 Durum 5 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



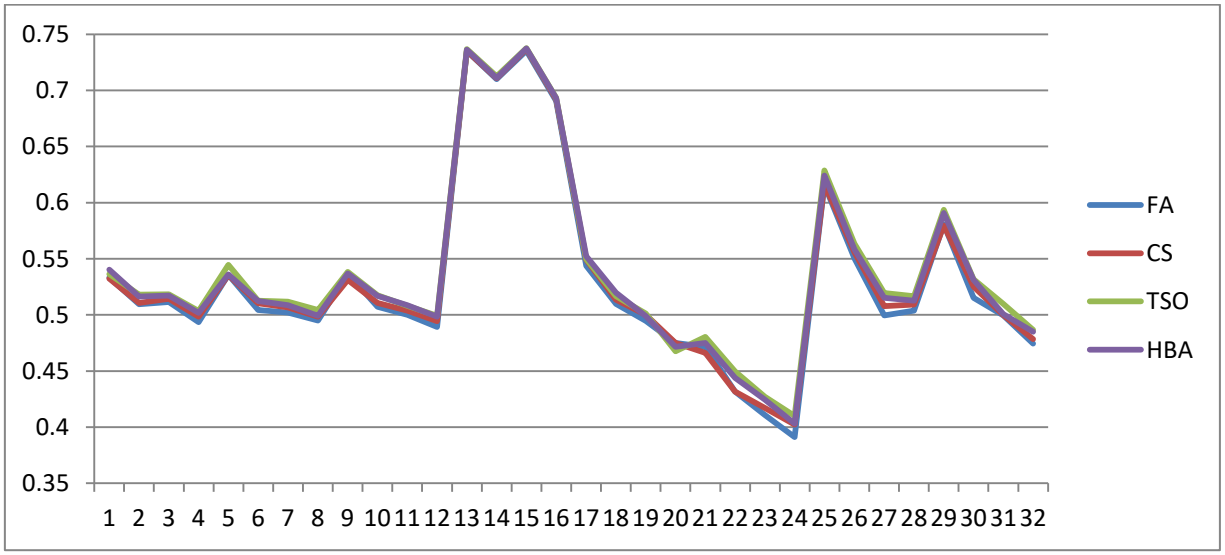
Tablo 3.12 Durum 6 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



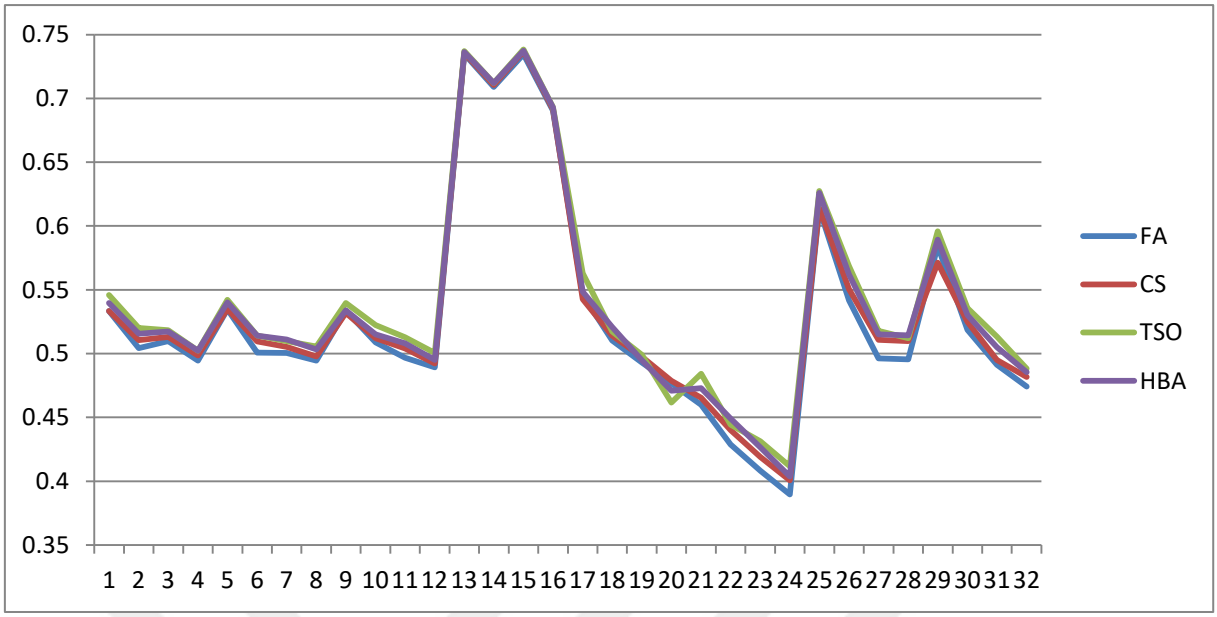
Tablo 3.13 Durum 7 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



Tablo 3.14 Durum 8 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



Tablo 3.15 Durum 9 için Doluluk Oranlarına göre Algoritmaların Kıyaslanması



Yukarıdaki grafiklerde görüldüğü gibi doluluk oranı değerleri oldukça yakın seyretmektedir. Ancak yöntemler tam anlamıyla aynı çözüm değerlerine de sahip değildir. Bu nedenle aralarındaki farkın istatiksel açıdan anlamlı bir fark olup olmadığının testine ihtiyaç duyulmaktadır.

Bu tez çalışmasında doluluk oranlarına göre yöntemler arasında anlamlı bir farklılık olup olmadığı Friedman testiyle analiz edilmiştir.

Durum 1 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 67{,}763,\ p < .05$).

Tablo 3.16 Durum 1 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	0,5640	0,07621	3,28
CS	0,5185	0,07633	1,00
TSO	0,5632	0,07570	3,31
HBA	0,5588	0,07720	2,41
$\chi^2 = 67{,}763$	$P = 0{,}000$		

Yöntemlerden hangileri arasında anlamlı farklılıklar olduğu ise Wilcoxon testi ile analiz edilmiş olup test sonuçları Tablo 3.17'de görülmektedir.

Durum 1 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, sadece TSO ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür.

Tablo 3.17 Durum 1 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	**Sıra Ortalaması**	**Sıra Toplamı**	***Z***	***p***
CS - FA	Negatif Sıralar	32[a]	16,50	528,00		
	Pozitif Sıralar	0[b]	,00	,00	-4,937[m]	,000
TSO - FA	Negatif Sıralar	17[c]	16,82	286,00		
	Pozitif Sıralar	15[d]	16,13	242,00	-,411[m]	,681
HBA - FA	Negatif Sıralar	24[e]	19,75	474,00		
	Pozitif Sıralar	8[f]	6,75	54,00	-3,927[m]	,000
TSO - CS	Negatif Sıralar	0[g]	,00	,00		
	Pozitif Sıralar	32[h]	16,50	528,00	-4,937[n]	,000
HBA - CS	Negatif Sıralar	0[i]	,00	,00		
	Pozitif Sıralar	32[j]	16,50	528,00	-4,937[n]	,000
HBA - TSO	Negatif Sıralar	27[k]	18,96	512,00		
	Pozitif Sıralar	5[l]	3,20	16,00	-4,637[m]	,000

a. CS < FA
b. CS > FA
c. TSO < FA
d. TSO > FA
e. HBA < FA
f. HBA > FA
g. TSO < CS
h. TSO > CS
i. HBA < CS
j. HBA > CS
k. HBA < TSO
l. HBA > TSO
m. Pozitif sıralamalara dayalı
n. Negatif sıralamalara dayalı

Tablodaki negatif ve pozitif sayılar ilk satırdan itibaren ele alınacak olursa; burada CS<FA durumu, FA'nın CS'den daha büyük değerleri olduğunu gösterir. Ancak bu tez çalışmasında amaç fonksiyonu minimizasyon yönlü olduğu ve istatistik testleri, konteynerde kalan boş alanı işaret ettiği için büyük değerler daha fazla boşluk kaldığı anlamına gelir. Bu istenmeyen durum sebebi ile daha küçük değerlere sahip yöntemlerin daha iyi sonuçlar verdiği ifade edilebilir. CS yöntemi FA, TSO ve HBA'dan daha iyi sonuçlar vermiştir. Sonraki durumlarda da değerler bu kapsamda incelenmiştir.

Durum 2 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 49{,}950,\ p < .05$).

Tablo 3.18 Durum 2 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	**SS**	**Sıra Ortalaması**
FA	,5500	,07891	3,06
CS	,5445	,07974	1,19
TSO	,5501	,07863	3,25
HBA	,5482	,07918	2,50

$\chi^2 = 49{,}950$ $P = 0{,}000$

Durum 2 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, TSO ile FA ve HBA ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.19'da görülmektedir.

Tablo 3.19 Durum 2 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	**Sıra Ortalaması**	**Sıra Toplamı**	***Z***	***p***
CS - FA	Negatif Sıralar	31[a]	16,81	521,00		
	Pozitif Sıralar	1[b]	7,00	7,00	-4,806[m]	,000
TSO - FA	Negatif Sıralar	15[c]	17,13	257,00		
	Pozitif Sıralar	17[d]	15,94	271,00	-,131[n]	,896
HBA - FA	Negatif Sıralar	20[e]	18,65	373,00		
	Pozitif Sıralar	12[f]	12,92	155,00	-2,038[m]	,042
TSO - CS	Negatif Sıralar	1[g]	7,00	7,00		
	Pozitif Sıralar	31[h]	16,81	521,00	-4,806[n]	,000
HBA - CS	Negatif Sıralar	4[i]	9,75	39,00		
	Pozitif Sıralar	28[j]	17,46	489,00	-4,207[n]	,000
HBA - TSO	Negatif Sıralar	24[k]	17,42	418,00		
	Pozitif Sıralar	8[l]	13,75	110,00	-2,880[m]	,004

a. CS < FA
b. CS > FA
c. TSO < FA
d. TSO > FA
e. HBA < FA
f. HBA > FA
g. TSO < CS
h. TSO > CS
i. HBA < CS
j. HBA > CS
k. HBA < TSO
l. HBA > TSO
m. Pozitif sıralamalara dayalı
n. Negatif sıralamalara dayalı

Durum 3 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 54{,}975,\ p < .05$).

Tablo 3.20 Durum 3 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	**SS**	**Sıra Ortalaması**
FA	,5454	,07971	2,66
CS	,5400	,08055	1,22
TSO	,5473	,07931	3,59
HBA	,5441	,08000	2,53

$\chi^2 = 54{,}975$ $P = 0{,}000$

Durum 3 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, HBA ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.21'de görülmektedir.

Tablo 3.21 Durum 3 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	**Sıra Ortalaması**	**Sıra Toplamı**	***Z***	***p***
CS - FA	Negatif Sıralar	29[a]	17,55	509,00		
	Pozitif Sıralar	3[b]	6,33	19,00	-4,581[m]	,000
TSO - FA	Negatif Sıralar	8[c]	12,63	101,00		
	Pozitif Sıralar	24[d]	17,79	427,00	-3,048[n]	,002
HBA - FA	Negatif Sıralar	16[e]	19,88	318,00		
	Pozitif Sıralar	16[f]	13,13	210,00	-1,010[m]	,313
TSO - CS	Negatif Sıralar	1[g]	3,00	3,00		
	Pozitif Sıralar	31[h]	16,94	525,00	-4,880[n]	,000
HBA - CS	Negatif Sıralar	3[i]	3,67	11,00		
	Pozitif Sıralar	29[j]	17,83	517,00	-4,731[n]	,000
HBA - TSO	Negatif Sıralar	28[k]	16,82	471,00		
	Pozitif Sıralar	4[l]	14,25	57,00	-3,871[m]	,000

a. CS < FA e. HBA < FA i. HBA < CS
b. CS > FA f. HBA > FA j. HBA > CS m. Pozitif sıralamalara dayalı
c. TSO < FA g. TSO < CS k. HBA < TSO n. Negatif sıralamalara dayalı
d. TSO > FA h. TSO > CS l. HBA > TSO

Durum 4 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 57{,}638,\ p < .05$).

Tablo 3.22 Durum 4 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	,5415	,08156	1,94
CS	,5391	,08043	1,44
TSO	,5477	,07896	3,66
HBA	,5449	,08016	2,97

$\chi^2 = 57{,}638$ $P = 0{,}000$

Durum 4 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, CS ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.23'de görülmektedir.

Tablo 3.23 Durum 4 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	Sıra Ortalaması	Sıra Toplamı	Z	p
CS - FA	Negatif Sıralar	20[a]	18,35	367,00		
	Pozitif Sıralar	12[b]	13,42	161,00	-1,926[m]	,054
TSO - FA	Negatif Sıralar	4[c]	7,75	31,00		
	Pozitif Sıralar	28[d]	17,75	497,00	-4,357[n]	,000
HBA - FA	Negatif Sıralar	6[e]	12,17	73,00		
	Pozitif Sıralar	26[f]	17,50	455,00	-3,571[n]	,000
TSO - CS	Negatif Sıralar	0[g]	,00	,00		
	Pozitif Sıralar	32[h]	16,50	528,00	-4,937[n]	,000
HBA - CS	Negatif Sıralar	2[i]	9,00	18,00		
	Pozitif Sıralar	30[j]	17,00	510,00	-4,600[n]	,000
HBA - TSO	Negatif Sıralar	25[k]	17,20	430,00		
	Pozitif Sıralar	7[l]	14,00	98,00	-3,104[m]	,002

a. CS < FA e. HBA < FA i. HBA < CS
b. CS > FA f. HBA > FA j. HBA > CS m. Pozitif sıralamalara dayalı
c. TSO < FA g. TSO < CS k. HBA < TSO n. Negatif sıralamalara dayalı
d. TSO > FA h. TSO > CS l. HBA > TSO

Durum 5 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 51{,}862,\ p < .05$).

Tablo 3.24 Durum 5 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	,5356	,08279	1,72
CS	,5358	,08171	1,75
TSO	,5428	,08055	3,69
HBA	,5404	,08119	2,84

$\chi^2 = 51{,}862 \qquad P = 0{,}000$

Durum 5 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, CS ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.25'de görülmektedir.

Tablo 3.25 Durum 5 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	Sıra Ortalaması	Sıra Toplamı	*Z*	*p*
CS - FA	Negatif Sıralar	15[a]	17,00	255,00		
	Pozitif Sıralar	17[b]	16,06	273,00	-,168[m]	,866
TSO - FA	Negatif Sıralar	2[c]	18,50	37,00		
	Pozitif Sıralar	30[d]	16,37	491,00	-4,245[m]	,000
HBA - FA	Negatif Sıralar	6[e]	6,17	37,00		
	Pozitif Sıralar	26[f]	18,88	491,00	-4,245[m]	,000
TSO - CS	Negatif Sıralar	1[g]	27,00	27,00		
	Pozitif Sıralar	31[h]	16,16	501,00	-4,432[m]	,000
HBA - CS	Negatif Sıralar	6[i]	5,33	32,00		
	Pozitif Sıralar	26[j]	19,08	496,00	--4,338[m]	,000
HBA - TSO	Negatif Sıralar	25[k]	18,08	452,00		
	Pozitif Sıralar	7[l]	10,86	76,00	-3,515[n]	,000

a. CS < FA
b. CS > FA
c. TSO < FA
d. TSO > FA
e. HBA < FA
f. HBA > FA
g. TSO < CS
h. TSO > CS
i. HBA < CS
j. HBA > CS
k. HBA < TSO
l. HBA > TSO
m. Pozitif sıralamalara dayalı
n. Negatif sıralamalara dayalı

Durum 6 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 35{,}362,\ p < .05$).

Tablo 3.26 Durum 6 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	,5391	,08138	2,22
CS	,5370	,08097	1,59
TSO	,5419	,08087	3,44
HBA	,5400	,08104	2,75

$\chi^2 = 35{,}362 \qquad P = 0{,}000$

Durum 6 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, CS ile FA ve HBA ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde

anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.27'de görülmektedir.

Tablo 3.27 Durum 6 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	Sıra Ortalaması	Sıra Toplamı	*Z*	*p*
CS - FA	Negatif Sıralar	22[a]	17,59	387,00		
	Pozitif Sıralar	10[b]	14,10	141,00	-2,300[m]	,021
TSO - FA	Negatif Sıralar	6[c]	10,67	64,00		
	Pozitif Sıralar	26[d]	17,85	464,00	-3,740[n]	,000
HBA - FA	Negatif Sıralar	11[e]	14,64	161,00		
	Pozitif Sıralar	21[f]	17,48	367,00	-1,926[n]	,054
TSO - CS	Negatif Sıralar	4[g]	10,50	42,00		
	Pozitif Sıralar	28[h]	17,36	486,00	-4,151[n]	,000
HBA - CS	Negatif Sıralar	5[i]	8,80	44,00		
	Pozitif Sıralar	27[j]	17,93	484,00	-4,114[n]	,000
HBA - TSO	Negatif Sıralar	24[k]	17,50	420,00		
	Pozitif Sıralar	8[l]	13,50	108,00	-2,917[m]	,004

a. CS < FA
b. CS > FA
c. TSO < FA
d. TSO > FA
e. HBA < FA
f. HBA > FA
g. TSO < CS
h. TSO > CS
i. HBA < CS
j. HBA > CS
k. HBA < TSO
l. HBA > TSO
m. Pozitif sıralamalara dayalı
n. Negatif sıralamalara dayalı

Durum 7 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 63{,}413,\ p < .05$).

Tablo 3.28 Durum 7 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	,5344	,08325	1,41
CS	,5364	,08207	1,91
TSO	,5436	,08065	3,75
HBA	,5402	,08129	2,94

$\chi^2 = 63{,}413$ $P = 0{,}000$

Wilcoxon testi sonuçları Tablo 3.29'da görülmektedir.

Tablo 3.29 Durum 7 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	Sıra Ortalaması	Sıra Toplamı	*Z*	*p*
CS - FA	Negatif Sıralar	8[a]	16,50	132,00		
	Pozitif Sıralar	24[b]	16,50	396,00	-2,468[m]	,014
TSO - FA	Negatif Sıralar	1[c]	6,00	6,00		
	Pozitif Sıralar	31[d]	16,84	522,00	-4,824[m]	,000
HBA - FA	Negatif Sıralar	4[e]	12,00	48,00		
	Pozitif Sıralar	28[f]	17,14	480,00	-4,039[m]	,000
TSO - CS	Negatif Sıralar	2[g]	13,50	27,00		
	Pozitif Sıralar	30[h]	16,70	501,00	-4,432[m]	,000
HBA - CS	Negatif Sıralar	3[i]	20,00	60,00		
	Pozitif Sıralar	29[j]	16,14	468,00	-3,815[m]	,000
HBA - TSO	Negatif Sıralar	27[k]	17,96	485,00		
	Pozitif Sıralar	5[l]	8,60	43,00	-4,132[n]	,000

a. CS < FA e. HBA < FA i. HBA < CS
b. CS > FA f. HBA > FA j. HBA > CS m. Pozitif sıralamalara dayalı
c. TSO < FA g. TSO < CS k. HBA < TSO n. Negatif sıralamalara dayalı
d. TSO > FA h. TSO > CS l. HBA > TSO

Durum 7 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, CS ile FA yöntemlerinin sonuçları arasında anlamlı bir farklılık bulunamamıştır. Diğer tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür.

Durum 8 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 63{,}413,\ p < .05$).

Tablo 3.30 Durum 8 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	,5299	,08383	1,25
CS	,5328	,08269	1,91
TSO	,5389	,08117	3,78
HBA	,5368	,08183	3,06
$\chi^2 = 63{,}413$	$P = 0{,}000$		

Durum 8 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.31'de görülmektedir.

Tablo 3.31 Durum 8 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	Sıra Ortalaması	Sıra Toplamı	Z	p
CS - FA	Negatif Sıralar	6[a]	12,50	75,00		
	Pozitif Sıralar	26[b]	17,42	453,00	-3,534[m]	,000
TSO - FA	Negatif Sıralar	1[c]	11,00	11,00		
	Pozitif Sıralar	31[d]	16,68	517,00	-4,731[m]	,000
HBA - FA	Negatif Sıralar	1[e]	9,00	9,00		
	Pozitif Sıralar	31[f]	16,74	519,00	-4,768[m]	,000
TSO - CS	Negatif Sıralar	1[g]	20,00	20,00		
	Pozitif Sıralar	31[h]	16,39	508,00	-4,563[m]	,000
HBA - CS	Negatif Sıralar	2[i]	12,50	25,00		
	Pozitif Sıralar	30[j]	16,77	503,00	-4,469[m]	,000
HBA - TSO	Negatif Sıralar	27[k]	16,44	444,00		
	Pozitif Sıralar	5[l]	16,80	84,00	-3,366[n]	,001

a. CS < FA e. HBA < FA i. HBA < CS
b. CS > FA f. HBA > FA j. HBA > CS m. Pozitif sıralamalara dayalı
c. TSO < FA g. TSO < CS k. HBA < TSO n. Negatif sıralamalara dayalı
d. TSO > FA h. TSO > CS l. HBA > TSO

Durum 9 için elde edilen verilere göre kullanılan yöntemler arasında anlamlı bir farklılık görülmektedir ($\chi^2 = 75{,}263,\ p < .05$).

Tablo 3.32 Durum 9 için Doluluk Oranlarına göre Friedman Testi Sonuçları

Algoritmalar	$\bar{X}$	SS	Sıra Ortalaması
FA	,5280	,08459	1,19
CS	,5323	,08203	1,97
TSO	,5402	,08124	3,75
HBA	,5371	,08164	3,09

$\chi^2 = 75{,}263$ $P = 0{,}000$

Durum 9 için elde edilen verilere göre yöntemler arasındaki farklılıklar incelendiğinde, tüm ikili yöntem karşılaştırmaları arasında $p < .01$ düzeyinde anlamlı farklılıklar olduğu görülmüştür. Wilcoxon testi sonuçları Tablo 3.33'de görülmektedir.

Tablo 3.33 Durum 9 için Doluluk Oranlarına göre Wilcoxon Testi Sonuçları

		N	Sıra Ortalaması	Sıra Toplamı	*Z*	*p*
CS - FA	Negatif Sıralar	4[a]	12,50	75,00		
	Pozitif Sıralar	28[b]	17,42	453,00	-3,964[m]	,000
TSO - FA	Negatif Sıralar	1[c]	11,00	11,00		
	Pozitif Sıralar	31[d]	16,68	517,00	-4,563[m]	,000
HBA - FA	Negatif Sıralar	1[e]	9,00	9,00		
	Pozitif Sıralar	31[f]	16,74	519,00	-4,731[m]	,000
TSO - CS	Negatif Sıralar	1[g]	20,00	20,00		
	Pozitif Sıralar	31[h]	16,39	508,00	-4,432[m]	,000
HBA - CS	Negatif Sıralar	2[i]	12,50	25,00		
	Pozitif Sıralar	30[j]	16,77	503,00	-4,301[m]	,000
HBA - TSO	Negatif Sıralar	26[k]	16,44	444,00		
	Pozitif Sıralar	6[l]	16,80	84,00	-3,347[n]	,001

a. CS < FA
b. CS > FA
c. TSO < FA
d. TSO > FA
e. HBA < FA
f. HBA > FA
g. TSO < CS
h. TSO > CS
i. HBA < CS
j. HBA > CS
k. HBA < TSO
l. HBA > TSO
m. Pozitif sıralamalara dayalı
n. Negatif sıralamalara dayalı

Ek19-Ek27'de, ele alınan 9 durum için minimum (Kon_no_min) ve ortalama (Kon_no_ort) konteyner sayıları verilmektedir. Ek19'da 100 iterasyon n popülasyon durumu için konteyner sayıları paylaşılmıştır. Bu değerlere göre; minimum konteyner sayıları için 8 sınıfın 1 tanesi hariç tüm 50 ve 100 kutulu alt örnekleri için her 4 algoritma da en iyi konteyner sayılarına ulaşmıştır. Kutu sayıları arttıkça, 150 ve 200 kutulu alt örnekler için 1.sınıfta CS ve FA, 2.sınıfta FA, 3.sınıfta CS ve HBA, 4.sınıfta FA, CS ve TSO, 5.sınıftan TSO ve HBA, 6.sınıfta HBA ve CS, 7.sınıfta CS'nin daha iyi konteyner sayılarına ulaştığı görülmektedir. Yalnızca 8.sınıfta tüm yöntemler en iyi değerleri elde etmiştir. Ortalama konteyner sayılarına bakıldığında ise algoritmalar arasındaki fark daha belirgin olarak ortaya çıkmaktadır. 32 alt grubun 24'ünde CS tek başına en iyi ortalama konteyner sayılarına ulaşmıştır, 5'inde HBA, 4'ünde TSO, 3'ünde FA iyi değerler elde etmiştir. 6'sında bu başarısını diğer algoritmalarla paylaşmıştır. Ona en yakın başarıyı yakalayan yöntem HBA 6 alt grupta en iyi değerleri elde etmiştir. Yalnızca 4. Sınıfın ilk 50-100-150 kutulu ilk üç örneği

için tüm algoritmalar en iyi çözüm değerlerini elde etmiştir. CS'nin ortalama değerlerde gösterdiği bu başarı algoritmanın tekrarlayan çözümler için tutarlı biçimde iyi sonuçlar verdiğini ifade etmektedir. Ek20'de 100 iterasyon 2n popülasyon durumu için konteyner sayıları paylaşılmıştır. Bu değerlere göre; minimum konteyner sayıları için 8 sınıfın 1 tanesi hariç hemen hemen tüm alt gruplarda algoritmalar en iyi değerleri elde etmiştir. Yalnızca 3.sınıfın 100 kutulu örneğinde CS diğerlerinden daha iyi bir sonuç elde etmiştir. Bu durum, algoritmaların popülasyondaki artıştan etkilendiğini göstermektedir. Ancak bu, beklenen bir durumdur. Popülasyon büyüklüğündeki artış, daha fazla bireyin aynı anda arama yapmasını sağlar, bu da arama uzayının daha kapsamlı bir şekilde keşfedilmesini sağlar. Daha büyük bir popülasyon, daha fazla çözüm adayını içerir ve olası daha iyi çözümler bulunma olasılığı artar. Bu nedenle ortalama konteyner sayıları daha iyi bir performans göstergesidir. Çünkü algoritmanın tekrarlayan çözümler içinde birkaç başarılı sonuç yakalaması ile değil tutarlı biçimde başarısını sürdürmesi ile ilgilidir. Ortalama konteyner sayılarına bakıldığında yalnızca 4 sınıfta ve 3.sınıfın 50 kutulu örneğinde tüm algoritmalar en iyi değerleri elde etmiştir. Buna göre, 32 alt grubun 23'ünde CS'nin en iyi değerleri bulmayı sürdüğü görülmektedir, 15'inde HBA, 12'sinde FA, 5'inde TSO en iyi değerleri elde etmiştir. Ek21'de 100 iterasyon 4n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında, 3.sınıfn 100 ve 150 kutulu örneği hariç hemen hemen tüm algoritmaların tüm alt gruplarda en iyi değerleri elde ettiği görülmektedir. Ortalama konteyner sayılarına bakıldığında ise 32 alt grubun 26'sında CS'nin, 14'ünde FA'nın, 12'sinde HBA'nın, 9'unda TSO'nun en iyi çözüm değerlerine ulaştığı görülmektedir.

Ek22'de 500 iterasyon n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında 1.sınıfın 200 kutulu örneği, 2.sınıfın 150 ve 200 kutulu örneği, 3.sınıfın 150 kutulu örneği ve 5.sınıfın 100 kutulu örneği dışında hemen hemen tüm algortimalar en iyi değerlere ulaşmıştır. Dışarda tutulan bu örneklerde CS'nin elde ettiği değerler diğer algoritmaların önüne geçmiştir. Ortalama değerlere bakıldığında, 32 alt grubun 23'ünde CS, 12'inde FA, 11'inde HBA, 8'sinde TSO iyi sonuçlar elde etmiştir. Ek23'de 500 iterasyon 2n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında 1.sınıfın 100 ve 200, 2.sınıfın 150, 3.sınıfın 100, 5.sınıfın 100 ve 200, 6.sınıfın 150 ve 200, 8.sınıfın 150 ve 200 kutulu örnekleri dışında hemen hemen tüm algoritmalar en iyi değerleri elde etmiştir. Ortalama değerlere bakıldığında, 32 alt grubun 23'ünde FA, 22'sinde CS, 12'sinde HBA, 9'unda TSO en iyi değerleri elde etmiştir. Ek24'de 500 iterasyon 4n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında 1.sınıfın 200, 3.sınıfın 100, 5.sınıfın 100 kutulu örneği

dışında hemen hemen bütün algoritmalar en iyi değerlere ulaşmıştır. Bu üç örnekte CS diğer algoritmaların önüne geçmiştir. Ortalama değerlere bakıldığında, 32 alt grubun 22'sinde CS, 15'inde FA, 13'ünde HBA, 8'inde TSO en iyi değerleri elde etmiştir.

Ek25'de 1000 iterasyon n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında 1.sınıfın 200, 2.sınıfın 100, 3.sınıfın 150, 5.sınıfın 100 ve 200, 6.sınıfın 150 ve 200, 8.sınıfın 100, 150 ve 200 kutulu örnekleri dışında hemen hemen tüm algoritmalar en iyi değerleri elde etmiştir. Ortalama değerlere bakıldığında, 32 alt grubun 25'inde FA, 15'inde CS, 10'unda HBA, 6'sında TSO en iyi değerlere ulaşmıştır. Ek26'da 1000 iterasyon 2n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında 1.sınıfın 200, 2.sınıfın 100, 3.sınıfın 200, 6.sınıfın 150, 8.sınıfın 100 ve 200 kutulu örnekleri dışında hemen hemen tüm algoritmalar en iyi değerleri elde etmiştir. Ortalama değerlere bakıldığında, 32 alt grubun 29'unda FA, 16'sında CS, 10'unda TSO, 9'unda HBA en iyi değerleri vermiştir. Ek27'de 1000 iterasyon 4n popülasyon durumu için konteyner sayıları paylaşılmıştır. Minimum konteyner sayılarına bakıldığında 1.sınıfın 200, 2.sınıfın 150, 3.sınıfın 150 ve 200, 5.sınıfın 50, 6.sınıfın 100,150 ve 200, 7.sınıfın 100, 8.sınıfın 200 kutulu örnekler dışında hemen hemen tüm algoritmalar en iyi değerleri elde etmiştir. Ortalama değerlere bakıldığında, 32 alt grubun 28'inde FA, 15'inde CS, 7'sibde TSO, 6'sında HBA en iyi değerleri vermiştir.

Tablo 3.34'de algoritmaların tüm durumlardan elde edilen toplam minimum ve ortalama konteyner sayıları ifade edilmiştir. Toplam değerlere bakıldığında CS'nin iterasyon sayısı 100 ve 500 olan ilk 4 durumda en iyi değerleri elde ettiği görülürken, iterasyon sayısı FA'nın hem minimum hem ortalama değerler için daha iyi sonuçlar elde ettiği görülmektedir. Ancak 9 durumun tamamı şeklinde düşünecek olursa 5'inde CS önde sayılacağından FA'ya göre üstün olduğu söylenebilir.

Tablo 3.34 Toplam Konteyner Sayıları

	FA Kon_min	CS Kon_min	TSO Kon_min	HBA Kon_min	FA Kon _ort	CS Kon _ort	TSO Kon _ort	HBA Kon _ort
Durum 1	916	**910**	916	914	997.8	**988.4**	997.3	994.5
Durum 2	909	**905**	909	908	987.3	**983.6**	987.6	985.5
Durum 3	906	**903**	906	904	984.8	**981.2**	984.9	984.1
Durum 4	905	**900**	906	905	982	**980.4**	986	984.2
Durum 5	**897**	900	904	902	**976.7**	977.1	982.5	980.7
Durum 6	904	**900**	905	904	979.3	**978.5**	981.9	980.6
Durum 7	**896**	899	905	902	**975.1**	978.1	983.1	981.1
Durum 8	**895**	896	900	901	**972.8**	975.1	979.7	978.4
Durum 9	**893**	899	903	899	**971.9**	975.2	981.7	979.3

Tablo 3.35'de EK1, EK10 ve EK19'da verilen 1.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 1.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir. Tabloda S süreyi, DO doluluk oranını, Kon_no konteyner sayısını ifade etmektedir.

Tablo 3.35 Durum 1 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	100	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	CS	CS	CS
	200	TSO	TSO	CS	CS	FA, CS	CS
2	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	100	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	CS	CS	FA	CS
3	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	HBA
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	CS	CS	CS, HBA	CS
4	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	TSO	TSO	FA	CS	FA, CS, TSO	CS, TSO
5	**50**	TSO	TSO	HBA	CS	TSO, HBA	TSO
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	CS
6	**50**	TSO	TSO	TSO	CS	HBA	CS
	100	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	HBA	CS	CS, HBA	CS
	200	TSO	TSO	CS	CS	CS	CS
7	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	CS	CS, TSO, HBA	CS, HBA
	200	TSO	TSO	CS	CS	CS	CS
8	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS, HBA
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS

Tablo 3.36'da EK2, EK11 ve EK20'da verilen 2.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 2.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.36 Durum 2 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	TSO	TSO	CS	HBA	FA, CS, TSO, HBA	CS
	100	FA	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS
	150	TSO	TSO	CS	CS	FA, CS, TSO	FA, CS
	200	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	HBA
2	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, HBA
	100	TSO	TSO	FA	CS	FA, CS, HBA	CS
	150	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	CS, HBA
	200	TSO	TSO	FA	CS	CS, HBA	CS
3	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	TSO	CS	CS	CS
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS, TSO
4	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	TSO	TSO	HBA	CS	FA, CS, HBA	CS
	100	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	CS, TSO, HBA
	150	TSO	TSO	CS	HBA	CS, TSO, HBA	HBA
	200	TSO	TSO	TSO	HBA	FA, CS, TSO, HBA	TSO, HBA
6	**50**	TSO	TSO	HBA	HBA	FA, CS, TSO, HBA	CS, HBA
	100	TSO	TSO	CS	CS	FA, CS, TSO	FA
	150	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	CS
7	**50**	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	FA, CS, HBA
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	CS	CS	CS, TSO, HBA	CS
8	**50**	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	HBA
	100	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	FA, HBA
	150	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA
	200	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS

Tablo 3.37'de EK3, EK12 ve EK21'da verilen 3.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 3.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.37 Durum 3 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	TSO	HBA	TSO	CS	FA, CS, TSO, HBA	FA, TSO
	100	TSO	TSO	FA	FA	CS, HBA	FA
	150	TSO	TSO	FA	CS	FA, CS, HBA	CS
	200	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS
2	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA, CS, HBA
	150	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	CS, HBA
	200	TSO	TSO	TSO	TSO	FA, CS, TSO, HBA	TSO
3	**50**	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	HBA	CS	TSO	CS
	150	TSO	TSO	CS	CS	CS	CS
	200	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA
4	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	TSO	TSO	HBA	CS	CS, TSO, HBA	CS, HBA
	100	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	FA, CS, HBA
	150	TSO	TSO	HBA	HBA	FA, CS, HBA	HBA
	200	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	CS
6	**50**	TSO	TSO	CS	HBA	FA, CS, TSO, HBA	CS
	100	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	CS
7	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA
	150	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	CS	CS	FA, CS, TSO	CS
8	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS
	150	TSO	TSO	FA	CS	FA, HBA	CS
	200	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS

Tablo 3.38'de EK4, EK13 ve EK22'da verilen 4.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 4.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.38 Durum 4 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	HBA	HBA	FA	FA	FA, CS, TSO, HBA	FA
	100	FA	HBA	FA	CS	FA, CS, TSO	TSO
	150	TSO	TSO	TSO	CS	FA, CS, TSO, HBA	FA, CS
	200	FA	FA	TSO	CS	TSO	CS
2	**50**	FA	FA	FA	CS	FA, CS, TSO, HBA	CS
	100	FA	FA	CS	CS	FA, CS, TSO, HBA	CS
	150	FA	FA	CS	CS	CS	CS
	200	FA	FA	CS	CS	CS, TSO	CS
3	**50**	TSO	FA	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	CS	FA	FA, CS, HBA	HBA
	150	FA	FA	CS	CS	CS	CS
	200	FA	FA	FA	FA	FA, CS, TSO, HBA	CS
4	**50**	FA	FA	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	FA	FA	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	FA	FA	CS	CS	FA, CS, HBA	HBA
	100	FA	FA	CS	FA	CS	CS, HBA
	150	FA	FA	FA	CS	CS, TSO, HBA	TSO, HBA
	200	TSO	FA	TSO	HBA	FA, CS, TSO, HBA	CS
6	**50**	TSO	FA	CS	CS	FA, CS, TSO, HBA	CS
	100	FA	FA	FA	FA	FA, CS, HBA	FA
	150	FA	FA	FA	FA	FA, CS, TSO, HBA	HBA
	200	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
7	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	FA	CS	FA, CS, TSO, HBA	CS
	150	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA, CS
	200	FA	FA	HBA	FA	FA, CS, HBA	FA
8	**50**	TSO	FA	CS	CS	FA, CS, TSO, HBA	CS
	100	TSO	FA	TSO	FA	FA, CS, TSO, HBA	CS
	150	TSO	FA	FA	FA	FA, CS, TSO, HBA	FA, CS
	200	FA	FA	CS	CS	FA, CS, TSO, HBA	CS

Tablo 3.39'da EK5, EK14 ve EK23'da verilen 5.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 5.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.39 Durum 5 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	TSO	FA	CS	CS	FA, CS, TSO, HBA	FA, CS, HBA
	100	FA	FA	CS	CS	FA, CS	CS
	150	TSO	FA	HBA	CS	FA, CS, TSO, HBA	CS
	200	FA	FA	FA	FA	FA	FA
2	**50**	FA	FA	FA	CS	FA, CS, TSO, HBA	FA, CS, HBA
	100	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
	150	FA	FA	FA	FA	FA	CS
	200	FA	FA	CS	FA	FA, CS, TSO, HBA	FA, CS
3	**50**	FA	FA	HBA	HBA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	CS	CS	FA, CS	FA
	150	FA	FA	FA	CS	FA, CS, HBA	FA, CS
	200	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
4	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	HBA	HBA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	HBA	HBA	CS	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	CS	CS	CS, HBA	CS, TSO, HBA
	150	FA	FA	TSO	FA	FA, CS, TSO	FA, CS
	200	HBA	FA	TSO	TSO	TSO	TSO
6	**50**	HBA	HBA	HBA	FA	FA, CS, TSO, HBA	FA
	100	HBA	HBA	FA	CS	FA, CS, TSO, HBA	CS
	150	HBA	TSO	CS	FA	CS	FA
	200	TSO	TSO	FA	FA	FA	FA
7	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA, CS
	150	CS	TSO	FA	FA	FA, CS, TSO, HBA	FA, HBA
	200	HBA	TSO	FA	FA	FA, CS, TSO, HBA	CS
8	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS, HBA
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	CS
	150	TSO	TSO	CS	FA	FA, HBA	FA, CS
	200	TSO	TSO	FA	FA	FA, HBA	FA

Tablo 3.40'de EK6, EK15 ve EK24'da verilen 6.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 6.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.40 Durum 6 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	HBA	TSO	FA	FA	FA, CS, TSO, HBA	FA, TSO, HBA
	100	TSO	TSO	FA	CS	FA, CS, TSO	FA, CS
	150	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS
	200	TSO	TSO	FA	CS	CS	CS
2	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	CS
	100	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA
	150	TSO	TSO	CS	HBA	FA, CS, TSO, HBA	HBA
	200	TSO	TSO	CS	FA	FA, CS, TSO, HBA	FA
3	**50**	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	CS	CS	CS	CS
	150	TSO	TSO	HBA	FA	CS, HBA	CS
	200	FA	TSO	CS	FA	FA, CS, TSO, HBA	CS
4	**50**	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	TSO	TSO	HBA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	FA	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	CS	CS	CS	CS
	150	TSO	TSO	TSO	HBA	FA, CS, HBA	CS
	200	TSO	FA	FA	TSO	FA, CS, TSO, HBA	CS, TSO
6	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	CS, HBA
	100	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA, HBA
	150	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA
	200	FA	FA	FA	HBA	FA, CS, TSO, HBA	HBA
7	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	CS
	100	FA	FA	CS	CS	FA, CS, TSO, HBA	FA, CS
	150	HBA	FA	FA	CS	FA, CS, TSO, HBA	FA, CS
	200	FA	FA	CS	TSO	FA, CS, TSO, HBA	TSO, HBA
8	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	CS
	100	FA	FA	CS	CS	FA, CS, TSO, HBA	CS
	150	FA	FA	CS	FA	FA, CS, TSO, HBA	FA, HBA
	200	FA	FA	CS	FA	FA, CS, TSO, HBA	FA

Tablo 3.41'de EK7, EK16 ve EK25'da verilen 7.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 7.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.41 Durum 7 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, HBA
	100	FA	FA	CS	FA	FA, CS, HBA	FA
	150	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
	200	HBA	HBA	TSO	FA	CS	CS
2	**50**	HBA	HBA	CS	CS	FA, CS, TSO, HBA	CS, HBA
	100	HBA	HBA	FA	CS	FA, CS, TSO, HBA	CS
	150	HBA	HBA	TSO	FA	FA, CS	FA, CS
	200	HBA	HBA	FA	FA	FA, CS, TSO, HBA	FA
3	**50**	HBA	HBA	TSO	CS	FA, CS, TSO, HBA	TSO
	100	HBA	HBA	CS	FA	FA, CS, HBA	FA
	150	HBA	HBA	FA	FA	FA, CS	FA, CS
	200	HBA	HBA	FA	FA	FA, CS, TSO, HBA	FA
4	**50**	HBA	HBA	CS	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	HBA	HBA	CS	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	HBA	HBA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	HBA	HBA	CS	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	HBA	HBA	HBA	HBA	FA, CS, HBA	FA, HBA
	100	HBA	HBA	FA	FA	FA, TSO	FA
	150	HBA	HBA	TSO	FA	FA, CS, TSO, HBA	FA
	200	FA	FA	HBA	HBA	HBA	HBA
6	**50**	TSO	FA	CS	CS	FA, CS, TSO, HBA	CS, HBA
	100	FA	FA	CS	CS	FA, CS, TSO, HBA	FA, CS
	150	FA	FA	FA	FA	FA	FA
	200	FA	FA	HBA	FA	FA, CS	FA
7	**50**	CS	CS	HBA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	CS	CS	FA	FA	FA, CS, TSO, HBA	FA
	150	FA	CS	CS	FA	FA, CS, TSO, HBA	FA, CS
	200	CS	CS	FA	FA	FA, CS, TSO	FA
8	**50**	CS	CS	TSO	CS	FA, CS, TSO, HBA	CS
	100	CS	CS	FA	FA	FA	FA
	150	CS	CS	FA	FA	FA, HBA	FA, CS
	200	CS	CS	FA	FA	CS	FA

Tablo 3.42'de EK8, EK17 ve EK26'da verilen 8.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 8.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.42 Durum 8 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	FA	FA	CS	CS	FA, CS, TSO, HBA	FA
	100	FA	FA	FA	FA	FA, CS, TSO, HBA	CS
	150	FA	TSO	FA	FA	FA, CS, TSO, HBA	FA
	200	TSO	TSO	FA	FA	FA	FA
2	**50**	TSO	TSO	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA
	150	TSO	TSO	FA	FA	FA, CS, TSO	FA
	200	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA, CS
3	**50**	TSO	HBA	CS	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	HBA	HBA	CS	FA	FA, CS, TSO	FA, CS
	150	TSO	TSO	FA	FA	FA, CS, HBA	FA
	200	TSO	FA	CS	FA	FA, CS	FA
4	**50**	FA	HBA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	TSO	FA	CS	FA	FA, CS, TSO, HBA	FA, CS, TSO
	150	TSO	FA	FA	FA	FA, CS, TSO	FA
	200	CS	CS	TSO	TSO	FA, CS, TSO, HBA	TSO
6	**50**	CS	CS	FA	CS	FA, CS, TSO, HBA	CS, TSO
	100	CS	CS	TSO	CS	FA, CS, TSO, HBA	FA, CS
	150	CS	CS	HBA	FA	FA, HBA	FA
	200	CS	CS	HBA	FA	FA, CS, TSO, HBA	FA
7	**50**	CS	CS	FA	FA	FA, CS, TSO, HBA	FA
	100	CS	CS	FA	FA	FA, CS, TSO, HBA	FA, CS, HBA
	150	CS	CS	FA	FA	FA, CS, TSO, HBA	FA
	200	FA	CS	FA	FA	FA, CS, TSO, HBA	FA
8	**50**	FA	FA	FA	CS	FA, CS, TSO, HBA	FA, CS
	100	FA	FA	FA	FA	FA, CS	FA
	150	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA, HBA
	200	FA	FA	FA	FA	FA, CS	FA, CS

Tablo 3.43'de EK9, EK18 ve EK27'da verilen 9.durum çıktıları çözüm süreleri, doluluk oranları, konteyner sayıları yan yana gelecek şekilde birleştirilmiş, algoritmaların 9.durumdaki iterasyon ve popülasyon değerlerine göre elde ettiği sonuçlar ifade edilmiştir. Böylece eklerde yer alan 27 tablonun özetlenmesi ve kolayca kıyaslanması hedeflenmiştir.

Tablo 3.43 Durum 9 için En İyi Değerler

Sınıf	n	S_min	S_ort	DO_min	DO_ort	Kon_no_min	Kon_no_ort
1	**50**	HBA	HBA	CS	CS	FA, CS, TSO, HBA	FA, CS
	100	HBA	FA	FA	FA	FA, CS, HBA	FA, CS
	150	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA, CS
	200	FA	FA	CS	FA	FA, CS	FA
2	**50**	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, HBA
	100	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
	150	FA	FA	FA	FA	FA, CS	FA
	200	FA	FA	CS	FA	FA, CS, TSO, HBA	CS
3	**50**	FA	FA	FA	CS	FA, CS, TSO, HBA	FA
	100	FA	FA	FA	FA	FA, CS, HBA	FA, CS
	150	FA	FA	FA	FA	FA, CS	FA
	200	FA	FA	FA	FA	FA, HBA	FA
4	**50**	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	150	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	200	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS, TSO, HBA
5	**50**	FA	FA	FA	CS	FA, CS, TSO, HBA	FA, CS, TSO, HBA
	100	FA	FA	CS	FA	FA	FA
	150	FA	FA	HBA	FA	FA, CS, TSO, HBA	FA, CS
	200	TSO	FA	TSO	TSO	TSO, HBA	TSO
6	**50**	FA	CS	FA	FA	FA, CS, TSO, HBA	CS, TSO
	100	FA	CS	TSO	FA	TSO	FA
	150	CS	CS	CS	FA	FA	FA
	200	CS	CS	FA	FA	FA, HBA	FA
7	**50**	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
	100	TSO	TSO	FA	FA	FA	FA
	150	TSO	FA	FA	FA	FA, CS, TSO, HBA	FA
	200	TSO	TSO	FA	FA	FA, CS, TSO, HBA	FA
8	**50**	FA	FA	FA	FA	FA, CS, TSO, HBA	FA, CS
	100	FA	FA	FA	FA	FA, CS, TSO, HBA	FA
	150	FA	FA	FA	FA	FA, CS, HBA	CS
	200	FA	FA	FA	FA	FA	FA

Algoritmaların üstün ve zayıf olduğu tarafları için özet bir bilgi şu şekildedir:

FA

- *Avantajları:* Basit yapısı, hızlı keşif yeteneği, esnek parametre ayarları.
- *Dezavantajları:* Daha az sömürü yeteneği, yakınsama hızında yavaşlama olabilir.

CS

- *Avantajları:* Yüksek keşif ve sömürü yeteneği, etkili çözüm alanı keşfi.
- *Dezavantajları:* Daha karmaşık parametre ayarları gerekebilir, hesaplama karmaşıklığı.

TSO

- *Avantajları:* Hızlı yakınsama, küresel keşif ve yerel sömürü yetenekleri.
- *Dezavantajları:* Parametre ayarlarına duyarlılık, yakınsama hızında yavaşlama.

HBA

- *Avantajları:* Yüksek keşif ve sömürü yeteneği, çeşitli problem türlerine uyum sağlama.
- *Dezavantajları:* Parametre ayarlarına duyarlılık, hesaplama karmaşıklığı.

SONUÇ

Kutuların konteynerlere paketlenmesi, paketleme ve nakliye endüstrilerinde yaygın bir işlemdir. Bazı paketleme görevlerinin hâlâ manuel olarak gerçekleştirilmesi, özellikle farklı boyutta ve çok sayıda kutu bulunan durumlarda zaman alıcı ve zorlayıcı olabilmektedir. Bu durum bazen, alanın iyi kullanılamamasına ve ek konteynerlere ihtiyaç duyulmasına sebep olmakta, işletme verimliliğini ve maliyetleri doğrudan etkilemektedir. 3-D BPP çözümü için literatürde pek çok algoritma kullanılmaktadır. Problemin çözüm için ilk etapta kullanılan kesin çözüm yöntemleri, problemin kombinatoryal yapısı ve karmaşıklığı nedeniyle, pek çok durum için yetersiz kalmaktadır. Bu zorluklar göz önüne alındığında, sezgisel veya metasezgisel yöntemler daha yaygın olarak tercih edilmeye başlanmıştır. Bu yöntemler, kesin olmasa da optimal çözümlere yakın sonuçlar elde edebilme yeteneğine sahiptir.

Bu tez çalışmasında 3-D BPP'nin çözümünde farklı metasezgisel yöntemlerin performansları araştırılmıştır. Bu kapsamda; FA, CS, TSO ve HBA olmak üzere dört farklı metasezgisel yöntem ile çözümler elde edilmiştir. Çalışmada kullanılan metasezgisel yöntemlerin performansları, dokuz farklı durum altında test edilmiştir. Her bir durum, farklı iterasyon (100-500-1000) ve popülasyon (n-2n-4n) sayılarına bağlı parametre kombinasyonlarına dayanmaktadır. Bu testlerde her durum için; çözüm sürelerinin minimum ve ortalama değerleri, doluluk oranlarının minimum ve ortalama değerleri, konteyner sayılarının minimum ve ortalama değerleri hesaplanmıştır. Her durum için sonuç değerleri 30 tekrarlı çözümlerle elde edilmiştir. Bu hesaplamalarda, 8 sınıf 32 alt gruptan oluşan 320 örneğin bulunduğu Martello vd., (2000)'nin veri setinden yararlanılmaktadır. Çalışmada kullanılan parametre değerleri, algoritmaların geliştiricileri tarafından kabul edilen varsayılan değerlerdir. Çözümde kullanılan yerleştirme stratejisi köşe noktalar algoritmasıdır. Tüm testler, 64,0 GB RAM ve Intel Core i9 10850K CPU 3.60GHz işlemciye sahip 64 bit Windows işletim sistemli bir bilgisayarda gerçekleştirilmiş olup algoritmalar Python dilinde kodlanmıştır.

İlk olarak 9 farklı durum için çözüm süreleri, doluluk oranları ve konteyner sayıları elde edilmiş, sonuçlar EK1-EK27'de paylaşılmıştır. Daha sonra 8 sınıfın 32 alt grubunun da yer aldığı bu tabloların toplam değerleri bulunmuş, 9 durum için kıyaslaması ve istatistik tesleri yapılmıştır. Son olarak, en iyi çözüm değerlerini elde eden algoritmalar, sütunda; çözüm süreleri, doluluk oranları ve konteyner sayıları olacak biçimde özetlenmiştir.

Çözüm süreleri için elde edilen bulgulara göre; 100 iterasyon ve n-2n-4n popülasyonlarının ele alındığı ilk 3 durumda TSO en iyi minimum ve ortalama çözüm sürelerini elde etmiştir. 500 iterasyon n-2n-4n popülasyonlarının ele alındığı sonraki 3 durumda TSO üstünlüğünü kaybetmiş n popülasyon için FA öne çıkmış, 2n popülasyon için FA öncülüğünü sürdürse de TSO ve HBA da başarılı değerler elde etmiştir. 4n popülasyon için TSO yeniden öne geçmiş arkasında FA başarılı çözüm süreleri elde etmişti. 1000 iterasyon ve n-2n-4n popülasyonlarının ele alındığı son 3 durumda ise, iterasyon sayısındaki artışla HBA n popülasyon durumunda öne geçmiştir. Ancak 2n-4n popülasyonlarında FA çözümü tamamen domine etmiştir. Algoritmalar aynı zamanda ortalama çözüm değerleri için de yakın sonuçlar vermişlerdir. Bu da tutarlı biçimde düşük çözüm sürelerine ulaşabildiklerini göstermektedir. Toplam çözüm süreleri incelendiğinde; ilk 3 durum için TSO, son durumlarda ise FA'nın başarısı görülmektedir, yalnızca 7.durumda HBA diğerlerinden düşük sonuç değerleri elde etmiştir.

Sonuçlar algoritmaların farklı iterasyon durumlarından etkilendiklerini göstermektedir. Bunun pek çok farklı sebebi olabilir. Algoritmaların çözüm değerlerini elde ederken kullandıkları formüllerin karmaşıklık düzeyleri, çözüme yakınsama hızlarını etkileyebilmektedir. Bir diğer sebep algoritmaların keşif ve sömürü aşamaları arasındaki dengedir. Keşif, çözüm alanında daha fazla keşif yaparak daha geniş bir arama yapmayı ifade ederken, sömürü, bulunan iyi çözümleri daha derinlemesine araştırmayı ifade eder. Daha fazla iterasyon, keşif ve sömürü arasındaki dengeyi ayarlamaya yardımcı olabilmektedir. Düşük iterasyonlarda TSO daha iyi çözüm süresi değerleri elde ederken, yüksek iterasyonlarda FA öne geçmiştir. FA aynı zamanda bu algoritmalar arasında, çözümü global ve yerel arama olmak üzere 2 aşamayla değil tek bir aşamayla gerçekleştiren tek algoritmadır. Hesaplama karmaşıklığının az olması, çok daha kısa sürede çözüm değerleri elde etmiş olmasına etken olan bir diğer sebep olabilir.

Doluluk oranları için elde edilen bulgulara göre; 100 iterasyon ve n-2n-4n popülasyonlarının ele alındığı ilk 3 durumda CS en iyi doluluk oranlarını elde etmiştir. 500 iterasyon n-2n-4n popülasyonlarının ele alındığı sonraki 3 durumda n ve 4n popülasyonları için CS önde olmayı sürdürmüş, yalnızca 2n popülasyonunda FA'nın gerisinde kalmıştır. 1000 iterasyon ve n-2n-4n popülasyonlarının ele alındığı son 3 durumda ise, FA tamamıyla öne geçmiş ve hem minimum hem ortalama değerlerde en iyi doluluk oranlarına ulaşmıştır. Toplam doluluk oranları incelendiğinde; ilk 6 durumda CS önde iken son 3 durumda FA'nın öne geçtiği görülmektedir.

Sonuçlar algoritmaların farklı iterasyon durumlarının yanı sıra, farklı popülasyon durumlarından da etkilendiklerini göstermektedir. Arama ajanı olarak da bilinen popülasyon büyüklüğündeki artış, arama uzayında daha geniş bir alanın keşfedilmesini sağlar. Zira büyük bir popülasyon, daha fazla çözüm adayını içerir ve bu da daha iyi çözümlerin bulunma olasılığını artırır. Bu nedenle popülasyon büyüklüğünün artışında algoritmaların daha iyi değerler vermesi beklenen bir durumdur. Ancak yüksek popülasyonlarda başarı elde eden bir algoritmanın düşük popülasyonlarda da iyi değerler elde edeceği varsayılamaz. Burada anlaşılan bir diğer husus, çözüm sürelerinde geride kalan algoritmaların doluluk oranların da öne geçebildiğidir. Bu da hızlı çözümlerin her zaman en iyi değerleri göstermediğinin ifadesidir.

Konteyner sayıları için elde edilen bulgulara göre; 100 iterasyon ve n-2n-4n popülasyonlarının ele alındığı ilk 3 durumda, CS ortalama doluluk oranları için en iyi değerleri vermiştir. 500 iterasyon n-2n-4n popülasyonlarının ele alındığı sonraki 3 durumda, n ve 4n popülasyonları için CS'nin ortalama doluluk oranlarındaki başarısını sürdürdüğü gözlenmektedir. Ancak 2n popülasyonunda FA daha iyi çözüm değerleri elde etmiştir. 1000 iterasyon ve n-2n-4n popülasyonlarının ele alındığı son 3 durumda ise, FA tamamıyla öne geçmiştir. Toplam konteyner sayıları incelendiğinde; ilk 5 durum ve 7.durumda CS önde iken, son 3 durumda FA'nın öne geçtiği görülmektedir.

Sonuçlar algoritmaların neredeyse tamamının, pek çok durumda, minimum konteyner sayılarını elde edebildiğini göstermektedir. Bu nedenle bu kısımda minimum değerlerden ziyade ortalama değerler üzerinde durulmuştur. Bunun bir başka sebebi, 30 tekrarlı çözümlerde algoritmaların minimum birkaç değeri yakalamasının aslında genel bir başarı göstergesi olmamasıdır. Çalışmada algoritmaların performansından beklenen durum istikrarlı biçimde en iyi değerleri bulmayı sürdürmelidir. Aynı zamanda birden fazla dizilimin aynı konteyner sayısını verebilecek olması algoritma durgunluğu adı verilen bir soruna da yol açabilmektedir.

Elde edilen sonuçlara dayalı olarak, yöntemler arasında anlamlı farklılıkların varlığını değerlendirmek için Friedman testi kullanılmıştır. Bununla birlikte, hangi yöntemler arasında bu anlamlı farklılıkların olduğunu belirlemek adına da Wilcoxon testi kullanılmıştır. Bu analizler, yöntemlerin performanslarını karşılaştırırken istatistiksel olarak sağlam sonuçlar elde edilmesine yardımcı olmuştur. Yapılan analiz, yöntemler arasında anlamlı farklılıkların varlığını vurgulamaktadır. Bu anlamlı farklılıkların detaylı bir şekilde ortaya çıkarılması, yöntemlerin performanslarını değerlendirirken daha derin bir perspektif sunmaktadır. Elde edilen bu sonuçların, tez çalışmasının genel katkısını artıracağı düşünülmektedir.

Tüm sonuçlar toparlanacak olursa kullanıcının hangi algoritmaya ağırlık vereceği tamamen elde etmek istediği sonuçlara bağlıdır. Eğer en hızlı çözümlerin elde edilmesi hedeflenirse, düşük iterasyonlar için TSO yüksek iterasyonlar için FA tercih edilebilir. Eğer en iyi doluluk oranlarının elde edilmesi hedeflenirse düşük iterasyonlar için CS yüksek iterasyonlar için FA tercih edilebilir. Eğer en iyi konteyner sayılarının elde edilmesi hedeflenirse düşük iterasyonlar için CS yüksek iterasyonlar için FA tercih edilebilir.

Ancak daha önce de belirtildiği gibi düşük iterasyon ve popülasyon durumunda dahi başarılı olan bir algoritmanın, sınırlı sayıda çözüm adayıyla bile etkili bir şekilde arama yapabilme yeteneği olduğu düşünülmektedir.

Bu tez, 3-D BPP çözümünde metasezgisel algoritmaların değerlendirilmesiyle ilgili kapsamlı bir çalışma sunmaktadır. Elde edilen sonuçlar, lojistik süreçlerde paketleme optimizasyonunun daha etkin ve verimli hale getirilmesine katkı sağlayabilir. Ayrıca, gelecekteki araştırmalar için bir temel oluşturarak, daha gelişmiş algoritma ve optimizasyon tekniklerinin kullanılmasını teşvik edebilir.

KAYNAKÇA

Abdel-Basset, M., Manogaran, G., Abdel-Fatah, L. ve Mirjalili, S. (2018). "An improved nature inspired metaheuristic algorithm for 1-D bin packing problems". *Personal and Ubiquitous Computing*, *22*, 1117-1132.

Abdul-Minaam, D. S., Al-Mutairi, W. M. E. S., Awad, M. A. ve El-Ashmawi, W. H. (2020). "An adaptive fitness-dependent optimizer for the one-dimensional bin packing problem". *IEEE Access*, *8*, 97959-97974.

Ali, S., Ramos, A. G., Carravilla, M. A. ve Oliveira, J. F. (2022). "On-line three-dimensional packing problems: A review of off-line and on-line solution approaches". *Computers & Industrial Engineering*, *168*, 108122.

Allen, S. D., Burke, E. K. ve Kendall, G. (2011). "A hybrid placement strategy for the three-dimensional strip packing problem". *European Journal of Operational Research*, *209*(3): 219-227.

Ashraf, H., Elkholy, M. M., Abdellatif, S. O. ve El-Fergany, A. A. (2022). "Synergy of neuro-fuzzy controller and tuna swarm algorithm for maximizing the overall efficiency of PEM fuel cells stack including dynamic performance". *Energy Conversion and Management: X*, *16*, 100301.

Azad, S.K., Azad, S.K. (2011). "Optimum design of structures using an improved firefly algorithm". International Journal of Optimization in Civil Engineering. 1(2): 327–340.

Baldi, M. M., Crainic, T. G., Perboli, G. ve Tadei, R. (2012). The generalized bin packing problem. "*Transportation Research Part E: Logistics and Transportation Review", 48*(6): 1205-1220.

Banati, H. ve Bajaj, M. (2011). "Fire fly based feature selection approach". *International Journal of Computer Science Issues (IJCSI)*, *8*(4): 473.

Baranovskii, E. P. (1971). "Packings, coverings, partitionings, and certain other distributions in spaces of constant curvature". In *Progress in Mathematics: Algebra and Geometry*. Boston, MA: Springer US, 209-253.

Basu, B. ve Mahanti, G. (2011). "Fire fly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna". *Progress In Electromagnetics Research B*, *32*, 169-190.

Bellman, R. (1966). "Dynamic programming". *Science*, *153*(3731): 34-37.

Blaisdell, B. E. ve Solomon, H. (1970). "On random sequential packing in the plane and a conjecture of Palasti". *Journal of Applied Probability*, *7*(3): 667-698.

Bortfeldt, A. ve Wäscher, G. (2013). Constraints in container loading–A state-of-the-art review. "*European Journal of Operational Research"*, *229*(1): 1-20.

Boschetti, M. A. (2004). "New lower bounds for the three-dimensional finite bin packing problem". *Discrete Applied Mathematics, 140*(1-3): 241-258.

Chandra, K. ve Singh, S. (2014). "Firefly algorithm to solve two dimensional bin packing problem". *International Journal of Computer Science and Information Technologies, 5*(4): 5368-5373.

Chandu, D. P. (2014). "A parallel genetic algorithm for three dimensional bin packing with heterogeneous bins". *arXiv: 1411.4565*.

Chatterjee, A., Mahanti, G. ve Chatterjee, A. (2012). "Design of a fully digital controlled reconfigurable switched beam concentric ring array antenna using firefly and particle swarm optimization algorithm". *Progress In Electromagnetics Research B, 36,* 113-131.

Chen, C. S., Lee, S. M., ve Shen, Q. S. (1995). "An analytical model for the container loading problem". *European Journal of operational research, 80*(1): 68-76.

Chen, R. F., Luo, H., Huang, K. C., Nguyen, T. T. ve Pan, J. S. (2022). "An improved honey badger algorithm for electric vehicle charge orderly planning". *Journal of Network Intelligence, 7*(2): 332-346.

Cid-Garcia, N. M. ve Rios-Solis, Y. A. (2020). "Positions and covering: A two-stage methodology to obtain optimal solutions for the 2d-bin packing problem". *Plos one, 15*(4): e0229358.

Civicioglu, P. ve Besdok, E. (2013). "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms". *Artificial intelligence review, 39,* 315-346.

Coffman Jr, E. G., Csirik, J., Johnson, D. S. ve Woeginger, G. J. (2004). "An introduction to bin packing". *Bibliographie.*

Crainic, T. G., Perboli, G. ve Tadei, R. (2009). "TS2PACK: A two-level tabu search for the three-dimensional bin packing problem". *European Journal of Operational Research, 195*(3): 744-760.

Delorme, M., Iori, M. ve Martello, S. (2016). "Bin packing and cutting stock problems: Mathematical models and exact algorithms". *European Journal of Operational Research, 255*(1): 1-20.

Dereli, T., Baykasoglu, A., Das, G. S. ve Göçken, T. (2004). "Üç Boyutlu Paketleme Problemlerıne Analıtık Yaklasımlar". *YA/EM*, 15-18.

Dorigo, M., Birattari, M. ve Stutzle, T. (2006). "Ant colony optimization". *IEEE computational intelligence magazine, 1*(4): 28-39.

Dyckhoff, H. (1990). "A typology of cutting and packing problems". *European journal of operational research, 44*(2): 145-159.

Eberhart, R. ve Kennedy, J. (1995, October). "A new optimizer using particle swarm theory. In *MHS'95". Proceedings of the sixth international symposium on micro machine and human science,* 39-43.

Elsayed, W. (2022). "Enhanced Honey Badger Algorithm With Lévy Flights for Solving Economic Load Dispatch Problems". *Engineering Research Journal-Faculty of Engineering (Shoubra), 51*(4): 19-29.

Erol, O.K. ve Eksin, I. (2006). "A new optimization method: big bang–big crunch". *Advances in engineering software, 37*(2): 106-111.

Fadhil, R. A. ve Hassan, Z. A. H. (2023). "A Hybrid Honey-Badger Intelligence Algorithm with Nelder-Mead Method and Its Application for Reliability Optimization". *International Journal of Intelligent Systems and Applications in Engineering, 11*(4s): 136-145.

Faroe, O., Pisinger, D. ve Zachariasen, M. (2003). "Guided local search for the three-dimensional bin-packing problem". *Informs journal on computing, 15*(3): 267-283.

Fatih Tasgetiren, M., Liang, Y. C., Sevkli, M. ve Gencyilmaz, G. (2006). "Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem". *International Journal of Production Research, 44*(22): 4737-4754.

Festa, P. (2014). "A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems". In *2014 16th International Conference on Transparent Optical Networks,* 1-20.

Fister, I., Yang, X. S., Fister, D. ve Fister, I. (2014). "Firefly algorithm: a brief review of the expanding literatüre". *Cuckoo Search and Firefly Algorithm: Theory and Applications,* 347-360.

Formato, R. A. (2007). "Central force optimization: a new metaheuristic with applications in applied electromagnetics". Prog Electromagn Res 77: 425–491.

Fuellerer, G., Doerner, K. F., Hartl, R. F. ve Iori, M. (2010). "Metaheuristics for vehicle routing problems with three-dimensional loading constraints". *European Journal of Operational Research, 201*(3): 751-759.

Gandomi, A. H., Yang, X. S. ve Alavi, A. H. (2011). "Mixed variable structural optimization using firefly algorithm". *Computers & Structures, 89*(23-24): 2325-2336.

Gao, W., Liu, S. ve Huang, L. (2012). "A global best artificial bee colony algorithm for global optimization". *Journal of Computational and Applied Mathematics, 236*(11): 2741-2753.

Geem, Z. W., Kim, J. H. ve Loganathan, G. V. (2001). "A new heuristic optimization algorithm: harmony search". *simulation, 76*(2): 60-68.

Gezici, H. ve Livatyalı, H. (2022). "İki boyutlu kutu paketleme probleminin çözümü için hibrit çiçek tozlaşma algoritması yaklaşımı". *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 37*(3): 1523-1534.

Gezici, H. ve Livatyalı, H. (2023). "Optimization of one-dimensional bin packing problem using a hybrid flower pollination algorithm". *Sigma, 41*(3), 545-564.

Gherboudj, A. (2019). "African buffalo optimization for one dimensional bin packing problem". *International Journal of Swarm Intelligence Research (IJSIR), 10*(4):38-52.

Gilmore, P.C. ve Gomory, R. E. (1961). "A linear programming approach to the cutting-stock problem". *Operations research, 9*(6): 849-859.

Glover, F. (1989). Tabu search—part I". *ORSA Journal on computing, 1*(3): 190-206.

Gonçalves, J. F. ve Resende, M. G. (2013). "A biased random key genetic algorithm for 2D and 3D bin packing problems". *International Journal of Production Economics, 145*(2): 500-510.

Graham, R. L. (1971). "Bounds on multiprocessing anomalies and related packing algorithms". *Spring joint computer conference*. Mayıs 16-18 1972, 205-217.

Grandcolas, S. ve Pain-Barre, C. (2022). "A hybrid metaheuristic for the two-dimensional strip packing problem". *Annals of Operations Research, 309*(1): 79-102.

Guo, B., Zhang, Y., Hu, J., Li, J., Wu, F., Peng, Q. ve Zhang, Q. (2022). "Two-Dimensional Irregular Packing Problems: A Review". *Frontiers in Mechanical Engineering*, 79.

Hadj Salem, K., ve Kieffer, Y. (2020,). "New symmetry-less ilp formulation for the classical one dimensional bin-packing problem". In *Computing and Combinatorics: 26th International Conference, Cham: Springer International Publishing.* 29–31 August 2020, Atlanta, 423-434.

Haouari, M. ve Serairi, M. (2011). "Relaxations and exact solution of the variable sized bin packing problem". *Computational Optimization and Applications, 48*, 345-368.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. ve Al-Atabany, W. (2022). "Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems". *Mathematics and Computers in Simulation*, *192*, 84-110.

He, C., Zhang, Y. B., Wu, J. W. ve Chang, C. (2009, December). "Research of three-dimensional container-packing problems based on discrete particle swarm optimization algorithm". In 2009 International Conference on Test and Measurement 2, 425-428.

He, S., Wu, Q. H. ve Saunders, J. R. (2009). "Group search optimizer: an optimization algorithm inspired by animal searching behavior". *IEEE transactions on evolutionary computation*, *13*(5): 973-990.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. ve Chen, H. (2019). "Harris hawks optimization: Algorithm and applications". *Future generation computer systems*, *97*, 849-872.

Horng, M. H., Lee, Y. X., Lee, M. C. ve Liou, R. J. (2012). "Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis". *Theory and new applications of swarm intelligence*, *4*(7): 115-132.

Iori, M., De Lima, V. L., Martello, S., Miyazawa, F. K. ve Monaci, M. (2021). "Exact solution techniques for two-dimensional cutting and packing". *European Journal of Operational Research*, *289*(2): 399-415.

Jiang, J. ve Cao, L. (2012). "A hybrid simulated annealing algorithm for three-dimensional multi-bin packing problems". *2012 international conference on systems and informatics*, 1078-1082.

Jin, Z., Ohno, K. ve Du, J. (2004). "An efficient approach for the three-dimensional container packing problem with practical constraints". Asia-Pacific Journal of Operational Research, 21(03): 279-295.

Johnson, D. S. (1973). *Near-optimal bin packing algorithms*. Doctoral dissertation. Massachusetts Institute of Technology, ABD.

Johnson, D. S. (1974). "Fast algorithms for bin packing". *Journal of Computer and System Sciences*, *8*(3): 272-314.

Kang, K., Moon, I. ve Wang, H. (2012). "A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem". Applied Mathematics and Computation, 219(3): 1287-1299.

Kantorovich, L. V. (1960). "Mathematical methods of organizing and planning production". *Management science*, *6*(4): 366-422.

Kaveh, A. ve Bakhshpoori, T. (2013). "Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights". *The Structural Design of Tall and Special Buildings, 22*(13): 1023-1036.

Kaveh, A. ve Talatahari, S. (2010). "A novel heuristic optimization method: charged system search". *Acta mechanica, 213*(3-4): 267-289.

Koza, J. R. (1994). "Genetic programming as a means for programming computers by natural selection". *Statistics and computing, 4*, 87-112.

Kucukyilmaz, T. ve Kiziloz, H. E. (2018). "Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem". *Computers & Industrial Engineering, 125*, 157-170.

Kumar, A. ve Chakarverty, S. (2011). "Design optimization for reliable embedded system using Cuckoo Search". In *2011 3rd international conference on electronics computer technology*, 1, 264-268.

Kumar, C. ve Mary, D. M. (2022). "A novel chaotic-driven Tuna Swarm Optimizer with Newton-Raphson method for parameter identification of three-diode equivalent circuit model of solar photovoltaic cells/modüles". *Optik, 264*, 169379.

Laabadi, S., Naimi, M., El Amri, H. ve Achchab, B. (2020). "A binary crow search algorithm for solving two-dimensional bin packing problem with fixed orientation". *Procedia Computer Science, 167*, 809-818.

Lawler, E. L. ve Wood, D. E. (1966). "Branch-and-bound methods: A survey". *Operations research, 14*(4): 699-719.

Layeb, A. ve Boussalia, S. R. (2012). "A novel quantum inspired cuckoo search algorithm for bin packing problem". *International Journal of Information Technology and Computer Science, 4*(5): 58-67.

Leon, P., Cueva, R., Tupia, M. ve Paiva Dias, G. (2019). "A Taboo-search Algorithm for 3D-binpacking Problem in Containers". In *New Knowledge in Information Systems and Technologies, 1*, 229-240.

Li, X. ve Zhang, K. (2015). "A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers". *Computers & Industrial Engineering, 90*, 305-313.

Liang, J. J., Pan, Q. K., Tiejun, C. ve Wang, L. (2011). "Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer". *The International Journal of Advanced Manufacturing Technology, 55*, 755-762.

Lim, A., Rodrigues, B. ve Yang, Y. (2005). "3-D container packing heuristics". *Applied Intelligence, 22*, 125-134.

Lin, C.C. ve Yu, C. S. (2006). "A heuristic algorithm for the three-dimensional container packing problem with zero unloading cost constraint". In 2006 IEEE International Conference on Systems, Man and Cybernetics 6, 4637-4642.

Liu, B., Wang, L., Qian, B. ve Jin, Y. (2008). "Hybrid particle swarm optimization for stochastic flow shop scheduling with no-wait constraint". *IFAC Proceedings Volumes*, *41*(2): 15855-15860.

Lodi, A., Martello, S. ve Vigo, D. (2002). "Heuristic algorithms for the three-dimensional bin packing problem". *European Journal of Operational Research, 141*(2): 410-420.

Loh, K. H. (2006). "*Weight annealing heuristics for solving bin packing and other combinatorial optimization problems: Concepts, algorithms, and computational results"*. University of Maryland, College Park, ABD.

Long, H., He, Y., He, Y., Song, C., Gao, Q. ve Tan, H. (2022). "Application of Improved Honey Badger Algorithm in Multi-objective Reactive Power Optimization". *IAENG International Journal of Applied Mathematics*, *52*(4).

Louzazni, M., Khouya, A., Amechnoue, K., Gandelli, A., Mussetta, M. ve Crăciunescu, A. (2018). "Metaheuristic algorithm for photovoltaic parameters: comparative study and prediction with a firefly algorithm". *Applied Sciences*, *8*(3), 339.

Mareli, M. ve Twala, B. (2018). "An adaptive Cuckoo search algorithm for optimisation". *Applied computing and informatics*, *14*(2): 107-115.

Martello, S. ve Vigo, D. (1998). "Exact solution of the two-dimensional finite bin packing problem". *Management science*, *44*(3): 388-399.

Martello, S., Pisinger, D. ve Vigo, D. (2000). The three-dimensional bin packing problem. *Operations research, 48*(2): 256-267.

Meir, A. ve Moser, L. (1968). "On packing of squares and cubes". *Journal of combinatorial theory*, *5*(2): 126-134.

Melega, G. M., de Araujo, S. A., Jans, R. ve Morabito, R. (2022). "Formulations and exact solution approaches for a coupled bin-packing and lot-sizing problem with sequence-dependent setups". *Flexible Services and Manufacturing Journal*, 1-37.

Mirjalili, S. ve Lewis, A. (2016). "The whale optimization algorithm". *Advances in engineering software, 95*, 51-67.

Moon, J. W. ve Moser, L. (1967). "Some packing and covering theorems". In *Colloquium Mathematicae*, 17(1): 103-110.

Moura, A., Pinto, T., Alves, C. ve Valério de Carvalho, J. (2023). "A Matheuristic Approach to the Integration of Three-Dimensional Bin Packing Problem and Vehicle Routing Problem with Simultaneous Delivery and Pickup". *Mathematics, 11*(3), 713.

Munien, C. ve Ezugwu, A. E. (2021). "Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications". *Journal of Intelligent Systems, 30*(1): 636-663.

Munien, C., Mahabeer, S., Dzitiro, E., Singh, S., Zungu, S. ve Ezugwu, A. E. S. (2020). "Metaheuristic approaches for one-dimensional bin packing problem: A comparative performance study", *8*, 227438-227465.

Naik, M. K. ve Panda, R. (2016). "A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition". *Applied Soft Computing, 38*, 661-675.

Nikhil, M., Ruchitha, B. N., Gotur, P., Ravinand, M. P. ve Reddy, G. H. (2022). Frequency Control of Renewables/Biodiesel/Biogas/Battery Energy Storage based Isolated Microgrid using the Honey Badger Algorithm. In *2022 4th International Conference on Energy, Power and Environment (ICEPE)*, 1-6.

Ozcan, S. O., Dokeroglu, T., Cosar, A. ve Yazici, A. (2016). "A novel grouping genetic algorithm for the one-dimensional bin packing problem on gpu". In *Computer and Information Sciences: 31st International Symposium, ISCIS 2016, Kraków, Springer International Publishing.* 27–28 October 2016, Poland, 31(52-60).

Pan, W. T. (2012). "A new fruit fly optimization algorithm: taking the financial distress model as an example". *Knowledge-Based Systems, 26*, 69-74.

Perdana, A. (2017). "Analisis Komparasi Genetic Algorithm dan Firefly Algorithm pada Permasalahan Bin Packing Problem". *Sinkron: jurnal dan penelitian teknik informatika, 1*(2).

Pisinger, D. ve Sigurd, M. (2007). "Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem". *INFORMS Journal on Computing, 19*(1): 36-51.

Prituja, A. V. (2020). State-of-the-art review on 3D bin packing problems. Nanyang Technological University, Singapore.

Qian, B., Wang, L., Hu, R., Wang, W. L., Huang, D. X. ve Wang, X. (2008). "A hybrid differential evolution method for permutation flow-shop scheduling". *The International Journal of Advanced Manufacturing Technology, 38*, 757-777.

Raidl, G. R. ve Kodydek, G. (1998). "Genetic algorithms for the multiple container packing problem". In *International Conference on Parallel Problem Solving from Nature*, 875-884.

Rajesh Kanna, S. K. ve Udaiyakumar, K. C. (2017). "A complete framework for multi-constrained 3D bin packing optimization using firefly algorithm". *International Journal of Pure and Applied Mathematics, 114*(6): 267-282.

Rakotonirainy, R. G. ve van Vuuren, J. H. (2020). "Improved metaheuristics for the two-dimensional strip packing problem". *Applied Soft Computing, 92,* 106268.

Rao, R. V., Savsani, V. J. ve Vakharia, D. P. (2012). "Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems". *Information sciences, 183*(1): 1-15.

Rashedi, E., Nezamabadi-Pour, H. ve Saryazdi, S. (2009). "GSA: a gravitational search algorithm". *Information sciences, 179*(13): 2232-2248.

Sayadi, M., Ramezanian, R., ve Ghaffari-Nasab, N. (2010). "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems". *International Journal of Industrial Engineering Computations, 1*(1): 1-10.

Sensarma, D. ve Sarma, S. S. (2014). "A novel graph based algorithm for one dimensional bin packing problem". *Journal of Global Research in Computer Science, 5*(8): 1-4.

Senthilnath, J., Omkar, S. N. ve Mani, V. (2011). "Clustering using firefly algorithm: performance study". *Swarm and Evolutionary Computation, 1*(3): 164-171.

Silva, E. F. ve Wauters, T. (2019). "Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems". *Computers & Operations Research, 109,* 12-27.

Silveira, M. E., Vieira, S. M. ve Da Costa Sousa, J. M. (2013). "An ACO algorithm for the 3D bin packing problem in the steel industry". In *Recent Trends in Applied Artificial Intelligence: 26th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013, Amsterdam, The Netherlands, June 17-21, 2013,* 535-544.

Simon, D. (2008). "Biogeography-based optimization". *IEEE transactions on evolutionary computation, 12*(6): 702-713.

Soak, S. M., ve Lee, S. W. (2012). "A memetic algorithm for the quadratic multiple container packing problem". Applied Intelligence, 36, 119-135.

Storn, R. ve Price, K. (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". *Journal of global optimization, 11,* 341-359.

Tan, M., Li, Y., Ding, D., Zhou, R. ve Huang, C. (2022). "An Improved JADE Hybridizing with Tuna Swarm Optimization for Numerical Optimization Problems". *Mathematical Problems in Engineering, 2022*.

Tein, L.H. ve Ramli, R. (2010). "Recent advancements of nurse scheduling models and a potential path".

Tuerxun, W., Xu, C., Guo, H., Guo, L. ve Yin, L. (2022). "Fault classification in wind turbine based on deep belief network optimized by modified tuna swarm optimization algorithm". *Journal of Renewable and Sustainable Energy, 14*(3): 033307.

Tuerxun, W., Xu, C., Guo, H., Guo, L., Zeng, N. ve Cheng, Z. (2022). "An ultra-short-term wind speed prediction model using LSTM based on modified tuna swarm optimization and successive variational mode decomposition". *Energy Science & Engineering, 10*(8): 3001-3022.

Valério de Carvalho, J. M. (1999). "Exact solution of bin-packing problems using column generation and branch-and-bound". *Annals of Operations Research, 86*(0): 629-659.

Viegas, J. P., Vieira, S. M., Sousa, J. M. ve Henriques, E. M. (2014). "Metaheuristics for the 3D bin packing problem in the steel industry". In *2014 IEEE Congress on Evolutionary Computation,* 338-343.

Virk, A. K. ve Singh, K. (2016). "Solving bi-objective two-dimensional rectangle packing problem using binary cuckoo search". *International Journal of Computer Science and Information Security, 14*(7): 165.

Virk, A. K. ve Singh, K. (2018). "Solving two-dimensional rectangle packing problem using nature-inspired metaheuristic algorithms". *Journal of Industrial Integration and Management, 3*(02): 1850009.

Walton, S., Hassan, O., Morgan, K. ve Brown, M. R. (2011). "Modified cuckoo search: a new gradient free optimisation algorithm". *Chaos, Solitons & Fractals, 44*(9): 710-718.

Wang, J., Zhu, L., Wu, B. ve Ryspayev, A. (2022). "Forestry Canopy Image Segmentation Based on Improved Tuna Swarm Optimization". *Forests, 13*(11): 1746.

Wang, W. ve Tian, J. (2022). "An Improved Nonlinear Tuna Swarm Optimization Algorithm Based on Circle Chaos Map and Levy Flight Operator". *Electronics, 11*(22): 3678.

Wäscher, G., Haußner, H., ve Schumann, H. (2007). "An improved typology of cutting and packing problems". *European journal of operational research, 183*(3): 1109-1130.

Wilson, R. C. (1965). “A packaging problem”. *Management Science, 12*(4): B-135.

Wu, H., Leung, S. C., Si, Y. W., Zhang, D. ve Lin, A. (2017). “Three-stage heuristic algorithm for three-dimensional irregular packing problem”. *Applied Mathematical Modelling, 41*, 431-444.

Xie, L., Han, T., Zhou, H., Zhang, Z. R., Han, B. ve Tang, A. (2021). “Tuna swarm optimization: a novel swarm-based metaheuristic algorithm for global optimization”. *Computational intelligence and Neuroscience,* 1-22.

Yachba, K., Roufaida, L. ve El Batoul, B. F. (2022). Cuckoo Search for the Resolution of the Container Storage Problem. “*International Journal of Organizational and Collective Intelligence (IJOCI)”*, *12*(1): 1-14.

Yan, Z., Yan, J., Wu, Y., Cai, S. ve Wang, H. (2023). “A novel reinforcement learning based tuna swarm optimization algorithm for autonomous underwater vehicle path planning”. *Mathematics and Computers in Simulation, 209,* 55-86.

Yang, X. S. ve Deb, S. (2009). “Cuckoo search via Lévy flights”. In *2009 World congress on nature & biologically inspired computing (NaBIC)* (pp. 210-214).

Yang, X.S. (2010). Nature-inspired metaheuristic algorithms. Luniver press, United Kingdom.

Yang, X.S. ve Deb, S. (2010). “Engineering optimisation by cuckoo search”. *International Journal of Mathematical Modelling and Numerical Optimisation, 1*(4): 330-343.

Yesodha, R. ve Amudha, T. (2013). “Effectiveness of firefly algorithm in solving bin packing problem”. International Journal of Advanced Research in Computer Science and Software Engineering, 3(5): 1003-10.

Yildiz, A. R. (2013). “Cuckoo search algorithm for the selection of optimal machining parameters in milling operations”. *The International Journal of Advanced Manufacturing Technology, 64,* 55-61.

Zendaoui, Z. ve Layeb, A. (2016). “Adaptive cuckoo search algorithm for the bin packing problem”. In *Modelling and Implementation of Complex Systems: Proceedings of the 4th International Symposium, MISC 2016, Constantine, Algeria, May 7-8, 2016, Constantine, Algeria,* 107-120.

Zhao, C., Jiang, L. ve Teo, K. L. (2020). “A hybrid chaos firefly algorithm for three-dimensional irregular packing problem”. *Journal of Industrial & Management Optimization, 16*(1): 409.

Zhao, X., Bennell, J. A., Bektaş, T. ve Dowsland, K. (2016). “A comparative review of 3D container loading algorithms”. *International Transactions in Operational Research, 23*(1-2): 287-320.

Zheng, H. ve Zhou, Y. (2012). "A novel cuckoo search optimization algorithm based on Gauss distribution". *Journal of Computational Information Systems, 8*(10): 4193-4200.

Zhou, J., Wang, D., Band, S. S., Mirzania, E. ve Roshni, T. (2023). "Atmosphere air temperature forecasting using the honey badger optimization algorithm: on the warmest and coldest areas of the World". *Engineering Applications of Computational Fluid Mechanics, 17*(1): 2174189.

Zudio, A., da Silva Costa, D. H., Masquio, B. P., Coelho, I. M. Ve Pinto, P. E. D. (2018). "BRKGA/VND hybrid algorithm for the classic three-dimensional bin packing problem". *Electronic Notes in Discrete Mathematics, 66*, 175-182.

EK 1

100 İterasyon n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	0.068	0.401	**0.022**	0.090	0.079	0.464	***0.030***	0.116
	100	0.261	1.664	**0.120**	0.341	0.300	1.753	***0.125***	0.461
	150	0.622	3.572	**0.266**	0.973	0.710	3.864	***0.287***	1.067
	200	1.163	6.603	**0.476**	1.667	1.228	6.914	***0.495***	1.833
2	50	0.052	0.341	**0.022**	0.080	0.079	0.459	***0.031***	0.115
	100	0.281	1.647	**0.122**	0.399	0.319	1.784	***0.130***	0.475
	150	0.614	3.834	**0.261**	0.993	0.711	4.001	***0.286***	1.110
	200	1.153	5.493	**0.487**	1.687	1.227	6.439	***0.508***	1.871
3	50	0.070	0.384	**0.031**	0.128	0.089	0.472	***0.039***	0.151
	100	0.311	1.689	**0.141**	0.472	0.351	1.789	***0.146***	0.538
	150	0.602	2.833	**0.294**	1.154	0.770	3.539	***0.314***	1.215
	200	1.294	5.858	**0.528**	1.995	1.404	6.540	***0.556***	2.134
4	50	0.050	0.253	**0.016**	0.094	0.057	0.293	***0.018***	0.102
	100	0.181	0.963	**0.078**	0.301	0.207	1.038	***0.086***	0.321
	150	0.443	1.938	**0.181**	0.620	0.473	2.157	***0.187***	0.695
	200	0.813	3.388	**0.325**	1.134	0.843	3.630	***0.335***	1.216
5	50	0.100	0.413	**0.042**	0.162	0.119	0.506	***0.048***	0.189
	100	0.431	2.097	**0.170**	0.662	0.517	2.365	***0.190***	0.726
	150	0.952	4.544	**0.374**	1.434	1.157	5.238	***0.432***	1.705
	200	1.688	7.933	**0.690**	2.437	2.008	9.198	***0.746***	2.847
6	50	0.080	0.261	**0.030**	0.110	0.094	0.374	***0.035***	0.147
	100	0.301	1.172	**0.123**	0.472	0.337	1.328	***0.135***	0.531
	150	0.642	2.538	**0.253**	1.012	0.719	2.894	***0.284***	1.100
	200	1.083	4.984	**0.485**	1.596	1.259	5.358	***0.504***	1.800
7	50	0.072	0.311	**0.031**	0.111	0.105	0.441	***0.042***	0.140
	100	0.383	1.455	**0.160**	0.431	0.414	1.796	***0.170***	0.527
	150	0.973	3.897	**0.365**	1.313	1.067	4.466	***0.414***	1.421
	200	1.404	5.460	**0.527**	2.059	1.618	6.650	***0.629***	2.275
8	50	0.070	0.299	**0.030**	0.099	0.095	0.348	***0.033***	0.126
	100	0.261	1.181	**0.120**	0.348	0.328	1.276	***0.132***	0.467
	150	0.703	2.809	**0.264**	0.944	0.784	3.077	***0.304***	1.089
	200	1.264	5.290	**0.517**	1.667	1.381	5.513	***0.545***	1.821

EK 2

100 İterasyon 2n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	0.807	1.641	**0.562**	0.672	0.916	1.790	***0.675***	0.721
	100	**2.414**	5.370	2.445	2.559	2.642	5.588	***2.637***	2.645
	150	5.384	11.648	**3.835**	5.436	5.714	12.108	***5.234***	5.862
	200	8.939	19.596	**4.232**	9.342	9.254	20.560	***4.970***	9.956
2	50	0.801	1.533	**0.311**	0.562	0.974	1.726	***0.341***	0.670
	100	2.551	5.287	**1.351**	2.455	2.790	5.566	***1.467***	2.614
	150	5.186	11.472	**2.890**	5.155	5.573	11.985	***3.657***	5.725
	200	8.828	19.328	**5.561**	10.080	9.275	20.339	***6.004***	11.471
3	50	0.929	1.716	**0.361**	1.032	1.070	1.966	***0.469***	1.210
	100	2.811	5.760	**1.664**	3.196	3.045	6.192	***1.990***	3.542
	150	5.582	12.177	**4.377**	6.857	5.991	12.948	***4.663***	7.318
	200	9.756	21.029	**6.872**	11.706	10.452	22.427	***7.524***	12.653
4	50	0.512	1.136	**0.251**	0.549	0.656	1.248	***0.296***	0.824
	100	1.577	3.532	**1.073**	1.876	1.893	3.896	***1.158***	2.111
	150	3.495	7.489	**2.198**	4.159	3.639	7.914	***2.506***	4.454
	200	5.836	12.792	**3.972**	7.256	6.230	13.573	***4.340***	7.598
5	50	1.019	2.246	**0.511**	1.344	1.238	2.486	***0.676***	1.482
	100	3.865	8.609	**2.550**	4.760	4.354	9.433	***2.885***	5.200
	150	8.588	17.984	**5.445**	10.041	9.419	20.074	***6.599***	11.235
	200	14.433	29.671	**8.473**	16.061	15.425	33.792	***10.909***	18.522
6	50	0.815	1.720	**0.383**	0.860	1.026	2.078	***0.479***	1.274
	100	2.654	5.411	**1.544**	3.104	2.923	5.894	***1.810***	3.549
	150	5.302	10.560	**3.513**	6.025	5.686	11.775	***3.895***	6.787
	200	9.289	18.839	**6.829**	10.964	9.678	19.713	***7.111***	11.702
7	50	0.985	1.662	**0.434**	1.093	1.172	2.033	***0.594***	1.532
	100	2.968	6.058	**1.906**	3.860	3.532	6.942	***2.282***	4.232
	150	7.609	14.742	**5.282**	9.273	8.214	15.717	***5.882***	9.890
	200	10.721	20.902	**7.744**	13.022	12.085	23.545	***9.118***	14.839
8	50	0.859	1.024	**0.361**	1.003	0.975	1.497	***0.478***	1.144
	100	2.600	4.495	**1.665**	2.907	2.920	5.090	***1.924***	3.543
	150	5.583	10.361	**3.530**	6.369	6.242	11.244	***4.068***	7.320
	200	8.607	19.831	**6.889**	12.632	10.486	20.301	***7.212***	13.206

EK 3

100 İterasyon 4n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	1.451	2.952	**1.263**	1.297	1.580	3.189	1.421	***1.381***
	100	4.724	10.029	**4.669**	4.864	5.077	10.770	***4.900***	5.121
	150	10.513	21.802	**5.223**	10.920	11.071	23.617	***6.650***	11.860
	200	17.627	38.279	**10.662**	19.649	18.542	39.722	***12.260***	21.258
2	50	1.320	2.911	**0.690**	1.646	1.524	3.168	***0.811***	1.870
	100	4.757	10.493	**3.097**	5.716	5.137	11.111	***3.825***	6.306
	150	9.883	21.952	**7.327**	11.791	10.732	23.573	***8.261***	13.213
	200	17.491	38.701	**14.063**	21.730	18.411	40.094	***14.540***	22.518
3	50	1.459	3.266	**0.870**	1.852	1.805	3.625	***1.032***	2.069
	100	5.230	11.101	**3.219**	6.233	5.506	11.899	***3.883***	6.771
	150	10.959	24.242	**7.509**	13.300	11.732	25.629	***8.545***	14.220
	200	19.291	41.608	**15.121**	22.893	20.205	43.930	***15.596***	24.674
4	50	1.012	2.047	**0.529**	1.215	1.152	2.410	***0.603***	1.498
	100	3.331	6.983	**2.115**	3.968	3.625	7.272	***2.310***	4.177
	150	6.998	14.550	**4.522**	8.220	7.209	14.916	***5.055***	8.619
	200	12.049	23.998	**8.493**	14.417	12.287	25.044	***9.066***	14.782
5	50	1.697	3.089	**1.193**	2.464	2.154	3.856	***1.385***	2.897
	100	7.456	15.335	**5.114**	8.507	8.051	15.947	***5.910***	9.868
	150	15.501	32.828	**11.004**	20.416	18.242	37.379	***12.785***	22.868
	200	15.758	54.664	**15.422**	20.007	25.613	60.775	***18.685***	32.358
6	50	0.713	2.588	**0.682**	0.943	0.838	2.875	***0.836***	1.070
	100	2.935	9.602	**2.778**	3.754	3.536	10.495	***3.069***	4.295
	150	6.298	18.165	**6.027**	8.244	7.176	21.482	***6.414***	9.137
	200	13.340	29.118	**11.035**	15.028	14.539	31.516	***11.502***	17.831
7	50	1.001	2.395	**0.822**	1.264	1.238	2.755	***1.055***	1.593
	100	4.252	9.311	**3.473**	4.892	4.914	10.713	***3.985***	5.998
	150	11.212	21.791	**8.805**	13.469	12.116	24.720	***9.712***	14.789
	200	16.829	29.601	**12.955**	19.737	18.519	33.176	***14.950***	22.742
8	50	0.842	1.687	**0.760**	0.950	0.984	1.869	***0.833***	1.264
	100	3.360	5.715	**2.678**	4.252	3.681	6.430	***2.976***	4.654
	150	7.931	12.597	**6.086**	9.648	8.864	14.949	***6.973***	10.921
	200	15.609	22.016	**12.040**	15.953	16.049	23.405	***12.949***	17.607

EK 4

500 İterasyon n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	0.879	2.707	0.742	**0.732**	0.956	3.133	0.887	***0.812***
	100	**2.653**	9.610	2.668	2.777	3.545	10.353	3.249	***2.950***
	150	6.669	21.502	**6.655**	7.465	7.613	22.718	***7.100***	8.051
	200	**12.350**	36.655	13.800	13.263	***14.010***	38.276	14.247	15.287
2	50	**0.682**	2.960	0.754	0.893	***0.879***	3.135	0.904	1.023
	100	**2.778**	9.916	3.100	3.269	***3.241***	10.729	3.607	3.910
	150	**7.160**	21.258	7.522	7.863	***7.642***	22.753	8.238	9.086
	200	**12.335**	37.189	13.369	14.462	***13.567***	38.548	13.925	15.797
3	50	0.842	3.144	**0.841**	1.004	***0.961***	3.385	1.019	1.127
	100	**3.292**	10.862	3.299	3.682	***3.663***	11.627	3.795	4.249
	150	**7.693**	22.523	7.871	9.217	***8.537***	24.359	8.736	9.963
	200	**14.133**	40.425	15.064	16.568	***15.434***	42.124	15.624	17.733
4	50	**0.471**	2.021	0.550	0.602	***0.541***	2.324	0.614	0.702
	100	**1.616**	6.714	1.823	2.114	***2.049***	7.123	2.295	2.590
	150	**4.153**	13.933	4.713	4.924	***4.623***	14.482	5.176	5.709
	200	**7.332**	23.816	8.387	8.955	***7.744***	24.278	8.871	9.392
5	50	**0.781**	3.677	1.115	1.242	***1.061***	3.947	1.273	1.398
	100	**3.601**	12.907	3.880	4.865	***4.103***	14.913	4.972	5.533
	150	**8.545**	32.152	9.527	10.274	***9.930***	34.595	10.983	12.049
	200	15.767	50.292	**15.055**	18.764	***17.586***	57.230	17.683	21.086
6	50	0.692	2.056	**0.664**	0.822	***0.786***	2.633	0.794	0.918
	100	**2.357**	8.445	2.708	3.202	***2.899***	9.270	3.003	3.537
	150	**5.709**	19.956	5.938	6.759	***6.210***	20.828	6.425	7.398
	200	**10.450**	29.714	11.173	12.346	***11.180***	32.353	11.563	13.122
7	50	**0.750**	2.367	0.752	0.903	***0.947***	2.824	1.038	1.188
	100	**3.240**	8.685	3.420	3.900	***3.729***	9.901	3.764	4.487
	150	**8.448**	22.889	8.530	10.160	***9.385***	24.582	9.739	11.162
	200	**12.292**	27.599	12.639	13.987	***14.116***	32.955	14.902	16.635
8	50	0.660	1.547	**0.642**	0.771	***0.764***	1.769	0.805	0.890
	100	2.506	5.497	**2.485**	2.711	***2.815***	6.160	3.012	3.207
	150	5.839	13.165	**5.798**	6.154	***6.351***	14.867	6.955	7.023
	200	**9.876**	23.254	11.331	10.333	***10.551***	24.473	12.014	12.253

EK 5

500 İterasyon 2n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	1.374	5.017	**1.372**	1.794	***1.467***	5.551	1.544	2.224
	100	**4.948**	19.994	5.778	9.258	***5.954***	20.870	6.023	9.642
	150	14.398	42.686	**13.476**	20.650	***15.860***	45.197	17.252	22.239
	200	**24.334**	73.077	33.029	38.455	***31.481***	77.088	35.710	40.404
2	50	**1.836**	5.104	2.744	2.854	***2.028***	5.838	2.969	3.276
	100	**7.706**	19.339	9.463	10.362	***8.358***	20.852	10.463	11.108
	150	**18.715**	43.436	19.585	22.279	***20.669***	45.103	21.939	23.868
	200	**33.072**	72.374	36.373	39.630	***34.692***	76.440	37.933	41.055
3	50	**2.462**	5.637	2.879	3.202	***2.668***	6.215	3.387	3.662
	100	**9.018**	19.157	10.445	11.336	***9.747***	20.740	11.255	12.144
	150	**19.435**	42.034	22.540	24.499	***21.398***	44.861	24.072	26.077
	200	**34.400**	71.885	39.341	40.461	***35.832***	76.245	41.863	45.480
4	50	**1.291**	3.280	1.984	1.526	***1.462***	3.510	2.161	2.066
	100	5.216	12.662	6.796	**3.454**	5.393	12.910	6.997	***4.012***
	150	11.431	21.819	14.477	**9.033**	11.850	23.997	15.054	***10.332***
	200	20.313	35.677	**13.000**	17.998	20.688	38.024	***19.046***	20.604
5	50	**2.357**	4.935	2.626	3.000	***2.762***	5.801	2.889	3.253
	100	10.744	20.064	**9.681**	12.528	11.871	22.812	***11.090***	13.387
	150	**23.435**	39.452	23.847	28.344	***26.926***	48.076	28.279	30.990
	200	38.689	57.398	39.465	**38.367**	***44.707***	64.820	46.282	47.462
6	50	1.796	2.543	1.775	**1.734**	2.347	2.951	2.210	***1.988***
	100	9.159	9.418	6.719	**6.442**	10.277	10.252	7.547	***7.025***
	150	19.355	18.297	13.421	**13.088**	20.741	20.284	***14.853***	14.934
	200	34.510	31.055	**23.114**	25.104	36.084	32.734	***23.929***	26.218
7	50	3.133	2.508	**1.655**	1.966	3.825	3.066	***2.216***	2.367
	100	11.558	9.310	**7.382**	7.996	12.731	10.503	***8.145***	9.078
	150	27.878	**14.442**	17.023	18.874	30.404	23.887	***20.375***	21.558
	200	40.987	34.520	26.293	**26.073**	45.672	37.248	***30.937***	32.316
8	50	2.662	2.026	**1.475**	2.167	3.017	2.279	***1.785***	2.486
	100	9.077	7.038	**5.439**	7.173	10.092	7.690	***6.207***	8.363
	150	20.100	15.472	**11.286**	16.980	22.430	16.935	***13.090***	23.350
	200	37.640	29.120	**19.033**	39.023	39.453	30.329	***27.361***	43.920

EK 6

500 İterasyon 4n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	***S_ort***
1	50	2.705	7.393	2.716	**2.656**	2.926	8.226	***2.917***	2.799
	100	9.545	28.569	**4.755**	10.090	9.919	30.221	***7.649***	10.483
	150	20.594	63.729	**12.612**	22.809	21.609	66.720	***14.109***	24.489
	200	34.944	108.638	**27.321**	42.123	36.481	113.799	***29.207***	43.783
2	50	2.563	8.225	**1.572**	3.072	2.942	8.812	***1.823***	3.367
	100	9.405	29.039	**6.348**	11.246	10.103	31.412	***7.331***	12.013
	150	19.774	60.990	**15.032**	23.740	21.492	66.144	***16.614***	25.971
	200	35.376	108.574	**28.867**	42.858	36.947	113.510	***29.643***	44.467
3	50	3.197	9.373	**1.737**	3.708	3.390	10.088	***2.195***	3.911
	100	10.314	32.004	**7.073**	11.882	10.948	33.707	***7.709***	13.096
	150	21.557	65.195	**13.684**	26.062	23.530	70.551	***15.237***	28.329
	200	**21.366**	111.792	24.452	25.865	34.035	117.040	***25.722***	41.393
4	50	0.996	4.153	**0.933**	1.233	1.145	4.809	***0.988***	1.339
	100	3.872	16.197	**3.722**	4.674	4.098	18.085	***3.946***	5.090
	150	8.795	36.648	**8.496**	11.874	10.225	39.467	***8.765***	13.097
	200	17.361	61.060	**14.802**	21.009	18.525	64.651	***15.097***	22.419
5	50	**2.234**	8.614	2.256	2.968	2.559	10.336	***2.409***	3.359
	100	9.336	42.511	**8.944**	12.987	10.639	46.900	***9.838***	14.414
	150	24.311	81.084	**20.667**	31.219	27.080	91.915	***22.947***	33.921
	200	32.092	111.983	**27.741**	38.155	***39.871***	133.580	37.435	46.574
6	50	**1.475**	4.763	1.795	1.817	***1.717***	5.419	2.357	2.145
	100	**5.587**	16.431	6.629	6.989	***6.200***	19.196	7.741	7.734
	150	**12.417**	34.408	18.975	14.835	***13.251***	36.348	20.347	16.067
	200	**21.927**	53.893	33.499	26.838	***23.222***	56.098	36.360	28.523
7	50	**1.687**	4.318	3.044	2.118	***2.102***	5.330	3.901	2.538
	100	**7.241**	17.059	11.457	7.717	***8.055***	19.500	12.989	9.026
	150	18.685	44.823	27.740	**18.582**	***19.904***	47.034	31.038	22.065
	200	**19.537**	58.708	37.692	40.456	***25.671***	66.153	45.324	49.311
8	50	**1.582**	3.449	2.428	2.885	***1.775***	4.138	2.718	3.250
	100	**5.606**	12.032	8.596	10.666	***7.138***	13.108	9.353	12.140
	150	**14.813**	26.352	19.460	25.209	***17.438***	28.895	21.438	27.859
	200	**36.106**	45.920	36.225	46.473	***37.434***	47.601	38.066	48.176

EK 7

1000 İterasyon n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	**1.566**	4.505	1.939	1.969	***1.769***	5.157	2.244	2.299
	100	**5.769**	17.769	8.191	7.041	***6.785***	18.803	9.010	7.868
	150	**13.810**	38.985	16.422	16.849	***16.065***	41.571	19.298	17.897
	200	29.695	67.216	31.953	**27.831**	32.670	70.518	35.961	***29.980***
2	50	1.816	4.990	2.357	**1.626**	2.127	5.459	2.577	***1.953***
	100	8.738	18.434	8.512	**6.792**	9.481	19.496	9.523	***7.763***
	150	19.186	38.915	19.492	**15.659**	20.708	41.320	20.926	***17.674***
	200	33.445	67.163	33.861	**23.415**	34.681	70.530	35.958	***26.226***
3	50	2.397	5.911	2.410	**1.645**	2.624	6.270	2.712	***1.818***
	100	9.097	19.690	9.099	**6.351**	9.601	21.184	9.865	***6.951***
	150	19.093	41.464	20.142	**14.401**	20.648	44.198	21.756	***15.544***
	200	33.145	66.795	36.061	**25.311**	35.751	73.148	37.878	***27.512***
4	50	1.404	2.935	1.562	**1.003**	1.468	3.137	1.631	***1.110***
	100	5.196	10.006	5.657	**4.163**	5.348	10.923	5.945	***4.239***
	150	11.695	22.293	12.358	**8.906**	11.847	24.052	12.913	***9.316***
	200	20.192	39.960	20.504	**15.775**	20.785	41.325	21.049	***16.207***
5	50	2.487	5.795	3.148	**2.268**	2.761	6.283	3.362	***2.408***
	100	9.551	23.562	10.954	**8.082**	11.179	26.144	12.661	***9.423***
	150	21.678	49.427	24.811	**18.591**	25.948	60.510	28.267	***24.041***
	200	**38.040**	67.396	41.699	49.518	***43.741***	82.403	52.542	55.833
6	50	2.817	2.937	**2.702**	2.871	***3.219***	3.349	3.445	3.379
	100	**8.926**	11.198	10.165	10.358	***9.996***	12.107	10.965	11.266
	150	**18.657**	22.480	20.179	20.706	***20.766***	25.303	21.790	22.303
	200	**32.478**	34.095	35.792	37.004	***35.838***	38.542	38.010	38.659
7	50	2.991	**2.695**	3.075	3.009	3.745	***3.251***	4.008	3.658
	100	10.845	**10.773**	11.138	11.467	12.324	***11.825***	13.669	12.619
	150	**27.167**	27.553	27.344	27.658	29.331	***29.204***	32.240	30.889
	200	39.534	**38.766**	40.612	42.234	44.493	***43.338***	48.159	46.665
8	50	2.528	**2.176**	2.952	2.528	2.945	***2.574***	3.346	2.922
	100	9.043	**7.715**	9.684	8.624	9.880	***8.314***	10.583	9.588
	150	20.477	**17.268**	20.794	19.723	22.552	***19.038***	23.274	21.428
	200	36.913	**30.482**	36.936	35.501	38.288	***31.507***	40.163	37.530

EK 8

1000 İterasyon 2n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	**2.697**	7.406	3.358	3.090	***3.095***	8.447	3.845	3.410
	100	**11.107**	28.968	12.858	12.523	***13.768***	29.805	14.755	14.296
	150	**30.653**	58.227	31.666	34.922	35.890	64.688	***34.651***	37.272
	200	62.012	99.436	**50.245**	54.875	68.904	104.399	***57.575***	62.243
2	50	4.454	6.190	**3.131**	3.139	4.742	7.021	***3.394***	3.639
	100	16.917	24.759	**11.145**	12.210	18.168	26.519	***12.327***	13.205
	150	35.298	52.362	**25.837**	28.700	38.297	59.993	***27.614***	30.017
	200	63.157	110.006	**47.792**	49.471	66.067	115.442	***49.063***	52.024
3	50	4.393	9.131	**3.168**	3.321	5.076	10.092	3.779	***3.757***
	100	16.383	31.870	12.169	**12.025**	17.557	33.878	13.146	***13.035***
	150	34.868	67.859	**21.477**	25.216	38.746	72.472	***29.561***	35.746
	200	63.837	110.370	**62.276**	80.594	***73.178***	120.532	76.114	85.257
4	50	**3.573**	4.964	3.584	3.653	3.953	5.229	3.915	***3.771***
	100	**12.104**	17.010	12.553	13.447	***12.809***	18.351	12.930	13.676
	150	**26.255**	36.810	27.621	27.449	***27.281***	38.136	27.949	29.193
	200	46.314	56.790	**44.059**	47.294	46.878	64.801	***45.974***	48.386
5	50	**6.577**	8.504	6.823	7.014	***7.164***	9.111	7.345	7.562
	100	26.125	31.523	**25.011**	28.402	***27.613***	36.503	28.448	30.356
	150	59.743	61.104	**54.300**	59.154	***63.240***	71.675	63.756	67.212
	200	93.710	**78.356**	90.208	106.542	100.424	***92.551***	107.037	126.450
6	50	4.555	**3.539**	5.610	6.270	5.095	***4.047***	6.489	7.139
	100	15.537	**12.491**	19.980	22.073	17.007	***13.615***	21.720	23.685
	150	33.187	**23.147**	40.993	45.250	35.222	***25.613***	44.375	47.814
	200	51.772	**35.321**	72.803	76.955	60.727	***41.645***	76.224	81.584
7	50	3.911	**3.634**	6.317	6.465	4.751	***4.560***	8.022	8.041
	100	16.309	**13.325**	23.601	24.665	17.927	***15.539***	26.830	28.156
	150	36.040	**32.688**	54.734	59.590	39.592	***34.882***	63.494	65.106
	200	**41.477**	46.204	79.088	77.410	53.135	***51.343***	90.308	94.018
8	50	**2.477**	2.701	4.562	4.505	***2.671***	3.076	4.944	5.197
	100	**8.176**	9.606	16.650	15.294	***9.064***	10.287	17.602	17.372
	150	**18.542**	21.631	36.596	33.243	***20.658***	23.607	43.162	37.918
	200	**35.745**	40.195	60.421	55.007	***36.815***	41.850	64.798	57.569

EK 9

1000 İterasyon 4n popülasyon için çözüm süreleri (sn)

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	S_min	S_min	S_min	S_min	S_ort	S_ort	S_ort	S_ort
1	50	4.924	13.611	6.741	**4.322**	5.647	14.895	7.362	***4.931***
	100	19.566	50.052	23.861	**17.850**	***20.999***	51.851	27.065	22.535
	150	**46.628**	100.808	57.430	50.052	***55.010***	112.077	62.564	58.120
	200	**98.467**	171.561	100.045	101.345	***103.208***	181.816	106.501	111.207
2	50	**6.350**	11.066	6.636	7.360	***7.071***	12.226	7.188	7.973
	100	**24.656**	42.995	24.991	26.512	***26.842***	46.366	27.460	29.762
	150	**52.651**	90.087	55.233	58.326	***57.191***	104.142	60.314	61.601
	200	**94.249**	194.557	94.957	100.503	***99.279***	201.126	101.637	106.042
3	50	**6.322**	16.167	6.786	7.118	***7.060***	17.270	7.850	7.971
	100	**24.608**	56.762	25.898	27.238	***26.154***	59.470	28.315	29.183
	150	**50.456**	118.974	53.672	55.597	***57.455***	126.864	63.408	61.997
	200	**92.063**	193.521	116.475	98.286	***111.321***	210.968	124.077	122.992
4	50	**5.170**	8.618	5.960	5.710	***5.674***	9.182	6.289	6.367
	100	**18.137**	29.594	20.201	20.389	***18.875***	31.788	20.819	21.079
	150	**39.301**	66.188	42.367	44.021	***40.891***	67.050	43.533	44.693
	200	**68.259**	98.226	71.580	73.614	***69.625***	112.225	73.914	76.475
5	50	**9.581**	14.379	12.039	12.099	***10.488***	15.623	12.949	12.831
	100	**37.461**	55.912	38.954	46.100	***41.868***	62.813	46.215	49.263
	150	**79.181**	102.543	88.768	96.398	***92.211***	121.483	101.394	109.881
	200	125.206	134.783	**114.209**	147.748	***144.101***	160.631	154.673	173.456
6	50	**5.417**	6.176	5.961	6.921	7.231	***7.035***	7.441	8.161
	100	**21.334**	21.641	24.316	26.327	24.332	***23.923***	27.686	29.651
	150	49.190	**41.089**	46.906	47.350	53.454	***44.995***	56.697	58.475
	200	77.079	**57.876**	77.807	80.497	86.292	***72.310***	84.465	86.726
7	50	**5.327**	5.859	5.824	5.915	***6.498***	7.782	7.308	7.335
	100	22.641	22.186	**20.852**	23.374	25.742	26.933	***24.905***	25.522
	150	50.547	56.769	**46.653**	52.533	***56.855***	60.577	59.454	60.418
	200	61.437	80.023	**58.162**	62.470	76.712	89.379	***72.854***	74.076
8	50	**3.193**	4.754	3.732	3.770	***3.637***	5.456	4.167	4.409
	100	**12.129**	16.605	13.366	13.759	***13.064***	17.925	14.733	15.060
	150	**27.755**	37.130	28.914	31.512	***30.294***	40.935	32.820	34.810
	200	**50.285**	70.092	50.607	52.637	***53.053***	72.822	55.224	55.288

EK 10

100 İterasyon n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	OC_std	OC_std	OC_std	OC_std
1	50	0.520	**0.493**	0.533	0.529	0.568	***0.493***	0.569	0.560	0.025	0.025	0.021	**0.015**
	100	**0.487**	0.489	0.492	0.489	0.534	***0.489***	0.537	0.534	0.028	**0.025**	0.029	0.031
	150	0.512	**0.489**	0.504	0.509	0.532	***0.489***	0.533	0.534	0.016	0.019	0.017	**0.015**
	200	0.500	**0.492**	0.498	0.497	0.515	***0.492***	0.519	0.516	**0.010**	0.011	0.011	0.011
2	50	0.522	**0.519**	0.544	0.541	0.568	***0.519***	0.569	0.569	0.021	0.027	**0.020**	0.021
	100	0.506	0.503	0.502	**0.495**	0.537	***0.503***	0.535	0.528	**0.020**	0.022	0.022	0.024
	150	0.511	**0.496**	0.501	0.504	0.528	***0.496***	0.528	0.526	**0.012**	0.015	0.017	0.014
	200	0.493	**0.486**	0.504	0.497	0.517	***0.486***	0.519	0.518	0.015	0.013	**0.010**	0.011
3	50	0.528	**0.513**	0.530	0.520	0.563	***0.513***	0.561	0.556	0.020	**0.019**	0.023	0.023
	100	0.505	**0.498**	0.504	0.502	0.536	***0.498***	0.536	0.531	0.023	**0.018**	0.020	0.023
	150	0.493	**0.485**	0.498	0.498	0.525	***0.485***	0.526	0.523	0.019	0.017	0.017	**0.016**
	200	0.493	**0.480**	0.498	0.489	0.513	***0.480***	0.514	0.511	0.014	0.014	**0.012**	0.013
4	50	**0.693**	0.695	0.698	0.696	0.742	***0.695***	0.741	0.742	0.027	0.027	0.027	**0.027**
	100	**0.661**	0.662	0.663	0.665	0.716	***0.662***	0.717	0.715	0.029	0.029	0.028	**0.028**
	150	0.724	0.721	0.723	**0.719**	0.741	***0.721***	0.742	0.741	0.010	**0.010**	0.010	0.011
	200	**0.673**	0.673	0.674	0.675	0.697	***0.673***	0.696	0.697	0.019	0.018	**0.018**	0.018
5	50	0.588	0.548	0.547	**0.528**	0.615	***0.548***	0.597	0.598	**0.019**	0.022	0.038	0.035
	100	0.522	**0.512**	0.529	0.522	0.560	***0.512***	0.559	0.553	0.017	0.017	**0.017**	0.020
	150	0.489	**0.488**	0.512	0.491	0.533	***0.488***	0.534	0.531	0.022	0.019	**0.013**	0.019
	200	0.498	0.492	0.490	**0.483**	0.513	***0.492***	0.513	0.506	0.010	**0.007**	0.013	0.010
6	50	0.505	0.479	**0.458**	0.475	0.534	***0.479***	0.520	0.514	0.022	**0.017**	0.032	0.024
	100	**0.426**	0.433	0.448	0.436	0.480	***0.433***	0.484	0.478	0.032	0.029	0.031	**0.027**
	150	0.418	0.421	0.425	**0.411**	0.460	***0.421***	0.458	0.444	**0.024**	0.024	0.025	0.028
	200	0.408	**0.379**	0.392	0.391	0.434	***0.379***	0.432	0.426	0.016	0.017	0.020	**0.015**
7	50	0.654	**0.608**	0.619	0.624	0.669	***0.608***	0.663	0.653	**0.014**	0.024	0.026	0.022
	100	0.567	**0.557**	0.572	0.567	0.600	***0.557***	0.599	0.592	0.023	0.018	0.018	**0.015**
	150	0.530	**0.492**	0.506	0.504	0.552	***0.492***	0.551	0.542	**0.013**	0.023	0.024	0.019
	200	0.501	**0.488**	0.512	0.504	0.538	***0.488***	0.544	0.540	**0.024**	0.027	0.027	0.029
8	50	0.561	**0.536**	0.548	0.538	0.625	***0.536***	0.625	0.620	**0.031**	0.034	0.036	0.042
	100	0.539	**0.515**	0.535	0.527	0.561	***0.515***	0.558	0.550	0.019	0.019	**0.017**	0.022
	150	0.496	**0.471**	0.495	0.476	0.534	***0.471***	0.535	0.531	0.023	**0.021**	0.022	0.023
	200	0.493	**0.476**	0.485	0.481	0.510	***0.476***	0.506	0.499	**0.009**	0.011	0.016	0.011

EK 11

100 İterasyon 2n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	OC_std	OC_std	OC_std	OC_std
1	50	0.529	**0.504**	0.522	0.515	0.550	0.549	0.556	***0.547***	**0.017**	0.023	0.023	0.019
	100	**0.477**	0.488	0.491	0.489	0.523	***0.522***	0.527	0.529	0.028	0.026	0.027	**0.025**
	150	0.488	**0.482**	0.484	0.498	0.522	***0.520***	0.527	0.527	0.016	0.017	0.019	**0.015**
	200	0.496	0.487	0.492	**0.487**	0.511	***0.506***	0.511	0.506	**0.009**	0.012	0.012	0.011
2	50	**0.495**	0.513	0.521	0.527	0.554	***0.547***	0.554	0.552	0.031	0.025	0.025	**0.021**
	100	**0.484**	0.496	0.499	0.499	0.523	***0.519***	0.525	0.522	0.023	0.021	**0.020**	0.021
	150	0.491	0.487	**0.486**	0.489	0.516	***0.512***	0.517	0.516	0.016	0.016	0.015	**0.014**
	200	**0.482**	0.484	0.484	0.488	0.509	***0.507***	0.511	0.509	**0.012**	0.013	0.013	0.012
3	50	**0.491**	0.500	0.522	0.504	0.545	***0.542***	0.552	0.544	0.028	0.025	**0.022**	0.026
	100	0.496	0.475	**0.471**	0.488	0.528	***0.517***	0.523	0.523	**0.020**	0.028	0.025	0.022
	150	0.492	**0.474**	0.489	0.489	0.514	***0.511***	0.516	0.515	0.014	0.018	**0.013**	0.016
	200	0.481	**0.478**	0.481	0.482	0.503	***0.500***	0.504	0.505	0.014	0.014	0.014	**0.013**
4	50	**0.692**	0.692	0.695	0.694	0.740	***0.738***	0.740	0.739	0.027	0.027	**0.026**	0.026
	100	0.663	0.661	**0.659**	0.662	0.714	***0.712***	0.714	0.714	0.029	**0.028**	0.029	0.029
	150	0.720	0.719	**0.718**	0.720	0.739	***0.739***	0.739	0.739	0.011	**0.011**	0.011	0.011
	200	0.675	**0.672**	0.674	0.674	0.694	***0.693***	0.695	0.694	0.018	0.019	**0.018**	0.019
5	50	0.550	0.530	0.547	**0.510**	0.584	***0.569***	0.582	0.575	0.029	**0.028**	0.030	0.037
	100	0.506	0.508	**0.505**	0.514	0.542	***0.532***	0.533	0.533	0.017	**0.012**	0.016	0.017
	150	0.483	**0.477**	0.485	0.484	0.521	0.509	0.517	***0.507***	0.020	**0.017**	0.019	0.017
	200	0.484	0.482	**0.462**	0.467	0.498	0.495	0.493	***0.490***	0.012	**0.010**	0.015	0.014
6	50	0.471	0.466	0.467	**0.465**	0.506	0.495	0.502	***0.495***	0.021	0.025	0.025	**0.019**
	100	0.420	**0.413**	0.416	0.440	0.456	***0.456***	0.464	0.467	0.031	0.028	0.031	**0.023**
	150	0.420	0.390	**0.386**	0.389	0.439	***0.431***	0.440	0.437	**0.019**	0.026	0.030	0.029
	200	0.387	0.376	**0.372**	0.372	0.417	***0.410***	0.416	0.416	**0.018**	0.018	0.021	0.019
7	50	0.602	0.610	**0.592**	0.602	0.638	***0.633***	0.637	0.641	0.024	**0.021**	0.030	0.022
	100	0.559	**0.538**	0.555	0.548	0.578	***0.568***	0.576	0.576	**0.009**	0.018	0.014	0.012
	150	0.490	**0.477**	0.496	0.506	0.533	***0.521***	0.533	0.534	0.024	0.021	0.020	**0.016**
	200	0.481	**0.475**	0.496	0.490	0.528	***0.520***	0.524	0.522	0.030	0.033	**0.024**	0.027
8	50	0.541	0.538	**0.538**	0.548	0.617	***0.604***	0.611	0.611	0.037	0.037	0.039	**0.031**
	100	0.523	0.514	0.524	**0.511**	0.543	***0.540***	0.545	0.541	0.017	0.020	**0.013**	0.021
	150	**0.471**	0.476	0.475	0.478	***0.513***	0.515	0.522	0.519	0.022	**0.020**	0.022	0.022
	200	**0.481**	0.484	0.484	0.484	0.499	***0.493***	0.495	0.496	0.009	0.008	**0.008**	0.009

EK 12

100 İterasyon 4n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	OC_std	OC_std	OC_std	OC_std
1	50	0.521	0.512	**0.479**	0.518	0.547	***0.542***	0.544	0.549	**0.019**	0.021	0.029	0.023
	100	**0.465**	0.484	0.487	0.488	***0.519***	0.521	0.525	0.525	0.031	0.027	0.029	**0.026**
	150	**0.482**	0.483	0.499	0.493	0.519	***0.516***	0.523	0.521	0.019	0.019	0.018	**0.015**
	200	**0.483**	0.489	0.495	0.484	0.507	***0.503***	0.508	0.507	0.013	**0.010**	0.010	0.013
2	50	0.521	**0.502**	0.510	0.505	0.550	***0.544***	0.553	0.547	**0.020**	0.021	0.030	0.026
	100	**0.486**	0.486	0.491	0.492	***0.513***	0.514	0.520	0.519	0.023	0.024	0.025	**0.020**
	150	0.488	0.490	0.496	**0.471**	0.516	***0.511***	0.516	0.513	0.016	**0.013**	0.014	0.020
	200	0.479	0.480	**0.478**	0.482	0.508	0.506	***0.505***	0.508	0.013	0.013	**0.013**	0.013
3	50	0.496	0.500	**0.494**	0.504	0.544	***0.538***	0.543	0.540	0.025	0.025	0.027	**0.024**
	100	0.496	0.490	0.489	**0.486**	0.523	***0.518***	0.522	0.523	0.022	**0.021**	0.024	0.025
	150	0.489	**0.480**	0.489	0.491	0.512	***0.509***	0.513	0.513	0.015	0.015	0.014	**0.014**
	200	**0.473**	0.477	0.482	0.475	***0.497***	0.499	0.503	0.499	0.015	0.014	**0.011**	0.014
4	50	0.694	**0.692**	0.692	0.694	0.738	***0.736***	0.738	0.737	0.026	0.027	0.027	**0.026**
	100	0.662	**0.659**	0.661	0.660	0.713	***0.712***	0.714	0.713	**0.028**	0.029	0.029	0.029
	150	0.721	**0.719**	0.720	0.721	0.738	***0.738***	0.739	0.738	0.010	0.011	**0.010**	0.011
	200	**0.672**	0.673	0.673	0.674	0.694	***0.693***	0.694	0.694	0.018	0.019	0.018	**0.018**
5	50	0.537	0.533	0.541	**0.531**	0.574	***0.557***	0.575	0.563	0.032	0.022	0.024	**0.020**
	100	0.485	0.506	0.516	**0.468**	0.524	***0.523***	0.535	0.525	0.021	0.014	**0.013**	0.025
	150	0.478	0.476	0.485	**0.461**	0.514	0.508	0.515	***0.506***	0.018	0.021	**0.018**	0.020
	200	0.473	0.469	**0.452**	0.470	0.491	***0.483***	0.486	0.486	0.014	**0.007**	0.016	0.014
6	50	0.467	**0.452**	0.460	0.452	0.493	0.482	0.494	***0.481***	0.021	0.023	0.025	**0.018**
	100	0.432	0.419	0.416	**0.410**	0.458	***0.446***	0.458	0.451	0.026	**0.024**	0.029	0.029
	150	**0.388**	0.398	0.396	0.406	0.433	***0.429***	0.440	0.436	0.027	0.027	0.026	**0.024**
	200	0.373	0.373	**0.368**	0.375	0.413	***0.408***	0.414	0.415	0.018	**0.015**	0.021	0.020
7	50	0.614	**0.603**	0.611	0.613	0.634	***0.625***	0.637	0.631	**0.015**	0.022	0.020	0.021
	100	**0.530**	0.544	0.540	0.548	0.571	***0.567***	0.571	0.570	0.019	**0.014**	0.016	0.014
	150	0.512	**0.491**	0.506	0.495	0.533	***0.517***	0.534	0.524	0.016	**0.015**	0.017	0.018
	200	0.498	**0.476**	0.485	0.494	0.524	***0.516***	0.528	0.521	**0.025**	0.028	0.030	0.026
8	50	0.530	**0.524**	0.548	0.533	0.606	***0.590***	0.610	0.609	0.035	0.040	0.035	**0.035**
	100	0.510	**0.509**	0.518	0.510	0.539	***0.535***	0.544	0.543	0.021	**0.017**	0.020	0.020
	150	**0.466**	0.470	0.470	0.472	0.511	***0.510***	0.516	0.514	0.023	**0.020**	0.020	0.022
	200	0.476	**0.473**	0.486	0.476	0.496	***0.486***	0.496	0.490	0.011	**0.007**	0.008	0.009

EK 13

500 İterasyon n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	OC_std	OC_std	OC_std	OC_std
1	50	**0.473**	0.514	0.492	0.519	***0.542***	0.545	0.550	0.546	0.027	0.020	0.029	**0.019**
	100	**0.467**	0.484	0.470	0.482	0.518	***0.517***	0.521	0.523	0.030	**0.026**	0.028	0.031
	150	0.491	0.482	**0.479**	0.486	0.518	***0.516***	0.519	0.523	**0.016**	0.019	0.019	0.018
	200	0.484	0.482	**0.480**	0.486	0.506	***0.502***	0.507	0.505	0.014	0.012	0.015	**0.011**
2	50	**0.487**	0.498	0.513	0.526	0.542	***0.540***	0.555	0.550	0.031	0.026	0.024	**0.022**
	100	0.480	**0.474**	0.488	0.490	0.515	***0.512***	0.519	0.518	0.023	0.024	**0.021**	0.023
	150	0.486	**0.481**	0.482	0.495	0.511	***0.509***	0.519	0.517	**0.015**	0.017	0.016	0.015
	200	0.482	**0.475**	0.490	0.485	0.507	***0.503***	0.506	0.509	0.012	0.014	**0.010**	0.012
3	50	0.505	**0.487**	0.508	0.506	0.543	***0.535***	0.547	0.541	0.024	0.028	**0.020**	0.026
	100	0.482	**0.481**	0.489	0.482	***0.517***	0.520	0.524	0.519	0.022	0.025	**0.021**	0.023
	150	0.487	**0.467**	0.493	0.489	0.511	***0.506***	0.517	0.515	**0.014**	0.018	0.017	0.016
	200	**0.473**	0.481	0.482	0.479	***0.496***	0.498	0.503	0.499	0.015	0.014	0.014	**0.012**
4	50	**0.690**	0.691	0.693	0.693	0.737	***0.737***	0.738	0.738	0.027	0.027	**0.026**	0.026
	100	0.661	**0.658**	0.663	0.659	0.712	***0.711***	0.714	0.713	0.029	0.029	**0.028**	0.029
	150	**0.718**	0.718	0.722	0.720	0.738	***0.738***	0.739	0.739	0.011	0.011	**0.010**	0.011
	200	0.672	0.673	0.674	**0.672**	***0.693***	0.693	0.695	0.694	0.018	0.018	**0.018**	0.019
5	50	0.547	**0.509**	0.544	0.529	0.573	***0.561***	0.574	0.568	0.022	0.024	0.021	**0.021**
	100	0.500	**0.499**	0.508	0.509	***0.523***	0.524	0.534	0.525	0.015	0.014	0.014	**0.013**
	150	**0.457**	0.478	0.470	0.480	0.507	***0.503***	0.506	0.507	0.023	0.017	0.019	**0.016**
	200	0.473	0.458	**0.451**	0.462	0.490	0.483	0.483	***0.477***	0.013	0.016	0.018	**0.013**
6	50	0.454	**0.448**	0.476	0.455	0.490	***0.474***	0.494	0.491	0.026	0.020	**0.019**	0.026
	100	**0.397**	0.406	0.429	0.431	***0.443***	0.452	0.465	0.460	0.032	0.027	0.030	**0.022**
	150	**0.376**	0.386	0.419	0.388	***0.425***	0.428	0.438	0.431	0.027	0.030	**0.025**	0.027
	200	**0.363**	0.365	0.386	0.368	***0.405***	0.408	0.420	0.415	**0.017**	0.018	0.018	0.019
7	50	0.619	**0.573**	0.603	0.614	0.634	***0.617***	0.632	0.638	**0.015**	0.025	0.020	0.019
	100	**0.530**	0.551	0.556	0.556	0.563	***0.560***	0.575	0.572	0.017	**0.008**	0.012	0.010
	150	0.486	0.496	0.508	**0.484**	***0.519***	0.522	0.535	0.525	0.023	**0.016**	0.018	0.023
	200	0.483	0.482	0.501	**0.478**	***0.515***	0.521	0.530	0.522	0.028	0.028	**0.026**	0.029
8	50	0.550	**0.515**	0.548	0.534	0.609	***0.590***	0.613	0.606	0.034	0.044	**0.032**	0.034
	100	0.507	0.512	**0.493**	0.523	***0.531***	0.532	0.541	0.543	0.017	**0.017**	0.025	0.019
	150	**0.462**	0.467	0.483	0.472	***0.508***	0.508	0.512	0.516	0.019	0.020	**0.018**	0.021
	200	0.477	**0.470**	0.488	0.471	0.487	***0.486***	0.499	0.490	**0.008**	0.010	0.009	0.010

EK 14

500 İterasyon 2n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	OC_std	OC_std	OC_std	OC_std
1	50	0.512	**0.499**	0.509	0.500	0.539	***0.537***	0.544	0.541	**0.022**	0.023	0.022	0.023
	100	0.469	**0.467**	0.479	0.470	0.515	***0.514***	0.522	0.518	**0.027**	0.030	0.029	0.031
	150	0.481	0.479	0.484	**0.479**	0.516	***0.514***	0.519	0.520	0.019	0.018	**0.017**	0.018
	200	**0.476**	0.485	0.489	0.487	***0.499***	0.502	0.507	0.501	0.015	0.011	0.012	**0.009**
2	50	**0.494**	0.503	0.506	0.506	0.540	***0.537***	0.548	0.540	0.029	**0.022**	0.023	0.026
	100	**0.470**	0.484	0.494	0.490	***0.509***	0.509	0.517	0.516	0.027	0.024	**0.017**	0.023
	150	**0.481**	0.486	0.495	0.486	***0.505***	0.509	0.517	0.510	0.016	0.015	0.014	**0.014**
	200	0.479	**0.475**	0.483	0.477	***0.500***	0.500	0.506	0.504	**0.011**	0.013	0.012	0.014
3	50	0.495	0.493	0.506	**0.487**	0.538	0.537	0.542	***0.536***	0.025	0.026	**0.022**	0.028
	100	0.478	**0.471**	0.486	0.487	0.514	***0.511***	0.524	0.521	0.026	0.025	0.025	**0.024**
	150	**0.474**	0.478	0.486	0.486	0.505	***0.504***	0.512	0.513	0.016	0.016	**0.015**	0.016
	200	**0.472**	0.476	0.482	0.477	***0.490***	0.496	0.499	0.499	**0.012**	0.013	0.014	0.013
4	50	0.691	**0.689**	0.693	0.691	0.737	***0.736***	0.738	0.737	0.027	0.027	**0.026**	0.027
	100	**0.657**	0.659	0.661	0.658	***0.711***	0.711	0.712	0.712	0.030	0.029	**0.028**	0.029
	150	0.717	**0.717**	0.721	0.719	***0.736***	0.737	0.740	0.737	0.011	0.011	0.011	**0.010**
	200	**0.671**	0.673	0.673	0.672	***0.692***	0.693	0.694	0.693	0.019	0.019	**0.018**	0.019
5	50	0.534	**0.507**	0.513	0.527	0.565	***0.554***	0.559	0.564	**0.019**	0.029	0.028	0.023
	100	0.503	**0.498**	0.507	0.506	0.522	***0.520***	0.524	0.522	0.013	0.013	**0.012**	0.015
	150	0.472	0.472	**0.460**	0.462	***0.497***	0.499	0.504	0.501	0.016	**0.014**	0.023	0.021
	200	0.470	0.463	**0.427**	0.461	0.484	0.481	***0.469***	0.480	0.012	**0.011**	0.019	0.013
6	50	0.455	0.442	0.441	**0.435**	***0.475***	0.477	0.482	0.478	0.023	**0.020**	0.026	0.024
	100	**0.384**	0.393	0.415	0.427	0.439	***0.436***	0.452	0.450	0.032	0.032	0.026	**0.023**
	150	0.391	**0.373**	0.414	0.398	***0.422***	0.424	0.436	0.423	0.025	0.031	0.026	**0.023**
	200	**0.357**	0.362	0.370	0.364	***0.395***	0.402	0.413	0.408	**0.015**	0.018	0.019	0.019
7	50	0.592	**0.583**	0.611	0.608	0.619	***0.616***	0.634	0.632	0.021	0.024	**0.020**	0.021
	100	**0.530**	0.545	0.554	0.542	***0.559***	0.561	0.574	0.567	0.018	**0.012**	0.013	0.015
	150	**0.484**	0.493	0.495	0.493	***0.512***	0.520	0.524	0.520	0.015	0.019	0.019	**0.014**
	200	**0.466**	0.482	0.477	0.494	***0.509***	0.510	0.518	0.517	0.030	0.026	0.031	**0.026**
8	50	**0.529**	0.532	0.538	0.544	0.593	***0.585***	0.596	0.598	0.039	0.040	0.039	**0.033**
	100	0.503	**0.476**	0.516	0.508	0.528	***0.523***	0.541	0.534	0.020	0.021	**0.017**	0.018
	150	0.469	**0.455**	0.471	0.477	***0.499***	0.506	0.515	0.512	0.019	0.022	0.022	**0.019**
	200	**0.456**	0.475	0.483	0.464	***0.474***	0.482	0.490	0.488	0.012	**0.006**	0.007	0.010

EK 15

500 İterasyon 4n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	OC_std	OC_std	OC_std	OC_std
1	50	**0.458**	0.514	0.472	0.514	***0.537***	0.540	0.540	0.541	0.033	**0.019**	0.031	0.020
	100	**0.459**	0.474	0.480	0.474	0.515	***0.514***	0.521	0.519	0.030	0.030	**0.027**	0.030
	150	**0.482**	0.487	0.485	0.487	0.516	***0.514***	0.520	0.519	0.020	**0.017**	0.018	0.018
	200	**0.479**	0.480	0.488	0.487	0.503	***0.500***	0.505	0.503	0.013	0.013	0.010	**0.010**
2	50	**0.501**	0.502	0.516	0.508	0.542	***0.541***	0.546	0.544	0.024	0.026	0.024	**0.024**
	100	**0.484**	0.487	0.489	0.488	***0.509***	0.514	0.515	0.515	**0.018**	0.021	0.020	0.021
	150	0.486	**0.484**	0.491	0.489	0.509	0.509	0.515	***0.509***	0.013	0.015	0.013	**0.012**
	200	0.480	**0.475**	0.476	0.484	***0.501***	0.503	0.506	0.505	0.012	0.014	0.013	**0.011**
3	50	0.506	**0.490**	0.496	0.505	0.541	***0.535***	0.539	0.540	**0.023**	0.031	0.028	0.024
	100	0.485	**0.485**	0.487	0.496	0.518	***0.514***	0.517	0.519	0.023	0.022	0.021	**0.016**
	150	0.484	0.472	0.488	**0.471**	***0.505***	0.506	0.512	0.506	**0.014**	0.017	0.014	0.021
	200	0.478	**0.477**	0.479	0.479	***0.495***	0.495	0.499	0.499	0.014	**0.011**	0.012	0.014
4	50	0.693	0.692	0.693	**0.689**	0.738	***0.737***	0.738	0.737	0.026	0.026	**0.026**	0.027
	100	0.660	**0.658**	0.661	0.659	0.712	***0.711***	0.712	0.713	0.029	0.029	**0.028**	0.029
	150	0.718	0.718	0.720	**0.716**	0.738	***0.737***	0.739	0.738	0.011	0.011	**0.010**	0.011
	200	**0.672**	0.672	0.674	0.672	0.693	***0.693***	0.694	0.693	0.019	0.019	**0.018**	0.019
5	50	**0.516**	0.518	0.525	0.522	0.561	***0.553***	0.567	0.557	0.029	0.025	0.031	**0.019**
	100	0.502	**0.492**	0.493	0.500	0.523	***0.515***	0.523	0.523	**0.013**	0.014	0.017	0.016
	150	0.481	0.475	**0.461**	0.474	0.508	0.502	0.507	***0.500***	0.018	0.020	0.020	**0.018**
	200	**0.448**	0.467	0.455	0.463	0.482	0.484	***0.479***	0.484	0.015	**0.013**	0.013	0.014
6	50	0.458	**0.442**	0.461	0.450	0.487	***0.474***	0.489	0.479	**0.016**	0.023	0.019	0.023
	100	0.408	0.412	0.414	**0.395**	***0.445***	0.450	0.445	0.446	0.029	0.025	**0.022**	0.030
	150	0.396	0.396	0.390	**0.389**	***0.424***	0.425	0.430	0.429	0.024	**0.023**	0.028	0.027
	200	**0.357**	0.373	0.367	0.360	0.407	0.406	0.408	***0.404***	0.019	**0.017**	0.017	0.019
7	50	0.599	**0.579**	0.594	0.597	0.627	***0.623***	0.630	0.627	0.024	0.024	0.024	**0.023**
	100	0.549	**0.536**	0.547	0.545	0.567	***0.559***	0.571	0.566	**0.013**	0.015	0.013	0.015
	150	**0.490**	0.494	0.507	0.490	0.517	***0.515***	0.526	0.523	0.017	0.014	**0.012**	0.023
	200	0.491	**0.473**	0.484	0.481	0.519	0.517	***0.515***	0.518	0.028	0.029	0.028	**0.028**
8	50	0.539	**0.506**	0.537	0.526	0.590	***0.580***	0.596	0.591	**0.036**	0.042	0.037	0.039
	100	0.511	**0.493**	0.513	0.514	0.533	***0.527***	0.538	0.532	0.016	0.023	**0.016**	0.017
	150	0.466	**0.464**	0.465	0.474	***0.508***	0.508	0.511	0.511	0.022	0.021	0.023	**0.016**
	200	0.471	**0.470**	0.475	0.476	***0.482***	0.485	0.490	0.489	0.007	0.007	0.010	**0.006**

EK 16

1000 İterasyon n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	Oc_std	Oc_std	Oc_std	Oc_std
1	50	**0.479**	0.514	0.515	0.510	***0.536***	0.540	0.545	0.542	0.029	**0.020**	0.021	0.022
	100	0.472	**0.469**	0.477	0.477	***0.513***	0.515	0.522	0.519	0.029	0.029	0.032	**0.028**
	150	**0.481**	0.481	0.483	0.495	***0.511***	0.517	0.522	0.522	**0.015**	0.018	0.019	0.015
	200	0.482	0.482	**0.466**	0.480	***0.498***	0.501	0.504	0.504	**0.011**	0.011	0.016	0.014
2	50	0.518	**0.500**	0.523	0.501	0.544	***0.539***	0.550	0.543	0.024	0.026	**0.019**	0.024
	100	**0.484**	0.492	0.493	0.487	0.512	***0.511***	0.520	0.516	0.022	**0.020**	0.021	0.022
	150	0.481	0.486	**0.471**	0.474	***0.504***	0.507	0.514	0.509	0.013	**0.012**	0.019	0.016
	200	**0.474**	0.482	0.480	0.478	***0.495***	0.500	0.506	0.504	**0.011**	0.012	0.013	0.013
3	50	0.501	0.488	**0.484**	0.492	0.538	***0.532***	0.539	0.536	0.026	**0.026**	0.027	0.029
	100	0.483	**0.472**	0.492	0.484	***0.515***	0.517	0.522	0.516	0.022	0.025	**0.020**	0.023
	150	**0.473**	0.477	0.484	0.476	***0.501***	0.502	0.512	0.512	0.017	**0.014**	0.014	0.016
	200	**0.475**	0.478	0.476	0.476	***0.491***	0.495	0.500	0.499	0.013	0.015	**0.011**	0.015
4	50	0.689	**0.688**	0.692	0.692	***0.735***	0.735	0.737	0.736	0.027	**0.027**	0.027	0.027
	100	0.659	**0.658**	0.662	0.659	***0.710***	0.711	0.713	0.712	**0.028**	0.029	0.029	0.029
	150	**0.716**	0.718	0.717	0.719	***0.736***	0.737	0.737	0.738	0.011	**0.011**	0.011	0.011
	200	0.671	**0.670**	0.675	0.674	***0.692***	0.692	0.695	0.694	0.018	0.019	**0.018**	0.018
5	50	0.518	0.515	0.513	**0.509**	0.558	0.557	0.569	***0.549***	0.029	0.029	0.036	**0.026**
	100	**0.472**	0.498	0.481	0.500	***0.517***	0.524	0.522	0.524	0.019	**0.015**	0.020	0.015
	150	0.476	0.459	**0.451**	0.459	***0.494***	0.496	0.501	0.501	**0.014**	0.014	0.019	0.020
	200	0.456	0.467	0.451	**0.418**	0.478	0.485	0.474	***0.469***	0.012	0.011	**0.011**	0.019
6	50	0.441	**0.429**	0.458	0.446	0.477	***0.468***	0.487	0.479	0.025	0.025	**0.021**	0.023
	100	0.403	**0.385**	0.417	0.419	0.437	***0.435***	0.451	0.450	0.029	0.036	**0.024**	0.029
	150	**0.385**	0.397	0.400	0.391	***0.416***	0.425	0.433	0.426	0.026	0.027	**0.025**	0.027
	200	0.365	0.366	0.369	**0.356**	***0.395***	0.404	0.413	0.407	**0.015**	0.019	0.019	0.021
7	50	0.599	0.605	0.613	**0.596**	0.625	***0.625***	0.639	0.629	0.022	**0.020**	0.022	0.021
	100	**0.531**	0.541	0.547	0.543	***0.554***	0.564	0.570	0.567	0.014	0.012	**0.011**	0.017
	150	0.494	**0.488**	0.509	0.493	***0.513***	0.517	0.530	0.524	**0.013**	0.015	0.014	0.020
	200	**0.474**	0.474	0.485	0.495	***0.506***	0.516	0.522	0.525	0.029	0.028	**0.027**	0.027
8	50	0.537	0.529	**0.529**	0.534	0.594	***0.586***	0.602	0.601	0.037	**0.035**	0.041	0.036
	100	**0.480**	0.505	0.517	0.504	***0.524***	0.529	0.537	0.534	0.020	**0.016**	0.017	0.017
	150	**0.457**	0.463	0.476	0.470	***0.500***	0.503	0.515	0.511	0.021	0.019	**0.019**	0.020
	200	**0.465**	0.469	0.478	0.469	***0.480***	0.481	0.490	0.488	0.010	**0.005**	0.009	0.011

EK 17

1000 İterasyon 2n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	Oc_std	Oc_std	Oc_std	Oc_std
1	50	0.498	**0.471**	0.494	0.515	0.533	***0.532***	0.536	0.540	0.025	0.029	0.024	**0.020**
	100	**0.457**	0.465	0.468	0.460	***0.510***	0.511	0.518	0.516	0.028	**0.027**	0.029	0.032
	150	**0.479**	0.484	0.494	0.489	***0.512***	0.514	0.518	0.517	0.018	0.018	**0.016**	0.017
	200	**0.470**	0.487	0.488	0.483	***0.494***	0.498	0.504	0.502	0.014	**0.010**	0.011	0.011
2	50	**0.497**	0.499	0.505	0.498	0.536	***0.535***	0.544	0.536	**0.025**	0.027	0.025	0.029
	100	**0.480**	0.481	0.485	0.489	***0.504***	0.510	0.513	0.512	**0.022**	0.023	0.023	0.022
	150	**0.480**	0.483	0.490	0.489	***0.502***	0.507	0.512	0.509	**0.010**	0.013	0.014	0.013
	200	**0.473**	0.476	0.478	0.476	***0.495***	0.498	0.504	0.499	0.012	**0.011**	0.013	0.014
3	50	0.492	**0.485**	0.494	0.496	0.537	***0.531***	0.538	0.537	0.026	0.029	0.026	**0.025**
	100	0.468	**0.466**	0.482	0.483	***0.507***	0.511	0.517	0.517	0.024	0.026	0.028	**0.023**
	150	**0.471**	0.473	0.476	0.482	***0.500***	0.504	0.508	0.509	0.018	0.018	0.017	**0.016**
	200	0.466	**0.466**	0.478	0.476	***0.489***	0.494	0.499	0.498	**0.012**	0.014	0.015	0.015
4	50	**0.689**	0.692	0.693	0.690	***0.735***	0.735	0.737	0.736	0.027	0.026	**0.026**	0.027
	100	**0.657**	0.658	0.660	0.658	***0.710***	0.711	0.713	0.711	0.030	0.029	**0.029**	0.029
	150	**0.714**	0.717	0.719	0.718	***0.735***	0.737	0.738	0.737	0.012	0.011	**0.010**	0.011
	200	**0.670**	0.671	0.673	0.672	***0.691***	0.692	0.693	0.693	**0.018**	0.018	0.019	0.018
5	50	**0.493**	0.517	0.506	0.505	***0.544***	0.549	0.550	0.552	0.027	0.024	**0.024**	0.025
	100	0.479	**0.446**	0.486	0.488	***0.510***	0.513	0.517	0.520	**0.016**	0.027	0.018	0.016
	150	**0.463**	0.475	0.472	0.473	***0.495***	0.500	0.501	0.499	0.021	0.017	0.015	**0.015**
	200	0.470	0.435	**0.423**	0.450	0.475	0.475	***0.468***	0.471	**0.006**	0.016	0.021	0.013
6	50	**0.425**	0.435	0.455	0.437	0.472	***0.466***	0.480	0.475	0.026	0.026	**0.023**	0.026
	100	0.391	0.392	**0.387**	0.388	0.432	***0.432***	0.450	0.444	0.031	0.033	**0.030**	0.033
	150	0.388	0.390	0.399	**0.385**	***0.411***	0.417	0.427	0.424	**0.024**	0.028	0.026	0.026
	200	0.355	0.356	0.367	**0.340**	***0.391***	0.402	0.410	0.403	**0.018**	0.018	0.019	0.025
7	50	**0.589**	0.592	0.608	0.596	***0.616***	0.616	0.629	0.624	0.026	**0.020**	0.020	0.023
	100	**0.527**	0.536	0.543	0.540	***0.550***	0.556	0.563	0.558	0.014	0.015	**0.012**	0.014
	150	**0.480**	0.488	0.498	0.492	***0.499***	0.508	0.520	0.515	0.016	**0.015**	0.017	0.015
	200	**0.465**	0.480	0.481	0.477	***0.504***	0.509	0.517	0.513	0.029	0.029	**0.024**	0.028
8	50	**0.494**	0.524	0.533	0.531	0.581	***0.580***	0.594	0.591	0.046	**0.039**	0.039	0.039
	100	**0.486**	0.490	0.506	0.512	***0.515***	0.525	0.532	0.531	0.017	0.021	0.019	**0.015**
	150	0.460	0.465	0.474	**0.454**	***0.500***	0.500	0.510	0.501	0.020	0.022	**0.019**	0.023
	200	**0.461**	0.466	0.470	0.473	***0.475***	0.479	0.487	0.485	0.010	**0.007**	0.011	0.008

EK 18

1000 İterasyon 4n popülasyon için doluluk oranları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	DO_min	DO_min	DO_min	DO_min	DO_ort	DO_ort	DO_ort	DO_ort	Oc_std	Oc_std	Oc_std	Oc_std
1	50	0.501	**0.493**	0.519	0.502	0.533	***0.533***	0.546	0.540	0.023	0.026	**0.019**	0.022
	100	**0.456**	0.460	0.475	0.464	***0.504***	0.511	0.520	0.516	**0.027**	0.033	0.028	0.030
	150	0.479	0.485	0.488	**0.475**	***0.510***	0.513	0.518	0.517	0.019	**0.016**	0.017	0.019
	200	0.479	**0.477**	0.485	0.488	***0.495***	0.499	0.502	0.502	0.011	0.012	0.014	**0.010**
2	50	**0.497**	0.502	0.497	0.499	***0.535***	0.535	0.542	0.540	0.027	**0.026**	0.027	0.026
	100	**0.467**	0.482	0.484	0.491	***0.501***	0.510	0.514	0.514	0.024	0.022	0.021	**0.021**
	150	**0.471**	0.485	0.489	0.487	***0.500***	0.505	0.510	0.511	0.017	0.015	0.015	**0.014**
	200	0.472	**0.464**	0.476	0.479	***0.495***	0.498	0.505	0.503	**0.012**	0.015	0.015	0.012
3	50	**0.481**	0.485	0.496	0.496	0.534	***0.532***	0.540	0.534	0.027	0.027	**0.025**	0.026
	100	**0.475**	0.483	0.490	0.476	***0.509***	0.512	0.522	0.515	0.025	0.023	**0.022**	0.025
	150	**0.471**	0.473	0.487	0.476	***0.497***	0.504	0.513	0.508	0.017	0.018	**0.014**	0.016
	200	**0.469**	0.473	0.476	0.471	***0.489***	0.492	0.501	0.495	**0.012**	0.016	0.014	0.017
4	50	**0.689**	0.691	0.692	0.693	***0.735***	0.736	0.737	0.736	**0.026**	0.026	0.027	0.026
	100	**0.655**	0.658	0.660	0.659	***0.709***	0.711	0.712	0.712	0.029	0.029	**0.029**	0.029
	150	**0.714**	0.719	0.720	0.718	***0.735***	0.737	0.738	0.737	0.012	0.011	**0.010**	0.011
	200	**0.669**	0.669	0.672	0.672	***0.691***	0.691	0.693	0.693	0.019	0.019	0.019	**0.018**
5	50	**0.490**	0.503	0.526	0.514	0.546	***0.543***	0.563	0.548	0.031	0.032	0.025	**0.024**
	100	0.495	**0.492**	0.494	0.497	***0.510***	0.514	0.517	0.521	**0.013**	0.014	0.015	0.016
	150	0.450	0.458	0.473	**0.440**	***0.493***	0.498	0.499	0.495	0.018	0.018	**0.012**	0.023
	200	0.459	0.464	**0.417**	0.417	0.477	0.479	***0.462***	0.471	**0.012**	0.012	0.021	0.020
6	50	**0.412**	0.428	0.464	0.440	***0.460***	0.465	0.484	0.473	0.027	0.023	**0.019**	0.021
	100	0.392	0.396	**0.379**	0.416	***0.429***	0.440	0.444	0.449	0.030	0.027	0.032	**0.026**
	150	0.389	**0.387**	0.413	0.390	***0.408***	0.419	0.431	0.427	0.025	0.027	**0.024**	0.026
	200	**0.355**	0.361	0.358	0.358	***0.390***	0.401	0.412	0.404	**0.018**	0.018	0.021	0.022
7	50	**0.578**	0.585	0.592	0.598	***0.613***	0.614	0.627	0.626	0.027	0.023	0.024	**0.021**
	100	**0.506**	0.523	0.550	0.536	***0.542***	0.551	0.569	0.562	0.022	0.017	**0.013**	0.020
	150	**0.479**	0.485	0.497	0.494	***0.496***	0.511	0.518	0.515	0.017	0.017	0.019	**0.014**
	200	**0.461**	0.466	0.475	0.483	***0.496***	0.510	0.512	0.514	**0.027**	0.032	0.028	0.029
8	50	**0.529**	0.533	0.537	0.530	0.584	***0.571***	0.596	0.589	**0.037**	0.039	0.038	0.041
	100	**0.502**	0.504	0.510	0.510	***0.519***	0.525	0.536	0.530	**0.014**	0.017	0.020	0.014
	150	**0.438**	0.449	0.470	0.463	***0.491***	0.495	0.513	0.505	0.022	0.021	0.020	**0.018**
	200	**0.463**	0.471	0.467	0.475	***0.474***	0.482	0.488	0.485	0.008	0.007	0.012	**0.007**

EK 19

100 İterasyon n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	13.4	***13.2***	13.4	13.3
	100	**24**	**24**	**24**	**24**	26.8	***26.4***	26.7	26.7
	150	36	**35**	36	36	37	***36.5***	37	36.9
	200	**49**	**49**	50	50	51.6	***51.4***	51.9	51.6
2	50	**11**	**11**	**11**	**11**	14	***13.9***	14.3	14.2
	100	**22**	**22**	**22**	**22**	26.3	***25.8***	26.1	25.9
	150	**35**	**35**	**35**	**35**	37.9	***37.5***	38	37.8
	200	**47**	48	48	48	50.6	***50.2***	50.5	50.5
3	50	**11**	**11**	**11**	**11**	13.5	13.5	13.6	***13.4***
	100	**23**	**23**	**23**	**23**	26.5	***26.4***	26.6	26.5
	150	**35**	**35**	**35**	**35**	38.5	***38.2***	38.4	38.5
	200	47	**46**	47	**46**	51.2	***50.7***	50.9	50.9
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	115	117.4	***117.3***	***117.3***	117.7
5	50	7	7	**6**	**6**	8.4	8	***7.9***	8
	100	**13**	**13**	**13**	**13**	15.4	***14.8***	15.3	15.2
	150	**19**	**19**	**19**	**19**	21	***20.8***	21.1	20.9
	200	**26**	**26**	**26**	**26**	28.2	***27.9***	28.2	28.1
6	50	8	**7**	8	**7**	10.1	***9.7***	10.1	9.9
	100	**17**	**17**	**17**	**17**	19.6	***19.5***	19.6	19.7
	150	29	**28**	29	**28**	30.2	***29.7***	30.3	29.8
	200	37	**36**	37	37	38.5	***37.9***	38.3	38.2
7	50	**5**	**5**	**5**	**5**	7.7	***7.4***	7.7	7.5
	100	**10**	**10**	**10**	**10**	13.5	***13***	13.5	13.2
	150	15	**14**	**14**	**14**	16.2	***15.9***	16.1	***15.9***
	200	22	**21**	22	22	24.8	***24.4***	24.9	25
8	50	**8**	**8**	**8**	**8**	9.7	***9.5***	9.7	9.7
	100	**17**	**17**	**17**	**17**	19.6	***19.4***	19.7	***19.4***
	150	**21**	**21**	**21**	**21**	24.8	***24.4***	24.9	24.9
	200	**29**	**29**	**29**	**29**	30.2	***29.9***	30.1	30

EK 20

100 İterasyon 2n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	13.2	***13.1***	13.2	13.2
	100	**24**	**24**	**24**	**24**	***26.3***	***26.3***	26.5	26.5
	150	**35**	**35**	**35**	36	***36.4***	***36.4***	36.6	36.6
	200	**49**	**49**	**49**	**49**	51.5	51.2	51.5	***51.1***
2	50	**11**	**11**	**11**	**11**	***13.9***	14	14	***13.9***
	100	**22**	**22**	23	**22**	25.8	***25.7***	25.9	25.8
	150	**35**	**35**	**35**	**35**	37.4	***37.3***	37.5	***37.3***
	200	48	**47**	48	**47**	50.2	***50***	50.3	50.1
3	50	**11**	**11**	**11**	**11**	***13.4***	***13.4***	***13.4***	***13.4***
	100	23	**22**	23	23	26.5	***26.1***	26.3	26.4
	150	**35**	**35**	**35**	**35**	38	***37.8***	38.1	38
	200	**46**	**46**	**46**	**46**	50.6	***50.5***	***50.5***	50.6
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	**6**	**6**	7	**6**	7.8	***7.6***	7.9	7.7
	100	**13**	**13**	**13**	**13**	14.9	***14.8***	***14.8***	***14.8***
	150	19	**18**	**18**	**18**	20.7	20.5	20.7	***20.4***
	200	**26**	**26**	**26**	**26**	27.9	27.7	***27.6***	***27.6***
6	50	**7**	**7**	**7**	**7**	9.9	***9.7***	9.9	***9.7***
	100	**16**	**16**	**16**	17	***19.1***	19.3	19.3	19.5
	150	**28**	**28**	**28**	**28**	29.6	***29.4***	29.7	29.5
	200	**36**	**36**	**36**	**36**	37.8	***37.5***	37.8	37.6
7	50	**5**	**5**	**5**	**5**	***7.4***	***7.4***	7.5	***7.4***
	100	**10**	**10**	**10**	**10**	13	***12.8***	12.9	13
	150	**14**	**14**	**14**	**14**	15.8	***15.5***	15.8	15.8
	200	22	**21**	**21**	**21**	24.6	***24.2***	24.3	24.3
8	50	**8**	**8**	**8**	**8**	9.5	9.4	9.3	***9.2***
	100	**17**	**17**	**17**	**17**	***19.3***	19.4	19.4	***19.3***
	150	**21**	**21**	**21**	**21**	***24.3***	24.4	24.5	24.5
	200	**29**	**29**	**29**	**29**	30	***29.7***	29.9	29.8

EK 21

100 İterasyon 4n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	***13.1***	13.2	***13.1***	13.2
	100	24	**23**	24	**23**	***26.3***	26.4	26.5	26.4
	150	**35**	**35**	36	**35**	36.3	***36.2***	36.4	36.4
	200	**49**	**49**	**49**	**49**	51.3	***51.1***	51.2	51.3
2	50	**11**	**11**	**11**	**11**	***13.8***	***13.8***	***13.8***	***13.8***
	100	**22**	**22**	**22**	**22**	***25.7***	***25.7***	25.8	***25.7***
	150	**35**	**35**	**35**	**35**	37.5	***37.3***	37.4	***37.3***
	200	**47**	**47**	**47**	**47**	50.1	50.1	***49.9***	50.2
3	50	**11**	**11**	**11**	**11**	***13.4***	***13.4***	***13.4***	***13.4***
	100	23	23	**22**	23	26.5	***26.2***	26.4	26.4
	150	35	**34**	35	35	38	***37.9***	38.1	38
	200	**46**	**46**	**46**	**46**	***50.2***	50.4	50.4	50.3
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	7	**6**	**6**	**6**	7.8	***7.6***	7.7	***7.6***
	100	**13**	**13**	**13**	**13**	***14.7***	***14.7***	14.8	***14.7***
	150	**18**	**18**	19	**18**	20.5	20.6	20.7	***20.4***
	200	**25**	**25**	**25**	**25**	27.5	***27.2***	27.4	27.3
6	50	**7**	**7**	**7**	**7**	9.7	***9.6***	9.7	9.7
	100	**16**	**16**	**16**	**16**	19.3	***18.9***	19.1	19
	150	**28**	**28**	**28**	**28**	29.5	***29.4***	29.6	29.7
	200	**36**	**36**	**36**	**36**	37.7	***37.5***	37.6	37.8
7	50	**5**	**5**	**5**	**5**	***7.4***	***7.4***	***7.4***	***7.4***
	100	**10**	**10**	**10**	**10**	***12.8***	13	12.9	12.9
	150	**14**	**14**	**14**	**14**	15.9	***15.4***	15.9	15.5
	200	22	**21**	**21**	22	24.5	***24.1***	24.3	24.3
8	50	**8**	**8**	**8**	**8**	9.4	***9.1***	9.5	9.5
	100	**17**	**17**	**17**	**17**	19.3	***18.9***	19.5	19.4
	150	**20**	21	21	**20**	***24.2***	***24.2***	24.3	24.3
	200	**29**	**29**	**29**	**29**	29.9	***29.4***	29.6	29.7

EK 22

500 İterasyon n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	***13.1***	13.2	13.3	13.3
	100	**23**	**23**	**23**	24	26.3	26.3	***26.2***	26.3
	150	**35**	**35**	**35**	**35**	***36.3***	***36.3***	36.4	36.4
	200	49	49	**48**	49	51.2	***50.9***	51.2	51.2
2	50	**11**	**11**	**11**	**11**	13.8	***13.7***	13.9	13.8
	100	**22**	**22**	**22**	**22**	25.7	***25.5***	25.6	25.6
	150	35	**34**	35	35	37.3	***37.2***	37.5	37.5
	200	48	**47**	**47**	48	50.1	***49.7***	49.8	50.2
3	50	**11**	**11**	**11**	**11**	***13.4***	***13.4***	***13.4***	***13.4***
	100	**22**	**22**	23	**22**	26.3	26.4	26.4	***26.2***
	150	35	**34**	35	35	37.9	***37.7***	38.3	38.1
	200	**46**	**46**	**46**	**46**	50.2	***50.1***	50.5	50.3
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	**6**	**6**	7	**6**	7.7	7.7	7.7	***7.6***
	100	13	**12**	13	13	14.8	***14.7***	14.8	***14.7***
	150	19	**18**	**18**	**18**	20.5	20.5	***20.4***	***20.4***
	200	**25**	**25**	**25**	**25**	27.5	***27.2***	27.5	27.3
6	50	**7**	**7**	**7**	**7**	9.8	***9.5***	9.8	9.7
	100	**16**	**16**	17	**16**	***18.8***	19.1	19.4	19.2
	150	**28**	**28**	**28**	**28**	29.4	29.4	29.6	***29.3***
	200	**36**	**36**	**36**	**36**	***37.3***	37.5	38.1	37.8
7	50	**5**	**5**	**5**	**5**	***7.4***	***7.4***	***7.4***	***7.4***
	100	**10**	**10**	**10**	**10**	12.8	***12.7***	13	13
	150	**14**	**14**	**14**	**14**	***15.5***	***15.5***	15.8	15.8
	200	**21**	**21**	22	**21**	***24***	24.4	24.6	24.3
8	50	**8**	**8**	**8**	**8**	9.5	***9.2***	9.5	9.6
	100	**17**	**17**	**17**	**17**	19.2	***19.1***	19.3	19.3
	150	**21**	**21**	**21**	**21**	***24.1***	***24.1***	24.2	24.4
	200	**29**	**29**	**29**	**29**	29.6	***29.5***	29.9	29.6

EK 23

500 İterasyon 2n popülasyon için konteyner sayıları

			FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf		n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50		**10**	**10**	**10**	**10**	***13.1***	***13.1***	13.2	***13.1***
	100		**23**	**23**	24	24	26.2	***26.1***	26.4	26.3
	150		**35**	**35**	**35**	**35**	36.3	***36.2***	36.3	36.4
	200		**48**	49	49	49	***50.8***	51	51.2	50.9
2	50		**11**	**11**	**11**	**11**	***13.7***	***13.7***	13.8	***13.7***
	100		**22**	**22**	**22**	**22**	***25.5***	25.6	25.6	25.7
	150		**34**	35	35	35	37.1	***37***	37.5	37.2
	200		**47**	**47**	**47**	**47**	***49.6***	***49.6***	50	49.9
3	50		**11**	**11**	**11**	**11**	***13.4***	***13.4***	***13.4***	***13.4***
	100		**22**	**22**	23	23	***25.9***	26	26.3	26.4
	150		**34**	**34**	35	**34**	***37.6***	***37.6***	38	37.8
	200		**46**	**46**	**46**	**46**	***49.9***	50.1	50.2	50.3
4	50		**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100		**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150		**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200		**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50		**6**	**6**	**6**	**6**	7.7	***7.6***	7.7	***7.6***
	100		13	**12**	13	**12**	14.7	***14.6***	***14.6***	***14.6***
	150		**18**	**18**	**18**	19	***20.1***	***20.1***	20.5	20.3
	200		25	25	**24**	25	27.4	27.3	***27***	27.3
6	50		**7**	**7**	**7**	**7**	***9.4***	9.6	9.6	9.5
	100		**16**	**16**	**16**	**16**	18.8	***18.7***	19.2	19.2
	150		28	**27**	28	28	***29.1***	29.3	29.6	29.2
	200		**35**	36	36	36	***37***	37.3	37.7	37.5
7	50		**5**	**5**	**5**	**5**	***7.4***	***7.4***	***7.4***	***7.4***
	100		**10**	**10**	**10**	**10**	***12.7***	***12.7***	12.8	12.8
	150		**14**	**14**	**14**	**14**	***15.4***	15.5	15.6	***15.4***
	200		**21**	**21**	**21**	**21**	24	***23.9***	24.1	24.1
8	50		**8**	**8**	**8**	**8**	9.3	***9.1***	9.2	***9.1***
	100		**17**	**17**	**17**	**17**	19	***18.8***	19.3	19.2
	150		**20**	21	21	**20**	***24***	***24***	24.2	24.3
	200		**28**	29	29	**28**	***29.1***	29.3	29.6	29.6

EK 24

500 İterasyon 4n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	***13.1***	13.2	***13.1***	***13.1***
	100	**23**	**23**	**23**	24	***26.1***	***26.1***	26.3	26.3
	150	**35**	**35**	**35**	**35**	36.3	***36.2***	36.4	36.4
	200	49	**48**	49	49	51	***50.8***	51.1	51
2	50	**11**	**11**	**11**	**11**	13.8	***13.7***	13.8	13.8
	100	**22**	**22**	**22**	**22**	***25.4***	25.6	25.6	25.7
	150	**35**	**35**	**35**	**35**	37.2	37.2	37.4	***37.1***
	200	**47**	**47**	**47**	**47**	***49.6***	49.9	50.1	49.7
3	50	**11**	**11**	**11**	**11**	***13.4***	***13.4***	***13.4***	***13.4***
	100	23	**22**	23	23	26.2	***26***	26.1	26.3
	150	35	**34**	35	**34**	37.7	***37.6***	38	37.7
	200	**46**	**46**	**46**	**46**	50.1	***50***	50.4	50.3
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	**6**	**6**	**6**	**6**	***7.6***	***7.6***	***7.6***	***7.6***
	100	13	**12**	13	13	14.7	***14.6***	14.7	14.8
	150	**18**	**18**	19	**18**	20.5	***20.2***	20.4	20.3
	200	**25**	**25**	**25**	**25**	27.4	***27.3***	***27.3***	27.4
6	50	**7**	**7**	**7**	**7**	9.7	***9.6***	9.7	***9.6***
	100	**16**	**16**	**16**	**16**	***18.9***	19.1	19.1	***18.9***
	150	**28**	**28**	**28**	**28**	***29.1***	29.2	29.5	29.5
	200	**36**	**36**	**36**	**36**	37.4	37.4	37.5	***37.3***
7	50	**5**	**5**	**5**	**5**	7.4	***7.3***	7.4	7.4
	100	**10**	**10**	**10**	**10**	***12.7***	***12.7***	12.9	12.9
	150	**14**	**14**	**14**	**14**	***15.5***	***15.5***	15.6	15.7
	200	**21**	**21**	**21**	**21**	24.2	24.1	***24***	***24***
8	50	**8**	**8**	**8**	**8**	9.2	***9***	9.1	9.1
	100	**17**	**17**	**17**	**17**	19.1	***19***	19.2	19.1
	150	**21**	**21**	**21**	**21**	***24.1***	24.2	24.2	***24.1***
	200	**29**	**29**	**29**	**29**	***29.4***	29.5	29.5	29.6

EK 25

1000 İterasyon n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	***13.1***	13.2	13.2	***13.1***
	100	**23**	**23**	24	**23**	***26***	26.2	26.3	26.2
	150	**35**	**35**	**35**	**35**	***36***	36.2	36.4	36.4
	200	49	**48**	49	49	50.8	***50.7***	51	51.1
2	50	**11**	**11**	**11**	**11**	13.8	***13.7***	13.9	***13.7***
	100	**22**	**22**	**22**	**22**	25.6	***25.5***	25.7	25.7
	150	**34**	**34**	35	35	***37***	***37***	37.3	37.1
	200	**47**	**47**	**47**	**47**	***49.4***	49.6	50	49.9
3	50	**11**	**11**	**11**	**11**	13.4	13.4	***13.3***	13.4
	100	**22**	**22**	23	**22**	***25.9***	26.2	26.4	26.1
	150	**34**	**34**	35	35	***37.5***	***37.5***	38	37.9
	200	**46**	**46**	**46**	**46**	***49.8***	50.2	50.4	50.3
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	**6**	**6**	7	**6**	***7.6***	7.7	7.9	***7.6***
	100	**12**	13	**12**	13	***14.5***	14.8	14.7	14.8
	150	**18**	**18**	**18**	**18**	***20.1***	20.2	20.2	20.3
	200	25	26	25	**24**	27.1	27.5	27.1	***27***
6	50	**7**	**7**	**7**	**7**	9.6	***9.4***	9.8	***9.4***
	100	**16**	**16**	**16**	**16**	***18.7***	***18.7***	19	19
	150	**27**	28	28	28	***28.9***	29.3	29.6	29.4
	200	**35**	**35**	36	36	***37***	37.3	37.7	37.4
7	50	**5**	**5**	**5**	**5**	***7.4***	***7.4***	***7.4***	***7.4***
	100	**10**	**10**	**10**	**10**	***12.7***	12.8	13	12.9
	150	**14**	**14**	**14**	**14**	***15.4***	***15.4***	15.7	15.7
	200	**21**	**21**	**21**	22	***23.9***	24.2	24.2	24.5
8	50	**8**	**8**	**8**	**8**	9.3	***9.1***	9.3	9.3
	100	**16**	17	17	17	***18.9***	19.1	19.2	19.2
	150	**20**	21	21	**20**	***24***	***24***	24.3	24.2
	200	29	**28**	29	29	***29.2***	29.3	29.6	29.6

EK 26

1000 İterasyon 2n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	***13***	13.1	13.1	13.2
	100	**23**	**23**	**23**	**23**	25.9	***25.8***	26.3	26.3
	150	**35**	**35**	**35**	**35**	***36***	36.2	36.3	36.3
	200	**48**	49	49	49	***50.4***	50.7	51.1	51
2	50	**11**	**11**	**11**	**11**	***13.7***	***13.7***	***13.7***	***13.7***
	100	**22**	**22**	**22**	**22**	***25.3***	25.5	25.6	25.4
	150	**34**	**34**	**34**	35	***36.9***	37	37.2	37
	200	**47**	**47**	**47**	**47**	***49.4***	***49.4***	49.8	49.6
3	50	**11**	**11**	**11**	**11**	***13.4***	***13.4***	***13.4***	***13.4***
	100	**22**	**22**	**22**	23	***25.8***	***25.8***	26.2	26.1
	150	**34**	**34**	35	**34**	***37.5***	37.6	37.7	37.8
	200	**45**	**45**	46	46	***49.7***	50	50.4	50.3
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	**6**	**6**	**6**	**6**	***7.6***	***7.6***	***7.6***	***7.6***
	100	**13**	**13**	**13**	**13**	***14.6***	***14.6***	***14.6***	14.7
	150	**18**	**18**	**18**	**18**	***20.1***	20.2	20.4	20.3
	200	25	**24**	**24**	25	27.1	27.1	***27***	27.2
6	50	**7**	**7**	**7**	**7**	9.6	***9.4***	***9.4***	9.6
	100	**16**	**16**	**16**	**16**	***18.7***	***18.7***	19	18.8
	150	**28**	**28**	**28**	**28**	***28.9***	29.1	29.4	29.3
	200	**35**	36	36	**35**	***36.9***	37.3	37.6	37.3
7	50	**5**	**5**	**5**	**5**	***7.3***	7.4	7.4	7.4
	100	**10**	**10**	**10**	**10**	***12.7***	***12.7***	12.8	***12.7***
	150	**14**	**14**	**14**	**14**	***15.2***	15.3	15.4	15.4
	200	**21**	**21**	**21**	**21**	***23.8***	23.9	24	24
8	50	**8**	**8**	**8**	**8**	***9***	***9***	9.2	9.1
	100	**16**	**16**	17	17	***18.7***	18.9	19.1	19.1
	150	**20**	**20**	**20**	**20**	***23.9***	24	24.1	***23.9***
	200	**28**	**28**	29	29	***29.2***	***29.2***	29.4	29.4

EK 27

1000 İterasyon 4n popülasyon için konteyner sayıları

		FA	CS	TSO	HBA	FA	CS	TSO	HBA
Sınıf	n	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_min	Kon_no_ort	Kon_no_ort	Kon_no_ort	Kon_no_ort
1	50	**10**	**10**	**10**	**10**	***13***	***13***	13.2	13.1
	100	**23**	**23**	24	**23**	***25.8***	***25.8***	26.3	26.2
	150	**35**	**35**	**35**	**35**	***36.1***	***36.1***	36.4	36.4
	200	**48**	**48**	49	49	***50.5***	50.7	50.9	51.1
2	50	**11**	**11**	**11**	**11**	***13.7***	***13.7***	13.8	***13.7***
	100	**22**	**22**	**22**	**22**	***25.2***	25.4	25.6	25.6
	150	**34**	**34**	35	35	***36.8***	36.9	37.2	37.3
	200	**47**	**47**	**47**	**47**	49.4	***49.3***	49.9	49.8
3	50	**11**	**11**	**11**	**11**	***13.3***	13.4	13.4	13.4
	100	**22**	**22**	23	**22**	***25.9***	***25.9***	26.5	26
	150	**34**	**34**	35	35	***37.5***	37.7	38.1	37.9
	200	**45**	46	46	**45**	***49.6***	50	50.4	50.1
4	50	**27**	**27**	**27**	**27**	***29.4***	***29.4***	***29.4***	***29.4***
	100	**57**	**57**	**57**	**57**	***58.8***	***58.8***	***58.8***	***58.8***
	150	**85**	**85**	**85**	**85**	***87***	***87***	***87***	***87***
	200	**114**	**114**	**114**	**114**	***117.3***	***117.3***	***117.3***	***117.3***
5	50	**6**	**6**	**6**	**6**	***7.6***	***7.6***	***7.6***	***7.6***
	100	**12**	13	13	13	***14.5***	14.7	14.7	14.7
	150	**18**	**18**	**18**	**18**	***20.1***	***20.1***	20.3	20.2
	200	25	25	**24**	**24**	27.3	27.2	***26.7***	27
6	50	**7**	**7**	**7**	**7**	9.5	***9.4***	9.7	***9.4***
	100	16	16	**15**	16	***18.6***	18.9	19	19
	150	**27**	28	28	28	***28.8***	29.1	29.5	29.4
	200	**35**	36	36	**35**	***37***	37.3	37.7	37.3
7	50	**5**	**5**	**5**	**5**	***7.3***	7.4	7.4	7.4
	100	**9**	10	10	10	***12.4***	12.7	12.9	12.8
	150	**14**	**14**	**14**	**14**	***15.2***	15.4	15.4	15.4
	200	**21**	**21**	**21**	**21**	***23.5***	23.9	24	24.2
8	50	**8**	**8**	**8**	**8**	***9.1***	***9.1***	9.3	9.2
	100	**17**	**17**	**17**	**17**	***18.8***	19	19.3	19.1
	150	**20**	**20**	21	**20**	23.7	***23.6***	24.4	24
	200	**28**	29	29	29	***29.2***	29.4	29.6	29.5

ÖZGEÇMİŞ

Adı ve SOYADI	Ahsen KÜÇÜK
EĞİTİM DURUMU	
Mezun Olduğu Lise	Antalya Anadolu Lisesi Türkçe-Matematik (2005 – 2009)
Lisans Diploması	Dokuz Eylül Üniversitesi İktisadi ve İdari Bilimler Fakültesi Ekonometri Bölümü (2009 – 2013)
Yüksek Lisans Diploması	Dokuz Eylül Üniversitesi Sosyal Bilimler Enstitüsü Ekonometri Bölümü (2013-2016)
Yabancı Dil / Diller	İngilizce (Intermediate)
BİLİMSEL FAALİYETLER	
"Hemşire Çizelgeleme Problemlerinin Genetik Algoritmalarla Optimizasyonu ve Bir Uygulama", (İpek Deveci Kocakoç ile), Manisa Celal Bayar Üniversitesi Sosyal Bilimler Dergisi, 19 (Armağan Sayısı), 203-210.	