ETKİLİ BİR YAPAY ARI KOLONİSİ ALGORİTMASININ ÜÇ BOYUTLU KUTU DOLDURMA PROBLEMİ ÜZERİNDE UYGULANMASI

2021
DOKTORA TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ

TUĞRUL BAYRAKTAR

ETKİLİ BİR YAPAY ARI KOLONİSİ ALGORİTMASININ ÜÇ BOYUTLU KUTU DOLDURMA PROBLEMİ ÜZERİNDE UYGULANMASI

Tuğrul BAYRAKTAR

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Endüstri Mühendisliği Anabilim Dalında
Doktora Tezi
Olarak Hazırlanmıştır

Tez Danışmanları
Prof. Dr. Filiz ERSÖZ
Prof. Dr. Cemalettin KUBAT

KARABÜK
Eylül 2021

Tuğrul BAYRAKTAR tarafından hazırlanan "ETKİLİ BİR YAPAY ARI KOLONİSİ ALGORİTMASININ ÜÇ BOYUTLU KUTU DOLDURMA PROBLEMİ ÜZERİNDE UYGULANMASI" başlıklı bu tezin Doktora Tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Filiz ERSÖZ .....
Tez Danışmanı, Endüstri Mühendisliği

Prof. Dr. Cemalettin KUBAT .....
Tez Danışmanı, Endüstri Mühendisliği

Bu çalışma, jürimiz tarafından Oy Birliği ile Endüstri Mühendisliği'nde Doktora tezi olarak kabul edilmiştir. 07/09/2021

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. Emel KIZILKAYA AYDOĞAN (ERÜ)	.....
Üye : Prof. Dr. Filiz ERSÖZ (KBU)	.....
Üye : Prof. Dr. Cemalettin KUBAT (İGÜ)	.....
Üye : Prof. Dr. Oğuz FINDIK (KBU)	.....
Üye : Doç. Dr. Özer UYGUN (SAU)	.....
Üye : Doç. Dr. Safiye SENCER (SAU)	.....
Üye : Dr. Öğr. Üyesi Buket KARATOP (İUC)	.....

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

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

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

Tuğrul BAYRAKTAR

ÖZET

Doktora Tezi

ETKİLİ BİR YAPAY ARI KOLONİSİ ALGORİTMASININ ÜÇ BOYUTLU KUTU DOLDURMA PROBLEMİ ÜZERİNDE UYGULANMASI

Tuğrul BAYRAKTAR

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Tez Danışmanları:
Prof. Dr. Filiz ERSÖZ
Prof. Dr. Cemalettin KUBAT
Ağustos 2021, 135 sayfa

Yapay Arı Kolonisi (YAK) algoritması sayısal temelli optimizasyon problemleri yanı sıra tamsayılı optimizasyon problemlerinin çözümünde de yaygın bir şekilde kullanılmaktadır. Bununla birlikte Sırt Çantası Problemi (SÇP) gibi yüksek kompleksliğe sahip tamsayılı problemlerin kabul edilir ölçülerdeki çözümlerini mümkün olduğunca kısa sürede elde edebilmek için güçlü algoritmaların geliştirilmesine ihtiyaç duyulmaktadır. Literatürde SÇP'lerin çözümünde, keşif ve derinlemesine arama yapabilme kabiliyetine sahip YAK algoritması birçok çalışmada kullanılsa da YAK algoritmasının hammadde ve malzeme taşıma sistemlerinin en temel problemlerinden birisi olan üç boyutlu kutu doldurma problemlerinin çözümünde kullanıldığına az rastlanmaktadır. Bu çalışmada, YAK algoritmasının derinlemesine araştırma yönü Tabu Araştırması (TA) tabanlı bir hafıza mekanizması ile, keşif yönü ise Genetik Algoritmada (GA) kullanılan genetik operatörler ile ayrı

ayrı geliştirilerek farklı iki algoritma elde edilmiştir. Ayrıca her iki takviye yaklaşımın YAK algoritmasına bir arada entegre edildiği bir üçüncü algoritma da geliştirilmiştir. Geliştirilen üç algoritma kendi aralarında literatürdeki kurallara göre türetilen ve hali hazırda literatürde yaygın olarak kullanılan tekli ve çoklu konteyner yükleme problemi veri setleri üzerinde uygulanarak karşılaştırılmış, ayrıca algoritma üçlüsü arasında her iki takviye yaklaşımı da kullanan ve en iyi performansa sahip Bütünleşik Hibrit YAK (BHYAK) algoritması literatürdeki diğer yaklaşımlar ile karşılaştırılmıştır. Elde edilen sonuçlara göre hafıza mekanizması hibrit YAK algoritması geliştirmek için takviye yaklaşım olarak kullanıldığında, faydası uygulandığı problemin kompleksliğine ve çözüm süresi kısıtına göre değişiklik gösterdiği gözlemlenmiştir. Genetik operatörlerin ise hibrit YAK algoritması oluşturmada hafıza mekanizmasına göre daha tercih edilir olduğu tespit edilmiştir. Ayrıca, literatürdeki diğer yaklaşımlar ile karşılaştırıldığında tamamen rassal arama temelli bir yaklaşım olan BHYAK algoritmasının, nesneleri bir ön sınıflandırma işlemine tâbi tutan diğer yaklaşımlarınkine yakın bir performans gösterdiği tespit edilmiştir.

Anahtar Sözcükler : Yapay arı kolonisi algoritması, tabu araştırması, genetik algoritma, üç boyutlu kutu doldurma problemi, sırt çantası problemi

Bilim Kodu **:** 90610**,** 90619

ABSTRACT

Ph. D. Thesis

AN EFFECTIVE ARTIFICIAL BEE COLONY ALGORITHM FOR SOLVING THREE-DIMENSIONAL BIN PACKING PROBLEM

Tuğrul BAYRAKTAR

Karabuk University
Institute of Graduate Program
Department of Industrial Engineering

Thesis Supervisors:
Prof. Dr. Filiz ERSÖZ
Prof. Dr. Cemalettin KUBAT
August 2021, 135 pages

The Artificial Bee Colony (ABC) algorithm is widely used to achieve optimum solution in a short time in integer-based optimization problems. However, the complexity of integer-based problems such as Knapsack Problems (KP) requires robust algorithms to avoid excessive solution search time. ABC algorithm that provides both the exploitation and the exploration approach is used as an alternative approach for various KP problems in the literature. However, it is rarely used for the Three-Dimensional Bin Packing Problem (3DBPP) which is an important part of the transportation systems. In this study, the exploitation and exploration aspects of the ABC algorithm are improved by using memory mechanisms and genetic operators to develop two different hybrid ABC algorithms. The developed algorithms and the basic ABC algorithm are applied to a generated 3DBPP dataset to observe the effects of the memory mechanism and the genetic operators separately. In addition, a joint hybrid

artificial bee colony (JHABC) algorithm, that uses both reinforcement approaches, has been developed. All three developed algorithm with the basic ABC algorithm has been tested on single container problem and multiple container problem data sets that widely used in the literature, and the performances of two reinforcement approaches has been compared by the test results. JHABC algorithm also has been compared with approaches from the literature. The results show that, the effect of the memory mechanism on the hybridized ABC algorithm, depends on the complexity of the problem to which it is applied, and the solution search time constraint. On the other hand, genetic operators are more useful reinforcement tools than the memory mechanism as the complexity of the problem increases, according to the test results. Compared with the results in the literature, JHABC algorithm obtains promising solution search results for container loading problems by using only complete random search method.

Keywords **:** Artificial bee colony algorithm, tabu search, genetic algorithm, three-dimensional bin packing problem, knapsack problem

Science Code : 90610**,** 90619

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocalarım Prof. Dr. Filiz ERSÖZ ve Prof. Dr. Cemalettin KUBAT'a sonsuz teşekkürlerimi sunarım.

Başta babam olmak üzere sevgili aileme doktora süreci boyunca verdikleri manevi destek için ayrıca teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

g_i : nesne genişliği

d_i : nesne derinliği

y_i : nesne yüksekliği

G_i : konteyner genişliği

D_i : konteyner derinliği

Y_i : konteyner yüksekliği

a_i : tekli konteyner yükleme probleminde nesnenin konteynere atanma değişkeni

h_i : tekli konteyner yükleme probleminde atanan nesnenin hacmi

H : tekli konteyner yükleme probleminde konteynerin hacmi

k_i : çoklu konteyner yükleme probleminde i indisli konteynerin kullanılma değişkeni

c_i : çoklu konteyner yükleme probleminde kullanılan konteynerin maliyeti

y_{ij} : çoklu konteyner yükleme probleminde i indisli konteynere atanan nesne sayısı

n_j : konteyner yükleme probleminde yerleştirilebilecek toplam nesne sayısı

a_j : çoklu konteyner yükleme probleminde i indisli konteynere nesnelerin atanma değişkeni

h_{ij} : çoklu konteyner yükleme probleminde i indisli konteynere atanan nesnenin hacmi

H_i : çoklu konteyner yükleme probleminde i indisli konteynerin hacmi

x_i : problem çözümlerinden oluşan popülasyon içerisinde geliştirilecek çözüm

x_k : geliştirilmek üzere ele alınan x_i çözümünün komşu çözümü

x_j : x_k komşuluğunda x_i'den türetilen yeni çözüm

p_i : popülasyon içinde x_i çözümünün geliştirilmek üzere seçilme olasılığı

Φ : problem türüne göre değişen aralıktan seçilen rassal değer

$\oplus$: x_i çözümünün x_k çözümü ile etkileşimini yöneten xor operatörü

iw_p : yerel araştırmada tercih ağırlığı

dw_p : çözüm çeşitlendirme aşamasında tercih ağırlığı

SN : popülasyon büyüklüğü

Eval : çözüm değerlendirme sayısı

Itr : iterasyon

e : elit arı sayısı

m : hafıza büyüklüğü

L : ele alınan aynı çözümü geliştirememe limiti

T_0 : başlangıç sıcaklığı

T_{min} : son sıcaklık

σ_N : komşu çözüm sayısı

τ : soğuma derecesi

P_m : mutasyon olasılığı

KISALTMALAR

KYP : Konteyner Yükleme Problemi
SÇP : Sırt Çantası Problemi
KPP : Kesme ve Paketleme Problemi
GA : Genetik Algoritma
TB : Tavlama Benzetim
AA : Ağaç Araştırması
YAK : Yapay Arı Kolonisi
ÇKYP : Çoklu Konteyner Yükleme Problemi
TKYP : Tekli Konteyner Yükleme Problemi
TA : Tabu Araştırması
GYAK : Genetik Operatör Temelli Yapay Arı Kolonisi
HYAK : Hafıza Entegre Yapay Arı Kolonisi
BHYAK : Bütünleşik Hibrit Yapay Arı Kolonisi
ŞPP : Şerit Paketleme Problemi
KARP : Kapasiteli Araç Rotalama Problemi
TSBSKP : Tek Stok Boyutlu Stok Kesme Problemi
TKBKDP : Tek Kutu Boyutlu Kutu Doldurma Problemi
ÇSBSKP : Çoklu Stok Boyutlu Stok Kesme Problemi
ÇKBKDP : Çoklu Kutu Boyutlu Kutu Doldurma Problemi
ASKP : Artık Stok Kesme Problemi
AKDP : Artık Kutu Doldurma Problemi
ÖNPP : Özdeş Nesne Paketleme Problemi
TBNYP : Tek Büyük Nesne Yerleştirme Problemi
TSÇP : Tekli Sırt Çantası Problemi
ÇÖBNYP : Çoklu Özdeş Büyük Nesnelerin Yerleştirilmesi Problemi
ÇHBNYP : Çoklu Heterojen Büyük Nesnelerin Yerleştirilmesi Problemi
ÇÖSÇP : Çoklu Özdeş Sırt Çantası Problemi
ÇHSÇP : Çoklu Heterojen Sırt Çantası Problemi

AçA : Açgözlü Araştırma

ARAAP : Açgözlü Rassal Adaptif Araştırma Prosedürü

DM : Dizin Modifikasyonu

AGA : Amaç Güdümlü Araştırma

KE : Kademeli Evrim

DA : Dağınık Araştırma

YAA : Yerel Araştırma Algoritması

BM : Bulanık Mantık

ArıA : Arı Algoritması

KKO : Karınca Kolonisi Optimizasyonu

PSO : Parçacık Sürü Optimizasyonu

YSA : Yapay Sinir Ağları

EGA : Evrimsel Gelişim Algoritması

BY : Blok Yerleştirme

MBD : Maksimum Boşluk Doldurma

KY : Katman Yerleştirme

ABD : Alt Boşluk Doldurma

DÖ : Duvar Örme

KT : Köşe Tutma

KO : Kademeli Oyma

GK : Giyotin Kesme

YO : Yığın Oluşturma

SO : Sütun Oluşturma

İUY : İlk Uygun Yerleştirme

DDBDR : Düzgün Dörtgen Boşluk Doldurma Referansı

UNR : Uç Noktalar Referansı

NTA : Nesne Temas Alanı

DDS : Dip Derin Sol

RY : Rassal Yerleştirme

ÜBY : Üst Boyut Yerleştirme

ST : Satış Tahmini

PKÜ : Prototip Kolon Üretme

TP : Tamsayılı Programlama

KVTL : Kısa Vadeli Tabu Listesi

OVTL : Orta Vadeli Tabu Listesi

UVTL : Uzun Vadeli Tabu Listesi

NYD : Nesne Yerleştirme Dizinleri

FIFO : İlk Giren İlk Çıkar (First In First Out)

KDO : Konteyner Dolum Oranı

LSD : En Küçük Önemli Fark (Least Significantly Difference)

BÖLÜM 1

GİRİŞ

Konteynerler taşıma sistemlerinin başta gelen temel bileşenlerinden birisini teşkil etmektedir. Farklı taşıma noktalarına (depo, liman, perakendeci vb.) ticari nesnelerin (mal, malzeme, yük vb.) ulaştırılması için yaygın olarak kullanılan konteynerler genellikle standart boyutlara sahiptir. Konteynerleri yük taşıma amaçlı kullanma misyonuna sahip olan firmaların en temel amacı, uluslararası rekabet koşullarında taşınılan her çeşit ve boyuttaki maddelerin minimum maliyetle imalatçı, perakendeci, müşteri vb. bir sonraki teslimat noktasına aktarmaktır. Verilen taşıma hizmetinin kalite ölçüsünü belirleyen faktörlerin başında ise taşınan nesneyi teslim alan tarafın memnuniyeti gelmektedir. Memnuniyeti en üst seviyede sağlamak için ise taşınan maddeler hiçbir fiziksel, kimyasal veya biyolojik zarara uğramadan ve taraflar arasında üzerinde anlaşılan tedarik süresini aşmadan teslim edilmelidir.

Üç boyutlu nesnelerin yine üç boyutlu konteynerlerin içerisine yerleştirilmesi, bir tür kombinatoryal optimizasyon problemi olan Konteyner Yükleme Probleminin (KYP) konusu olup, Sırt Çantası Probleminin (SÇP) alt başlığı olarak değerlendirilmektedir. SÇP her türlü boyuttaki (bir, iki ve üç boyutlu) nesnelerin veya verilerin belli kapasitedeki alana yerleştirilmesinin genel bir problem başlığı olup, uzun süredir çeşitli bilim dallarından birçok araştırmada çalışılmış ve Lin tarafından yapılan derleme çalışmasında bunların SÇP ile ilişkileri incelenmiştir [1]. KYP problem çeşidi bakımından Kesme ve Paketleme Problemi (KPP) ile de ilişkilendirilmektedir. KPP, belli boyutlardaki alanların mümkün olan en fazla sayıda parçaya (nesne veya veri) bölünmesini problem edinmektedir. KPP sınıflandırması ile ilgili ilk çalışma Dyckhoff tarafından [2] yapıldıktan sonra Wascher tarafından bu sınıflandırma geliştirilmiştir [3].

KYP özelinde onlarca çalışma hakkında ilk derinlemesine sınıflandırma Bortfeldt ve Wascher [4] tarafından yapılmıştır. Bu sınıflandırmada ilgili KYP çalışmalarındaki

çözümleri etkileyen problem kısıtları incelenmiş, KYP için önerilen nesne yerleştirme yöntemi ile sezgisel çözüm yöntemleri hususunda genel bir değerlendirme yapılmıştır. Zhao ve ark. ise çözüm yöntemleri yönünden daha ileri bir irdelemeyle KYP için önerilen sezgisel yaklaşımları ayrıntılı şekilde sınıflandırarak performans bakımından karşılaştırmalarını yapmıştır [5]. Zhao ve arkadaşlarının yaptığı sınıflandırmada görüleceği üzere KYP için önerilen sezgisel yaklaşımlar arasında öne çıkanlar Genetik Algoritma (GA), Tavlama Benzetim (TB) ve Ağaç Araştırması (AA) olmuştur. Buna rağmen temelde SÇP için yaygın olarak kullanılan ve etkin çözümler sunan Yapay Arı Kolonisi (YAK) algoritması [6–8] KYP için önerilen sezgisel yaklaşımlar arasında yer almamaktadır. YAK algoritmasının Çoklu KYP (ÇKYP) için çözüm yaklaşımı olarak ele alındığı tek bir araştırma [9] mevcutken, Tekli KYP (TKYP) için çözüm yaklaşımı olarak kullanıldığı herhangi bir araştırma bilinen literatürde yer almamaktadır.

YAK algoritması sürü zekasına dayalı bir meta-sezgisel algoritma olup, çeşitli sürekli ve kesikli optimizasyon problemine etkin ve hızlı çözüm sunması yönüyle son yıllarda birçok araştırma tarafından sezgisel çözüm yaklaşımı olarak tercih edilmiştir. Temel YAK algoritması yanı sıra problem çözümü arayışını geliştirme maksadıyla birçok dönüştürülmüş YAK algoritması veya hibrit YAK algoritması önerilmiştir [10,11]. YAK algoritmasının keşif yönünü güçlendirmek maksadıyla GA'nın kullandığı genetik operatör (çaprazlama ve mutasyon) temelli YAK algoritmaları geliştirilirken [10,12,13], YAK algoritmasının yerel arama yönünü güçlendirmek maksadıyla Tabu Araştırmasından (TA) faydalanarak hafıza temelli YAK algoritmaları geliştirilmiştir [14–16]. Genetik operatörler ve hafıza mekanizması hibrit YAK algoritması geliştirilmesi hususunda takviye unsur olarak kullanılmasına rağmen, bu iki unsurun temel YAK algoritmasının performansına etkisinin karşılaştırıldığı bir çalışma literatürde yer almamaktadır. Bu eksikliğin sebeplerinden birisi olarak, bu iki takviye unsurunun literatürde hibrit YAK algoritması türetmede ayrı ayrı kullanılıp uygulandıkları problemi çözmek için kullanılan literatürdeki diğer çözüm yaklaşımlarıyla performans bakımından karşılaştırılmalarının ötesine geçilmemesi gösterilebilir.

Bu çalışmada YAK algoritmasının TKYP için çözüm sunma hususunda nasıl bir performans sergilediğinin incelenmesi yanı sıra, genetik operatör temelli YAK

(GYAK) ve hafıza entegre YAK (HYAK) algoritmalarının da TKYP için ne gibi çözümler sunduğu araştırılmıştır. YAK algoritması için her iki takviye yaklaşımının bir arada kullanıldığı bütünleşik bir hibrit YAK (BHYAK) algoritması da TKYP için bir çözüm yaklaşımı olarak ayrıca önerilmektedir. Bu şekilde TKYP üzerinden ilgili takviye yaklaşımlardan hangisinin YAK algoritmasının performansına daha fazla katkı sağladığı anlaşılabilecektir.

Bu çalışma genel itibariyle KYP ile ilgili literatür, YAK algoritması ile ilgili literatür, YAK algoritmasının KYP üzerinde uygulanabilmesi için kullanılan yaklaşım, özgün olarak geliştirilen hibrit YAK algoritmaları ve bu hibrit algoritmaların deneysel sonuçlarının incelendiği bölümler olmak üzere beş ana kısımdan oluşmaktadır. Literatür kısmı kendi içerisinde konteyner yükleme problemi ve yapay arı kolonisi algoritması başlıkları altında ikiye ayrılmaktadır. Metot kısmı ise konteyner yükleme yaklaşımı ve çözüm arama yaklaşımları başlıkları altında iki bölüm halinde incelenmektedir. Deneysel çalışma kısmı ise iki başlık altında sunulmaktadır. Birinci bölümde yapılan çalışma hakkında özet bilgi verilmiştir.

İkinci bölümde, çalışmaya konu olan KYP'nin tanımı, kısıtları ve türleri yanı sıra KYP çözmek için uygulanan nesne yerleştirme yaklaşımları ile çözüm araştırması için uygulanan sezgisel yaklaşımlar hakkında geniş bir literatür bilgisi yer almaktadır. Üçüncü bölümde ise TKYP'yi çözmek için önerilen YAK algoritması hakkında ayrıntılı bilgi verilmekte ve YAK algoritmasının TKYP'nin çatı problemi olarak düşünülebilecek olan SÇP üzerindeki uygulamaları örnek çalışmalarla anlatılmaktadır.

Çalışmanın dördüncü bölümünde, nesne yerleştirme sezgiseli ve YAK algoritmasının KYP için nasıl uygulanabileceği hakkında bilgi verildikten sonra önerilen üç farklı hibrit YAK algoritmasının çalışma mantığı beşinci bölümde anlatılmaktadır.

Altıncı bölümde, literatürde yaygın olarak kullanılan bir TKYP problem seti üzerinde önerilen üç farklı hibrit YAK algoritmalarının uygulanabilmesi için ön çalışma olarak uygun parametre değerleri belirlenip deneysel çalışmaya geçilmiştir. Deneysel çalışma sonucunda elde edilen çözüm değerleri, TKYP çözümü için temel alınan GA ve TA çözüm değerleri ile aynı çizelgede karşılaştırılmıştır. Takviye yaklaşım olarak

kullanılan genetik operatörler ve hafıza sisteminin iterasyonlara bağlı performansı, önerilen GYAK ve HYAK algoritmaları üzerinden karşılaştırılmak üzere çözüm arama süreci ve çeşitlilik grafikleri ile gösterilmiştir. Ayrıca temel YAK algoritması, GYAK, HYAK ve BHYAK algoritmalarının performansları bu grafikler aracılığı ile karşılaştırılmıştır.

Deneysel çalışma sonuçlarının özetlendiği ve bu sonuçlardan çıkarımda bulunulduğu yedinci ve son bölümde, önerilen algoritmaların TKYP için ne gibi çözümler sunabildiği ve kullanılan takviye yaklaşımların YAK algoritmasının performansını geliştirmesi hususundaki etkileri araştırılmış ve sonuçlar yorumlanmıştır.

BÖLÜM 2

KONTEYNER YÜKLEME PROBLEMİ

2.1. PROBLEMİN TANIMI

KYP, SÇP'nin alt problemlerinden birisi olup dıştan dışa genişlik (g_i), derinlik (d_i) ve yükseklik (y_i) ebatlarına sahip üç boyutlu düzgün veya düzgün olmayan nesnelerin yine dıştan dışa genişlik (G_i), derinlik (D_i) ve yükseklik (Y_i) ebatlarına sahip üç boyutlu düzgün veya düzgün olmayan konteynerin/konteynerlerin içerisine yerleştirilmesini konu edinmektedir [17]. Problem tanımına göre $g_i \times d_i \times y_i$ çarpımının eşit olduğu herhangi bir nesne hacmi olan h_i, $G_i \times D_i \times Y_i$ çarpımına eşit olan konteyner hacmi H_i'den büyük olamaz. Ayrıca hiçbir nesnenin g_i, d_i, y_i boyutlarından herhangi birisi yerleştirilecekleri konteynerin ilgili G_i, D_i, Y_i boyutlarının herhangi birisinden büyük olamaz [18].

KYP'de sadece tek bir konteyner ele alındığında temel amaç konteynerin içini nesneler ile mümkün olan en yüksek oranda doldurmak ve bu şekilde nesne başına düşen taşıma maliyetini minimize etmektir. Eşitlik 2.1'de de gösterildiği üzere tek konteyner için amaç konteyner yükleme yüzdesini maksimize etmektir.

$$Z = \ maksimum \sum_{i=1}^{I} \frac{a_i . h_i}{H} \tag{2.1}$$

$$kısıtlar \sum_{i=1}^{n} \ a_i . h_i \ \leq H \tag{2.2}$$

$$a_i \in [0, 1], i = 1, \dots, \text{I}.$$

Eğer birden fazla konteynerin doldurulması gerekiyorsa doldurma yüzdesi yanı sıra konteyner maliyeti de dikkate alınması gerekmektedir. Bu durumda Eşitlik 2.3'te

gösterildiği üzere amaç bütün nesneleri herhangi bir konteynere atadıktan sonra toplam konteyner yükleme maliyetini minimize etmektir [19].

$$Z = minimum \sum_{i=1}^{p} k_i . c_i \tag{2.3}$$

$$kısıtlar \sum_{i=1}^{I} k_i . y_{ij} \geq n_j, \qquad \forall j \tag{2.4}$$

$$\sum_{j=1}^{J} a_j . h_{ij} \leq k_i . H_i, \qquad \forall i \tag{2.5}$$

$$k_i \; ve \; a_j \in [0,1], i = 1, \dots, \text{I} \; ve \; j = 1, \dots, \text{J}.$$

Eşitlik 2.3'te k değişkeni i indisli konteynerin kullanılıp kullanılmayacağını, c değişkeni ise bu konteynerin kullanım maliyetini belirtmektedir. Eşitlik 2.4'te y_{ij} değişkeni i indisli konteynere yüklenen nesneleri göstermektedir. Buna göre bütün kullanılan konteynerlere yüklenen nesne sayısı en az n_j değişkeni ile gösterilen toplam nesne sayısı kadar olmalıdır. Eşitlik 2.5'te ise kullanılan her i indisli konteynerde, a_j değişkeni ile o konteynere atama durumu belirlenen nesnelerin toplam h_{ij} hacimlerinin ilgili konteynerin hacminden küçük olması gerektiğini belirtmektedir.

Konteyner ve nesne şekillerine göre KYP dört grup altında incelenebilir:

1. Düzgün dörtgen konteyner ve nesneler,
2. Düzgün dörtgen konteyner ve düzgün dörtgen olmayan nesneler,
3. Düzgün dörtgen olmayan konteyner ve düzgün dörtgen olan nesneler,
4. Düzgün dörtgen olmayan konteyner ve nesneler.

Düzgün dörtgen nesnelerin yine düzgün dörtgen konteynerlerin içerisine yerleştirilmesi problemi en yaygın çalışılan KYP türüdür. Bazı durumlarda ise düzgün dörtgen şekle sahip olmayan bir grup nesne konteyner içerisindeki boşluklardan daha iyi faydalanabilmek amacıyla bir araya getirilip dıştan dışa düzgün dörtgen şekle sahip nesne öbekleri oluşturulabilmektedir [20]. Nesnelerin bu şekilde öbekler halinde

konteynere yerleştirilmeleri yaklaşımı özellikle düzgün dörtgen olmayan ürünlerin taşınmasında kullanılabilmektedir [21]. Bu çalışmada düzgün dörtgen olan nesneler yine düzgün dörtgen olan konteyner içerisine, nesne yüzeyleri konteyner yüzeylerine paralel olacak şekilde yerleştirilmeye çalışılmıştır.

2.2. PROBLEMİN KISITLARI

Konteyner yükleme işlemi nesne yerleştirme yaklaşımlarını yönlendiren birtakım kısıtlar tarafından etkilenmektedir. KYP'nin en baştaki kısıtı konteynerin belli bir kapasiteye sahip olması ve nesnelerin ise daha küçük parçalara bölünmeden bütün haline konteyner içerisine yerleştirilmeleri gerekliliğidir. Bu temel kısıt yanı sıra nesnelerin konteynere yerleştirilmelerini kısıtlayan diğer faktörler kaynak kısıtı, geometrik kısıtlar, fiziksel kısıtlar, nesneler arası etkileşim kısıtı, zaman kısıtı, gönderi kısıtı ve çevresel vb. kısıtlar olarak sıralanabilir.

KYP'de nesne ve konteyner sayısı kaynak kısıtı olarak düşünülebilir. Çoğu çalışmada nesne sayısının bir veya daha fazla konteyneri doldurabilecek miktarda olduğu varsayılır. Eğer perakendeci gibi bir kaynaktan tedarik edilen nesne sayısı bir konteyneri dolduracak miktarda değilse, birden fazla perakendeci katlanılan gönderi taşıma maliyetini düşürebilmek tek konteyneri ortak kullanabilirler. Ayrıca perakendecinin de kendi içinde gönderi kısıtları mevcutsa bu konteyner yükleme işlemini daha karmaşık hale getirir [22,23]. Öte yandan yine çoğu çalışmada birden fazla konteynerin yüklenmesi gerekliliği halinde, problem karmaşıklığını artıran olursuz nesne yerleştirme çözümlerinin üretilmesini engellemek adına konteyner sayısı sonsuz olarak varsayılır. Yine de bir kısım çalışmada KYP'yi daha gerçekçi şartlar altında ele almak için konteyner sayısı da sınırlı tutulmuştur [24].

KYP iki temel geometrik kısıt içermektedir: Nesneler birbirinin sınırlarının içine giremez ve konteyner hacmini belirleyen sınırların dışına taşacak şekilde yerleştirilemez. Ancak halka şeklinde orta kısmında başka bir nesnenin yerleştirilmesine yetecek kadar boşluk bulunan nesnelerin mevcut olduğu durumda nesne sınırlarının birbiri içerisine girememe kısıtı ihlal edilmiş olmaz [20]. Ayrıca, konteynerin bir boyutunun hayali olarak sınırlandırılmadığının varsayıldığı Şerit

Paketleme Probleminde (ŞPP) referans alınan boyutun sınırının fazladan yerleştirilen nesneler tarafından ihlal edilmesi göz ardı edilebilmektedir [25].

Geometrik kısıtların yanı sıra fiziksel kısıtlar da nesnelerin konteyner içerisine nasıl yerleştirilmeleri gerektiğini belirlemektedir. Nesnelerin yük taşıma kapasitesi, çevrilebilir yön sayısı, diğer nesnelerle temas yüzeyinin oranının yanı sıra nesne yığının sabitlik durumu, konteynerin denge durumu ve alabileceği en fazla ağırlık kapasitesi [26] ile nesnenin kırılganlığı [27] ve nesne yığınının ağırlık merkezi [28] gibi fiziksel kısıtlar nesnelerin konteyner içerisine yerleştirilmelerindeki öncelikleri belirlemektedir.

Nesneler arası etkileşim bir diğer KYP kısıtı olarak değerlendirilebilir. Birbirini kimyasal veya başka yolla etkileyecek iki nesnenin aynı konteyner içerisine yerleştirilmesi gerektiğinde bu nesnelerin konteyner içerisinde mümkün olan en uzak noktalara yerleştirilmeleri gerekmektedir. Bu kısıtı göz önüne alan ender çalışmalardan birinde, Mustafee ve Bischoff konteynere yerleştirilecek olan zararlı etkiye sahip nesneleri sınıflandırarak konteynerdeki diğer nesnelerin zararlı yönde etkilenmelerini minimize eden bir konteyner yükleme simülasyonu yapmıştır [29].

KYP'de zaman kısıtı iki türlü ele alınabilir. Nesnelerin konteyner içerisine yerleştirilme süreci için gerekli minimum süre KYP için kısıtlayıcı nitelikte olabilir [30]. Öte yandan KYP için önerilen çözüm yaklaşımlarının optimum konteyner yükleme çözümünü elde edinceye kadar geçen süre bir diğer zaman kısıtı olarak değerlendirilebilir [31].

Nesne gönderim önceliği ise taşıma sistemleri içinde belki de en önemli kısıtlardan birisi olarak karşımıza çıkmaktadır. Özellikle Kapasiteli Araç Rotalama Probleminde (KARP) gönderim önceliği en başta dikkate alınan kısıttır. Bunun sebebi ise en kısa sürede ve en az nesne yükleme/boşaltma işlemi ile nesnelerin teslimatının yapılmak istenmesidir [32].

Çevresel faktörler son zamanlarda tüm taşıma sistemi problemlerinde olduğu gibi KYP için de bir diğer kısıt olarak öne çıkmaktadır. Konteynerin taşıma sistemlerinin

temel bileşenlerinden birisi olduğu kabul edildiğinde, depoların bulunduğu yerler, konteyner türü ve özellikleri ile gönderi teslimat rotası yüklenilen biyolojik veya kimyasal tehlike arz eden nesnelerin taşınmaları esnasında çevreye ne derecede zararlı olacaklarını belirlemektedir [33]. Bu bağlamda karbon emisyonu da son zamanlarda taşıma sistemleri için önem arz eden öncelikli kısıtlardan birisi olarak ele alınmıştır. Yüklenilen nesne öbeğinin toplam ağırlığı ve teslimat mesafesi konteyneri taşıyan aracın sebep olduğu karbon emisyonunu etkilemektedir [34].

Bu çalışmada sezgisel yaklaşım olarak kullanılacak olan YAK algoritmasının temel performansını ölçmek için nesne yerleştirme işlemi esnasından bütün kısıtlar göz ardı edilmektedir. Öte yandan, kısıt türü uygulanacak sezgisel yaklaşımın performansının diğer yaklaşımların performansından olumlu veya olumsuz yönde ayrışmasına sebep olabileceği için geometrik kısıtlar haricindeki kısıtlar bu sebeple ihmal edilmiştir.

2.3. PROBLEMİN TÜRLERİ

KYP temelde iki farklı problem türü altında incelenebilir. Bunlardan birinci minimizasyon problemi olup birden fazla konteynerin yüklendiği durumlarda konteyner sayısının minimize edilmesi bu türe örnek olarak gösterilebilir. İkincisi ise maksimizasyon problemi olup her bir konteyner içerisine yerleştirilen nesnelerin boşluk kullanma oranı, ağırlıkları ve kıymetleri gibi değerlerinin maksimize edilmesi de bu türe örnek teşkil eder [5]. KYP, konteyner sayısının minimize edilmeye çalışıldığı altı minimizasyon problemi başlığı altında aşağıdaki gibi incelenebilir;

1. *Tek Stok Boyutlu Stok Kesme Problemi (TSBSKP)*: Konteyner boyutları birbirine eş ve nesne boyutlarının çeşitliliğinin az olduğu durumlar için geçerlidir. Nesneler şekil bakımından neredeyse birbirinin aynısıdır. Bu problem türünü minimize etme süreci diğer problemlere göre daha kolay olmaktadır [35].
2. *Tek Kutu Boyutlu Kutu Doldurma Problemi (TKBKDP)*: Konteynerler yine birbirine eş fakat bu sefer nesne boyutlarının çok çeşitli olduğu durumlar için geçerlidir. Nesne boyutlarının dağılımının daha heterojen olması yönüyle TSBSKP'ye göre daha karmaşık bir problem yapısına sahiptir. Öte yandan her

iki problemde de birbirine eş konteynerler kullanıldığı için konteyner yükleme maliyeti ihmal edilmektedir [36].

3. *Çoklu Stok Boyutlu Stok Kesme Problemi (ÇSBSKP)*: Konteyner ve nesne boyutlarının çeşitliliğinin az olduğu durumlar için geçerlidir. Çeşitli boyutlardaki konteynerler nesnelerin yerleştirilmesi aşamasında minimum yükleme maliyeti sağlayan konteyner türünün kullanılması için seçenek sunmaktadır. Bu tür problemde boyutsal olarak kısıtlı çeşitteki nesneler en uygun konteyner çeşidi içerisine yerleştirilmeye çalışılır. Bu yaklaşımdaki temel amaç; problemi alt kısımlara bölerek basitleştirmek ve üçüncü boyutta çözülmesi zor olan herhangi bir KYP'yi önce alt boyutlarda çözerek üçüncü boyuttaki en uygun çözümü elde edebilmektir [19].
4. *Çoklu Kutu Boyutlu Kutu Doldurma Problemi (ÇKBKDP)*: Boyutsal bakımdan nesne çeşitliliğinin daha heterojen olduğu fakat konteyner çeşitliliğinin homojene yakın olduğu durumlarda geçerlidir. Nesneler fazla çeşitlilikten ötürü ÇSBSKP'de olduğu kadar kolayca gruplandırılıp bir araya getirilemezler. Bu tür problemlerde problemin alt kısımlara ayrılarak çözülmesi yaklaşımının uygulanması her ne kadar kolay olmasa dahi az da olsa boyutsal benzerlik taşıyan nesneler gruplandırılarak bloklar halinde konteynere yerleştirilebilirler. Böylelikle problemi alt kısımlara bölme yaklaşımından kısmen de olsa istifade edilmiş olur [37]. Bu problem türü en yaygın çalışılan KYP problemlerinden birisi olup, Çoklu Konteyner Yükleme Problemi (ÇKYP) adı altında da incelenebilmektedir [24,37,38].
5. *Artık Stok Kesme Problemi (ASKP)*: Nesne çeşitliliğin homojene yakın olduğu konteyner çeşitliliğin heterojen yapıda olduğu durumlarda geçerlidir. Bu problem türü ÇSBSKP ile benzerlik göstermekle beraber konteyner seçeneğinin fazlaca olması yönüyle ayrışmaktadır. Bu problem türünde nesnelerin yerleştirilebileceği en uygun konteyneri bulmak ÇSBSKP'ye göre daha kolay olmaktadır [39].
6. *Artık Kutu Doldurma Problemi (AKDP)*: Nesne ve konteyner çeşitliliğin heterojen yapıda olduğu durumlarda geçerlidir ve problem bu çeşitliliğe bağlı olarak son derece karmaşık bir hal alır. Bu tür problemi çözmek için deterministik modeller yeterli olmayıp, sezgisel yaklaşımların kullanılması gerekebilir [40].

Diğer yandan KYP, konteyner doluluk oranının maksimize edilmeye çalışıldığı yedi maksimizasyon problemi başlığı altında aşağıdaki gibi incelenebilir.

1. *Özdeş Nesne Paketleme Problemi (ÖNPP)*: Birbiriyle özdeş boyutta olan nesnelerin tek bir konteyner içerisine yerleştirildikleri durumlarda geçerlidir. Nesnelerin basit bir hesaplamayla yerleştirilebilmeleri yönüyle ÖNPP'nin en basit maksimizasyon problemi olduğu söylenebilir [41].
2. *Tek Büyük Nesne Yerleştirme Problemi (TBNYP)*: Boyutsal çeşitliliği homojene yakın olan nesnelerin tek bir konteynere yerleştirildiği durumlarda geçerlidir. Bu tür TSBSKP'nin konteyner seviyesindeki bir alt problemi olarak değerlendirilebilir. TBNYP'de konteyneri optimum şekilde yüklemenin en iyi yolu benzer nesnelerin bir araya getirilerek bloklar halinde yerleştirilmesidir. Konteyner yükleme işleminde küçük nesne bloklarının büyük nesne blokları üzerine yerleştirilmeleri gerekmektedir [42].
3. *Tekli Sırt Çantası Problemi (TSÇP)*: Tek konteyner içerisine boyutsal olarak heterojen çeşitliliğe sahip nesnelerin yerleştirilmeleri durumunda geçerlidir ve konteyner düzeyinde TKBKDP'nin alt problemi olarak değerlendirilebilir. Bu tür problemlerde çeşitliliğin fazla olmasından dolayı nesneler bloktan ziyade katmanlar şeklinde yerleştirilmesi halinde daha etkin çözümler elde edilebilir [43].
4. *Çoklu Özdeş Büyük Nesnelerin Yerleştirilmesi Problemi (ÇÖBNYP)*: Boyutsal çeşitliliği homojene yakın olan nesnelerin özdeş konteynerler içerisine yerleştirilmeleri durumunda geçerlidir. Bu tür problemde yapılan temel hesaplama belli boyutlardaki nesneler için gerekli olan en az konteyner sayısını bulmak şeklindedir [44].
5. *Çoklu Heterojen Büyük Nesnelerin Yerleştirilmesi Problemi (ÇHBNYP)*: Boyutsal olarak homojen dağılıma yakın nesnelerin yine boyutsal olarak homojen dağılıma yakın ya da heterojen boyut çeşitliliğine sahip konteynerler içerisine yerleştirilmeleri durumunda geçerlidir. Bu tür, kısmen ÇSBSKP ile benzerlik göstermektedir ve aynı şekilde çözüm için problemi alt problemlere bölme yaklaşımı kullanılabilmektedir [45].

6. *Çoklu Özdeş Sırt Çantası Problemi (ÇÖSÇP)*: Boyutsal çeşitliliği heterojen olan nesnelerin özdeş konteynerler içerisine yerleştirilmeleri durumunda geçerlidir ve bu şekliyle TKBKDP ile benzerlik göstermektedir. Aradaki fark ise bu türde nesneler konteynerlere dengeli olarak dağıtılarak kullanılan her konteynerde optimum doluluk oranı sağlamanın amaçlanmasıdır [27].
7. *Çoklu Heterojen Sırt Çantası Problemi (ÇHSÇP)*: Boyutsal çeşitliliği heterojen olan nesnelerin yine boyutsal olarak homojen dağılıma yakın ya da heterojen boyut çeşitliliğine sahip konteynerler içerisine yerleştirilmeleri durumunda geçerlidir. Bu tür, ÇKBKDP ile AKDP'nin bir arada çözülmeye çalışıldığı bir problem çeşidi olarak değerlendirilebilir. Bu yönüyle ÇHSÇP ileri derecedeki karmaşıklığı ile en gerçekçi problem türü olarak öne çıkmaktadır. ÇHSÇP ile alakalı çalışmalarda, önerilen yaklaşımın performansının ölçülmesinden ziyade problemin karmaşıklığının nasıl basite indirgenebileceği üzerine odaklanılmaktadır [37].

Bu çalışmada önerilen yaklaşımlar, performanslarının daha iyi test edilebilmesi için nesne çeşitliliğin azdan çoğa doğru değişiklik gösterdiği problem setleri üzerinde ayrı ayrı uygulanmaktadır. Önerilen yaklaşımların temel performanslarının incelenebilmesi için nesnelerin tek konteyner içinde maksimum alan kaplayacak şekilde yerleştirilmeleri üzerine odaklanılmaktadır. Çoklu konteyner yükleme problemi bu çalışmada test edilen ve sonuçlarının uzun sürelerde alınabildiği problem setine konu olan problemin mevcut karmaşık yapısını daha da karmaşık hale getireceği için tercih edilmemiştir. Bu yönüyle çalışmada referans aldığımız problem türü TSÇP olup, literatürde diğer adıyla tekli konteyner yükleme problemi (TKYP) olarak bilinmektedir.

2.4. NESNE YERLEŞTİRME YAKLAŞIMLARI

Bu çalışmada, üç boyutlu, düzgün, doğrusal ve dış bükey şekle sahip ham madde, malzeme, son ürün gibi bir noktadan diğerine taşınan şeylere nesne adı verilmiştir. Nesne yerleştirme yaklaşımları bir önceki alt başlıkta anlatılan problem türlerine göre değişiklik göstermektedir. Bazı yaklaşımlar birkaç farklı problem türüne birden uygulanabilirken bazıları ise problem türüne özel yaklaşımlar olarak geliştirilmiş

olabilmektedir. Nesne yerleştirme yaklaşımı seçimi yapılırken konteyner ve nesnelerin boyutsal çeşitliliği ve miktarı dikkate alınmalıdır.

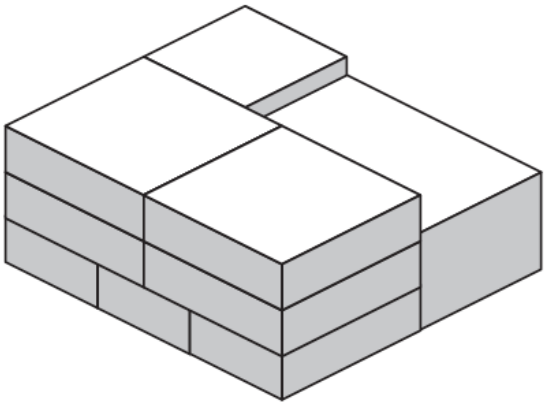
Nesne yerleştirme yaklaşımlarının temel amacı konteyner içerisindeki düzgün dörtgen alanlar ile en uygun nesneyi eşleştirebilmektir. Nesneler, en uygun yaklaşıma karar verilebilmesi için ilgili problem karakteristiğine göre sıralanarak konteyner yükleme sürecinden önce bir nesne dizini oluştururlar. Nesne yerleştirme yaklaşımları ele alınan problem için iki farklı nesne dizininden birisini kullanır:

1. Nesnelerin kıymet, miktar ya da boyutlarına göre önceden sıralanarak oluşturdukları sabit dizinler,
2. Herhangi bir nesne yerleştirildikten sonra konteyner içerisinde yerleştirilen nesneden arta kalan aday boşlukların ve yerleştirilmeyi bekleyen nesnelere ait bilgilerin sıralandığı dinamik dizinler [5].

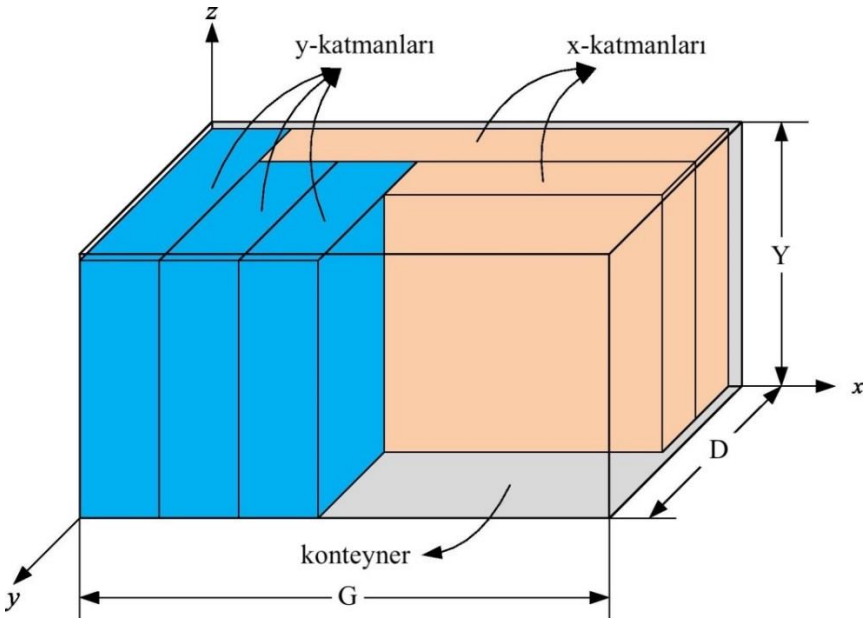
En çok uygulanan nesne yerleştirme yaklaşımları sırasıyla blok oluşturma, katman oluşturma, duvar örme ve giyotin kesmedir. Bunlarla birlikte, birçok çalışmada bu yaklaşımların elverişli yönlerinden faydalanmak için bir arada kullanıldığı nesne yerleştirme yaklaşımları da kullanılmaktadır [46–48].

Nesnelerin konteynere teker teker değil bir benzer olanların bir araya getirilerek bloklar halinde yerleştirilmesi yaklaşımı ilk olarak Eley [49] tarafından önerildikten sonraki yıllarda farklı şekillerde geliştirilmiştir. Blok oluşturma temel olarak, nesnelerin teslimat adresi gibi paylaşılan öncelikler, kırılganlık ve ağırlık taşıma kapasiteleri gibi fiziksel özellikler ve boyutsal benzerliklerine göre bir araya getirilip grup halinde konteynere yerleştirilmeleri yaklaşımıdır. Şekil 2.1'de nesne bloklarının nasıl oluşturulabileceği basit bir şekilde gösterilmiştir. Oluşturulan blokların dıştan dışa belli bir boyuttaki kübik şekle sahip olduğu varsayılmakta ve temel amaç olarak önceden ölçüleri belirlenen bu bloklar içerisine mümkün olan en fazla sayıda nesnenin yerleştirilmesi hedeflenmektedir [50]. Ayrıca bloklar başka nesne yerleştirme yaklaşımları açısından büyük ebatlı nesneler olarak değerlendirilebilirler [51,52].

Katman oluşturma, referans alınan bir nesnenin dikey katmanın derinliğini belirlediği ve referans nesneden sonra yerleştirilecek olan bir kısım nesnenin bu katman içerisine yerleştirilmeye çalışıldığı bir yaklaşımdır. Referans nesneye göre belirlenen katman diğer nesneler tarafından doldurulduğunda, yeni bir referans nesne ile birlikte yeni bir katman oluşturulur [53]. Şekil 2.2'de basit haliyle x ve y düzlemi boyunca katmanların nasıl oluşturulduğu gösterilmiştir [54].



Şekil 2.1. Blok oluşturma yaklaşımına göre nesnelerin bir araya getirilmesi



Şekil 2.2. X ve Y düzleminde katmanların basit görünümü

Katmanlara nesne yerleştirilmesi işlemi iki aşamada gerçekleşmektedir. Birinci aşama, yükleme öncesi süreç olarak değerlendirilir ve yığın, katman gibi sanal konteyner altı düzgün dörtgen alanların oluşturulmasını kapsar. İkinci aşamada ise bu sanal alanları optimum seviyede dolduracak en uygun nesneler seçilip yerleştirilir. Bununla birlikte

duvar örme yaklaşımı, konteyner yükleme öncesi sürecinin uygulanmaması yönüyle katman oluşturma yaklaşımına göre farklılık gösterir. Duvar örme yaklaşımında henüz konteynere yerleştirilmemiş olan nesneler, en küçük boyutları büyükten küçüğe sıralanacak şekilde bir sıra oluştururlar ve bir sonraki atamada sıradaki nesne yerleştirilir [55].

Giyotin kesme yaklaşımı, nesne yerleştirme yaklaşımlarının en eskilerinden birisi olarak bilinmektedir. Bu yaklaşım katman oluşturma yaklaşımı ile benzerlik gösterse de dikey katmanlar yanı sıra yatay katmanların da oluşturulmasını sağlaması yönüyle farklılık göstermektedir [56].

Yukarıda bahsedilen nesne yerleştirme yaklaşımları yanı sıra literatürde köşe tutma [26], kademeli oyma [57], yığın oluşturma [44], düzgün dörtgen paketleme alanı minimizasyonu [58], maksimum alan dokunuşu [32] ve nesne dizini modifikasyonu [59] vb. nesne yerleştirme yaklaşımları da yer almaktadır.

Nesne yerleştirme yaklaşımlarında en iyi nesne – boş dörtgen alanı çiftini belirleyebilmek için nesne ağırlıklandırma yöntemi kullanılabilmektedir [60]. Ağırlıklandırma süreci, KYP'de değişen problem tanımları ve kısıtlarına karşılık nesne, konteyner ve teslimat güzergâhı gibi problem elemanlarını göz önünde bulundurarak konteyner yükleme sürecinin bütünüyle sürdürülebilir hale getirilmesi için problem elemanlarını normalize edici bir görev üstlenmektedir [61]. Ele alınan KYP'de sürdürülebilirliği sağlayabilmek için, nesne – yerleştirme yüzeyinin pozisyonu, nesne – konteynerdeki aday boşluğun boyutu, nesne – konteynerde kullanılamayacak boşluklar [62], nesne – iki veya üç boyutlu düzlemde boşluk kullanma oranı [63] ve nesne – teslimat önceliği [52] eşleşmelerinden en uygun olanını seçmek gerekmektedir. Burada nesne ağırlıklandırma fonksiyonları, aday nesne ve boşluklarla alakalı en elverişli eşleşme kriterlerini belirleme rolü üstlenir. Bu fonksiyonlar, nesnenin veya konteynerin taşıma sistemi içerisindeki toplamsal veya çarpımsal etkisi dikkate alınarak oluşturulmalıdır.

Konteyner yükleme sürecinde genellikle teslimat uzaklığı, teslimat önceliği veya nesnelerin boyutları faktörlerinden bir veya birkaçını dikkate alarak nesneler

yerleştirilmek üzere sıraya konulur. Literatürde nesne – boşluk eşleşme kriterlerini dikkate alan birkaç çalışmada da gözlemlendiği üzere konteyner yükleme sürecinde ağırlıklandırma fonksiyonu değerlerine göre de nesne dizinleri oluşturulabilir [61,64,65]. Seçilen nesne – boşluk eşleştirme yaklaşımı aynı zamanda nesnelerin ağırlıklandırılmış değerlerini de etkileyebilmektedir [20].

Bu çalışmada önerilen çözüm geliştirme yaklaşımlarının performansı temel düzeyde ölçülmek ve sadece konteyner içindeki boşluktan maksimum seviyede faydalanılmak istendiği için nesne – konteyner eşleşmesi dikkate alınmamıştır. Bölüm 4'te de anlatıldığı üzere nesneler rassal olarak sıralanarak dizin oluşturmakta ve rassal oluşturulan bu dizine göre sırayla konteyner içerisine duvar örme yaklaşımıyla yerleştirilmektedirler.

2.5. ÇÖZÜM GELİŞTİRİCİ SEZGİSEL YAKLAŞIMLAR

KYP'nin ilk aşamasında ilgili problemin kısıtlarına göre hangi nesne yerleştirme yaklaşımının uygulanacağı belirlendikten sonra ikinci aşamada problem karakteristiğine göre rastgele veya belli bir kurala bağlı oluşturulan nesne dizinleri üzerinde değişiklikler yapılıp mevcut KYP çözümleri iyileştirilmeye çalışılır.

KYP ile alakalı ilk çözüm geliştirme sezgiselleri 1990'lı yıllarda çalışılmıştır. AA ve Açgözlü Araştırma (AçA) gibi temel sezgiseller muhtemel nesne yerleştirme şablonlarını değerlendirdikten sonra içlerinden en iyisini seçer. Morabito ve Arenales (1994) AA ve AçA sezgiselleri ile giyotin kesme yaklaşımına göre yerleştirilen nesneler için en iyi kesme şablonunu bulmak üzere katman yerleştirme, yığın yerleştirme ve giyotin kesme yaklaşımlarını bir arada kullanan bir yaklaşım önermektedirler. Buna göre, ilk olarak her bir nesne için alt ve üst hacimsel sınırları içeren giyotin kesme düğüm şablonları oluşturulur. Ardından AA mevcut çözümün alt sınırına ait şablona 0 üst sınırına ait şablona 1 ağırlık değerini vererek oluşturulan yeni düğüm şablonunun elde ettiği çözüm ile karşılaştırır. Eğer mevcut şablona ait sınırlar karşılaştırılan şablon sınırlarından daha iyi ağırlık değerine sahipse mevcut şablon daha iyisi bulunana kadar diğer şablonlar ile karşılaştırılır [66].

AA ve AçA'nın yanı sıra GA, TA ve TB gibi evrimsel gelişim temelli yaklaşımlar da KYP çözümü için uygulanan sezgisel yaklaşımlar arasında yer almaktadır. Komşu araştırması aşamasında gelişme sağlamayan çözümleri belli bir periyot boyunca yasaklayan, gelişme gösteren çözümlerin ise sonraki aramalarda tekrar referans çözüm olarak kullanılmak üzere öncelikli çözümler listesine yine belli bir periyot için kaydedip hafızaya alan TA [67], TKYP ve ÇKYP için verimli çözümler önerebilmektedir. Gendreau ve ark. (2006) iki aşamalı TA yaklaşımı kullanarak araç boyunu minimize edip dolu araç sayısını optimum sayıda tuttukları çalışma örnek olarak gösterilebilir. Dolu araç sayısını optimum hale getirebilmek için araçlar arasında nesne alışverişi yapıp, nesnelerin kendileri için en uygun araca yerleştirilmeleri sağlanmaktadır. TA ise nesne alışveriş hareketini listeleyip verimli nesne alışverişleri üzerinden optimum çözüme ulaşmaktadır. Önerilen yaklaşım literatürdeki gerçek verilere dayanan problemler üzerinde uygulandığında dikkat çekici sonuçlar elde ettiği görülmüştür [68].

KARP'deki nesne yerleştirme aşamasında ise konteyner yapısının yanı sıra nesnelerin teslimat adresleri de önem taşımaktadır. Ceschia ve Schaerf (2013), nesneler konteynere yerleştirildikten sonra mevcut yerleştirme düzeninden daha iyisini bulabilmek için komşu araştırması esnasında TA ve TB'yi bir arada kullanmaktadırlar. Gerçek verilere dayalı problem üzerine yapılan ilgili çalışmada temel amaç konteyner maliyetlerini daha da aşağıya çekebilmek için mevcut konteynerlere sığmayan nesne sayısını minimize ederken ortalama konteyner doluluk oranını maksimize etmektir. Bunun için TB, TA yapısında mevcut olan dinamik kısa dönem tabu listelerini kullanmak yolu ile komşu araştırması aşamasında gelişme sağlamayan çözümleri rassal uzunlukta bir periyot için hafızaya almaktadır [37].

Evrimsel gelişim mantığına göre belli bir grup çözümü nesil aktarımı yolu ile geliştirmeye çalışan popülasyon temelli GA, KYP için ilk olarak uzun süre önce çözüm önermesiyle beraber [26] halen yaygın olarak uygulanan [23,69–74] bir diğer sezgisel yaklaşımdır. TKYP için ise Huang ve ark. (2016) çalışması örnek olarak gösterilebilir. İlgili çalışmada konteyner gönderilerin dağıtılacağı perakendeci sayısına göre değişen uzunluklara sahip sanal alt konteynerlere bölünüp, bu uzunlukların minimize edilmesi sayesinde dağıtım yapılan perakendeci sayısı maksimize edilmeye

çalışılmaktadır. Bunun için ise, nesne ve nesnelerin konteyner içerisindeki pozisyonlarını dizinler halinde kodlayan çözümleri rassal olarak üreten ve bunları nesiller boyunca geliştiren GA, çözüm geliştirici sezgisel yaklaşım olarak uygulanmaktadır [75].

Çizelge 2.1'de KYP ile ilgili çalışmalar ve bu çalışmaların hangi tür KYP için hangi nesne yerleştirme yaklaşımı ve çözüm geliştirici sezgisel yaklaşım ile çözüm elde ettiği gösterilmektedir.

Çizelge 2.1 KYP ile ilgili yapılmış çalışmaların sınıflandırması

Yazar ve Yayın Yılı	Problem Türü	Nesne Yerleştirme Yaklaşımı	Çözüm Geliştirici Sezgisel Yaklaşım
Bischoff & Marriott – 1990[76]	TBNYP	KY	-
Morabito & Arenales – 1994[66]	TBNYP	KY, GK, YO, DDS	AçA, AA
Chen ve ark. – 1995[39]	ÖNPP	TP	-
Bischoff & Ratcliff – 1995[77]	TSBSKP, TBNYP	KY, DÖ	-
Bischoff ve ark. – 1995[42]	TSBSKP, TBNYP	BY	-
Gehring & Bortfeldt – 1997[78]	TSÇP	KT	GA
Raidl & Kodydek – 1998[79]	TKBKDP	İUY	GA
Chien & Wu – 1998[80]	TSÇP	TP	-
Ratcliff & Bischoff – 1998[81]	TBNYP	BY, KY	-
Raidl – 1999[82]	TKBKDP	İUY	AçA, GA
Scheithauer – 1999[83]	TSBSKP, TSÇP	KY	-
Davies & Bischoff – 1999[84]	TBNYP	MBD, KY	-
Bortfeldt & Gehring – 2001[85]	TSÇP	MBD	AçA, GA
Gehring & Bortfeldt – 2002[53]	TSÇP	KY	GA
Lins ve ark. – 2002[86]	TBNYP	BY, ABD, DÖ	DM
Hifi – 2002[87]	TBNYP	ABD, GK, YO, SO	AA, YAA
Pisinger – 2002[88]	TSÇP	GK, YO, SO	AA
Eley – 2002[89]	TSBSKP, TBNYP	BY	AçA, AA
Eley – 2003[90]	ÇKBKDP	TP	AçA, AA
Bortfeldt ve ark. – 2003[91]	TBNYP	KY	TA
Lins ve ark. – 2003[92]	TSÇP	GK	AA
Mau Yeh ve ark. – 2003[93]	TSÇP	KY	GA, YSA
Mack ve ark. – 2004[94]	TBNYP	BY	TA, TB
Lau ve ark. – 2004[95]	ÇHSÇP	DÖ	AA, YSA
Pimpawat & Chaiyaratana – 2004[96]	ÇÖSÇP	GK	GA

Çizelge 2.1. (devam ediyor)

Brunetta & Gregoire – 2005[97]	ÇSBSKP, ÇKBKDP	KY	AA
Lewis ve ark. – 2005[98]	TSÇP	MBD	GA
Moura & Oliveira – 2005[99]	TBNYP	MBD, KY	ARAAP
Yeung & Tang – 2005[100]	TSÇP	KY	GA
Birgin ve ark. – 2005[101]	ÖNPP	GK	AA
Takahara & Miyamoto – 2005[102]	ÇKBKDP	DÖ, YO	EGA
Tsai & Li – 2006[103]	TBNYP	TP	-
Bischoff – 2006[62]	TBNYP	KY, TP	-
Ertek & Kılıç – 2006[104]	ÇKBKDP	KY, GK	AçA, AA
Gendreau ve ark. – 2006[105]	TKBKDP	YO	TA
Takahara – 2007[106]	TKBKDP	MBD, İUY	AçA, YAA, TB
Bortfeldt & Mack – 2007[107]	ŞPP	KY, GK	GA, TA
Xue & Li – 2007[108]	TKBKDP	KY, DÖ	DA
Huang & He – 2007[109]	TSÇP	KT	AçA
Wang & Li – 2007[110]	ÖNPP	BY, KY	-
Nepomuceno ve ark. – 2007[111]	ÖNPP, ÇÖSÇP	KY, TP	GA
Kovacs & Beck – 2008[30]	ÇSBSKP	TP	-
Wang ve ark. – 2008[112]	TBNYP	MBD	AA
Parreno ve ark. – 2008[113]	TBNYP, TSÇP	BY, MBD	ARAAP
Soak ve ark. – 2008[114]	TKBKDP, ÇKBKDP	RY	YAA, EGA
Leung ve ark. – 2008[115]	ŞPP	ST	GA
Yan ve ark. – 2008[116]	TKBKDP	TP	-
Chang & Liao – 2008[117]	ÖNPP	YO, TP	-
Huang & He – 2009[118]	TSÇP	KO	AÇA
Tarantilis ve ark. – 2009[119]	TKBKDP	YO	AA, TA, YAA
Chien ve ark. – 2009[120]	TBNYP	DÖ, GK, YO	-
Moura & Oliveira – 2009[121]	TSBSKP	DÖ	ARAAP
Huang & He – 2009[122]	TBNYP, TSÇP	KO	-
Egeblad & Pisinger – 2009[123]	TSÇP	BY, DÖ	DM, TB
Christensen & Rousøe – 2009[124]	TBNYP	BY	AçA, AA
He & Huang – 2010[125]	TSÇP	KO	-
Parreno ve ark. – 2010[126]	TKBKDP	MBD	ARAAP
Dereli ve Das – 2010[127]	TSÇP	KY	TB
Almeida & Figueiredo – 2010[40]	ÇKBKDP, AKDP	KT	-
He & Huang – 2010[128]	ÇÖSÇP	KO	-
de Araujo & Pinheiro – 2010[129]	ÇÖSÇP	BY	GA, AA
Fanslau & Bortfeldt – 2010[130]	TBNYP, TSÇP	BY, GK	AA
Shiau & Lee – 2010[131]	ÇKBKDP	MBD, TP	AçA
Parreno ve ark. – 2010[132]	TBNYP	BY, MBD, KY	YAA
Ortmann ve ark. – 2010[133]	ÇKBKDP, AKDP	İUY, DDS	-
Torra ve ark. – 2010[134]	ÇÖSÇP	MBD	YAA, TB

Çizelge 2.1. (devam ediyor)

Dereli ve Das – 2010[135]	TBNYP	DÖ	ARIA
Egeblad ve ark. – 2010[136]	TSÇP	DÖ	AçA, AA
Kang ve ark. – 2010[137]	TSBSKP, TBNYP	BY	-
Cano & Torra – 2010[138]	ÇÖSÇP	MBD	PSO
He & Huang – 2010[139]	ÖNPP, TBNYP, TSÇP	KY, DÖ, KO	AçA
Wu ve ark. – 2010[140]	ÇHBNYP	BY, UNR	GA
Koloch & Kaminski – 2010[141]	TKBKDP, TSÇP	MBD	GA
Liu ve ark. – 2011[142]	ÇÖSÇP	BY, MBD	TA
Ren ve ark. – 2011[143]	TBNYP	BY	AçA, AA
He & Huang – 2011[144]	TSÇP	KT	AçA, YAA, ARIA
Dereli & Das – 2011[145]	TBNYP	DÖ	ARIA
Che ve ark. – 2011[146]	ÇSBSKP, ÇKBKDP	KY	ARAAP
Ma ve ark. – 2011[147]	TSBSKP	TP	TA, YAA
Junqueira ve ark. – 2012[148]	TSÇP	KY	-
He ve ark. – 2012[149]	TSBSKP	DÖ	GA
Zhang ve ark. – 2012[150]	ÇÖSÇP	KY	AA
Kang ve ark. – 2012[151]	ÇÖBNYP, MKP	DDBDR	GA
Wei ve ark. – 2012[152]	ÇÖSÇP	BY, KY	AçA, AA
Bortfeldt ve Jungmann – 2012[153]	ÇÖSÇP	GK	AA
Zhu & Qin ve ark. – 2012[154]	ÇKBKDP	NTA, DDS	TA
Lim ve ark. – 2012[155]	TSÇP	BY, KY	AçA
Burke ve ark. – 2012[156]	TSBSKP, TKBKDP, TBNYP, TSÇP	ÜBY	GA
Yap ve ark. – 2012[157]	ÇÖSÇP	YO	KKO
Liu ve ark. – 2012[158]	ÇÖSÇP	MBD	TA
Mack & Bortfeldt – 2012[159]	TSBSKP, TKBKDP	DÖ, GK, İUY	AA
Zhu & Lim – 2012[160]	TBNYP, TSÇP	BY	AA
Gonçalves & Resende – 2012[161]	TBNYP, TSÇP	KY	GA
Zhu & Huang ve ark. – 2012[162]	ÇSBSKP, ÇKBKDP	BY, PKÜ, TP	-
Allen ve ark. – 2012[163]	TSÇP	ABD, TP	-
Thapatsuwan ve ark. – 2012[164]	TKBKDP	KY, DÖ	GA
Zhu & Zhang ve ark. – 2012[165]	TKBKDP	İUY, UNR	-
Zhu & Oon ve ark. – 2012[166]	TBNYP, TSÇP	BY	ARAAP, AA
Junqueira ve ark. – 2012[167]	ÖNPP, TBNYP, TSÇP	MBD	-
Ceschia & Schaerf – 2013[168]	ÇKBKDP, ÇHSÇP	BY, KY	AA, TB
Bortfeldt – 2013[169]	TSÇP	DDBDR	GA, AA
Ho ve ark. – 2013[170]	ÇÖSÇP	KY	-
Junqueira ve ark. – 2013[171]	ÇHBNYP	TP	-
Mustafee & Bischoff – 2013[172]	TBNYP	MBD, GK	-
Hasni & Sabri – 2013[71]	TBNYP, TSÇP	KY	GA

Çizelge 2.1. (devam ediyor)

Alvarez-Valdes ve ark. – 2013[173]	ÇKBKDP	MBD	ARAAP
Tian ve ark. – 2013[174]	ÇKBKDP	MBD	AçA
Wang ve ark. – 2013[175]	TBNYP, TSÇP	BY	AA
Gonçalves & Resende – 2013[176]	TSBSKP, TKBKDP	MBD, DDS	GA
Lim ve ark. – 2013[177]	ÇÖSÇP	BY, KY, DÖ	ARAAP
Alonso ve ark. – 2014[178]	ÇÖSÇP	BY, MBD, KY, DÖ, GK, YO	ARAAP
Araya & Riff – 2014[64]	TSÇP	BY	AçA
Sheng ve ark. – 2014[179]	TSÇP	DÖ	AA
Ramos ve ark. – 2015[180]	ÇÖSÇP	MBD	-
Zheng ve ark. – 2015[181]	ÇÖSÇP	MBD	GA, BM
Wei ve ark. – 2015[182]	ÇSBSKP, ÇKBKDP	PKÜ, TP	AGA
Martinez ve ark. – 2015[183]	TBNYP, TSÇP	MBD	ARAAP
Li & Zhang – 2015[184]	ÇKBKDP	MBD	KE
Saraiva ve ark – 2015[28]	ÇKBKDP	MBD, KY	AçA
Karoonsoontawong & Heebkhoksung – 2015[185]	ÇHBNYP	DÖ	AA
Junqueira & Morabito – 2015[186]	TSBSKP, TKBKDP, TBNYP, TSÇP	KY, DÖ	TB
Feng ve ark. – 2015[74]	TKBKDP	DDS	GA
Escobar-Falcon ve ark. – 2015[187]	TBNYP, TSÇP	MBD, TP	ARAAP,
Zeineldin & Morsy – 2015[9]	ÇKBKDP	DÖ, KY	YAK
Iwasawa ve ark. – 2016[188]	ÇKBKDP	BY, GK	AçA
Wang ve ark. – 2016[189]	ÇSBSKP, ÖNPP	TP	-
Ramos ve ark. – 2016[190]	ÇÖSÇP	MBD	-
Huang ve ark. – 2016[191]	ÇÖSÇP	ABD	GA
McDonald – 2016[192]	ÇSBSKP	KY, TP	YAA
Gonzalez ve ark. – 2016[193]	TBNYP	GK	EGA
Brinker & Gündüz – 2016[194]	ÇKBKDP	DÖ, TP	-
Galrao Ramos ve ark. – 2016[195]	TKBKDP, TBNYP, TSÇP	MBD	GA
Jamrus & Chien – 2016[196]	TKBKDP	KY	GA
Tian ve ark. – 2016[197]	ÇKBKDP, ÇHSÇP	BY, MBD	AçA
Costa & Captivo – 2016[198]	TSBSKP	KT	-
Moura & Bortfeldt – 2016[199]	ÇÖBNYP	KY, DÖ	ARAAP, AA
Toffolo ve ark. – 2017[200]	ÇKBKDP	KY, YO, PKÜ	AçA
Correcher ve ark. – 2017[201]	ÇSBSKP, ÇKBKDP	KY, YO, SO, DDS	ARAAP
Araya ve ark. – 2017[202]	TSÇP	BY, MBD	AA

TSBSKP: Tekli Stok Boyutlu Stok Kesme Problemi, **TKBKDP:** Tekli Kutu Boyutlu Kutu Doldurma Problemi, **ÇSBSKP:** Çoklu Stok Boyutlu Stok Kesme Problemi, **ÇKBKDP:** Çoklu Kutu Boyutlu Kutu Doldurma Problemi, **ASKP:** Artık Stok Kesme Problemi, **AKDP:** Artık Kutu Doldurma Problemi, **ÖNPP:** Özdeş Nesnelerin Paketlenmesi Problemi, **TBNYP:** Tekli Büyük Nesnelerin Yerleştirilmesi Problemi, **TSÇP:** Tekli Sırt Çantası Problemi, **ÇÖBNYP:** Çoklu Özdeş Büyük Nesnelerin

Yerleştirilmesi Problemi, **ÇHBNYP:** Çoklu Heterojen Büyük Nesnelerin Yerleştirilmesi Problemi, **ÇÖSÇP:** Çoklu Özdeş Sırt Çantası Problemi, **ÇHSÇP:** Çoklu Heterojen Sırt Çantası Problemi, **ŞPP:** Şerit Paketleme Problemi, **BY:** Blok Yerleştirme, **MBD:** Maksimum Boşluk Doldurma, **KY:** Katman Yerleştirme, **ABD:** Alt Boşluk Doldurma, **DÖ:** Duvar Örme, **KT:** Köşe Tutma, **KO:** Kademeli Oyma, **GK:** Giyotin Kesme, **YO:** Yığın Oluşturma, **SO:** Sütun Oluşturma, **İUY:** İlk Uygun Yerleştirme, **DDBDR:** Düzgün Dörtgen Boşluk Doldurma Referansı, **UNR:** Uç Noktalar Referansı, **NTA:** Nesne Temas Alanı, **DDS:** Dip Derin Sol, **RY:** Rassal Yerleştirme, **ÜBY:** Üst Boyut Yerleştirme, **ST:** Satış Tahmini, **PKÜ:** Prototip Kolon Üretme, **TP:** Tamsayılı Programlama, **ARAAP:** Açgözlü Rassal Adaptif Araştırma Prosedürü, **AçA:** Açgözlü Araştırma, **DM:** Dizin Modifikasyonu, **AGA:** Amaç Güdümlü Araştırma, **GA:** Genetik Algoritma, **AA:** Ağaç Araştırması, **KE:** Kademeli Evrim, **DA:** Dağınık Araştırma, **TA:** Tabu Araştırması, **YAA:** Yerel Araştırma Algoritması, **TB:** Tavlama Benzetim, **BM:** Bulanık Mantık, **ArıA:** Arı Algoritması, **YAK:** Yapay Arı Kolonisi algoritması, **KKO:** Karınca Kolonisi Optimizasyonu, **PSO:** Parçacık Sürü Optimizasyonu, **YSA:** Yapay Sinir Ağları, **EGA:** Evrimsel Gelişim Algoritması

Çizelge 2.1'de de görülebileceği üzere KYP çözmek için en fazla uygulanan sezgisel yaklaşımların başında; GA, AA ve TB gelmektedir. Öte yandan KYP'de çok sık uygulanmasa da, etkili çözümler sunabilen algoritmaların da çalışmalarda yer aldığı görülebilmektedir. Zeineldin ve Morsy'nin YAK algoritması önerdiği çalışma da örnek olarak gösterilebilir. İlgili çalışmada sadece çözüm arama yaklaşımı dönüştürülmüş bir YAK algoritması, ÇKBKDP üzerinde uygulandığında bu problem türü için önerilen diğer temel sezgisel yaklaşımlara göre daha verimli sonuçlar elde ettiği görülmektedir [9].

YAK algoritmasının KYP türü problemlerin çözümünde etkili olduğundan hareketle bu çalışmada da KYP çözümü için hibrit bir YAK algoritmasının uygulanması önerilmektedir. Önerilen algoritma farklı iki sezgisel yaklaşımın çözüm arama sürecine olumlu katkı sağlayacak yönlerini barındırmaktadır. Ayrıca Zeineldin ve Morsy'nin ele aldıkları ÇKYP yerine bu çalışmada önerilen YAK algoritması TKYP üzerinde uygulanmakta ve bu problem türü için önerilen temel sezgisel yaklaşımlar ile karşılaştırılmıştır.

BÖLÜM 3

YAPAY ARI KOLONİSİ (YAK) ALGORİTMASI

Özellikle son çeyrek asırda hızla gelişen teknoloji ile büyümesini daha ileri boyutlara taşıyan endüstri kolları, geleneksel optimizasyon teknikleriyle çözülmesini zor kılan yeni tür imalat ve ürün taşıma problemlerini beraberinde getirmektedir. Alternatif yaklaşım olarak benzer problemlere getirilen çözümler ile elde edilen tecrübeler veyahut doğadan ilham alınarak geliştirilen algoritmalar bu tür problemler için iyi birer çözüm aracı olarak kullanılabilir [203].

Tüm tecrübe ve tabiat kaynaklı yaklaşımlar, optimum çözümü elde edemese de optimuma yakın faydalı çözümleri göreceli olarak daha kısa sürede yakınsayabilen sezgisel yaklaşımlara dönüştürülebilir. Bu bağlamda karınca ve arılar gibi sosyal iletişim içerisinde çalışarak besin arayan canlıların davranışları kompleks problemlerin çözümünü sağlayan algoritmaların geliştirilmesi için ilham kaynağı olabilmektedir. Bu yönüyle sürü zekâsı kavramı, etkin çalışabilen algoritmaların geliştirilebilmeleri için temel unsurlardan birisi olarak öne çıkmaktadır [204].

3.1. TEMEL YAK ALGORİTMASI

Doğada özellikle böcekler gibi küçük boyuttaki canlılar çevrelerindeki zorlukların üstesinden gelebilmek için birlikte hareket etmek zorundadırlar. Birey olarak başa çıkamayacakları tehlikeleri veya problemleri koloni halinde hareket ederek aşabilirler. Sürü içerisinde her bireyin üstlendiği farklı görevler sayesinde dışarıdan gelen etkilere karşı sürü bir bütün halinde tepkisel davranış sergileyebilmektedir. Aynı zamanda bireylerin üstlendiği görevler koloni içerisinde karşılıklı fayda sağlamayı da beraberinde getirecektir. En basitinden bireylerin keşfettiği besin kaynakları hakkında diğer bireyleri bilgilendirmesi sayesinde sürü halinde o besin kaynağından faydalanabilmektedirler. Bu yönüyle sürü oluşumu bu tür canlıların çevrelerindeki

kaynaklardan maksimum düzeyde faydalanıp aynı zamanda çevresel şartlara en güvenli derecede uyum göstermelerini sağlamaktadır [205].

Sürü zekâsına dayanan algoritmaların en büyük avantajı, bireylerin birçok noktada araştırma yapmaları yolu ile çeşitli bölgelerde ortaya çıkan engellerin ve faydalı kaynakların bilgisinin diğer bireyler ile paylaşılarak sürünün daha hızlı karar almasını sağlamaktır. Çok yönlü araştırma yapabilme kabiliyetine sahip sürü zekâsı ile birden fazla seçenek elde edilmekte ve bireyler bu seçenekler arasında en faydalı olanı birlikte çalışarak keşfetmektedirler. Seçeneklerin çeşitlendirildiği sürü zekâsına dayalı algoritmalar diğer sezgisel yaklaşımların çoğuna göre daha hızlı ve etkili sonuç elde edebilmektedirler. Ayrıca araştırmanın çeşitli yönlerde yapılması ile sürü zekâsına dayalı algoritmaların belli bir araştırma bölgesinde saplanıp kalmalarını da engellemektedir [205].

YAK algoritması da sürü zekâsına dayalı nöro-ilham bir algoritma olup bal arılarının besin bulma davranışlarını taklit etmektedir. YAK algoritması 2005 yılında Karaboğa tarafından geliştirilmiştir ve basit çözüm arama yönü sayesinde birçok sayısal ve tam sayı tabanlı optimizasyon problemi üzerinde kolaylıkla uygulanabilmektedir. Komşu arama yaklaşımı ile derinlemesine araştırma ve rassal arama mekanizması ile yeni çözümler keşfetme yönlerinin bir araya gelmesiyle etkin bir YAK algoritması meydana gelmektedir [206].

Diğer sürü zekâsına dayalı algoritmalarda çeşitli şekillerde olduğu gibi YAK algoritmasında da sürü için verimli kaynak arayan bireyleri bal arıları temsil etmektedir. Besin kaynaklarının da olurlu problem çözümlerini temsil ettiği düşünüldüğünde YAK algoritması temel olarak bal arılarının besin kaynaklarını keşfetmesi, tekrar ziyaret etmesi ve bu ziyaretlerde besin kaynağını değerlendirmesi şeklinde gelişmektedir. YAK algoritması bal arılarını *kâşif arı, işçi arı* ve *gözcü arı* olmak üzere üç görev altında sınıflandırır. Koloni için besin aramasına başlandığı ilk anda bal arılarının bir kısmı kâşif arı olarak görev alır ve rastgele olarak yeni besin kaynakları bulmaya çalışırlar. Kâşif arı adedince belli bir alt ve üst parametrik sınır arasında Eşitlik 3.1'deki hesaba göre rastgele belirlenen besin kaynakları YAK algoritması için başlangıç popülasyonu olarak belirlenir.

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

Eşitlikteki $i = 1, \dots, SN$ $i.$ besin kaynağını, SN ise araştırma yapılan alanda toplam besin kaynağı sayısını belirtmektedir. Öte yandan $j = 1, \dots, D$ ise $i.$ besin kaynağının belli bir alt ve üst limit arasında değişen $j.$ parametre değerini, D ise $i.$ besin kaynağının optimize edilmesi gereken toplam parametre sayısını belirtmektedir. Başlangıç aşamasında bütün besin kaynaklarının olumlu yönde geliştirilememe sayaçları $failure_i$ sıfırlanır.

Başlangıç popülasyonu için gerekli olan rastgele besin kaynakları belirlendikten sonra, kâşif arılar kovana dönerler ve ziyaret ettikleri besin kaynağının nektar miktarı, kalitesi ve koordinatı hakkındaki bilgileri *waggle dansı* aracılığı ile işçi arılara aktarır. *Waggle dansı* esnasında bilgiyi aktaran arı dansı ne kadar uzun sürdürürse ziyaret ettiği besin kaynağının o kadar verimli olduğunu anlatmaktadır. Arının dikey eksenle dansı icra ederken yaptığı açı güneş ile besin kaynağının kaç derecelik açı yaptığını gösterir. Böylelikle dansları takip eden işçi arılar hangi besin kaynağına gideceklerine karar verir [207]. Her bir işçi arı sadece tek bir besin kaynağını ziyaret eder ve kaynağın verimliliğini ölçtükten sonra mevcut besin kaynağı ile Eşitlik 3.2'ye göre komşusu olarak türettiği v_{ij} kaynağını karşılaştırır ve daha verimli olanı seçerek diğer besin kaynağı ile alakalı bilgileri hafızasından siler.

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \quad (3.2)$$

Eşitlik 3.2'de v_{ij} çözümü x_{ij} çözümünün komşusu olup, x_i 'nin $j.$ parametresinin rassal olarak değiştirilmesi yolu ile elde edilmektedir. Öncelikle v_{ij} türetebilmek için x_i'nin $j.$ parametresi ile popülasyon içerisinde x_i 'den farklı rassal olarak seçilen bir x_k çözümünün ($i \neq k \in 1, \dots, SN$) $j.$ parametresi belirlenir. Ardından iki j parametresi arasındaki fark [-1,1] arasında değişen rassal bir değer ile ağırlıklandırılarak mevcut x_i çözümünün $j.$ parametre değerine ilave edilir. Eğer elde edilen yeni $j.$ parametre değeri önceden belirlenen alt ve üst sınır değerlerini aşarsa Eşitlik 3.3'teki gibi normalize edilir.

$$v_{ij} = \begin{cases} x_j^{min} & , \quad v_{ij} < x_j^{min} \\ v_{ij} & , \quad x_j^{min} \leq v_{ij} \leq x_j^{max} \\ x_j^{max} & , \quad v_{ij} > x_j^{max} \end{cases} \tag{3.3}$$

Eşitlik 3.3'te görüleceği üzere eğer v_i çözümünün j. parametresi alt sınırın altında bir değer alırsa alt sınır değerini, üst sınırın üstünde bir değer alırsa da üst sınır değerini alarak normalize edilir. Türetilen yeni v_{ij} besin kaynağının nektar miktarının karşılığı olan performans değeri işçi arı tarafından Eşitlik 3.4'e göre hesaplanarak değerlendirilir.

$$fitness_i = \begin{cases} 1/(1+f_i) & , \quad f_i \geq 0 \\ 1 + abs(f_i) & , \quad f_i < 0 \end{cases} \tag{3.4}$$

Eşitlik 3.4'te $fitness_i$ değeri, $i.$ işçi arının x_i besin kaynağının verimliliğini ölçmesi ile elde edilir. Eşitlikteki f_i değeri ise x_i besin kaynağının, amaç fonksiyonu ile elde edilen geçiş değeridir $\left(f_i = F(\overrightarrow{x_i})\right)$ ve negatif veya pozitif değer alması durumunda eşitliğe göre normalize edilir. Eğer v_i komşu besin kaynağının performans değeri x_i besin kaynağının performans değerinden daha verimli ise v_i çözümü mevcut x_i çözümünün yerini alır. Daha verimli olanı ile değiştirilen çözüm ise hafızadan silinir ve geliştirememe sayacı $failure_i$ sıfırlanır. Eğer mevcut x_i çözümü geliştirilemezse geliştirememe sayacı $failure_i$ bir artırılır.

Besin kaynağı adedi kadar atanmış olan işçi arıların her biri sadece bir kaynağın etrafında komşu araması yaptıktan sonra kovana geri döner ve aynı şekilde gözcü arılara atanmış oldukları besin kaynaklarının koordinatları, verimliliği ve kovandan uzaklığı hakkında bilgi verir. Yine besin kaynağı adedince gözcü arılar ise birbirinden bağımsız olarak işçi arılardan aldıkları bilgilere göre besin kaynaklarını verimliliklerine göre değerlendirip, aralarından birisini seçerler. Besin kaynağı tercihi aşamasında işçi arıların aksine birden fazla gözcü arı bir kaynağı seçebilir. Bu tercih işlemi her ne kadar tabiatta bal arısının kararı ile neticelense de, YAK algoritmasında çözümler arasındaki tercih işlemi, ilgili çözümlerin verimliliğine bağlı değişen ve Eşitlik 3.5'e göre hesaplanan p_i olasılık değerinin rulet tekerleği üzerinde oluşturduğu

açılar arasından rassal olarak bir açı aralığına denk gelen çözümün seçilmesi yolu ile gerçekleştirilir.

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{3.5}$$

Eşitlik 3.5'e göre her bir çözümün p_i olasılık değerleri, x_i çözümünün performans değerinin bütün çözümlerin performans değerleri toplamına bölünmesi ile hesaplanır. Ardından her bir gözcü arı için [0,1] aralığında rassal değer üretilir. Hangi p_i olasılık değeri üretilen rassal değerden büyükse rulet tekerleğinde ilgili açıya denk düşen x_i çözümü ziyaret edilmek üzere gözcü arı tarafından seçilmiş olur. Gözcü arılar da tercih ettikleri x_i çözümleri çevresinde Eşitlik 3.2'ye göre komşu araştırması yapar. Eğer mevcut x_i çözümünü geliştirebilirse komşu v_i çözümünü hafızaya alır ve eski çözümü hafızasından siler ve çözüm geliştirememe sayacı $failure_i$ sıfırlanır. Aksi takdirde geliştirilemeyen çözümün sayacı $failure_i$ bir artırılır. Rulet tekerleğine göre besin kaynağına atama süreci bütün gözcü arılar atanana kadar sürdürülür.

Her iterasyon sonunda sayaçlar kontrol edilir. Eğer en yüksek sayaç değerine sahip $failure_i$ önceden belirlenen geliştirememe sınırına eşit veya ondan büyükse ve eğer popülasyon içerisinde en iyi performans değerine sahip değilse ilgili x_i çözümü hafızadan silinir ve x_i çözümünü ziyaret eden son gözcü arı kâşif arıya dönüşerek yeni bir rassal çözümü Eşitlik 3.1'e göre belirler. Her iterasyonda yalnız bir gözcü arının kâşif arıya dönüşmesine izin verilir. Algoritma 3.1'de temel YAK algoritmasının nasıl işlediği gösterilmektedir.

Algoritma 3.1: Temel YAK Algoritması

01	İlk popülasyonu oluşturmak için Eşitlik (3.1)'e göre SN adedince rastgele besin kaynakları türet
02	Bütün geliştirememe sayaçlarını sıfırla ($failure_i = 0$)
03	Toplam besin kaynağı değerlendirme sayısını $Eval_{max}$'ı belirle
04	Besin kaynaklarının geçiş ($f_i = F(\overrightarrow{x_i})$) ve verimlilik değerlerini hesapla
05	**while** $Eval < Eval_{max}$
06	**for** i = 1 to SN **do** **//İşçi Arı Aşaması**
07	Mevcut $\boldsymbol{x_i}$ besin kaynağı çevresinde Eşitlik (3.2)'ye göre $\boldsymbol{v_i}$ komşu besin kaynağını rassal olarak belirle

Algoritma 3.1 (devam ediyor)

08	$\boldsymbol{v_i}$ komşu besin kaynağının verimliliği $fitness(v_i)$'yi Eşitlik (3.4)'e göre hesapla
09	**if** $fitness(v_i) > fitness(x_i)$
10	x_i besin kaynağını v_i besin kaynağı ile değiştir
11	$failure_i = 0$
12	**else**
13	$failure_i = failure_i + 1$
14	**end if**
15	**if** $Eval \geq Eval_{max}$
16	En iyi verimliliğe sahip besin kaynağını hafızaya al
17	Araştırmayı bitir
18	**end if**
19	**end for** // **İşçi Arı Aşamasını Bitir**
20	Eşitlik (3.5)'i kullanarak her bir besin kaynağının $\boldsymbol{p_i}$ değerini hesapla
21	**for** t = 1 to SN **do** // **Gözcü Arı Aşaması**
22	**if** rassal sayı > p_i
23	Mevcut $\boldsymbol{x_i}$ besin kaynağı çevresinde Eşitlik (3.2)'ye göre $\boldsymbol{v_i}$ komşu besin kaynağını rassal olarak belirle
24	$\boldsymbol{v_i}$ komşu besin kaynağının verimliliği $fitness(v_i)$'yi Eşitlik (3.4)'e göre hesapla
25	**if** $fitness(v_i) > fitness(x_i)$
26	x_i besin kaynağını v_i besin kaynağı ile değiştir
27	$failure_i = 0$
28	**else**
29	$failure_i = failure_i + 1$
30	**end if**
31	$t = t + 1$
32	**end if**
33	**if** $Eval \geq Eval_{max}$
34	En iyi verimliliğe sahip besin kaynağını hafızaya al
35	Araştırmayı bitir
36	**end if**
37	**end for** // **Gözcü Arı Aşamasını Bitir**
38	En iyi verimliliğe sahip besin kaynağını hafızaya al
39	**if** $failure_{i_{max}} > limit$
40	x_i besin kaynağı yerine Eşitlik (3.1)'e göre yeni bir rassal besin kaynağı belirle
41	**end if**
42	**end while**
43	Araştırmayı Bitir

3.2. YAK ALGORİTMASI ve SÇP UYGULAMALARI

YAK algoritması sürekli problemlerin yanı sıra kesikli (tamsayılı) problemlerin optimizasyonu için de başvurulan bir meta-sezgisel yaklaşımdır. Literatürde genel itibariyle KYP üzerinde YAK algoritmasının uygulanması ile alakalı çalışmalar oldukça kısıtlıdır. Bilinen literatürde sadece Zeineldin ve Morsy'nin çalışmasında [9] YAK algoritmasını ÇKYP üzerinde uygulamıştır. Her ne kadar KYP üzerinde YAK algoritmasının uygulanması ile ilgili çalışmalar nadir olsa da KYP'nin çatı problemi olarak değerlendirilebilecek olan SÇP üzerinde YAK algoritmasının uygulanması ile ilgili birçok çalışma mevcuttur. SÇP için sezgisel yaklaşım olarak YAK algoritmasının önerildiği çalışmalar bu çalışmanın konusu olan YAK algoritmasının KYP üzerinde uygulanması hususunda yol göstermektedir.

YAK algoritması ile ilgili çalışmalara bakıldığında algoritma hususunda önerilen geliştirmeler algoritmanın parametre değerlerinin belirlenmesine, komşu çözüm belirleme yaklaşımlarına veya verimsiz çözümlerin nasıl eleneceğine yöneliktir. Pulikanti ve Singh başlangıç çözümlerini türetirken, tek boyutlu olan nesnelerin maddi değerleri ve ağırlıklarına göre değişen mutlak değer yoğunluklarını dikkate alarak çift halinde seçtikleri nesneler arasından birisi turnuva yöntemi ile mutlak değer yoğunluğuna bağlı değişen ikili olasılığa göre seçilmekte ve nesne yerleştirme dizinine eklenmektedir. Bununla birlikte, her defasında bir sonraki seçilen nesne dizine yerleştirilirken bütün dizinin olurlu çözüm sağlayıp sağlamadığı kontrol edilmektedir. Bu ise dizin oluşturma sürecini uzatabilmektedir. Öte yandan bu yaklaşım sayesinde ön eleme ile nesne yerleştirme dizinine seçilen nesnelerin oluşturacağı dizinler tamamen rassal olarak türetilen dizinlere göre daha hızlı bir optimum çözüm yakınsaması sağlanabilir [208].

Turnuva yöntemi, nesneler arasında tercih yapılması amacıyla uygulanması yanı sıra muhtemel çözümler arasında da seçim yapılması için uygulanabilir. Sundar ve arkadaşları gözcü arı aşamasında arıları besin kaynaklarına yönlendirirken rulet tekerleği yöntemi yerine turnuva yöntemini kullanmışlardır. Önerilen yaklaşıma göre çiftler halinde seçilen besin kaynakları arasından birisi verimliliklerine bağlı değişen ikili olasılığa göre seçilmekte ve besin kaynağına atanmaktadır. Rulet tekerleri

yöntemi ile kıyaslandığında turnuva yöntemini uygulamanın daha iyi performans gösterdiği belirtilmektedir. Çalışmada ayrıca diğerlerine göre daha verimli olan çözüm dizinlerindeki nesnelerin diğer dizinlerde daha fazla yer alması da sağlanmakta ve v_i çözümlerinin sadece parametre değerlerinin değiştirilerek değil dizinde olmayan nesnelerden rassal olarak seçilenlerin de yer alması yolu ile de türetilebildiği bir komşu arama yaklaşımı önerilmektedir [6].

Sundar ve Singh'in bir diğer çalışmasında ise gözcü arıların komşu araştırması için problem çözümlerine yine turnuva yöntemiyle atanmasından sonra mevcut çözümlerden yenileri türetilirken ÇSÇP'ye göre tasarlanan dizinler arasında sırt çantası alış-verişinin yapılması önerilmektedir. Buna göre bir çözüm çevresinde komşu çözüm türetirken başka bir çözümden rastgele bir sırt çantası içindeki bütün nesneler ile beraber mevcut çözüme eklenir ve aynı şekilde mevcut çözümdeki rastgele bir sırt çantası da sırt çantası alınan çözüme eklenir. Türetilen geçiş çözümünde nesne tekrarı oluşursa birisi dizinden çıkarılıp yerine dizinde bulunmayan nesnelerden rastgele birisi yerleştirilir. Önerilen bu yaklaşım ile de mevcut çözümlerden daha fazla çeşitte çözümler üretilerek optimum çözüm araştırmasının daha geniş alanda yapılması sağlanmaktadır [209].

Gözcü arıların besin kaynaklarına atanmaları turnuva yöntemi dışında rulet tekerleğine alternatif başka yöntemler ile de sağlanabilmektedir. Sabet ve arkadaşlarının çalışmasında dizinlerdeki nesneler maddi değer ve hacimlerine bağlı olarak değişen normalize değerlerine göre dizinlere seçilmektedirler. Gözcü arı aşamasında ise önerilen bir formülasyona göre arıların aldığı değer hangi verimlilik düzeyine sahip olan besin kaynağına atanacaklarını belirler. Formülasyonda önceden belirlenen eğilim katsayısı gözcü arıların çoğunluğunun daha verimli besin kaynaklarına mı yoksa daha verimsiz besin kaynaklarına mı gönderileceklerini belirlemektedir. Buradaki amaç, gözcü arıların tamamen verimli kaynaklara yönelip verimsiz kaynakların çevresindeki verimlilik potansiyeline sahip besin kaynakları göz ardı etme ihtimalini en aza indirmektir [210].

3.3. YAK ALGORİTMASI ÜZERİNDE YAPILAN DEĞİŞİKLİKLER

Temel YAK algoritması SÇP dışında birçok alanda uygulanabilmektedir. TSÇP dikkate alındığında, nesneler sırt çantasına atanırsa çözüm dizininde 1, atanmaz ise 0 değerini almaktadırlar. Buradan hareketle TSÇP ile ikili optimizasyon (binary optimization) problemlerinin birbiri ile benzeştiği düşünülebilir. Bu çalışmada da ele alınan problem TKYP olduğu için YAK algoritmasının ikili optimizasyon problemleri üzerindeki uygulamaları ve bu uygulamalar için YAK algoritmasının nasıl değiştirildiği önem arz etmektedir. Bölüm 3.2'de YAK algoritmasının SÇP üzerinde uygulandığı çalışmalarda daha çok gözcü arıların besin kaynaklarına nasıl dağıtılacaklarının belirlenmesi üzerinde yoğunlaşılmıştır.

Komşu araştırmasında hangi besin kaynağına ne kadar ağırlık verilmesi gerektiğinin yanı sıra komşu araştırması sürecinde yeni çözümlerin mevcut çözüme nasıl daha az benzer şekilde türetilebileceği meselesi de önem arz etmektedir. Kıran ve Gündüz'ün çalışmasında mevcut x_i çözümünden x_j komşu çözümünü türetmek için öncelikle popülasyondan rastgele seçilen x_k çözümü ile x_i çözümü arasındaki benzerliğe ve önceden belirlenen v katsayısına bağlı olarak değişen optimum değişim belirlenmektedir. Bu işlemden sonra ise x_i çözümünü oluşturan dizindeki nesne atanma indisini belirleyen sıfır ve bir sayıları belirlenen değişim değerlerine göre değiştirilerek yeni x_j çözümü oluşturulur. Önerilen komşu arama yaklaşımı sayesinde benzer olmayan çözümler türetilerek verimlilik potansiyeline sahip çözümlere daha hızlı yakınsama yapılabilmektedir. Benzeşmezlik (dissimilarity) hususundan ayrı olarak aynı çalışmada sürekli optimizasyon için geliştirilen Eşitlik 3.2, ikili (0-1) optimizasyonunda x_j türetmek için xor operatörü kullanılarak uyarlanmıştır [211]. Jia ve arkadaşları da komşu arama aşamasında yine xor operatörünü kullanmaktadırlar [212]. Bu çalışmada da Kıran ve Gündüz'ün kullandığı xor operatörü TKYP için tekrar uyarlanarak ele alınan probleme uygun şekilde x_j çözümleri türetilmektedir.

Çözümlerin benzeşmezliği ile ilgili bir diğer çalışmada Hançer ve arkadaşları optimum değişim değerlerinin, komşu araştırmasının hangi iterasyonda gerçekleştiğine bağlı olarak değiştiği bir formulasyon önermektedirler [213]. Öztürk ve arkadaşları ise optimum değişim değerleri için birden fazla alternatif bulunduğunda hangi değerlerin

seçilmesi gerektiği hususunda öneride bulunmaktadırlar [214]. Bu çalışmada ise mevcut çözüm uzayını genişletmek ve mevcut çözümlerden benzer çözümlerin tekrar tekrar türetilmesini engellemek için YAK algoritmasına genetik operatörlerin ve tabu araştırması temelli bir hafıza mekanizmasının takviye sezgisel yaklaşım olarak entegre edilmesi önerilmiştir.

3.4. HAFIZA ENTEGRE YAK ALGORİTMASI ÖRNEKLERİ

Yerel araştırma YAK algoritmasına önceden belirlenen geliştirememe limitine erişene kadar mevcut çözümlerin geliştirilebilmesi için tekrar tekrar deneme imkânı sağlamaktadır. Önceden belirlenen çözüm geliştirememe limiti aşıldıktan sonra geliştirilemeyen çözüm koloni hafızasından silinir ve yerine rastgele başka bir çözüm türetilir. Geliştirilemeyen çözüm hafızadan silinirken beraberinde önceki iterasyonlarda yerel araştırma aşamasında ilgili çözüm çevresinde verimli şekilde gerçekleşen geliştirme adımları da silinmektedirler. Geçmiş tecrübelerin bu şekilde silinmesini engellemek için çeşitli çalışmalar Tabu Araştırması (TA) yardımı ile YAK algoritmasına bir hafıza mekanizmasının entegre edilmesini önermektedirler [14–16].

TA mevcut çözümden türetilen yeni çözüme geçiş hareketlerini sınıflandırmak ve bu hareketlerin belli bir dönem boyunca ilgili listelerde saklanmasını sağlamak yolu ile çözüm araştırmasını yönlendiren bir stratejidir. Eğer geçiş hareketi mevcut çözümü geliştiren bir hareket değilse bu hareket belli bir dönem için yasaklanır ve Kısa Vadeli Tabu Listesi'ne (KVTL) kaydedilerek hafızaya alınır [215]. Öte yandan geçiş hareketi çözümü mevcut çözümü yerel araştırma aşamasında geliştirirse Orta Vadeli Tabu Listesi'ne (OVTL) kaydedilirken, global araştırma aşamasında belli bir kıstasa göre belirlenen verimlilik eşik değerinin aşılmasını sağlarsa Uzun Vadeli Tabu Listesi'ne (UVTL) kaydedilerek yine önceden belirlenmiş bir iterasyon sayısı için hafızaya alınır [67]. İleriki iterasyonların yerel ve global araştırma aşamalarında hafızaya alınan geçiş hareketlerinin yönlendirmesiyle optimum çözümün daha hızlı elde edilmesi hatta geliştirilmesi mümkün olabilmektedir.

TA stratejisi daha önceden mevcut çözüm üzerinde gelişme sağlayamamış geçiş hareketlerinin tekrar tekrar denenmesini engellemektedir. Li ve Yang başarılı geçiş

hareketlerindeki k ve Φ parametre çiftlerini kaydeden bir hafıza mekanizmasını YAK algoritmasına entegre etmektedirler. Eğer hafızaya alınan hareket bir sonraki kullanılışında tekrar başarılı olursa hafızada kalmaya devam eder. Aksi takdirde hafızadan silinir [15]. Chengli ve ark. da aynı mantık çerçevesinde başarılı olan geçiş hareketlerindeki $l_{best(i)}$, $k_{i(j)}$ ve $\Phi_{i(j)}$ parametre üçlülerinin daha sonra tekrar faydalanmak üzere hafızaya alınmasını önermektedirler. Bu yolla hafızaya alınan parametre değerleri çözüm araştırmasını yönlendirerek araştırmanın yerel optimum değerlerinden sakınmasını sağlamaktadırlar [16]. Li ve Yang'ın önerdiği hafıza mekanizmasının performansı hafıza listesinin genişliğine bağlı olmamakla beraber hangi parametrelerin hafızaya alındığından etkilenmektedir.

Yin ve Chuang ise teslimat önceliğine göre araç rotalarının optimize edildiği bir KARP üzerinden uyguladıkları YAK algoritmasına takviye olması için hafıza mekanizması önermektedirler. Yapılan çalışmada [14] problem kısıtlarının gevşetilmesiyle meydana gelen olursuz çözümlerin tekrar olurlu hale getirilerek, daha iyi çözümler elde edilmesi yaklaşımı olan stratejik salınımın, YAK algoritmasının hem işçi hem de gözcü arı aşamalarını kullanması sağlanmaktadır. Problem çözümlerinin salınım hareketi boyunca değişen çözüm değerleri hafızaya alınmaktadır. Eğer geçiş hareketi mevcut çözümü geliştirirse UVTL'ye, geliştiremez ise de KVTL'ye kaydedilir.

Hafızaya alınan hareketler çözüm araştırmasını faydasız geçiş hareketlerinin tekrar edilmesini azaltırken faydalı olanlarının daha fazla tekrar edilmesi şeklinde yönlendirmektedir. Önceki çalışmada, tek boyutlu KDP için faydasız geçiş hareketlerinden sakınmak üzere bu hareketlerin belli iterasyon sayısı boyunca KVTL'ye kaydedildiği bir hafıza entegre edilmiş YAK algoritması önermiştik. Önerilen hibrit YAK algoritması sayesinde mevcut optimum çözümün daha hızlı elde edilmesi sağlanmıştır [204].

3.5. GENETİK OPERATÖR ENTEGRE YAK ALGORİTMASI ÖRNEKLERİ

Genetik Algoritma (GA), bir çözüm popülasyonu içerisinde rastgele türetilmiş çözümlerin iterasyonlar boyunca kalıtsal mirasın aktarımı mantığına dayalı olarak geliştirilmesini sağlayan bir Evrimsel Gelişim (EG) algoritmasıdır. GA çaprazlama ve

mutasyon aracılığı ile problem çözümleri olarak temsil edilen popülasyondaki bireylerin performans değerlerini çeşitlendirmektedir. GA her ne kadar çözüm çeşitlendirmeye dayalı güçlü bir keşif yeteneğine sahipse de yerel araştırma aşamasında yeterli seviyede nokta odaklı araştırma yeteneğine sahip değildir. Bu yönüyle YAK algoritması, GA'nın çeşitli yerel optimum bölgelerindeki muhtemel global çözümleri elde edebilme şansını sağlamak için faydalı bir araç olarak kullanılabilir.

Singh ve ark. genetik operatörler tarafından türetilen çözümler çevresindeki en verimli performansa sahip olanlarının YAK algoritması operatörleri yardımı ile bulunmasını önermektedirler. Buna göre öncelikle popülasyondaki bir çözüm ve rastgele seçilen bir diğer çözümden sırasıyla çaprazlama ve mutasyon operatörleri kullanılarak oğul ve torun çözümler (off-springs) türetilmektedir. Ardından YAK algoritmasının işçi ve gözcü arı operatörleri genetik operatörler ile türetilen oğul ve torun çözümler çevresinde yeni çözümler türetmek üzere iki aşama halinde sırasıyla kullanılmaktadır. Türetilen çözümler arasından performans bakımından en iyi olanı mevcut çözümden daha verimli ise ilgili çözümün yerini alır. Singh ve ark. tarafından elde edilen sonuçlara göre önerilen bu yaklaşımın uygulandığı bütün vakalarda GA'ya göre daha verimli olduğu ortaya konulmaktadır [11].

GA her zaman mevcut çözümlerden yeni çözümler türetme işlemi tamamlandıktan sonra en verimsiz fazlalık çözümleri popülasyondan elemek yolu ile gelişim sağlamaktadır. Diğer taraftan YAK algoritmasında ise eğer bir çözüm etrafında ondan daha verimli başka bir çözüm elde edilirse daha verimli olan çözüm o anda mevcut çözümün yerini alır. Yani YAK algoritmasında geliştirilen çözümler GA'nın aksine iterasyonun sonlanması beklenmeden hafızadan silinirler. Ayrıca geliştirememe sayacı en yüksek olan ve geliştirememe sınırını geçmiş çözüm tamamen hafızadan silinerek yerine rastgele başka bir çözüm türetilir. Bundan dolayı YAK algoritmasında terk edilen çözümler, başka çözümlerin geliştirilmesi ile türetilmişlerse bile verimli yönde yapılan bu ilerlemeler de hafızadan silinmiş olur ve bu bilgiler sonraki iterasyonlara aktarılamaz.

Öztürk ve ark. genetik operatörleri YAK algoritmasının hem işçi hem de gözcü arı aşamalarında kullanmaktadırlar. Öncelikle herhangi bir mevcut çözüm olan x_i'nin rastgele seçilmiş olan komşuları x_j ve x_k, kolonideki en iyi çözümü temsil eden x_{best} ve sıfır çözümü temsil eden x_{zero} belirlenir. Ardından çaprazlama operatörü belirlenen bu beş çözüm arasında rastgele seçilen çözüm çiftlerinden oğul çözüm türetmek için kullanılır. Son aşamada ise türetilen oğul çözümlerden torun çözümler türetmek için mutasyon operatörü kullanılır. Belirlenen beş çözüm ve bu beş çözümden türetilen çözümler arasından en iyi olanı eğer mevcut çözüm olan x_i'den daha verimli ise onun yerini alır [10]. Panahi ve Navimipour da Öztürk ve ark. tarafından önerilen bu yaklaşımı, basit ve ikili optimizasyon problemlerinin geneli için kolaylıkla uygulanabilir olması sebebiyle çalışmalarında kullanmaktadırlar [13].

Mevcut çözüm x_i çevresinde komşu çözüm olan x_j, $x_i \neq x_k$ olmak şartıyla x_i'nin aynı popülasyon içinde başka bir x_k çözümü ile genetik operatörlerin kullanıldığı etkileşiminden türetilebilmektedir. Chaurasia ve ark., Öztürk ve ark.'nın genetik operatörler aracılığı ile türetilmiş çözümler arasından en iyi çözümü seçme yaklaşımı yerine direkt olarak x_j çözümünü x_i ve x_k çözümleri arasındaki genetik operatör temelli etkileşimden elde etme yaklaşımını önermektedirler. İlk olarak x_i ve x_k'dan çoklu nokta yerleştirme metodu ile x_j geçiş çözümü elde edilir. Ardından üç nokta takas metodu ile de x_j geçiş çözümü komşu çözüm olarak son halini alır. Chaurasia ve ark. önerdikleri bu yaklaşım ile YAK algoritmasının yerel araştırma odaklı yönünü kullanarak aynı çalışmada önerilen bir diğer yaklaşım olan hibrit GA'ya göre daha verimli sonuçlar elde etmektedirler [12].

Bu çalışmada ise YAK algoritmasının hem nokta odaklı araştırma hem de keşif yönünün geliştirilmesine yönelik bir hibrit YAK algoritması önerilmektedir. Önerilen yaklaşıma göre genetik operatörler ve hafıza mekanizması ayrı birer takviye araçları olarak belirlenmiştir. Bu takviye araçları arasından birisinin seçilmesi için öncelikle bu araçlar ağırlıklandırılıp, mevcut çözümlerin geliştirilmesine olan katkılarına göre puanlandırılmakta ve bu puanlamalar ile ilgili takviye aracının ağırlığı değişmektedir. YAK algoritması ise yerel ve global araştırma aşamasında takviye araçlarından birisini ağırlığını da göz önünde bulundurarak rastgele seçmektedir.

Ayrıca takviye aracı olarak kullanılan genetik operatör ve hafıza mekanizmasının YAK algoritması üzerindeki etkilerini ölçmek ve bu iki takviye aracını karşılaştırmak maksadıyla sırasıyla genetik operatör entegre YAK (GYAK) algoritması, hafıza entegre YAK (HYAK) algoritması TKYP üzerinde ayrı ayrı uygulanmaktadır. Son olarak her iki takviye aracının bir arada YAK algoritmasına entegre olarak kullanıldığı bütünleşik bir hibrit YAK (BHYAK) algoritması da TKYP üzerinde uygulanmaktadır.

BÖLÜM 4

KONTEYNER YÜKLEME PROBLEMİ ÜZERİNDE YAK ALGORİTMASININ UYGULANIŞI

4.1. PROBLEM ÇÖZÜMLERİNİN TEMSİL EDİLME ŞEKİLLERİ

Bölüm 3'te belirtildiği üzere temel YAK algoritması yapısal olarak sayısal temelli optimizasyon problemlerinin çözümüne uygun durumdadır. Bununla birlikte, YAK algoritması birçok çalışmada tamsayılı problemlerinin de çözülebilmesini sağlayacak şekilde dönüştürülmüş ve ilgili problemlere uygun çözüm üretme mekanizmaları geliştirilmiştir [6,10,14].

Temel YAK algoritması ile KYP'nin nasıl çözülmesi gerektiğinden önce popülasyon çözümlerinin ne şekilde temsil edileceği önem arz etmektedir. Bu çalışmada, TKYP ve ÇKYP için ayrı ayrı çözüm temsilleri önerilmektedir.

4.1.1. Tekli Konteyner Yükleme Problemi

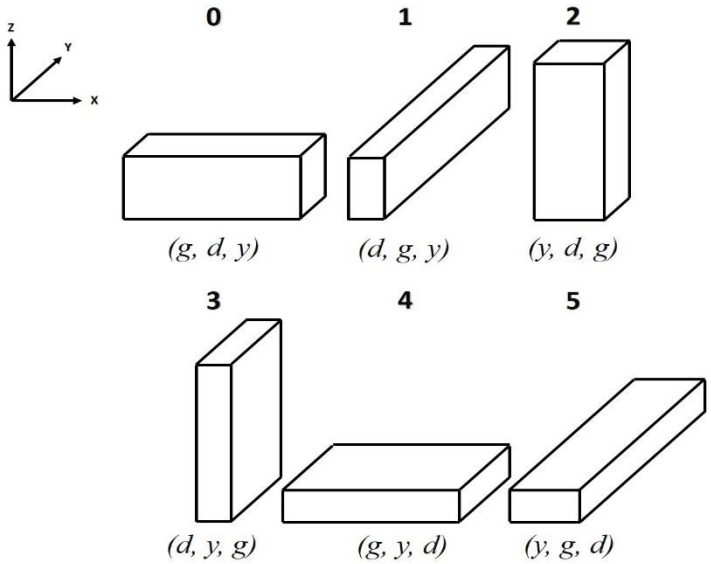
Bu çalışmada, TKYP için çözüm sunulurken birincil amaç nesnelerin yerleştirilecekleri konteyneri mümkün olan en üst doluluk oranı ile doldurabilmektir. Problem çözümü için oluşturulan "Nesne Yerleştirme Dizinleri" (NYD) ler konteynere hangi nesnenin hangi yönde atandığı Şekil 4.1'de gösterilmektedir. NYD'lerde nesnelerin konteynere atandığını göstermek için "1", atanmadığını göstermek için ise "0" değişkeni kullanılmaktadır. Nesne konteynere atanmasa dahi çözüm araştırması esnasında kullanıldığı için yönlendirme değişkeni atanmaktadır.

Her bir nesne Şekil 4.2'de gösterildiği gibi numaralandırılan altı farklı yönden (*g-d-y*, *d-g-y*, *y-d-g*, *d-y-g*, *g-y-d*, ve *y-g-d*) biri ile kısıtlama olmaksızın yerleştirilebilirken, bu

Şekil 4.1. TKYP'de nesne yerleştirme dizinlerinin yapısı

çalışmada önerilen yaklaşımların TKYP uygulamalarını gerçekleştirebilmek için kullanılan veri setinde konteynere atanacak nesne adayları için üç farklı yönlendirme kısıtı mevcuttur:

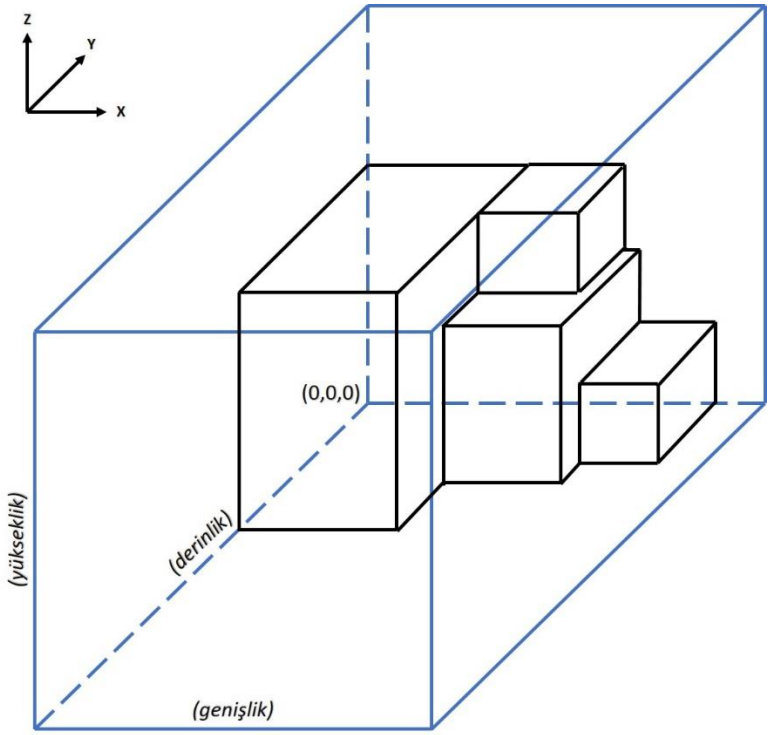
1. Bütün yönlendirmelerin izin verildiği durum,
2. 0, 1, 2 ve 3 nolu yönlendirmelere izin verildiği durum,
3. 0, 1, 4 ve 5 nolu yönlendirmelere izin verildiği durum.



Şekil 4.2. Nesne yönlendirme türleri

NYD'lerde nesnelere ait değişken atamaları gerçekleştirildikten sonra konteynere atanmış olan nesneler kendi aralarında genişlik, yükseklik, derinlik ölçüleri önceliğine göre büyükten küçüğe doğru sıralandıktan sonra konteynere Derin-Dip-Sol (DDS) yaklaşımı ile Şekil 4.3'teki gibi yerleştirilmektedirler. Nesneler konteynere

yerleştirilirken boyutsal ve yönlendirme kısıtları haricindeki kısıtlar bu çalışmada göz ardı edilmektedir.



Şekil 4.3. Nesnelerin konteynere DDS ile yerleştirilmesi

4.1.2. Çoklu Konteyner Yükleme Problemi

ÇKYP'de TKYP'den farklı olarak amaç konteynerlerdeki doluluk oranını maksimize etmekten ziyade kullanılan toplam konteyner sayısını veya toplam konteyner kullanım maliyetini minimize etmektir. Eşitlik 2.3 – 2.5'te konteynerlerin toplam kullanım maliyetinin minimizasyon modeli verilmektedir. Bu çalışmada ÇKYP uygulaması için kullanılan veri setinde toplam kullanım maliyeti değil toplam kullanılan konteyner sayısı Eşitlik 4.1 ve 4.2'de gösterilen modele göre minimize edilmektedir. Diğer bir deyişle ortalama kullanım oranı maksimize edilmektedir.

$$maksimize \ \frac{\sum_{b=1}^{C}\sum_{i=1}^{n} a_{ib}.v_{ib}}{\sum_{b=1}^{C} V_b} \tag{4.1}$$

$$kısıtlar \sum_{i=1}^{n} a_{ib}.v_{ib} \leq V_b \tag{4.2}$$

Eşitlik 4.1 kullanılan konteynerlerin ortalama kullanım oranının göstermekte ve yerleştirilen nesnelerin toplam hacimlerinin kullanılan konteynerlerin toplam hacimlerine bölünmesiyle bulunmaktadır. $a_{ib} \in [0,1]$ değişkeni nesne $i = 1, \dots, n$'nin konteyner $b = 1, \dots, C$'ye atanıp atanmadığını, v_{ib} ise b konteynerine atanacak aday nesnenin kapladığı hacmi göstermektedir. Bu çalışmada kullanılan veri setinde konteynerler eş boyutlara sahiptir ve V_b ise kullanılan konteynerlerin her birinin hacmini belirtmektedir.

Eşitlik 4.2'de ise konteyner b 'ye atanacak nesnelerine toplam hacminin ilgili konteynerin hacmini geçemeyeceği belirtilmektedir. Nesnelerin konteynerlere rastgele bir şekilde atanması, bazı konteynerlerin kapasitesini aşacak olursuz nesne yükleme dizinlerinin oluşturulmasına sebep olabilir. Bu sebeple, bu çalışmada NYD'lerde nesne atama değişkeni TKYP'deki gibi kullanılmazken Şekil 4.4'te gösterildiği üzere sadece nesne türü ve yönlendirme değişkeni kullanılmaktadır.

Şekil 4.4. ÇKYP'de nesne yerleştirme dizinlerinin yapısı

Şekil 4.4'teki NYD, ÇKYP için üretilen çözümleri temsil etmektedir. İlk satırdaki değişkenler nesne türü t'yi, ikinci satırdaki değişkenler ise TKYP'de olduğu gibi nesnelerin yönlendirme türünü temsil etmektedir. ÇKYP çözümü için Şekil 4.4 'deki haliyle çözümler üretildikten sonra nesneleri konteynerlere yerleştirme aşamasında birinci sıradan başlayarak ilgili sütundaki nesne türü ilgili yönlendirme ile birinci konteynere DDS yaklaşımı ile yerleştirilmektedir.

NYD boyunca yerleştirme sırası kendisine gelen nesneler, her defasında öncelikle ilk konteynerden başlayarak sonraki konteynerlere doğru yerleştirilmeye çalışılması yaklaşımı olan "İlk Uygun Yerleştirme" (İUY) yaklaşımı ile konteynerlere yerleştirilirler. Bu şekilde herhangi bir nesnenin herhangi bir konteynerin kapasitesini aşacak şekilde yerleştirilmeye çalışılması engellenmiş olur. TKYP'nin aksine ÇKYP'de ele alınan bütün nesnelerin konteynerlere yerleştirilir. Bu çalışmada ÇKYP için kullanılan veri setlerinde en fazla beş farklı türdeki nesnelerin eş boyutlardaki konteynerlere yerleştirilmesi ele alınmaktadır.

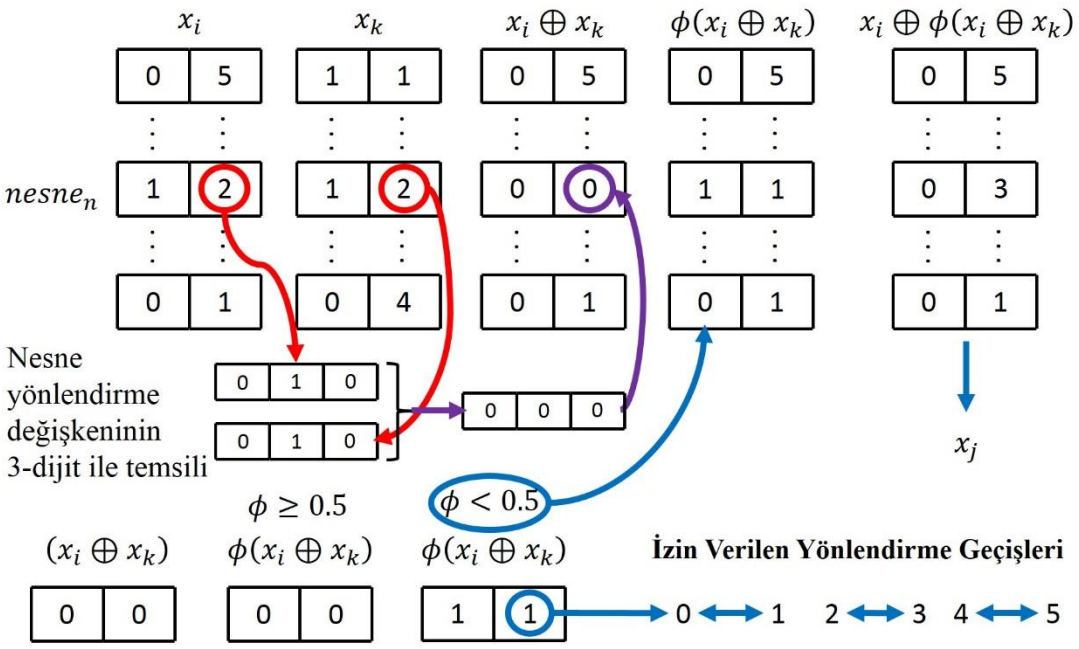
4.2. YAK ALGORİTMASININ KYP ÜZERİNDE UYGULANIŞI

TKYP ve ÇKYP çözümü için Eşitlik 3.1'deki formülden bağımsız şekilde başlangıç popülasyonunun bireyleri olan ve tamamen rassal olarak türetilen NYD'ler, temel YAK algoritması ve önerilen hibrit yaklaşımlar aracılığı ile iterasyonlar boyunca gelişim göstererek optimum NYD'lere dönüşür ve konteyner/konteynerlere yerleştirilecek nesneler için aslî yükleme planlarını teşkil eder.

Bu çalışmada, çözüm arama popülasyonundaki başlangıç çözümlerinin ve kâşif arı aşamasında terk edilen çözümün yerine türetilen yeni çözümün oluşturulmasını kapsayan çözüm çeşitlendirme işlemi tamamen rassal olarak yürütülmektedir. Öte yandan, çözüm yoğunlaştırma süreci çerçevesinde ise popülasyondaki mevcut çözümlerden yeni çözümler üretmek için Kıran ve Gündüz tarafından önerilen xor operatörü Şekil 4.5'te gösterildiği şekilde kullanılmaktadır [211].

TKYP'de başlangıç çözümlerinin her iki sütunu da tamamen rassal olarak belirlenirken, ÇKYP'de ise ilk sütunun temsil ettiği her bir nesne türünde kısıtlı sayıda nesne atanabileceği hesaba katılarak rassal değişken atanmakta ve ikinci sütunda yine tamamen rassal ilk atama yapılmaktadır. Bu çalışmada, temel YAK algoritması her bir alternatif çözüm arama hamlesinde mevcut çözümlerin tek bir elementini değiştirmeye çalışır. Şekil 4.5'te en soldaki NYD geliştirilmeye çalışılan mevcut çözüm x_i'yi, soldan ikincisi ise aynı popülasyon içerisinde yer alan ve x_i'nin komşuluğunda başka bir rastgele çözüm olan x_k'yı temsil etmektedir. Mevcut çözüm x_i ve komşu çözüm

x_k'nın xor operatörü yardımıyla etkileşiminden türetilen yeni çözüm ise x_j ile temsil edilmektedir.



Şekil 4.5. YAK algoritması XOR operatörüyle yeni çözüm oluşturma süreci

Kıran ve Gündüz, çalışmalarında [211] xor operatörünü yalnız 0 – 1 değişkenlerinin kullanıldığı ikili optimizasyon problemlerine göre geliştirmişlerdir. Öte yandan bu çalışmada, ÇKYP'deki NYD'ler için kullanılan nesne türü ve her iki problem türünde de kullanılan nesne yönlendirme değişkenleri 0 ve 1 ile sınırlı olmadıkları için, Şekil 4.5'te de gösterildiği biçimde, kullanılan değişken değerleri öncelikle üç dijite dönüştürülerek xor operatörünün uygulanabilir hale gelmesi sağlanmaktadır. Değişken dönüştürme işleminin ardından Eşitlik 3.2'deki yeni çözüm türetme formülü Eşitlik 4.3'teki şekliyle uygulanmaktadır.

$$x_j = x_i \oplus \phi(x_i \oplus x_k) \tag{4.3}$$

Eşitlik 4.3'teki $\oplus$ sembolü xor operatörünü temsil etmektedir. Mevcut çözüm x_i ve komşu çözümü x_k'nın n. elementleri seçildikten sonra TKYP için nesne atanma değişken değerleri (0 – 1), ÇKYP için nesne türü değişken değerleri (0 – 5) ve nesne yönlendirme değişken değerleri xor operatörü ile şu kurala göre etkileşime girerler:

Eğer x_i ve x_k'nın n. elementlerinin arasındaki dijit farkı sıfır ise işlem çıktısı 0 olarak belirlenir. Aksi takdirde işlem çıktısı 1 değerini alır.

TKYP'deki nesne atanma değişkenleri açısından xor işlemi çıktısı yine 0 veya 1 olacağı için sorun teşkil etmemektedir. Bununla birlikte, ÇKYP'deki nesne türü değişkenleri ve nesne yönlendirme değişkenlerinin yalnızca 0 – 5 arasında değer alabildiği değerlendirildiğinde üç dijit dönüşümleri sonrasında işlem çıktısı olarak 6 ve 7 değerlerinin elde edilirse mod 6'ya göre tekrar değer alırlar. Öte yandan TKYP'de kullanılan veri setinden kaynaklı olarak nesne yönlendirme geçişleri de Şekil 4.5'te gösterildiği üzere kısıtlanmaktadır. Örneğin; sadece 0, 1, 2 ve 3 nolu nesne yönlendirmelerine izin verildiği durumda x_i'nin n. nesne yönlendirme elementi 2 nolu yönlendirmeden 0, 1, 4 veya 5 nolu yönlendirmeye geçiş yapamamaktadır. Bunun yerine yalnızca Şekil 4.5'te belirtilen geçişlere izin verildiği yönde yani yalnızca 3 nolu nesne yönlendirmesine geçiş yapabilecektir. Böylelikle problemin kısıtlarının dışına çıkılmayarak olursuz çözümlerin türetilmesi engellenmektedir.

İlk aşamada $x_i \oplus x_k$ işleminin çıktısı olan değer rassal olarak belirlenen ϕ değeri ile ağırlıklandırılır. Eğer ϕ değeri 0,5'e eşit veya büyükse $x_i \oplus x_k$ işleminin çıktı değeri değişikliğe uğratılmaz. Aksi takdirde ilgili değişken değeri 0 – 1 dijitli karşılığının tam tersi haline dönüştürülür. Örneğin; yönlendirme kısıtının olmadığı durumda 1 nolu yönlendirme önce üç dijit olarak 1-0-0'a dönüştürülür ve ardından tersi alındığında bu değer 0-1-1'in karşılığı olan 6'ya dönüşür. 6 nolu yönlendirme mevcut olmadığından 1 nolu yönlendirmenin tersi 6(mod 6) = 0 değerini alır. Eğer yönlendirme kısıtı mevcutsa örneğin 4 nolu yönlendirmenin tersi direkt olarak 5 nolu yönlendirme olarak belirlenir. ÇKYP'de her bir türe ait nesne sayısı sınırlı olduğu için seçilen n. elementteki nesne türü değişirse NYD'de şu şekilde düzeltme yapılmaktadır: Yeni değişken değeri kendi türünden bir azaltacağı için eski değişken değeri NYD'nin sonundan başlayarak başa doğru ilk rastladığı elementte yeni değişken değerinin yerini alır. Başka bir deyişle eski değişken NYD içinde en sonda yer alan yeni değişken ile yer değiştirir. Burada amaçlanan şey, çözüm geliştirme aşamasında eğer değiştirilen nesne türü NYD sonuna taşındığında çözüm gelişiyorsa ilgili nesne türünün yerleştirilmesini sona bırakarak o nesne türünü bir nevi cezalandırmaktır.

Son aşamada ise mevcut çözüm x_i'nin $n.$ elementi, kendisini yeni çözüme taşıyacak adım olan $\phi(x_i \oplus x_k)$ işleminin çıktısı ile xor etkileşimine girerek yeni çözüm x_j'nin $n.$ elementi halini alır. Böylelikle komşu çözüm arama süreci tamamlanmış olur. KYP için temel YAK algoritması ile çözüm geliştirme sürecinin geri kalan kısmı, algoritmanın sayısal optimizasyon problemlerine uygulanma süreci ile aynıdır.

BÖLÜM 5

KYP İÇİN ÖNERİLEN YAKLAŞIMLAR

Bu çalışmada, temel YAK algoritmasının problem çözümlerini geliştirme sürecinde kullandığı çözüm arama yoğunlaştırma ve çözüm çeşitlendirme yaklaşımlarını geliştirmek için sırasıyla TA ve GA'nın yapısında barındırdıkları araçlar takviye yaklaşım olarak kullanılmaktadır. İki farklı aracın takviye olarak kullanılmasındaki amaç temel YAK algoritmasının geliştirilmesinde hangisinin daha etkili olacağını tespit etmektir. Ayrıca her iki aracın temel YAK algoritmasında bir arada takviye olarak kullanıldığı Bütünleşik Hibrit YAK (BHYAK) algoritması ise karşılaştırılan çalışmalarda KYP üzerinde uygulanan sezgisel yaklaşımlara göre etkinlik derecesi tespit edilmeye çalışılmaktadır.

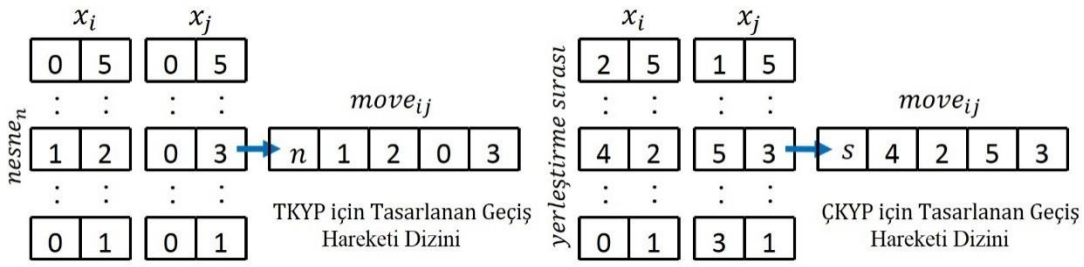
5.1. HAFIZA ENTEGRE YAK ALGORİTMASI

Temel YAK algoritmasında bir çözümün geliştirilememe sayacı $failure_i$, belirlenen geliştirememe limitini aştığında ilgili çözümü ile beraber, önceki döngülerde geliştirildiyse o çözümü terk edilmeden önceki haline taşıyan çözüm geliştirme adımları da hafızadan silinmektedir. Eğer mevcut x_i çözümünü daha iyi bir performans değerine sahip olan x_j çözümüne taşıyan hareketler ($move_{ij}$) belli bir döngü boyunca hafızaya alınırlarsa mevuct x_i çözümünün yerini alan x_j'nin terk edilmesi durumunda hafızaya alınan hareketler, popülasyon içindeki başka çözümlerin geliştirilmesi için de kullanılarak çözüm aram sürecinin hızlandırılmasına ve hatta temel YAK algoritmasının elde edebileceği optimum performans değerlerinin geliştirilmesine de katkı sağlayabilirler.

5.1.1. Geçiş Hareketlerinin Yapısı

Hafıza mekanizmasını kullanan benzer çalışmalarda [15,16] uygulandığı çözümlerde gelişme sağlayan parametre değerleri diğer hareketlerde de kullanılmak üzere hafızaya alınır. Eğer hafızaya alınan herhangi bir parametre çifti başka bir çözümün geliştirilmesi için kullanıldığında gelişme sağlayamazsa hafızadan silinir. Diğer taraftan, Yin ve Chuang ise parametre çiftleri yerine problem çözümlerini öncelikle olursuz hale getirip ardından problem kısıtlarına göre tekrar olurlu çözüm haline getirmeye çalışılmasını kapsayan salınma hareketlerinin [14] mevcut çözümü geliştirip geliştirememelerine göre ayrı ayrı hafızaya alınmasını tercih etmektedirler.

Bu çalışmada, çözüm arama hareketleri $move_{ij}$'ler Şekil 5.1'de gösterildiği üzere geçiş dizileri halinde hafızaya alınmaktadır. TKYP için tasarlanan $move_{ij}$'deki ilk öge nesne numarasını, ikinci ve dördüncü ögeler konteynere atanma değişkenlerini, üçüncü ve beşinci ögeler ise ilgili nesnenin yönlendirme türünü belirtmektedir. ÇKYP için tasarlanan $move_{ij}$'deki ilk öge ise ilgili nesnenin yerleştirilme sırasını, ikinci ve dördüncü ögeler nesne türlerini, üçüncü ve beşinci ögeler ise yine ilgili nesnenin yönlendirme türünü belirtmektedir.



Şekil 5.1. Geçiş hareketleri dizinlerinin yapısı

Bir önceki bölümde de belirtildiği üzere TKYP'deki NYD'ler üzerinde yapılacak olan değişiklikler dizinin yapısını düzeltmeyi gerektirmese de ÇKYP'deki NYD'ler üzerinde yapılacak olan değişiklikler esnasında her bir türdeki nesne sayısı sınırlı olduğu için her değişiklikte iki adet $move_{ij}$ hafızaya alınmaktadır.

5.1.2. Tabu Listeleri

Tabu listeleri, çözüm araştırma sürecinde rassal olarak belirlenen parametre değerlerinin araştırma için faydalı veya faydasız olmalarına göre sınıflanıp belli bir periyot için kaydedildikleri listeler olarak nitelendirilmektedir. Hafızaya alınan hareketlerin tabu listelerinde kalma süresi, ilgili tabu listesi için başta belirlenen kapasiteye göre değişebilmektedir. Dolana kadar hafızaya alınacak her hareket ilgili tabu listesine kaydedilir. Tabu listesi, diğer bir deyişle hafıza dolduktan sonra hafızaya alınacak yeni hareketlere listede yer açmak için ilk giren ilk çıkar (FIFO) prensibine göre en eskiden başlayarak hareketler sırasıyla silinirler. Eğer kısa bir tabu liste mevcutsa hafızadaki hareketler hızlı bir şekilde değişebilmektedir. Aksi takdirde, hareketler uzun süreli olarak hafızada yer alabilmektedir.

Algoritma 5.1 hafıza entegre YAK (HYAK) algoritmasının yerel araştırma safhasını anlatmaktadır. HYAK algoritmasında Kısa Vadeli Tabu Listesi (KVTL), Orta Vadeli Tabu Listesi (OVTL) ve Uzun Vadeli Tabu Listesi (UVTL) işçi ve gözcü arı safhasında alternatif çözümler bulmak için yapılan yerel araştırmayı yönlendirebilecek hareketleri tekrar kullanmak üzere kayıt altında tutmaktadırlar.

Algoritma 5.1: HYAK'da işçi ve gözcü arı safhasında geliştirilen yeni çözümün değerlendirilmesi

```
Eşitlik (2.1) veya (4.1)'e göre x_j'nin performans değerini hesapla
if f_{x_j} > f_{x_i}
    x_i'yi x_j ile değiştir
    failure_i = 0
    intscore_p = intscore_p + 1
    Hareket move_{ij}'yi OVTL'ye kaydet
else
    failure_i = failure_i + 1
    intscore_p = intscore_p − 1
    if move_{ij} ∈ OVTL or UVTL
        Hareket move_{ij}'yi ilgili listeden sil
    else
        Hareket move_{ij}'yi KVTL'ye kaydet
    end if
    if intscore_p < 1
        intscore_p = 1
    end if
end if
```

Eğer geçiş hareketi $move_{ij}$ mevcut çözümü geliştiremiyorsa başka çözümlerin geliştirilmesi aşamasında tekrar tekrar kullanılmasını engellemek maksadıyla Algoritma 5.1, 13. satırda da belirtildiği üzere KVTL'ye kaydedilmektedir. Eğer türetilen hareketler KVTL'de kayıtlı ise, kayıtlı olmayan bir hareket bulununcaya kadar tekrar hareket türetme işlemi yapılır. Böylelikle çözüm araştırması daha önceden denenmemiş hareketleri bulmaya zorlanmaktadır. HYAK algoritması KVTL'deki hareketlere ek olarak OVTL ve UVTL'deki faydalı olarak belirlenen hareketleri de bir olasılık değerine göre rassal olarak kullanabilmektedir. Böylelikle HYAK algoritması bir yandan faydasız hareketleri kısıtlayarak diğer yandan da faydalı hareketlerin kullanımını teşvik ederek temel YAK algoritmasının yerel araştırma yönünü takviye etmekte ve çözüm araştırma odağını daha iyi performans değerlerine sahip çözümlerin yer aldığı çözüm bölgelerine taşıyarak daha iyi optimum değerleri daha kısa sürede elde etme imkânı sağlayabilmektedir.

5.1.3. Yeni Çözüm Türetmek için Kullanılan Hareketin Seçilmesi

HYAK algoritmasında mevcut x_i çözümünden yeni x_j çözümüne geçmek için Eşitlik 4.3 ile türetilen rassal bir hareket veya OVTL ve UVTL'den birisinde hafızaya alınmış rassal bir hareket kullanılabilir. Bu durumda her bir x_i çözümünü geliştirmek üzere kullanılacak sıradaki hareket için üç farklı seçenek bulunmaktadır:

1. OVTL'den rassal bir hareket seçmek,
2. UVTL'den rassal bir hareket seçmek,
3. Rassal bir hareket türetmek.

Bu üç seçenek arasından sıradaki hareketin tercihi Eşitlik 5.1'e göre belirlenmektedir.

$$iw_p = \frac{intscore_p}{\sum_{p=1}^{PI} intscore_p} \tag{5.1}$$

Eşitlik 5.1'de, $p = 1, \dots, PI$ toplam seçenek sayısını göstermekte, iw_p ise yerel araştırmada yeni çözüm türetmek üzere kullanılan hareket için tercih edilen seçeneğin ağırlığını belirtmekte ve [0, 1] aralığında değer almaktadır. $intscore_p$ değeri ise tercih

edilen seçeneğin net başarı puanını belirtmektedir. iw_p değeri her bir seçeneğin net başarı puanının bütün seçeneklerin başarı puanına bölünmesiyle bulunmaktadır.

Benzer şekilde Moayedikia ve ark. [216] seçenekler arasında tercihte bulunmak için kullanılan rulet tekerleği üzerinde birtakım değişiklikler yoluyla az ziyaret edilen çözümlerin tercih edilme ağırlığını artırarak çözümlerin dengeli şekilde ziyaret edilmelerini sağlamaktadır. Bu çalışmada ise, önerilen HYAK ve diğer algoritmalarda türetilecek hareketlerin tercih edilmesi için geliştirilen seçenekler ağırlıklandırılıp aralarından birisine rulet tekerleğinde denk gelen olasılık değerine göre karar verilmektedir. Eğer tercih edilen $move_{ij}$ hareketi mevcut x_i çözümünü geliştirebilirse bu hareketin bağlı olduğu seçeneğin puanı $intscore_p$, Algoritma 5.1, 5. satırda belirtildiği üzere bir artırılır. Aksi takdirde ilgili seçeneğin puanı 9. satırda belirtildiği üzere bir azaltılır. Bununla birlikte, herhangi bir seçeneğin tercih kapsamından çıkmasını engellemek için hiçbir seçeneğin puanı 16. satırda belirtildiği üzere 1'den az olamamaktadır.

Eğer tercih edilen $move_{ij}$ hareketi OVTL veya UVTL'den seçilmesine rağmen uygulandığı x_i çözümünü geliştiremezse, bu tür hareketler 11. satırda belirtildiği üzere ilgili tabu listesinden silinir. Böylelikle gelişme sağlayamayan hareketler elenerek ilgili tabu listesinde uzun süre fayda sağlayabilecek hareketlere yer açılması sağlanır.

UVTL yerel araştırma safhasında keşfedilen iyi çözüm parametreleri veya geçiş hareketlerini kaydetmek yerine çözüm çeşitlendirme safhasında, mevcut x_i çözümünü, araştırmanın yoğunlaştığı bölgeden daha farklı ve daha iyi çözüm değerlerinin bulunma ihtimali olan bölgelere taşıyan parametre değerleri veya hareketlerini kaydetmektedir [215]. Temel YAK algoritması kâşif arı aşamasında gerçekleştirilen çözüm çeşitlendirme sürecinde terk edilecek çözümleri, göreceli olarak daha iyi bir bölgeye taşıyan hareketlerin hafızaya alınması için en uygun tabu listesi UVTL olacaktır. HYAK algoritmasında UVTL'ye kaydedilen hareketlerin, popülasyon içerisinde yer alan diğer çözümleri de daha iyi araştırma bölgelerine taşıyabileceği düşünülmektedir. Algoritma 5.2, 4. satırda UVTL'ye kaydedilecek olan hareketlerin seçilme kriteri belirtilmektedir.

Algoritma 5.2: HYAK algoritması kâşif arı aşamasında yeniden çözüm türetme süreci	
01	**while** $move_{ij} \in$ STTL
02	Mevcut x_i çözümünü Eşitlik (4.3)'e göre rassal türetilen x_j ile değiştir
03	**end while**
04	**if** $\left\|f_{x_{maks}} - f_{x_j}\right\| < \left\|f_{x_j} - f_{x_{ort}}\right\|$ ve $f_{x_j} > f_{x_i}$
05	Hareket $move_{ij}$'yi UVTL'ye kaydet
06	**end if**

Eğer kâşif arı aşamasında terk edilecek çözümün taşındığı yeni x_j çözümünün performans değeri, popülasyon ortalamasından yüksek ve popülasyonun en iyi performans değerine sahip çözüme diğerlerinden göreceli olarak daha yakınsa, terk edilecek çözümü x_j 'ye taşıyan $move_{ij}$ hareketi, popülasyon içerisinde başka çözümlerin performans değerlerini de geliştirebileceği düşünülerek UVTL'ye kaydedilir.

5.2. GENETİK OPERATÖR ENTEGRE YAK ALGORİTMASI

Temel YAK algoritmasında istenilen seviyede performans değerine sahip olmayan çözümlere erken yakınsama ihtimali, algoritmanın dezavantajlarından birisidir. Bu durumda temel YAK algoritmasındaki özüm çeşitlendirme sürecini takviye ederek aslında optimum olmayan çözüme yakınsama ihtimalini azaltmak için GA'nın sağlayacağı araçlar kullanılabilir.

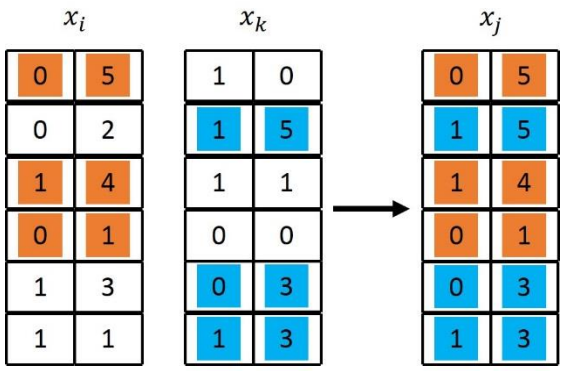
GA, popülasyondaki çözümleri çeşitlendirmeyi çaprazlama ve mutasyon operatörleri aracılığı ile sağlamaktadır. Bu çalışmada, işçi ve gözcü arı safhasında rassal çözüm türetme yaklaşımı yanı sıra genetik operatörlerin ayrı ayrı veya bir arada kullanılması ile dört farklı çözüm türetme seçeneği oluşmaktadır:

1. Yalnız çaprazlama operatörünün kullanılması,
2. Yalnız mutasyon operatörünün kullanılması,
3. Çaprazlama ve mutasyon operatörlerinin bir arada sırayla kullanılması,
4. Rassal çözüm türetme yaklaşımının kullanılması.

5.2.1. Genetik Operatörler

GA kapsamındaki çaprazlama operatörü, normal şartlarda mevcut x_i ve komşuluğundaki x_k çözümlerinden yine oğul (child) olarak belirtilen iki yeni birey çözüm türetmektedir ve sonraki aşamalarda bu çözümlerden birisini elemektedir. Bu çalışmada ise önerilen algoritmaların uygulandığı problemlerin kompleksliği sebebiyle, çaprazlama işlemi neticesinde performans değerlendirmesine alınması gereken çözüm sayısını azaltmak maksadıyla mevcut x_i ve komşuluğundaki x_k çözümlerinden yalnız bir oğul çözümün türetilmesi önerilmektedir.

Chaurasia ve ark. [12] önerdikleri genetik operatör takviyeli bir YAK algoritmasında, çaprazlama işlemi neticesinde yalnız tek oğul çözümün türetilmesini sağlamak için Çoklu-Nokta Yerleştirme yaklaşımını Şekil 5.2'de gösterildiği gibi kullanmışlardır. Bu çalışmada ise önerilen genetik operatör takviyeli YAK (GYAK) algoritmasında, Çoklu-Nokta Yerleştirme işlemi dört alternatif çözüm türetme seçeneğinden birisi olarak yer almaktadır.



Şekil 5.2. Çoklu-Nokta yerleştirme yaklaşımı

İkinci seçenek olan mutasyon operatörü ile mevcut x_i çözümünden yeni x_j çözümü türetmek için x_i çözümündeki rassal iki sıradaki değişken değerleri değiş-tokuş yapılmaktadır. Üçüncü seçenek ise Çoklu-Nokta Yerleştirme işlemi ile elde edilen ara çözüm x_z'den, mutasyon operatörü ile yeni çözüm x_j türetilmektedir. Son seçenek de HYAK algoritmasında olduğu gibi tamamen rassal bir x_j çözümünün türetilmesidir.

5.2.2. Yeni Çözüm Türetmek için Genetik Operatörlerin Seçilmesi

GYAK algoritması işçi ve gözcü arı safhasında, yeni x_j çözümlerinin türetilmesi için dört seçenekten birisi, HYAK algoritmasında kullanılan Eşitlik 5.1'de belirtildiği gibi başarı puanları ile ağırlıklandırılıp rulet tekerleği olasılığına göre tercih edilmektedir.

GYAK algoritması kâşif arı safhasında da çözüm araştırmasında kullanılacak yaklaşımlar da yerel araştırma safhasında olduğu gibi ağırlıklandırılıp rulet tekerleği olasılığına göre aralarından birisi tercih edilmektedir. Singhal ve ark. [217] mevcut döngüde türetilen çözümlerden performans değeri en iyi olan ile popülasyonun en iyi performans değerine sahip çözümü genetik operatörler aracılığı ile etkileşime girmesi önerilmektedir. Ayrıca kâşif arı safhasında terk edilen çözümün yerini elde edilen en iyi çözümün alması önerilmektedir. Bu fikirden hareketle, GYAK algoritması kâşif arı safhasında terk edilecek çözüm, popülasyonun en iyi performans değerlerine sahip elit grup e çözümlerden birisi ile önce Çoklu-Nokta Yerleştirme ve ardından mutasyon operatörünü kullanarak etkileşime girmesi önerilmektedir. Öte yandan terk edilecek çözüm başka bir seçenek olarak yerini tamamen rassal türetilen bir çözüme bırakabilir. Bu durumda GYAK algoritması kâşif arı safhasında yeni çözüm türetmek için Algoritma 5.3'te de belirtildiği üzere kullanılabilecek iki seçenek sunmaktadır:

1. Rassal çözüm türetme $rçt$,
2. Elit çözüm yönlendirmeli türetme $eçyt$.

Algoritma 5.3: GYAK algoritması kâşif arı safhası

Çeşitlendirme operatörlerinin dw_p ağırlıklarını Eşitlik (5.2)'ye göre hesapla
if $rassal < dw_{\boldsymbol{p}_{rçt}}$ **//Rassal Çözüm Türetme Seçeneği**
    Yeni xj çözümünü rassal olarak türet
    if $\left|f_{x_{maks}} - f_{x_j}\right| < \left|f_{x_j} - f_{x_{ort}}\right|$ ve $f_{x_j} > f_{x_i}$
        $divscore_p = divscore_p + 1$
    end if
else if $rassal < dw_{\boldsymbol{p}_{eçyt}}$**//Elit Çözüm Yönlendirmeli Türetme Seçeneği**
    Elit çözümler arasından x_k çözümünü rassal olarak seç
    Terk edilecek x_i çözümü ile $x_k$çözümünden "Çoklu-Nokta Yerleştirme" yaklaşımı ile geçiş çözümü olan x_z'yi türet
    x_z çözümünün rassal iki elemanının yerini değiştirerek yeni x_j çözümünü türet
    if $\left|f_{x_{maks}} - f_{x_j}\right| < \left|f_{x_j} - f_{x_{ort}}\right|$ ve $f_{x_j} > f_{x_i}$
        $divscore_p = divscore_p + 1$
    end if
end if

GYAK algoritması kâşif arı safhasında çözüm türetme operatörünün net başarı puanını belirten $divscore_p$ Algoritma 5.3, 5. ve 12. satırlarda gösterildiği üzere, ilgili seçenek ile türetilen yeni çözümün performans değerinin, popülasyon ortalamasından ziyade en iyi performans değerine yakın olduğu durumda bir artırılmaktadır. Her bir seçeneğin sahip olduğu ve her yeni türetilen çözümün değerlendirilmesinde tekrar hesaplanan $divscore_p$ puanı ile Eşitlik 5.2'de gösterildiği gibi ilgili seçeneğin çözüm çeşitlendirme ağırlığı hesaplanır.

$$dw_p = \frac{divscore_p}{\sum_{p=1}^{PD} divscore_p} \tag{5.2}$$

Eşitlik 5.2'ye göre seçeneklerin ağırlıkları hesaplandıktan sonra rulet tekerleği olasılığına göre seçeneklerden birisi yeni x_j çözümünü türetecek operatör olarak tercih edilir.

5.3. BÜTÜNLEŞİK HİBRİT YAK ALGORİTMASI

Bu çalışmada, temel YAK algoritmasının hem çözüm arama hem de çözüm çeşitlendirebilme özelliğini aynı anda geliştirerek güçlü bir algoritma geliştirmek amaçlanmaktadır. Önceki bölümlerde de belirtildiği üzere, HYAK algoritması yerel araştırmada keşfedilmemiş çözümleri türetmek için bir hafıza mekanizması ile temel YAK algoritmasını takviye ederken, GYAK algoritması ise daha iyi performans değerlerine sahip çözümlerin bulunduğu muhtemel yerel araştırma sahalarını, genetik operatörlerin çözüm çeşitlendirme özelliği ile keşfederek, araştırmanın optimal olmayan bölgelerde sonlanmasını engellemek yolu ile takviye etmektedir. Bütünleşik Hibrit YAK (BHYAK) algoritması ise temel YAK algoritmasının çözüm araştırma ve çeşitlendirme özelliklerini kuvvetlendirmek için HYAK ve GYAK algoritmalarının bir araya gelerek oluşturduğu üçlü bir algoritma olarak Algoritma 5.4'te ayrıntılı olarak anlatıldığı üzere geliştirilmiştir.

BHYAK algoritmasında işçi ve gözcü arı safhalarında, ikisi HYAK algoritması, üçü GYAK algoritması kaynaklı ve birisi de rassal çözüm türetme olmak üzere toplam altı

farklı stratejiden birisi kullanılarak popülasyon içerisindeki mevcut çözümler geliştirilirler.

Algoritma 5.4: BHYAK algoritması

SN popülasyon adedince rassal çözüm türet
$\forall\ failure_i = 0$ **//Başlangıç Safhası**
$\forall\ intscore_p = 1$ ve $divscore_p = 1$
Toplam değerlendirme sayısını ($Eval_{maks}$) belirle
İlk popülasyondaki (x_i) çözümlerinin performans değerlerini Eşitlik (2.1) veya (4.1)'e göre hesapla
    while $Eval < Eval_{maks}$
        for $i = 1$ to SN do **//İşçi Arı Safhası**
            Çözüm Türetme Operatörü Seçimi;
            Algoritma 5.1;
        end for **//İşçi Arı Safhasını Bitir**
        for $t = 1$ to SN do **//Gözcü Arı Safhası**
            if $rassal < p_i$
                Çözüm Türetme Operatörü Seçimi;
                Algoritma 5.1;
            end if
            if $Eval \geq Eval_{maks}$
                En iyi performans değerine sahip çözümü koru
                Araştırmayı Bitir
            end if
            $t = t + 1$
        end for **// Gözcü Arı Safhasını Bitir**
        En iyi performans değerine sahip çözümü koru
        if $failure_{i_{maks}} > L$ **//Kâşif Arı Safhası**
            Çözüm çeşitlendirme operatörlerinin dw_p ağırlıklarını Eşitlik (5.2)'ye göre hesapla
            if $rassal < dw_{\boldsymbol{p}_{rçt}}$ **//Rassal Çözüm Türetme Seçeneği**
                while $move_{ij} \in$ STTL
                    Mevcut x_i çözümünü Eşitlik (4.3)'e göre rassal türetilen x_j ile değiştir
                end while
                if $\left|f_{x_{maks}} - f_{x_j}\right| < \left|f_{x_j} - f_{x_{ort}}\right|\ and\ f_{x_j} > f_{x_i}$
                    $divscore_p = divscore_p + 1$
                    Hareket $move_{ij}$'yi UVTL'ye kaydet
                end if
            else if $rassal < dw_{\boldsymbol{p}_{eçyt}}$ **//Elit Çözüm Yönlendirmeli Türetme Seçeneği**
                while $move_{ij} \in$ STTL
                    Terk edilen x_i çözümü, her iki genetik operatörü kullanarak x_j çözümünü türetmek için elit x_k çözümlerinden biriyle eşleştir
                end while
                if $\left|f_{x_{maks}} - f_{x_j}\right| < \left|f_{x_j} - f_{x_{ort}}\right|\ ve\ f_{x_j} > f_{x_i}$
                    $divscore_p = divscore_p + 1$
                    Hareket $move_{ij}$'yi UVTL'ye kaydet
                end if
            end if
        end if **//Kâşif Arı Safhasını Bitir**
    end while
    Araştırmayı Bitir

Her bir x_i çözümünün elementleri üzerinde yapılan değişiklik faydalı olup olmamasına göre farklı tabu listelerinde (KVTL veya OVTL) kaydedileceği için, $move_{ij}$ geçiş hareketleri kullanılan stratejiye göre teker teker veya blok halindeki hafızaya alınacaklardır. Bu durumda, özellikle GYAK kaynaklı stratejilerden olan çaprazlama operatörünün kullanımı ile ortaya çıkan değişiklikler tabu listelerini daha hızlı dolduracaktır. Yerine göre çaprazlama operatörü ile on – on beş elementin bir arada yer değiştirebildiği durumlar düşünüldüğünde, olumlu veya olumsuz gelişimin kaynağını tespit etmek zorlaşacak ve tabu listelerine kaydedilecek olan geçiş hareketlerinin hacmi genişleyeceği için hafızanın kullanımı son derecede hızlanacaktır. Bu sebeple BHYAK algoritmasında çaprazlama kaynaklı değişiklikler blok halinde tek bir geçiş hareketiymiş gibi ilgili tabu listelerine kaydedilmektedirler.

Seçilen strateji ele alınan mevcut çözümü geliştirebilirse başarı puanı $intscore_p$ bir artırılır. Aksi takdirde bir azaltılır ki bu da ilgili stratejinin aleyhine iw_p ağırlığını etkilemektedir. Öte yandan seçilen strateji neticesinde türetilen çözümün performans değeri, Algoritma 5.4, 29 ve 37. satırdaki şartı sağlaması halinde ilgili çözüme geçişi sağlayan hareket UVTL'ye kaydedilerek çözüm çeşitlendirme puanı olan $divscore_p$ bir artırılır ki bu da ilgili stratejinin lehine dw_p ağırlığını etkilemektedir.

Bu çalışmada önerilen BHYAK algoritması, ikili (0-1) optimizasyon temelli KYP için geliştirilmiş olup, çaprazlama ve mutasyon gibi evrimsel gelişim algoritmalarında kullanılan operatörleri de bünyesinde barındırmaktadır. Bu operatörlerin ayrı ayı veya bir arada kullanılması yolu ile elde edilen farklı stratejiler, yerel araştırma ve çözüm çeşitlendirme safhalarındaki performanslarına göre ağırlıklandırılıp rulet tekerleği olasılığına göre bu stratejilerden birisi seçildikten sonra problem çözümlerini geliştirmek için kullanılmaktadır. Evrimsel gelişim operatörlerinin kullanıldığı bir diğer çalışmada, Öztürk ve ark. [10] ele aldıkları ikili optimizasyon problemini çözmek maksadıyla, yerel araştırma safhasında çaprazlama ve mutasyon operatörlerini bir arada kullanarak birkaç adımda mevcut x_i çözümü ve komşuluğundaki x_k çözümünden birden fazla alternatif x_j çözümü türetmekte ve aralarından en iyi performansa sahip olanı x_i'nin performansından daha iyi ise ilgili alternatifi mevcut çözümün yerine koymaktadırlar. Luo ise Öztürk ve ark. tarafından önerilen bu

yaklaşımı basitleştirerek geliştirdikleri tek-nokta çaprazlama ve mutasyon yaklaşımı ile x_i ve x_k çözümlerinden sadece iki oğul çözüm türetmekte ve ardından yine mevcut çözüm ile türetilen çözümler arasında bir tercihte bulunmaktadırlar [218]. BHYAK algoritması ise Çoklu-Nokta Yerleşimi yaklaşımı ile her x_i-x_k etkileşiminden yalnız bir yeni çözüm türetmekte, böylelikle özellikle bu çalışmada da ele alınan KYP gibi NP-hard türde olan problemlerin kompleks yapısı düşünüldüğünde çözüm değerlendirme süresini azaltarak katkıda bulunmaktadır.

Elghamrawy, gözcü arı aşamasında mevcut x_i çözümüne alternatif olacak x_j çözümünün, komşuluktaki rastgele bir x_k çözümü ile popülasyonun en iyi performans değerine sahip çözüm arasında yalnız çaprazlama işlemi yapılarak türetilmesini önermektedir [219]. Cao ve ark. ise öncelikle komşu x_k çözümünü, popülasyonun en iyi performans değerine sahip çözümleri arasından bir olasılıkla seçileninin mutasyon işlemine tâbi tutulmasıyla elde ederek, mevcut x_i çözümü ile bu x_k çözümünün çaprazlanmasından yeni x_j çözümünü türetmektedir [8]. BHYAK algoritması ise kâşif arı safhasında terk edilecek x_i çözümüne yerel optimumdan kaçabilmesi için son bir şans vermek amacıyla geliştirilen yeni çözüm türetme stratejisiyle, öncelikle elit arı olarak tanımlanan popülasyonun en iyi performans değerine sahip arı sınıfından rastgele birisini komşu x_k çözümü olarak seçer. Ardından x_i ve x_k çözümleri sırasıyla Çoklu-Nokta Yerleştirme temelli çaprazlama ve mutasyon sürecinden geçirilerek yeni x_j çözümü türetilir. Eğer terk edilecek olan x_i çözümü araştırma uzayında daha verimli bir bölgeye taşınabilirse, çözümü taşıyan hareket, son derece verimli bir geçiş hareketi olarak değerlendirilir ve daha sonra da kullanılmak üzere UVTL'ye kaydedilmek suretiyle hafızaya alınır.

Mevcut x_i çözümlerinin yerel optimumdan kurtarılabilmesi için çeşitli yaklaşımlar denenmektedir. Temel YAK algoritmasında çözümlerin geliştirilmesi, çözüm elementlerinin teker teker değiştirilmesi ile sağlanırken, Chu ve ark. her çözüm geliştirme hamlesinde birden fazla elementin bir arada değiştirilmesini önermektedir. Ayrıca, BHYAK algoritmasının kâşif arı safhasında uyguladığı, elit çözümlerden biriyle terk edilecek çözümün etkileşime girmesi stratejisine benzer şekilde Chu ve ark. da x_i çözümleri için x_k komşu çözümü seçerken her zaman performans değeri daha iyi olan çözümün tercih edilmesini önermektedirler. Bu çalışma ile benzeşmeyen

yönü ise her bir iterasyondaki en iyi performans değerine sahip çözümü Cauchy mutasyonu ile tekrar geliştirmeye çalışmalarıdır [220]. Chen ve ark. ise erken yakınsamadan kaçınmak maksadıyla, gözcü arı safhasında rulet tekerleği olasılığı yerine iterasyonlar boyunca dinamik olarak değişen Boltzmann Seçimi olasılığını kullanarak göreceli olarak kötü performansa sahip çözümlere geliştirilebilmeleri için daha çok şans verilmektedir [221]. BHYAK algoritması ise yerel optimumdan kurtulmak için genetik operatörleri kullanırken erken yakınsamayı engellemek için ise hafıza mekanizmasını kullanmaktadır. Ayrıca, YAK algoritmasının her safhasında çözüm geliştirme başarılarına göre ağırlıklandırılmış stratejilerin, rulet tekerleği olasılığına göre uygulanması ile de dengeli bir çözüm arama süreci sağlanmaktadır.

Yerel optimum bölgesinden kurtulup erken yakınsamayı engellemenin bir diğer yolu da çözüm geliştirme hareketlerinin ölçüsünü, ne optimum çözüme yakınsama süresini aşırı derece uzatacak kadar kısa ne de verimli çözüm parametrelerini atlayacak kadar uzun tutmamaktır. Bansal ve ark. ile Kumar ve ark. Eşitlik 3.2'deki ϕ_{ij} rassal değerini Altın Sahası Araştırması metodu ile belirleyip en verimli değer ile çözüm geliştirme hareketlerinin elde edilebileceğini öngörmektedirler [222, 223]. Kumar ve ark. ek olarak, gözcü arı safhasında sadece ilgili x_i çözümünün performans değerine göre değil ayrıca ϕ_{ij} değerine göre dinamik bir p_{ij} olasılık değeri önermektedirler. Jia ve ark. da benzer şekilde performans değerlerinin, Eşitlik 3.2'deki yeni v_{ij} çözümünün belirlenmesinde etkili olduğu bir yaklaşım önermektedirler. Bu yaklaşıma göre λ ile belirtilen faktör değeri ilgili çözümün performans değerine göre değişmekte ve popülasyon evrildikçe, popülasyondaki çözümlerin yeni çözümlerin türetilmesindeki ağırlığı artmaktadır. Bu da yerel araştırmanın daha da derinleştirilebilmesini sağlamaktadır. Diğer taraftan, kontrol parametresi olarak kullandıkları modifikasyon oranı ile de çözümlerin çeşitlendirilmesini sağlamışlardır [224].

Kıran ve Gündüz ele aldıkları ikili optimizasyon probleminde, mevcut x_i ve komşuluğundaki x_k çözümleri arasındaki benzeşmezliğe göre $move_{ij}$ hareketini belirlemektedir. Burada x_i ve x_k arasındaki 0-1 takaslarının sayısı rassal v ve benzeşmezlik değeri dikkate alınarak optimize edilmekte ve ona göre yeni x_j çözümü türetilmektedir [211]. Hançer ve ark. da benzer şekilde yeni x_j çözümünün türetilmesi

sürecinde benzeşmezlik değerini kullanmakta, ilaveten v değerinin mevcut iterasyon numarasını ve toplam iterasyon sayısının hesaba katılarak optimize edilmesini önermektedirler [213].

BHYAK algoritmasında ise tercih edilen çözüm geliştirme stratejisine göre geçiş hareketinin boyutu değişiklik göstermektedir. Eğer yeni çözüm rastgele veya mutasyon operatörü vasıtasıyla türetilecek olursa geçiş hareketleri tek satırlık dizinler halinde oluşur. Öte yandan çaprazlama işleminin sonucunda ise birden fazla çözüm elementinin değişebileceği durumlar ortaya çıkabilmektedir. Yukarıda da belirtilen sebeplerden ötürü bu tür hareketler hafızaya alınması gerektiğinde blok halinde tek bir hareket olarak ilgili tabu listesine kaydedilmesi gerekmektedir.

BÖLÜM 6

DENEY VE BULGULAR

Bu çalışmada, KYP üzerinde uygulanmak üzere temel YAK algoritması yanı sıra HYAK, GYAK ve BHYAK algoritmaları geliştirilmiş, kodlama çalışmaları MATLAB R2016 yazılımı üzerinde gerçekleştirilmiştir. Deneysel testlerin yapıldığı bilgisayar özellikleri Windows 7 işlemci, 4 GB RAM hafıza ve 3,10 GHz işlemci hızı şeklindedir. Temel YAK algoritması ve önerilen diğer algoritmalar literatürde sıkça kullanılan veri setleri üzerinde test edilmiştir. Genel olarak KYP'lerin komplekslik ölçüsünün $O(n^3)$ olduğu belirtilmektedir [225].

6.1. TKYP DENEYLERİ ve BULGULARI

Bu çalışmada, önerilen algoritmaları TKYP üzerinde uygulamak için literatürde birçok çalışma tarafından uzun yıllar boyunca kullanılan BR01-15 veri seti tercih edilmiştir. Bu veri setindeki 15 farklı problem, yerleştirilmesi gereken nesne sayısına göre (3'ten 100'e kadar) sınıflandırılmış olup, her birisi 100 alt problemden oluşmaktadır. İlk yedi problemdeki (BR01-BR07) nesneler homojene yakın bir dağılıma sahipken, sonraki problemler (BR08-BR15) heterojen dağılıma sahip nesnelerin tek bir konteynere yerleştirilmesini amaçlamaktadır. TKYP'de amaç konteyner dolum oranını (KDO) maksimize etmektir. Önerilen BHYAK algoritması, literatürden seçilen aşağıdaki çalışmalarda önerilen yaklaşımlar ile karşılaştırılmıştır:

- GA_GB: Gehring ve Bortfeldt tarafından GA (1997), Pentium/l30 MHz PC [26]
- TS_BG: Bortfeldt ve Gehring tarafından TA (1997), Pentium/200 MHz PC [226]
- CBGAS: Bortfeldt ve Gehring tarafından hibrit GA (2001), N/A [43]

- HCLPMC: Sheng ve ark. tarafından TA (2017), Intel Core i7 870 @2.93 GHz [54]
- GRASP: Parreno ve ark. tarafından GRASP yaklaşımı (2008), Pentium IV 1.7 GHz [113]
- HTS: Liu ve ark. tarafından hibrit TA (2011), Intel Centrino Duo CPU 1.66 GHZ, 1 GB RAM [142]
- Hibrit-BA: Dereli ve Daş tarafından hibrit ArıA (2011), N/A [145]
- SOA: Zhou ve Liu tarafından Sürü Optimizasyonu Algoritması (2017), N/A [227]

BR01-15'i önerdikleri yaklaşımlar için uygulama veri seti olarak kullanan yukarıdaki çalışmalar, önerilen yaklaşımın çözüm araştırma performansını daha iyi ölçebilmek için nesnelerin konteynere yerleştirilmelerinde sadece yönlendirme ve denge kısıtlarını dikkate almaktadırlar. Bu çalışmada da aynı şekilde sadece yönlendirme ve denge kısıtları dikkate alınarak nesnelerin yerleştirilmesi amaçlanmaktadır.

Karşılaştırma yapmak için ele alınan hiçbir çalışmada uygulanan algoritmanın hesaplama kompleksliği ile alakalı bilgi verilmemiştir. Yalnız Liu ve ark. [142] çalışmasında problem kompleksliğinin genişleyen çözüm uzayı ile birlikte arttığını belirtmişlerdir. Bu yüzden çözümleme için kullandıkları NYD'leri toplam nesne sayısı yerine nesne türünü dikkate alarak tasarlamışlardır. Önerilen BHYAK algoritması ve karşılaştırılan algoritmaların BR veri setindeki problemleri çözebilmesi aşırı uzun test sürelerine ihtiyaç duyduğundan dolayı, istatistiksel açıdan anlamlı sonuç çıkarmak için gereken test tekrar sayısı 30 olması gerekirken karşılaştırma için incelenen araştırmalardan bazılarında 3 tekrar [145], 10 tekrar [113] ve 20 tekrar [54] olarak belirlenmiştir. Karşılaştırılan diğer çalışmalarda ise test tekrar sayısı belirtilmemiştir. Bu çalışmada ise yine toplam iterasyon sayısı ve problem kompleksliğine bağlı olarak çözüm süresinin uzun olmasından dolayı test tekrar sayısı 20 olarak belirlenmiştir.

6.1.1. Parametre Ayarları

Temel YAK algoritması, sürü zekasına dayalı bir sezgisel yaklaşım olup, ele alınan problemin kabul edilebilir optimum çözümünü makul bir süre zarfında elde edebilmesi parametre değerlerinin mümkün olan en doğru şekilde belirlenmesi gerekmektedir. YAK algoritmasının parametre ölçütleri; popülasyon büyüklüğü *SN*, toplam çözüm

değerlendirme *Eval* veya iterasyon *Itr* sayısı ve her bir arı için izin verilen başarısız olma sınırı *L* olarak sıralanmaktadır. Bu temel parametre ölçütlerine ek olarak, genetik operatör tabanlı GYAK ve BHYAK algoritmalarının optimum çözüme yakınsama süresini azaltmak amacıyla, popülasyon içerisinde etkileşime girdiği çözümlerin performans değerlerini geliştirme kapasitesine sahip olduğu düşünülen ve popülasyon ortalamasına göre daha iyi performans değerine sahip *e* sayıdaki elit çözümler belirlenmelidir. Ayrıca KVTL, OVTL ve UVTL'lerin kapasitesi *m* ise verimli veya verimsiz hareketlerden en fazla kaçının ilgili listeye kaydedilebileceğini belirtmektedir. Bu çalışma için karşılaştırma amaçlı seçilen ve TKYP üzerinde farklı algoritmalar uygulayan çalışmalarda seçilen parametre değerleri farklılık gösterebilmektedir. Bununla birlikte, karşılaştırma çalışmalarında seçilen parametre değerleri bu çalışmada parametre ayarlarının en doğru şekilde yapılabilmesi için önemli fikir vermektedir.

Gehring ve Bortfeldt 50'den küçük popülasyon hacmine sahip çözüm araştırmalarının genel olarak hedeflenen optimum değerlerinin elde edilmesinde istenilen performansı gösteremediklerine dikkat çekmişler ve yaptık çalışmada popülasyon büyüklüğünü 50 olarak belirlemişlerdir [26]. Bortfeldt ve Gehring bir başka çalışmasında popülasyon büyüklüğünü yine 50 olarak belirlemişlerdir [226]. Dereli ve Daş, optimum parametre değerlerini belirlemek üzere yaptıkları faktöriyel analiz ile yaptıkları çalışma için popülasyon büyüklüğünün 10 yerine 20 olması gerektiği ve özel seçilmiş elit arıların sayısı arttıkça çözüm kalitesinin düştüğü tespitinde bulunup elit arı sayısını 4 olarak belirlemişlerdir [145]. Zhou ve Liu, popülasyon içerisindeki her bir çözümün rassal olarak belirlenen sayıda diğer çözümler ile etkileşime girmesine bağlı olarak çözümlerin pozisyonlarının her iterasyonda değiştiği bir yaklaşım önermeleri sebebiyle toplam çözüm değerlendirme sayısının popülasyon büyüklüğüne bağlı olarak değişmediği çalışmalarında popülasyon büyüklüğünü 10 olarak belirlemişlerdir [227]. TKYP dışında birçok problem üzerinde YAK algoritmasını uygulayan diğer çalışmalarda [6–10] popülasyon büyüklüğü 20 ile 100 arasında değişmektedir. Bu çalışmada ise, 50-75-100 olarak belirlenen popülasyonlar denendiklerinde performans/test süresi bakımından popülasyon büyüklüğü *SN*'nin en verimli olduğu değer 50 ve çözüm geliştirememe limiti ise popülasyon büyüklüğünün yarısı 25 olarak belirlenmiştir.

Çizelge 6.1 Önerilen ve karşılaştırılan yaklaşımların parametre detayları

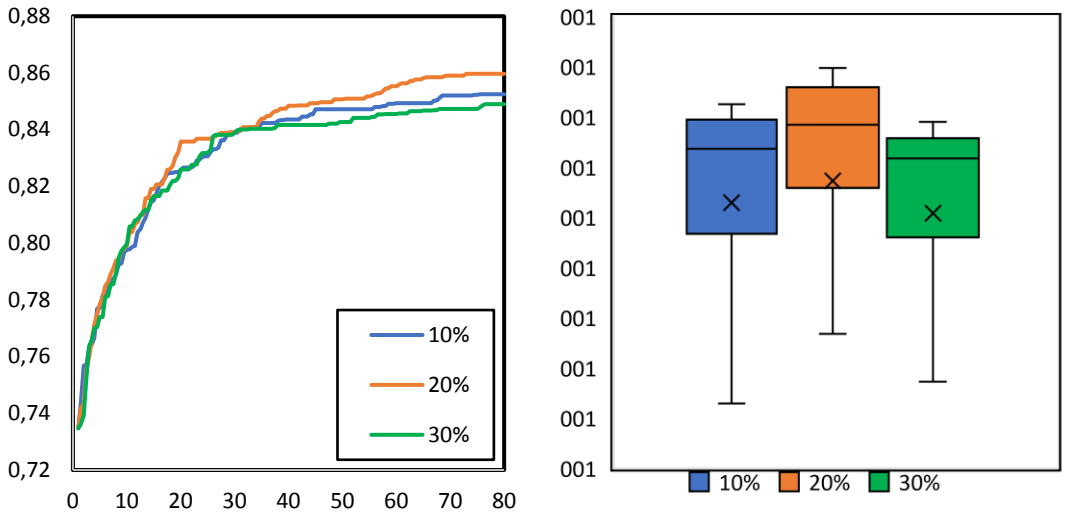
Algoritma	*SN*	*Eval*	*Itr*	*e*	*m*	*L*
GA_GB	50	–	500	–	–	–
TS_BG	–	–	120	–	–	–
CBGAS	50	–	500	–	–	–
GRASP	–	–	50000	–	–	–
Hibrit-BA	20	–	1000	4	–	–
SOA	10	–	20	–	–	–
YAK	50	$5n^2$	$n^2/20$	–	–	25
HYAK	50	$5n^2$	$n^2/20$	–	$n \times SN$	25
GYAK	50	$5n^2$	$n^2/20$	10	–	25
BHYAK	50	$5n^2$	$n^2/20$	10	$n \times SN$	25

Çizelge 6.1'de de görülebileceği üzere TKYP'yi çözmek için uygulanan yaklaşımların yapıları kaynaklı olarak amaç fonksiyonu değerlendirme sayısı *Eval*, ilgili karşılaştırma çalışmasına göre çeşitlilik göstermektedir. Bununla birlikte, eğer ilgili karşılaştırma çalışmaları sağlar ise en azından optimum çözüme yakınsama süreleri açısından bu çalışmada önerilen BHYAK algoritması ile kıyaslanabilirler. Karşılaştırma çalışmaları çözüm araştırmasını belirli bir süre veya iterasyon sayısı ile sınırlandırmaktadırlar. Gehring ve Bortfeldt çözüm araştırmasını hem 500 saniye hem de 500 iterasyon ile sınırlandırmışlardır [26]. Diğer çalışmalarında TS_BG için araştırmayı 120 iterasyon veya yine 500 saniye ile sınırlandırmaktadırlar [226]. Parreno ve ark. kullandıkları GRASP yaklaşımının kaç test ve iterasyonla uygulanması gerektiği ile alakalı parametre tasarımı ön çalışması yapmışlar, 5 ve 10 test seviyelerinde 5000 iterasyon denenmiştir. Son olarak tek test 50000 iterasyon seviyesini de denedikten sonra test sayısındaki değişikliğin optimum çözümün geliştirilmesi açısından bir katkısının olmadığını, fakat iterasyon sayısındaki artışın optimum KDO'yu geliştirmekte etkili olduğu sonucuna varmışlardır [113]. Dereli ve Daş da iterasyon sayısındaki artışın çözüm kalitesini artırdığına dikkat çekmişler, iterasyon sayısını 1000 olarak belirlemişlerdir [145]. Zhou ve Liu ise çalışmalarında iterasyon sayısını 20 olarak belirlemişlerdir [227]. Bu çalışmada, birçok denemeden sonra toplam fonksiyon değerlendirme sayısının en fazla $5n^2$, yani konteynere yerleştirilmesi gereken nesne sayısının karesinin 5 katı olması gerektiği belirlenmiştir. Öte yandan, çalışmada kullanılan yaklaşımlara ait parametre değerlerinin diğer çalışmalardaki yaklaşımlara ait değerlerle karşılaştırılabilmesi maksadıyla, toplam fonksiyon değerlendirme sayısı *Eval*, her 100 değerlendirmenin 1 iterasyona denk

olduğu var sayılarak toplam iterasyon sayısı Itr'ye dönüştürülerek bu değer de $n^2/20$ olarak belirlenmiştir. Ayrıca, bu çalışmada çözülmeye çalışılan veri seti BR01-BR07 ve BR08-BR15 olmak üzere iki kısım halinde ele alındığında, genel olarak ilk kısım için 400 ve ikinci kısım için 700 iterasyondan sonra elde edilen optimum KDO değerinin çok fazla gelişme göstermediği tespit edilmiştir.

Çeşitli algoritmaların uygulandığı çalışmalarda parametre değerlerinin sabit veya araştırma süreci boyunca değişen şekillerde belirlendiği görülebilmektedir. Ele alınan problemin büyüklüğü ve komplekslik seviyesi optimum çözümün bulunabilmesi için gerekli olan asgarî süreyi etkilemektedir. Li ve Yang çalışmalarında iterasyon sayısını ve hafıza büyüklüğünü ele aldıkları problem büyüklüğü ile ilişkilendirerek ona göre belirlemişlerdir [15]. Bu çalışmada ise problem büyüklüğü, tek bir konteynere yerleştirilmesi gereken toplam nesne sayısı n'dir. Hafıza büyüklüğü m ise popülasyon büyüklüğü ile problem büyüklüğünün çarpımı $n \times SN$'den elde edilen değer olarak belirlenmiştir. Bu çalışmada önerilen ve karşılaştırılan çalışmalarda kullanılan yaklaşımlara ait parametre değerleri Çizelge 6.1'de verilmiş olup HCLPMC ve HTS yaklaşımlarının kullanıldığı çalışmalarda herhangi bir parametre değeri belirtilmemiştir.

Bu çalışmada kullanılan bir diğer parametre değeri de kâşif arı safhasında terk edilecek çözümün etkileşime gireceği seçkin çözümlerin sayısı yani elit arı sayısı e'dir. Elit arı sayısını belirlemek için GYAK algoritması, popülasyona oranlı olarak farklı üç seviyede (%10, %20 ve %30) ve BR00 veri seti üzerinde test edilmiş, Şekil 6.1'deki araştırma geçmişi ve seviyeler arası farklılığı gösteren kutu grafiği ile üç farklı seviyenin çözüm araştırmasını nasıl etkilediği gösterilmiştir.



Şekil 6.1. Üç farklı elit arı oranında elde edilen KDO değerleri

Şekil 6.1'de yatay çizgideki değerler iterasyon sayısını, dikey çizgideki değerler ise KDO'yu göstermektedir. Elde edilen sonuçlara göre hiçbir oran seviyesi diğerlerine göre belirgin bir farklılık ($p > .05$) göstermemektedir. Bununla birlikte, %20 seviyesi %85,97 ile en yüksek doluluk oranını elde ederek araştırma geçmişi ve kutu grafiğinde de görüleceği üzere diğerlerinden az da olsa ayrışmaktadır. Sonuç olarak elit arıların popülasyon içerisinde %20 oranına denk gelecek sayıda olmasına karar verilmiştir.

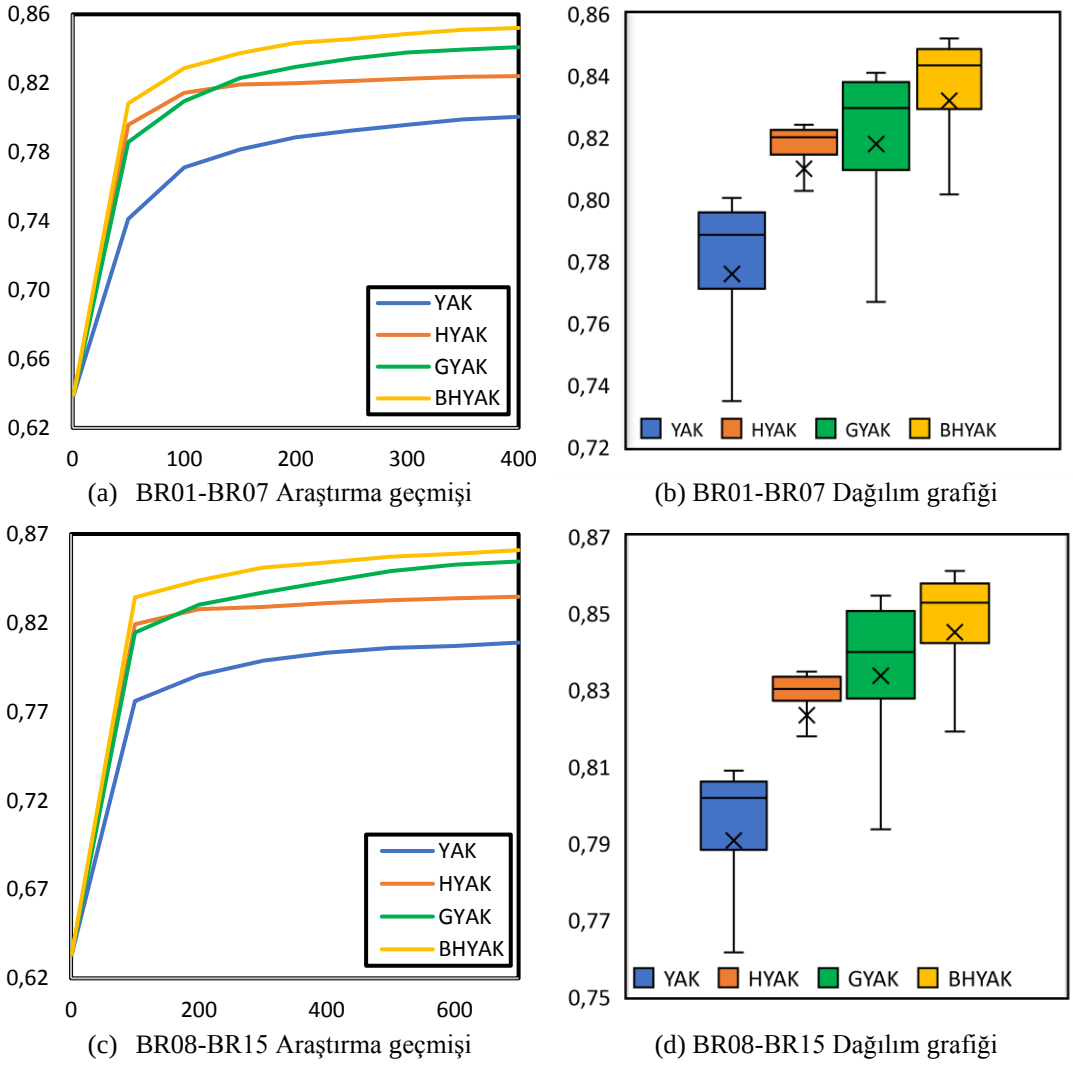
6.1.2. Temel YAK ve Önerilen Algoritmaların Karşılaştırılması

Temel YAK algoritması tek bir çözümden türetilen ilk popülasyonun iterasyonlar boyunca problem çözümlerini geliştirmesini sağlamaktadır. Bu çalışmada her bir BR problem seti için ayrı ayrı ilk popülasyonlar türetilmiş, ardından uygulanan algoritmalar aynı popülasyon üzerinden tekrarlı testlere tâbi tutulmuştur. Çizelge 6.2'de her bir problem seti için elde edilen ortalama optimum KDO değeri, oranlardaki standart sapma ve saniye cinsinden ortalama test süresi verilmektedir. Çizelgede her bir problem setine karşılık gelen değerler, bir önceki bölümde belirtilen kriterlere göre ayarlanan parametre değerlerine bağlı olarak sürdürülen araştırmalar sonucunda elde edilmiştir. Bununla birlikte, problem setlerinin ilk kısmı (BR01-BR07) ortalama olarak optimuma yakın değerlere 400 iterasyon, ikinci kısmı ise (BR08-BR15) 700 iterasyon civarında ulaşmaktadır.

Çizelge 6.2 Temel YAK ve önerilen algoritmaların BR problem setleri için elde ettikleri KDO değerleri

Problem Seti	Temel YAK			HYAK			GYAK			BHYAK		
	KDO	St. Sapma	Test Süresi (sn.)	KDO	St. Sapma	Test Süresi (sn.)	KDO	St. Sapma	Test Süresi (sn.)	KDO	St. Sapma	Test Süresi (sn.)
BR01 (3)	0,8012	0,0015	91,03	0,7882	0,0011	122,67	0,8219	0,0012	117,45	**0,8442**	0,0010	138,33
BR02 (5)	0,7811	0,0011	48,76	0,7980	0,0008	61,12	0,8139	0,0007	62,61	**0,8330**	0,0012	73,55
BR03 (8)	0,8059	0,0018	82,81	0,8424	0,0012	111,34	0,8545	0,0014	81,63	**0,8684**	0,0010	97,24
BR04 (10)	0,8239	0,0014	104,56	0,8344	0,0017	163,76	0,8398	0,0017	122,89	**0,8526**	0,0017	138,67
BR05 (12)	0,7977	0,0022	83,61	0,8218	0,0013	115,11	**0,8420**	0,0009	107,71	0,8417	0,0009	125,41
BR06 (15)	0,8248	0,0020	215,54	0,8583	0,0010	298,45	0,8784	0,0013	282,43	**0,8933**	0,0011	286,81
BR07 (20)	0,8013	0,0010	131,78	0,8354	0,0009	187,79	0,8570	0,0010	144,91	**0,8587**	0,0014	175,14
BR08 (30)	0,8241	0,0012	312,12	0,8500	0,0012	442,23	0,8731	0,0021	342,11	**0,8766**	0,0022	405,05
BR09 (40)	0,8198	0,0017	374,15	0,8525	0,0008	547,65	**0,8697**	0,0016	405,10	0,8693	0,0015	500,25
BR10 (50)	0,8199	0,0015	311,10	0,8467	0,0014	441,12	0,8670	0,0011	341,32	**0,8729**	0,0023	426,69
BR11 (60)	0,8042	0,0009	252,44	0,8246	0,0017	356,89	0,8512	0,0010	273,15	**0,8608**	0,0010	347,29
BR12 (70)	0,8108	0,0015	316,78	0,8347	0,0021	450,12	0,8461	0,0014	325,25	**0,8626**	0,0012	441,28
BR13 (80)	0,8014	0,0010	248,11	0,8248	0,0012	378,54	0,8461	0,0008	270,33	**0,8541**	0,0010	335,61
BR14 (90)	0,8089	0,0017	191,92	0,8300	0,0014	272,14	0,8517	0,0020	210,76	**0,8557**	0,0012	260,52
BR15 (100)	0,7995	0,0014	197,32	0,8266	0,0018	294,78	0,8543	0,0015	232,94	**0,8569**	0,0016	278,10
Ort. BR01-07	0,8051	–	108,30	0,8255	–	151,46	0,8439	–	131,38	0,8560	–	147,88
Ort. BR08-15	0,8111	–	275,49	0,8362	–	397,93	0,8574	–	300,12	0,8636	–	374,35
Ort.	0,8083	–	197,47	0,8312	–	282,91	0,8511	–	221,37	0,8601	–	268,66

Şekil 6.2'deki grafikler, iki sınıf olarak (BR01-BR07 ve BR08-BR15) incelenen problem seti üzerinde uygulanan temel YAK ve önerilen algoritmalar ile elde edilen ortalama KDO değerlerini göstermektedir. Sol taraftaki grafikler algoritmaların optimum çözümü arama sürecini gösterirken sağ taraftaki grafikler ise bu süreç bütününde algoritma performanslarının birbirlerinden nasıl ayrıştığını göstermektedir. Temel YAK ve önerilen algoritmalar kendi aralarında tek yönlü varyans analizi (ANOVA) ve Fisher'in önemli en küçük fark (LSD) post-hoc testi istatistiksel olarak analiz edilmiş [228] ve uygulanan algoritmaların her birinin çözüm araştırma geçmişi dağılımlarının birbirinden farkının, her iki problem sınıfı için de önemli derecede olduğu tespit edilmiştir ($p < .05$). KDO dağılımı farklılıkları ise, sağ tarafta algoritmaların renkli kutularla temsil edildiği farklılık grafiklerinde gösterilmektedir.



(a) BR01-BR07 Araştırma geçmişi

(b) BR01-BR07 Dağılım grafiği

(c) BR08-BR15 Araştırma geçmişi

(d) BR08-BR15 Dağılım grafiği

Şekil 6.2. Temel YAK ve önerilen algoritmaların BR sonuçları genel durumu

Şekil 6.2'de (a) ve (b) grafikleri BR veri setinin ilk yedi probleminde elde edilen KDO değerlerinin iterasyona bağlı ortalama değişimini anlatırken, (c) ve (d) grafikleri ise son sekiz problemin çözümünde elde edilen ortalama KDO değerlerinin iterasyonlara bağlı değişimini anlatmaktadır. Sol tarafta bulunan (a) ve (c) grafikleri, her bir algoritmanın çözüm araştırma sürecinde elde ettiği KDO değerlerinin iterasyonlara bağlı olarak değişimi gösterilmektedir. Sağ tarafta ise çözüm araştırma süreci boyunca her bir iterasyonda elde edilen KDO değerlerinin merkezi eğilim ölçülerini ve dağılımlarını özet olarak göstermektedir.

Şekil 6.2'de sağ tarafta yer alan, iterasyon sayısına göre değişen ve popülasyon ortalama performans değerlerinden oluşan anakütlenin dağılımının yapısına ait merkezi eğilim ölçülerinin gösterildiği kutu grafiğinde; alt ve üst sınır çizgileri (bıyıklar) alt ve üst uç gözlem değerlerini, çarpı işareti anakütle ortalamasını, kutunun ortasındaki çizgi ise medyanı göstermektedir. Anakütle karşılaştırmasında önemli bir ölçüt olarak yer alan medyanlar karşılaştırıldığında Şekil 6.2 (b) ve (d)'de algoritmalara ait gözlem dağılımlarının medyanının her iki problem sınıfında da farklı değerlere sahip olduğu görülmektedir.

Mavi renkle temsil edilen temel YAK algoritması bariz bir şekilde en kötü performansa sahipken, turuncu renkle temsil edilen HYAK algoritması ile yeşil renkle temsil edilen GYAK algoritmasının uygulandıkları her iki problem sınıfında da çözüm araştırma sürecinin ilk çeyreği boyunca başa baş performans gösterseler de, araştırma sürecinin bütününe bakıldığında GYAK algoritması lehine iki algoritmanın birbirinden önemli seviyede farklı düzeyde çözümler elde ettiği ($p < .05$) görülmektedir. Temel YAK algoritmasının yerel araştırma kabiliyetini geliştirerek, yerel araştırma bölgelerindeki verimli çözümlerin hızlıca keşfedilmesini sağlayan HYAK algoritması her ne kadar ilk çeyrekte kısa bir süreliğine GYAK algoritmasına üstün gelse de YAK algoritmasının çözüm çeşitlendirme kabiliyetini geliştiren GYAK algoritması ilerleyen iterasyonlarda tekrar üstünlük sağlamaktadır. Ayrıca HYAK algoritmasının TKYP üzerinde uygulamasında, yerel araştırmada gösterdiği performansı çözüm çeşitlendirmede gösteremeyerek, araştırmanın erken dönemlerinde elde edilebilir optimum çözümün altında bir değerin ötesinde çözümü geliştiremediği görülmektedir. Çözüm araştırma süresinin çok kısıtlı olduğu durumlarda hızlı ve etkili

çözüm için GYAK algoritması yerine HYAK algoritması tercih edilebilir. Bu da göstermektedir ki YAK algoritmasının TKYP üzerinde uygulamasında genetik operatörler daha etkin bir takviye aracıdır.

Sarı renkle temsil edilen BHYAK algoritması hem yerel araştırma hem de çözüm çeşitlendirme araçlarını bir arada kullanarak kısa sürede yerel ve global araştırmada diğer üç algoritmaya göre önemli ölçüde ($p < .05$) daha iyi sonuçlar elde etmiş, araştırmanın başından sonuna kadar diğer algoritmalara olan üstünlüğünü korumuştur. Bununla birlikte hafıza mekanizmasının uzun vadede etkisini yitirmesi sebebiyle, GYAK algoritması ile BHYAK algoritması arasındaki fark araştırma sonlarına doğru iyice kapanmakta ve hatta BR05 ve BR09 problem setlerinde GYAK algoritması ortalamada BHYAK algoritmasından daha iyi sonuçlar elde etmektedir. Bu durum hafıza mekanizmasının uzun vadede etkisini yitirdiğinin yanı sıra genetik operatör ile birlikte entegre edildiği BHYAK algoritmasının çözüm araştırma kapasitesini de kısıtladığını göstermektedir. Hafıza mekanizmasının bu olumsuz etkisine rağmen BHYAK algoritmasının LSD post-hoc testi sonuçlarına göre çoğu problem setinin çözümünde diğer üç algoritmaya üstün geldiği görülmektedir. Sonuç olarak temel YAK algoritmasının geliştirilmesi hususunda hafıza mekanizmasının katkısı genetik operatörlerin katkısı yanında son derece düşük kalmaktadır.

Algoritmaların ele alınan problemi çözmedeki etkinliği yanı sıra problemi ne kadar sürede çözdükleri de bir diğer karşılaştırma ölçütüdür. Konteynere yerleştirilecek nesnelerin heterojen dağılımından ve sayısından bağımsız olarak, bütün algoritmalar optimum sonuçları elde etmek için en kısa süreye BR02, en uzun süreye ise BR09 problem setlerini çözmek için ihtiyaç duymuşlardır. HYAK algoritması, hafıza mekanizmasının yapısından dolayı optimum çözümleri, diğer algoritmalarla karşılaştırıldığında en yüksek test süresinde elde ettiği görülmektedir. Buna sebep olan unsurlardan en öncelikli olanı ise KVTL'nin her yeni çözüm türetilmesi sürecinde tekrar kontrol edilmesi, eğer mevcut çözümü türetilen yeni çözüme taşıyan hareket KVTL'de mevcut ise kabul edilmeyip tekrar yeni bir hareket türetilmesidir. Ayrıca KVTL, elverişli olmayan hareketlerin türetilmesi ihtimalinin daha yüksek olması sebebiyle, tabiî olarak diğer tabu listelerine göre daha çok hareket ihtiva etmektedir. Bu da KVTL'nin her kontrolünde daha çok hareketin gözden geçirilmesi ve kontrol

için gereken sürenin diğer araştırma araçlarının kullanılması için gereken süreden daha fazla olacağı anlamına gelmektedir. Diğer yandan OVTL, elverişli hareketlerin türetilme ihtimalinin göreceli olarak daha az olması sebebiyle araştırma başlangıcında belirlenen hafızanın ortalama olarak yarısını kullanmaktadır. UVTL ise yalnızca kâşif arı safhasında terk edilecek çözümleri, popülasyonun performans ortalamasından daha iyi ve popülasyonun en iyi çözümlerine yakın bir performans gösteren çözümlere taşıyan hareketleri kaydettiği ve bu tür hareketlerin nadiren türetilebilmesi sebebiyle çok az sayıda geçiş hareketi ihtiva etmektedir. HYAK algoritmasında bu üç tabu listesi de tekrar tekrar kontrol edildiği için optimum çözüme yakınsama süresi uzamaktadır. Hafıza kapasitesi düşürüldüğünde ise elde edilen optimum değerler kötüleşmektedir.

Hafıza mekanizmasının, çözüm araştırmasının erken safhasında YAK algoritmasını, elde edebileceği optimum çözüm değerinden daha kötü bir çözüm elde etmesini sağlayıp araştırmanın ilerleyen safhalarında da bu kötü çözümü geliştirememesi ve çözüm araştırması için GYAK ve BHYAK algoritmalarına göre daha fazla süreye ihtiyaç duyması sebebiyle, genetik operatörlerin kullanımı ile karşılaştırıldığında, TKYP'nin çözümünde temel YAK algoritması için verimli bir takviye aracı olamayacağı görülmektedir. Öte yandan BHYAK algoritmasının performansına bakıldığında, hafıza mekanizması her ne kadar etkisini kısa sürede yitirse de genetik operatörler ile birlikte kullanıldığında etkisinin süresi arttığı görülmektedir.

6.1.3. BHYAK Algoritmasının Literatürdeki Yaklaşımlarla Karşılaştırılması

Temel YAK ve önerilen algoritmalar kendi içerlerinde karşılaştırıldıklarında BHYAK algoritmasının TKYP'yi çözmede bariz bir şekilde daha iyi performans gösterdiği görülmektedir. Bu bölümde ise BHYAK algoritmasının TKYP'yi çözmedeki performansı yukarıda listesi verilen ve yakın zamana kadar literatürde kullanılmış yaklaşımların performansı ile karşılaştırılmaktadır. Karşılaştırma amaçlı ele alınan çalışmaların birçoğunda test süreleri verilmese de birtakım çalışmalarda önerdikleri yaklaşımların test edilmesi süresini kısıtlı tutmuşlardır. Bu yüzden Çizelge 6.3'te sadece karşılaştırılan çözüm yaklaşımlarının elde ettikleri KDO değerleri verilmektedir.

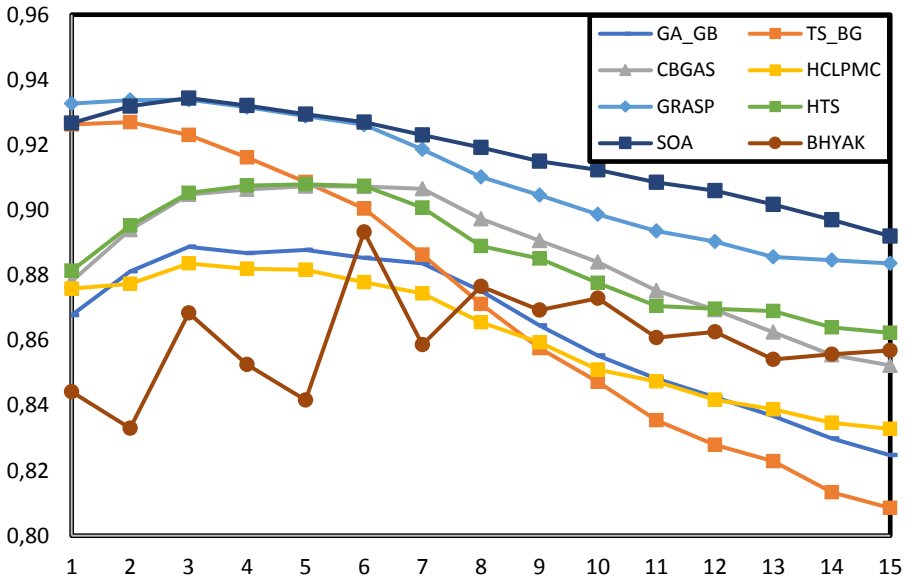
HCLPMC yaklaşımı dışındaki yaklaşımlar ve önerilen BHYAK algoritması sadece yönlendirme ve nesnelerin yığından düşmeden durabilmesi kısıtını dikkate alarak çözüm arayışında bulunurken, HCLPMC bu iki kısıta ek olarak konteynerin dengeli yüklenmesi, toplam yük ağırlığı ve nesnelerin üstüne binen yük ağırlığı kısıtlarını da dikkate almaktadır. Çizelge 6.3'te kalın olarak gösterilen oranlar ilgili problem seti için elde edilen en iyi değeri göstermektedir. Temel YAK ve önerilen algoritmaların tek yönlü ANOVA ve LSD testleri ile istatistiksel olarak karşılaştırılabilirken, BHYAK algoritması ve literatürdeki diğer yaklaşımların istatistiksel karşılaştırması, problem seti sayısının az olması ve verilerin normal dağılım göstermemesi sebebiyle parametrik olmayan istatistik tekniklerinden Mann Whitney-U testi ile yapılabilmektedir [229].

Çizelge 6.3 BHYAK algoritması ve literatürdeki yaklaşımların BR01-15 sonuçları

Problem Seti	GA_GB	TS_BG	CBGAS	HCLPMC	GRASP	HTS	Hibrit-BA	SOA	BHYAK
BR01 (3)	0,8677	0,9263	0,8781	0,8759	**0,9327**	0,8814	0,8341	0,9267	0,8442
BR02 (5)	0,8812	0,9270	0,8940	0,8773	**0,9338**	0,8952	0,8460	0,9319	0,8330
BR03 (8)	0,8887	0,9231	0,9048	0,8836	0,9339	0,9053	0,8542	**0,9344**	0,8684
BR04 (10)	0,8868	0,9162	0,9063	0,8820	0,9316	0,9075	0,8519	**0,9321**	0,8526
BR05 (12)	0,8878	0,9086	0,9073	0,8817	0,9289	0,9079	0,8511	**0,9294**	0,8417
BR06 (15)	0,8853	0,9004	0,9072	0,8778	0,9262	0,9074	0,8469	**0,9270**	0,8933
BR07 (20)	0,8836	0,8863	0,9065	0,8744	0,9186	0,9007	0,8399	**0,9231**	0,8587
BR08 (30)	0,8752	0,8711	0,8973	0,8655	0,9102	0,8889	–	**0,9192**	0,8766
BR09 (40)	0,8646	0,8576	0,8906	0,8593	0,9046	0,8851	–	**0,9150**	0,8693
BR10 (50)	0,8553	0,8473	0,8840	0,8510	0,8987	0,8776	–	**0,9123**	0,8729
BR11 (60)	0,8482	0,8355	0,8753	0,8474	0,8936	0,8706	–	**0,9085**	0,8608
BR12 (70)	0,8425	0,8279	0,8694	0,8418	0,8903	0,8697	–	**0,9059**	0,8626
BR13 (80)	0,8367	0,8229	0,8625	0,8388	0,8856	0,8690	–	**0,9017**	0,8541
BR14 (90)	0,8299	0,8133	0,8555	0,8347	0,8846	0,8640	–	**0,8970**	0,8557
BR15 (100)	0,8247	0,8085	0,8523	0,8328	0,8836	0,8623	–	**0,8920**	0,8569
Ort. BR01-07	0,8830	0,9126	0,9006	0,8790	**0,9294**	0,9008	0,8463	0,9292	0,8560
Ort. BR08-15	0,8471	0,8355	0,8734	0,8464	0,8939	0,8734	–	**0,9065**	0,8636
Ort.	0,8639	0,8715	0,8861	0,8616	0,9105	0,8862	–	**0,9171**	0,8601

Çizelge 6.3'te Hibrit-BA algoritması haricindeki çözüm yaklaşımlarının 15 problem seti için elde ettikleri KDO değerleri yer almaktadır. 01-15 arası numaralandırılan problem setlerinin yanında parantez içerisinde yer alan değerler ise 3 ile 100 arasında artan şekilde değişen, konteynere yerleştirilebilir nesne çeşidi sayısını göstermektedir. Şekil 6.3'te de gösterildiği üzere, karşılaştırılan yaklaşımların elde ettikleri sonuçlar,

ilk problem sınıfında önce artan trendde iken, ikinci problem sınıfında ise problem heterojenliği arttıkça elde edilen çözüm değerlerinin azaldığı bir trende girmektedir. BHYAK algoritması için ise elde edilen çözümlerin problem setine göre değişimi diğer yaklaşımlara göre farklılık göstermekte, ilk sınıf problem setinde düzensiz değişimler gözlemlenirken ikinci problem sınıfında ise karşılaştırılan yaklaşımlara nispetle daha kararlı ve yataya yakın bir trende girmektedir. Karşılaştırılan algoritmaların çoğu, ilk problem sınıfının daha homojen yapıda olması neticesinde nesneleri öbekler halinde kolayca konteynere yerleştirebiliyorken BHYAK algoritmasının tamamen rassal arama yaklaşımıyla çözüm araştırmasını bir yerde yanlış yönlendirmesi sebebiyle çalışmada önerilen algoritmadan daha iyi sonuçlar elde etmektedir. İkinci problem sınıfında nesne çeşitliliğinin artması neticesinde nesne gruplandırma yaklaşımının etkisinin azalmasıyla BHYAK algoritmasının TKYP çözümündeki etkisi daha görünür hale gelmekte ve nesne çeşitliliği arttıkça aradaki makas kapanarak karşılaştırılan yaklaşımlara nispetle BHYAK algoritması ortalama bir performans sergilemektedir.



Şekil 6.3. BHYAK algoritması ve karşılaştırılan yaklaşımların BR sonuç trendleri

GA_GB ve TS_BG algoritmaları sırasıyla genetik operatörlerin ve tabu listelerinin TKYP için çözüm araştırmasını nasıl yönlendirdiği ile alakalı verdiği temel fikirler, bu çözüm araştırma araçlarının temel YAK algoritmasının geliştirilebilmesi için ne

şekilde kullanılabileceği hususunda bu çalışmayı yönlendirmektedir. Bu sebeple önerilen BHYAK algoritmasının GA_GB ve TS_BG algoritmaları karşısındaki durumu üzerinde özellikle durulmaktadır. GA_GB ve TS_BG kendi aralarında karşılaştırıldıklarında TS_BG'nin özellikle ilk sınıf problem setlerinde (BR01-BR07) GA_GB'ye göre daha sonuçlar elde ederken, nesne çeşitliliği arttıkça ikinci sınıf problem setlerinde (BR08-BR15) GA_GB'nin gerisine düştüğü görülmektedir. Öte yandan, GA_GB ortalama 360 saniyeden az bir sürede optimum çözüme yakınsayabilirken, TS_BG ise araştırma süresi limiti olarak belirlenen 500 saniye sonunda optimum KDO değerlerini elde edebilmektedir. Bu da GA_GB'nin özellikle ikinci sınıf problem setlerinde TS_BG'ye karşı bariz üstün konumda olduğunu göstermektedir. Bu iki temel yaklaşımın kullandığı araçların, temel YAK algoritmasını geliştirmek için bir arada kullanıldığı BHYAK algoritmasının elde ettiği KDO değerleri ise GA_GB ve TS_BG ile karşılaştırıldığında önemli seviyede ($p < .05$) daha iyi konumdadır. Ayrıca test sürelerine bakıldığında ise BHYAK algoritmasının GA_GB algoritmasına üstün geldiği ikinci sınıf problem setlerinde KDO değerlerini ortalama olarak 269 saniyede elde ettiği görülmektedir. Ayrıca GA_GB'nin ikinci bir kısıt olarak uyguladığı 500 iterasyon sınırı dikkate alınacak olsa dahi, BHYAK algoritması %85,72'lik bir doluluk yüzdesi ile GA_GB'den daha iyi performans gösterebildiğini kanıtlamaktadır. BHYAK algoritması herhangi bir nesne gruplandırma yaklaşımı kullanmadan, tamamen rassal türetilen nesne atama ve yönlendirme değişkenlerinden oluşan NYD'ler ile problemi çözme yolu benimserken, CBGAS algoritması, GA_GB algoritmasının geliştirilmiş hali olup, kullandığı nesneleri katmanlar halinde gruplayıp konteynere yerleştirme yaklaşımı ile BR14 ve BR15 problem seti dışında diğer problem setlerinde ortalama olarak önerilen algoritmadan daha iyi sonuçlar elde etmiştir.

HCLPMC tüm beş kısıtı dikkate almak yerine diğer karşılaştırma yaklaşımlarında olduğu gibi temel iki kısıtı dikkate alarak nesneleri yerleştirmeye çalışması, BHYAK algoritmasından daha iyi KDO değerleri elde etmesini sağlayabilirdi. Mevcut duruma rağmen HCLPMC sadece nesne yerleştirme yaklaşımını iyileştirerek dahi GA_GB ve TS_BG algoritmalarından daha iyi KDO değerleri elde etmektedir. GRASP ve HTS ise bir nesne yerleştirildikten sonra sıradaki adayı seçerken önceki nesne yerleştirildikten sonra konteyner içerisinde arta kalan dikdörtgensel alanları en üst

seviyede doldurabilecek en uygun olan nesneyi bulmaya çalıştığı için BHYAK algoritmasına göre daha iyi çözümler elde edebilmektedirler.

Nesneleri, katmanlar halinde yerleştirmeye çalışan Hibrit-BA ve bloklar halinde yerleştirmeye çalışan SOA ise GRASP ve HTS'den farklı olarak sürü optimizasyonu temelli çözüm arama yaklaşımı kullanmaktadırlar. Hibrit-BA, ilk sınıf problem seti için elde ettiği KDO değerleri ile TKYP çözümü hususunda elverişli bir yaklaşım olduğunu göstermesine rağmen daha iyi çözümler elde edebilmesi için kullandığı çözüm arama stratejisinin geliştirilmesi gerekmektedir. YAK algoritması ile benzer yönleri olan Arı Algoritmasını temel alan Hibrit-BA algoritmasının TKYP için elde ettiği sonuçlar sadece ilk sınıf problem seti ile sınırlı olduğu için, BHYAK algoritması ile yeterli seviyede karşılaştırılması mümkün değildir. Bununla birlikte, iki algoritma karşılaştırıldıklarında BHYAK algoritmasının ortalama olarak daha iyi sonuçlar elde ettiği görülmektedir. Öte yandan yakın zamanda geliştirilmiş bir sezgisel yaklaşım olan SOA, farklı türdeki nesneleri devasa büyüklükteki bloklar halinde konteynere yerleştirebilmesini sağlayan kompleks bir stratejisi ile, hemen hemen bütün problem setlerinde en iyi çözüm değeri elde ederek güçlü bir algoritma olduğunu kanıtlamaktadır. Güçlü bir çözüm yaklaşımı olduğu Şekil 6.3'teki grafikte SOA'nın elde ettiği çözümlerin problem setine göre değişiminin çok kısıtlı olmasından ve nesne çeşitliliğindeki artıştan çok fazla etkilenmemesinden de anlaşılmaktadır.

Sonuç olarak, BHYAK algoritması literatürdeki belli başlı yaklaşımlar ile karşılaştırıldığında çalışmada önerilen 15 problem setinin hiçbirinde en iyi çözüm değerini sunamamıştır. Ayrıca diğer algoritmalara karşı bariz bir üstünlük sağlayan SOA'nın elde ettiği değerlere bakılacak olursa, BHYAK algoritmasının TKYP'nin çözümü için çok da etkili bir yaklaşım olmadığı görülmektedir. Bunun başlıca sebebi olarak, nesneleri hiçbir sınıflandırma işlemine tâbi tutmaksızın tamamen rassal türetilen değişken değerlerine sahip NYD'lerin evrimsel gelişimi yolu ile çözümleri geliştirmeye çalışmak gösterilebilir. Bu da yerleştirilecek nesne sayıları ve rassal aramaya bağlı hızla değişebilen çözümler dikkate alındığında, bütün NYD boyunca doğru nesne atama ve yönlendirme çiftlerini sıralı halde bulmayı zorlaştırmaktadır. Bununla birlikte GA_GB, TS_BG ve HCLPMC yaklaşımları ile karşılaştırıldığında, kabul edilebilir bir zaman dilimi içerisinde BHYAK algoritması TKYP için elverişli

çözümler türetebilmektedir. Ayrıca, BHYAK algoritmasından daha iyi sonuçlar elde ede yaklaşımlar, kullandıkları konteyner yükleme stratejileri sayesinde hızlıca KDO değerlerini geliştirip farklarını ortaya koydukları düşünüldüğünde, BHYAK algoritmasının diğer algoritmalarla TKYP'ye çözüm türetme hususunda denk bir şekilde karşılaştırılabilmesi için öncelikle nesne sınıflandırmaya dayalı konteyner yükleme stratejilerinden de faydalanabileceği şekilde geliştirilmesi gerekmektedir.

6.2. ÇKYP DENEYLERİ VE BULGULARI

Bu çalışmada önerilen yaklaşımlar öncelikle kendi içlerinde, Martello ve ark. tarafından belirtilen kurallara göre [36] türetilmiş problem setleri üzerinden karşılaştırmalar yapılmıştır. Türetilen veri setinde D = G = Y = 100 olan konteynerlere bahsi geçen kurallara göre boyutları beş tür halinde rassal olarak türetilmiş nesneler, Bölüm 4.1.2'de belirtilen şekilde NYD'lerde sıralanarak ilk uygun konteynere yerleştirme (First-Fit) yaklaşımı ile rassal olarak belirlenen yönlendirme değişken değeri dikkate alınarak yerleştirilir. Problem setleri üç sınıf halinde olup her sınıftaki nesne tür sayısı $k = 5$, nesne sayıları ise sırasıyla 20, 50 ve 100'dür. Her bir problem seti türetilirken bir nesne türü toplam nesne sayısının %60'ını, diğer nesne türleri ise toplam nesne sayısının %10'unu oluşturmaktadırlar. Ayrıca her nesne türü için rassal olarak beş farklı boyut belirlenip her bir sınıf için 25 toplamda ise 75 problem seti türetilmiştir.

Ardından, Bölüm 6.1.3'te olduğu gibi önerilen algoritmalar arasında türetilen problem setlerini çözmede en iyi performansı gösteren BHYAK algoritması, yine literatürde sıkça kullanılan ve Ivancic tarafından türetilen IMM problem setini [230] çözmedeki performansı üzerinden, literatürden seçilen aşağıdaki çalışmalarda önerilen yaklaşımlar ile karşılaştırılmıştır:

- BR: Bischoff ve Ratcliff tarafından Duvar Örme (DÖ) (1995), N/A [77]
- EE: Eley tarafından Eşzamanlı Yerleştirme (2002), Pentium/200 MHz PC [49]
- SA: Ceschia ve Schaerf tarafından Tavlama Benzetim (TB) (2011), N/A [37]
- 2C: Menghani ve Guha tarafından GA (2016), N/A [231]

IMM veri seti 47 problem setinden oluşmaktadır. 2C yaklaşımı yönlendirme kısıtını dikkate alırken, BHYAK algoritması ve karşılaştırıldığı diğer yaklaşımlar yönlendirme kısıtını dikkate almadan nesneleri yerleştirmeye çalışmaktadırlar. Karşılaştırılan çalışmalarda net bir test sayısı belirtilmese de BHYAK algoritması her bir problem seti için 20 defa test edilmiştir.

6.2.1. Parametre Ayarları

Karşılaştırılan çalışmalar arasında sadece SA ve 2C evrimsel gelişim temelli yaklaşımlar olduğu için BHYAK algoritmasının parametre ayarları yapılırken yalnız bu iki çalışma dikkate alınmıştır. Bu çalışmada TA yaklaşımının kullandığı çözüm araştırma araçlarını da kullanan BHYAK algoritması önerilmiş olsa da Ceschia ve Schaerf'in çalışmasında [37] TB yaklaşımı temel alınan IMM'den farklı bir veri setinde TA'ya üstün geldiği için TA yaklaşımının IMM veri seti üzerinde test edilmesini gerekli görmemişlerdir. TB için başlangıç sıcaklığı $T_0 = 50$, son sıcaklık $T_{min} = 0{,}001$, her bir sıcaklıkta mevcut çözümün komşu çözüm sayısı $\sigma_N = 1000$, ve soğuma derecesi $\tau = 0{,}998$ olarak belirlenmiştir. Buradan hareketle her sıcaklıkta araştırmanın yapıldığı çözüm sayısı popülasyon büyüklüğü olarak değerlendirildiğinde bu değer 1000 olmaktadır. Bölüm 6.2.3'te de bahsedildiği üzere SA çözüm değerlerini ortalama olarak 290 saniyede elde etmektedir. Bu da ikili karşılaştırmada BHYAK algoritması için bir kriter olarak kabul edilebilir.

Menghani ve Guha ise yine bu çalışmanın temel aldığı yaklaşımlardan birisi olan GA'yı kullanarak IMM problem setlerini çözmeye çalışmaktadırlar [231]. 2C yaklaşımı için seçilen parametre değerleri; popülasyon büyüklüğü $pop = 100$, yeniden popülasyon türetme tekrar sayısı $gen = 20$, popülasyonda çaprazlama işlemine girecek olan birey sayısı $ncross = 50$ ve mutasyon olasılığı ise $P_m = \%5$ olarak belirlenmiştir.

Bölüm 6.1'de TKYP için literatürde birçok çalışma evrimsel gelişim temelli yaklaşım önerirken aynı durum ÇKYP için geçerli değildir. BHYAK algoritmasını karşılaştırmak için seçilen dört çalışmadan sadece SA ve 2C evrimsel gelişim temelli yaklaşım önerirken, nesne yerleştirme sezgiseli temelli çözüm öneren BR ve EE

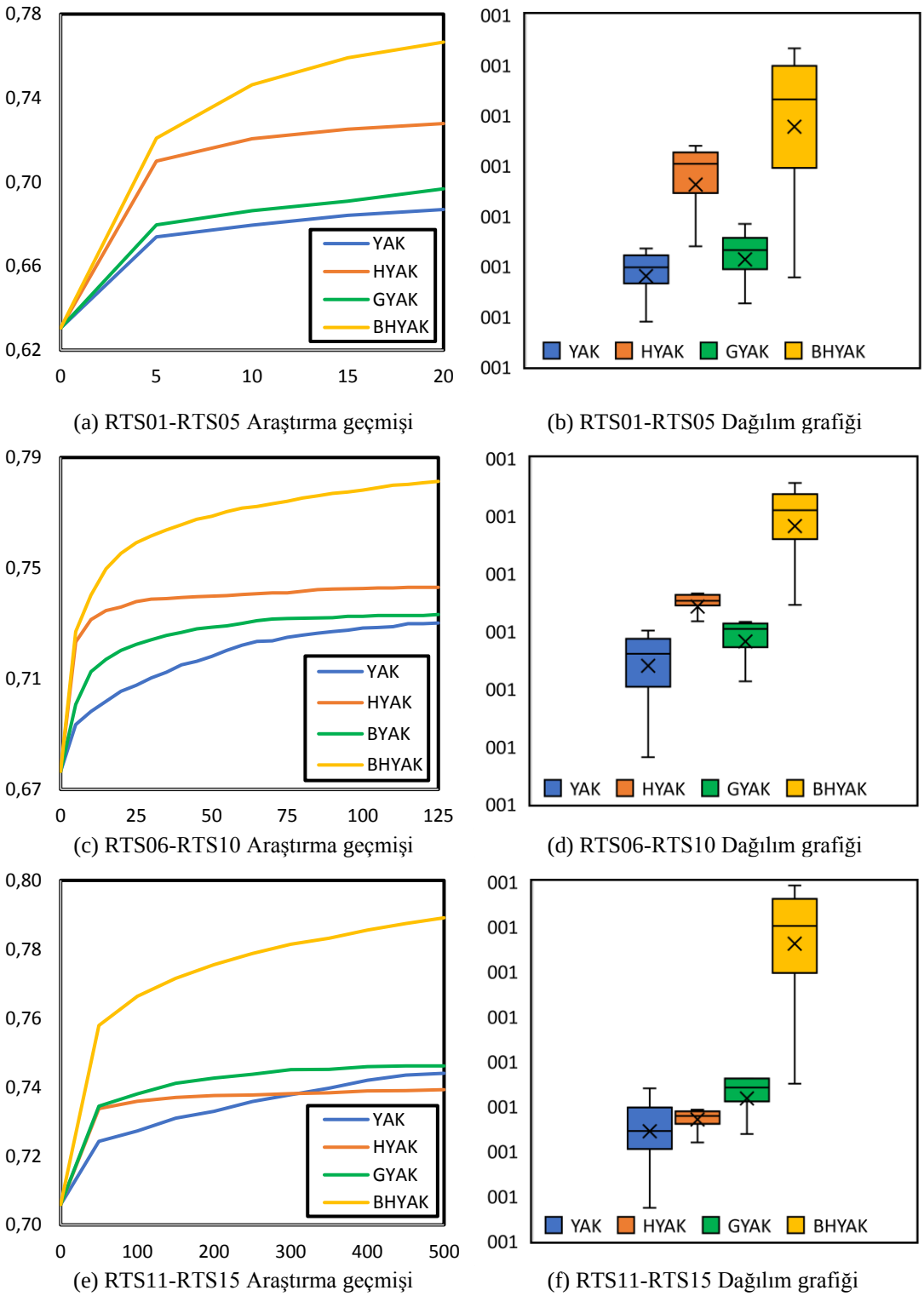
yaklaşımları, literatürde IMM veri seti için çözüm üreten referans çalışmalar oldukları karşılaştırma yaklaşımları olarak seçilmişlerdir. Öte yandan, sürü zekâsı temelli olması itibariyle BHYAK algoritması ile sadece 2C yaklaşımının benzerlik gösterdiği değerlendirildiğinde, TKYP çözümünde kullanılan parametre değerlerinin aynı şekilde kullanılmasının daha iyi olacağı görülmektedir. Ayrıca, bu çalışmada önerilen algoritmaların kendi içlerinde aynı parametre değerleri ile karşılaştırılması durumunda, algoritmaların problem farklılığından kaynaklanan muhtemel karakteristik değişimleri, parametre değerleri farklılığından bağımsız şekilde gözlemlenebilecektir.

6.2.2. Temel YAK ve Önerilen Algoritmaların Karşılaştırılması

ÇKYP'de esas olan mümkün olan en az konteynerin kullanılmasıdır. Bu aynı zamanda konteynerlerin ortalama doluluk oranlarının da maksimize edilmeye çalışılması anlamına gelir. Bu bölümde temel YAK ve önerilen algoritmaların karşılaştırılmasında KDO değerleri dikkate alınmaktadır. Kendi içinde değerlendirilmeye alınan temel YAK ve önerilen algoritmaların, uygulandıkları rassal türetilmiş problem setleri (RTS01-RTS15) için elde ettikleri KDO değerleri, standart sapmaları ve testler için harcanan ortalama süre saniye cinsinden Çizelge 6.4'te verilmektedir. RTS problem setlerine ait değerler, her birisinin beş alt problem setinde elde edilen KDO değerlerinin ortalama değerini ve bu ortalama değerlerin standart sapmalarını göstermektedir. Bölüm 6.1.2'de olduğu gibi ÇKYP için yapılan karşılaştırmada da kullanılan algoritmalar tek bir çözüm değerinden türetilen aynı popülasyon üzerinden çözüm aramaya başlamaktadırlar. İlk beş problem setinde (RTS01-05) 20, ikinci beş problem setinde (RTS06-10) 50 ve son beş problem setinde (RTS11-15) ise 100 nesneden tek bir yerleştirilmeyen kalmayıncaya dek eş boyuttaki konteynerlere atanmaktadır.

Temel YAK ve önerilen algoritmaların RTS üzerinde uygulandıklarında elde ettikleri ve parametrik varsayımı sağlamayan KDO değerleri ikili olarak Mann-Whitney U testi ile analiz edildiğinde, bütün problem setlerinde en iyi performansı göstermesi yanı sıra RTS veri setinin bütünü için BHYAK algoritmasının diğerlerinden önemli seviyede ($p < .05$) farklı olduğu görülmektedir. Öte yandan problem sınıfları özelinde bakılıp

elde edilen sonuçlar değerlendirildiğinde hiçbir yaklaşımın farklılık göstermediği ortaya çıkmaktadır.



(a) RTS01-RTS05 Araştırma geçmişi

(b) RTS01-RTS05 Dağılım grafiği

(c) RTS06-RTS10 Araştırma geçmişi

(d) RTS06-RTS10 Dağılım grafiği

(e) RTS11-RTS15 Araştırma geçmişi

(f) RTS11-RTS15 Dağılım grafiği

Şekil 6.4. Temel YAK ve önerilen algoritmaların RTS sonuçları genel durumu

Test süreleri açısından Çizelge 6.4'teki değerlerin birbirine oranı, Çizelge 6.2'deki test sürelerinin birbirine oranları ile paralellik göstermekle beraber, algoritmaların üzerinde uygulandığı RTS veri setindeki problemlerin BR problemlerine nazaran daha az kompleks yapıda olması sebebiyle ihtiyaç duyulan süreler daha azdır.

Şekil 6.4'te (a) ve (b) grafikleri RTS veri setinin ilk beş probleminde elde edilen KDO değerlerinin iterasyona bağlı ortalama değişimini anlatırken, (c) ve (d) grafikleri ikinci beş problemin çözümünde elde edilen ortalama KDO değerlerinin iterasyonlara bağlı değişimini ve (e) ile (f) grafikleri de son beş problem için elde edilen ortalama KDO değerlerinin iterasyonlara bağlı değişimini anlatmaktadır. Yine sol tarafta bulunan (a), (c) ve (e) grafikleri, her bir algoritmanın çözüm araştırma sürecinde elde ettiği KDO değerlerinin iterasyonlara bağlı olarak değişimi gösterilmektedir. Sağ tarafta ise çözüm araştırma süreci boyunca her bir iterasyonda elde edilen KDO değerlerinin merkezi eğilim ölçülerini ve dağılımlarını özet olarak göstermektedir. BR veri setinde gösterilen davranışların aksine RTS veri setindeki problemler üzerinde uygulanan temel YAK ve önerilen algoritmaların farklı davranışta bulundukları ve problem karmaşıklığı arttıkça özellikle HYAK ve GYAK algoritmaların performanslarında bariz değişimlerin olduğu görülmektedir.

TKYP'nin aksine, temel YAK algoritması bir ÇKYP veri seti olan RTS problem setlerinde diğer algoritmalara göre belirgin seviyede kötü bir performans göstermemekle beraber, son problem sınıfında HYAK algoritmasına üstün gelerek GYAK algoritmasının son sınıf için elde ettiği ortalama KDO değerine yakın doluluk yüzdesinin sağlandığı çözümler bulmuştur.

HYAK algoritmasının performansı ise problem sınıflarına göre farklılık göstermektedir. Şekil 6.4 (a)'daki çözüm araştırma süreçleri ikili t-testi ile analiz edildiğinde ve (b)'deki süreç dağılım grafiğine bakıldığında HYAK algoritmasının, temel YAK ve HYAK algoritmasından önemli ölçüde ($p < .05$) daha iyi performans gösterdiği görülmektedir. Diğer problem sınıflarında problem kompleksliğinin, yerleştirilmesi gereken nesne sayısı ile artmasıyla, önce ikinci problem sınıfında temel YAK ve HYAK algoritmalarından önemli ölçüde olmasa bile ($p > .05$) yine de iyi performans göstermekte, fakat son problem sınıfında problem kompleksliğinin

Çizelge 6.4 Temel YAK ve önerilen algoritmaların RTS problem setleri için elde ettikleri KDO değerleri

	Temel YAK			HYAK			GYAK			BHYAK		
Problem Seti	**KDO**	**St. Sapma**	**Test Süresi (sn.)**	**KDO**	**St. Sapma**	**Test Süresi (sn.)**	**KDO**	**St. Sapma**	**Test Süresi (sn.)**	**KDO**	**St. Sapma**	**Test Süresi (sn.)**
RTS01	0.7491	0.0456	0.41	0.7739	0.0198	0.59	0.7472	0.0202	0.46	**0.7919**	0.0251	0.56
RTS02	0.7461	0.0496	0.31	0.7642	0.0147	0.44	0.7496	0.0302	0.35	**0.7791**	0.0205	0.42
RTS03	0.7262	0.0391	0.29	0.7632	0.0121	0.41	0.7394	0.0375	0.32	**0.7957**	0.0132	0.39
RTS04	0.5397	0.0151	0.49	0.5980	0.0197	0.69	0.5479	0.0212	0.54	**0.6801**	0.0252	0.66
RTS05	0.6739	0.0296	0.29	0.7401	0.0477	0.42	0.7000	0.0324	0.33	**0.7866**	0.0582	0.40
RTS06	0.7755	0.0156	6.43	0.7754	0.0452	9.21	0.7764	0.0178	7.22	**0.8085**	0.0044	8.75
RTS07	0.7530	0.0241	4.82	0.7501	0.0301	6.91	0.7630	0.0229	5.41	**0.7869**	0.0396	6.56
RTS08	0.7929	0.0429	4.48	0.7927	0.0370	6.41	0.7922	0.0447	5.03	**0.8192**	0.0422	6.09
RTS09	0.5451	0.0352	7.58	0.5778	0.0239	10.85	0.5451	0.0208	8.50	**0.6611**	0.0143	10.31
RTS10	0.7934	0.0294	4.59	0.7877	0.0339	6.58	0.7895	0.0319	5.15	**0.8311**	0.0520	6.25
RTS11	0.7843	0.0255	51.66	0.7614	0.0389	73.95	0.7934	0.0407	57.95	**0.8121**	0.0239	70.28
RTS12	0.7634	0.0409	38.62	0.7539	0.0508	55.28	0.7689	0.0469	43.33	**0.8015**	0.0202	52.54
RTS13	0.8120	0.0157	35.82	0.8105	0.0396	51.28	0.8060	0.0411	40.18	**0.8380**	0.0439	48.73
RTS14	0.5496	0.0264	60.47	0.5730	0.0340	86.57	0.5490	0.0299	67.84	**0.6640**	0.0174	82.27
RTS15	0.8107	0.0124	37.33	0.8092	0.0446	53.44	0.8135	0.0163	41.88	**0.8403**	0.0155	50.79
RTS01-05 Ort.	0.6870	–	0.36	0.7279	–	0.51	0.6968	–	0.40	**0.7667**	–	0.49
RTS06-10 Ort.	0.7320	–	5.58	0.7367	–	7.99	0.7332	–	6.26	**0.7814**	–	7.59
RTS11-15 Ort.	0.7440	–	44.78	0.7416	–	64.10	0.7462	–	50.24	**0.7912**	–	60.92
Genel Ort.	0.7210	–	16.91	0.7354	–	24.20	0.7254	–	18.97	**0.7797**	–	23.00

daha da artmasıyla her ne kadar istatistiksel açıdan temel YAK algoritmasından farklı bir performans göstermese de süreç sonunda elde edilen KDO değerleri bakımından GYAK algoritması ile birlikte temel YAK algoritmasının da gerisine düşmektedir. BR problem setinde de olduğu gibi RTS problem setinde de HYAK algoritması her ne kadar diğer algoritmaların yanında ortalamanın üzerinde bir performans sergilese de çözüm araştırma sürecinin erken safhalarında yerel optimum çözümde takılıp kalarak çözüm geliştirmeyi ilerletememektedir.

GYAK algoritması problem kompleksliğinden bağımsız olarak ortalama bir performans göstermektedir. İlk problem setinde istatistiksel bakımdan temel YAK algoritmasının araştırma sürecinde farklı ($p < .05$) olsa da Şekil 6.4 (b)'ye bakıldığında genetik operatörlerin etkisinin kısıtlı olduğu görülmektedir. Bununla birlikte, problem kompleksliği arttıkça genetik operatör etkisini tam olarak gösterebilmekte ve böylece GYAK algoritması temel YAK algoritmasından önemli ölçüde ($p < .05$) ayrışmaktadır.

BHYAK algoritması ise bütün problem sınıflarında diğer üç algoritmadan da önemli ölçüde ($p < .05$) daha iyi performans göstermektedir. Diğer üç algoritma her üç problem sınıfında yaptıkları çözüm araştırmalarını, son safhalarda buldukları yerel optimum değeri ile sonlandırmış gibi gözükseler de BHYAK algoritmasının Şekil 6.4 (a), (c) ve (e)'deki iterasyon – KDO eğrisi, önerilen yaklaşımın süreç uzadıkça hâlâ daha iyi çözüm bulabilme potansiyelini taşıdığını göstermektedir.

HYAK ve GYAK algoritmalarının RTS problem setlerinin genelinde gösterdiği performanslar değerlendirildiğinde, HYAK algoritması RTS problem setlerinden GYAK algoritmasına göre daha iyi KDO değerleri elde ettiklerinde dahi bu çözüm değerlerini araştırmanın erken safhasında elde edip daha fazla geliştirememiş ve BR problem setlerinde gösterdiği performansa benzer şekilde araştırma bitene kadar yerel optimum çözümde takılı kalmaktadır. Buna rağmen, HYAK algoritması RTS01-RTS10 problem setlerinde gösterdiği performans ile düşük kompleksliğe sahip problemlerde hafıza sisteminin genetik operatörlere göre temel YAK algoritması üzerinde daha etkin olduğunu göstermektedir. Bununla birlikte, çözüm araştırmasının

erken safhasında etkisini yitiren hafıza sistemi ayrıca komplekslik arttıkça da işlevsiz hale gelmektedir.

Öte yandan genetik operatör kullanma yaklaşımı çözüm araştırmalarının erken safhalarında pek etkin değilken, istikrarlı bir şekilde ilerleyen safhalarda etkinliğini artırmakta, hatta problem kompleksliği artmasından da olumsuz etkilenmemektedir. RTS11-RTS15 problem setlerinin çözüm araştırma süreçlerinde görüleceği üzere, HYAK ve GYAK algoritmaları ileri safhada neredeyse temel YAK algoritması ile aynı performansı gösterir hale gelmesine rağmen, hafıza ve genetik operatörlerin YAK algoritması için takviye yaklaşım olarak bir arada kullanıldığı BHYAK algoritması bünyesinde bir sinerji oluşturarak çözüm araştırmasını çok daha farklı ve verimli bir noktaya taşıyabilmişlerdir.

6.2.3. BHYAK Algoritmasının Literatürdeki Yaklaşımlarla Karşılaştırılması

Literatürden karşılaştırma için seçilen çalışmalarda önerilen yaklaşımların IMM veri seti üzerinde uygulanmaları neticesinde elde edilen değerler Çizelge 6.5'te kullanılan toplam konteyner sayısı cinsinden olduğu için, BHYAK algoritmasının aynı veri seti üzerindeki uygulamasında çözüm değeri olarak KDO yerine ilgili problem seti için kullanılan konteyner sayısı verilmektedir. BR veri setinin aksine IMM veri setinde yerleştirilmeye çalışılan nesne türü sayısı kısıtlı olup problem setine göre iki ile beş arasında değişmektedir. Kullanılan konteynerlerin ölçüleri de nesnelerin ölçüleri ile hemen hemen orantılı olması sebebiyle IMM veri setinin BR veri setine göre daha homojen yapıda olduğu söylenebilir.

BHYAK ve Çizelge 6.5'te yer alan diğer karşılaştırma yaklaşımları tek yön ANOVA ve ardından LSD testi ile analiz edildiğinde SA yaklaşımının önemli ölçüde ($p < .05$) en iyi konteyner sayısı değerine sahip olduğu görülmektedir. SA'nın ardından EE ve 2C yaklaşımları benzer konteyner sayılarıyla ikinci grubu oluşturmaktadırlar. BHYAK ve BR yaklaşımları da benzer konteyner sayıları ile en kötü çözümleri elde eden grubu oluşturmaktadırlar.

Çizelge 6.5 BHYAK algoritması ve literatürdeki yaklaşımların IMM sonuçları

Problem Seti	BR	EE	SA	2C	BHYAK
IMM01	27	26	**25**	**25**	28
IMM02	11	**10**	**10**	**10**	11
IMM03	21	22	**19**	**19**	20
IMM04	29	30	**26**	**26**	27
IMM05	61	**51**	**51**	**51**	**51**
IMM06	10	10	10	10	10
IMM07	16	16	16	16	16
IMM08	4	4	4	4	4
IMM09	**19**	**19**	**19**	20	21
IMM10	**55**	**55**	**55**	**55**	58
IMM11	19	18	**17**	18	22
IMM12	55	**53**	**53**	**53**	54
IMM13	**25**	**25**	**25**	26	27
IMM14	**27**	**27**	**27**	28	31
IMM15	**11**	12	**11**	13	14
IMM16	28	**26**	**26**	29	29
IMM17	8	**7**	**7**	8	9
IMM18	3	**2**	**2**	**2**	3
IMM19	**3**	**3**	**3**	**3**	5
IMM20	5	5	5	5	5
IMM21	**24**	26	20	30	**24**
IMM22	11	9	**8**	9	12
IMM23	22	21	**20**	23	24
IMM24	**6**	**6**	**6**	**6**	7
IMM25	**5**	**5**	**5**	**5**	6
IMM26	**3**	**3**	**3**	**3**	4
IMM27	5	5	5	5	5
IMM28	11	**10**	**10**	11	13
IMM29	**17**	18	**17**	21	23
IMM30	24	23	**22**	27	25
IMM31	**13**	14	**13**	14	16
IMM32	**4**	**4**	**4**	**4**	6
IMM33	5	5	**4**	5	5
IMM34	9	9	**8**	9	9
IMM35	3	**2**	**2**	3	**2**
IMM36	19	**14**	**14**	**14**	**14**
IMM37	27	**23**	**23**	**23**	24
IMM38	56	**45**	**45**	**45**	47
IMM39	16	**15**	**15**	**15**	16
IMM40	10	9	**8**	9	9
IMM41	16	**15**	**15**	**15**	17
IMM42	5	**4**	**4**	5	6
IMM43	**3**	**3**	**3**	**3**	4
IMM44	4	4	**3**	4	4
IMM45	3	3	3	3	2
IMM46	**2**	**2**	**2**	**2**	3
IMM47	**3**	**3**	**3**	4	4
Kullanılan Toplam Konteyner Sayısı	763	721	696	738	776

BR yaklaşımı, temel duvar örme sezgiseli ile nesneleri yerleştirme yolunu tuttuğu için aynı şekilde nesnelerin yerleştirilmesini sağlayan BHYAK algoritması ile hemen hemen aynı sonuçları elde etmiştir. Öte yandan, BR yaklaşımı nesneleri öncelikle sınıflandırıp NYD'leri bu sınıflandırmaya göre tasarlamasının ardından yerleştirme sürecini başlatırken BHYAK algoritmasının uyguladığı duvar örme sezgiseli önceden nesnelerin gruplandırılmasını içermemekte, tamamen rassal olarak belirlenen

yerleştirme sırası ve yönlendirme türü ile nesnelerin konteynerlere yerleştirilmesi sağlanmaktadır.

EE yaklaşımında nesneler önce bloklar haline getirilip, ardından nesne yerleştirme esnasında her bir yerleştirme sonrasında sıradaki nesne blokları ağaç araştırması (AA) ile belirlenmekte ve böylelikle nesne yerleştirme süreci BHYAK algoritmasının kullandığı rassal çözüm arama yerine bilinçli arama ile yönlendirilmektedir. Ayrıca BR yaklaşımının uyguladığı önceden belirlenen dizine bağlı kalarak nesne yerleştirilmesi yaklaşımı yerine, AA ile anlık karar verme yolu benimsenerek BR yaklaşımına göre daha az konteynerin kullanımını sağlamıştır. SA yaklaşımı ise EE yaklaşımını da geliştirerek, TB sezgisel yaklaşımı yardımıyla her bir nesne yerleştirme aşamasında bir sonraki nesne veya nesne grubunu seçerken birçok alternatif çözüm - ki SA bunu 1000 olarak belirlemiştir- oluşturup aralarından en iyisini seçmek yolu ile en az konteyner kullanım sayısını elde etmiştir.

Literatürden seçilen yaklaşımlar arasında BHYAK algoritması ile yapısal olarak denk şekilde karşılaştırılabilecek 2C yaklaşımı GA'yı kullanarak çözüm türetmeye çalışmaktadır. Oluşturulan kromozomlardan birinin elementlerinin her biri nesnelerin farklı bir permütasyonunu temsil ederken diğer kromozomun her bir geni ise konteynerlerin farklı bir permütasyonunu temsil etmektedir. Nesne ve konteyner permütasyonları arasında yapılan eşleştirme ise çözümleri oluşturmaktadır. 2C yaklaşımında her bir kromozomdan jenerasyonlar boyunca genetik operatörler yardımıyla yeni nesne ve konteyner permütasyonları türetilmektedir. BHYAK algoritmasının rassal çözüm türetme yaklaşımı yerine diğer yaklaşımlara benzer şekilde çözümler üzerinde bilinçli bir yönlendirme benimsenmiştir. Böylelikle sadece GA kullanılarak bu çalışmada önerilen BHYAK algoritmasına göre önemli ölçüde ($p < .05$) daha az sayıda konteynerin kullanılmasına ihtiyaç duyulmuştur.

BHYAK algoritmasının, TKYP'ye örnek olarak uygulandığı BR veri setinde homojen yapıya yakın nesne dağılımına sahip problemlerde (BR01-BR07) karşılaştırılan diğer yaklaşımlara göre daha kötü performans gösterirken, problem heterojenliği arttıkça diğer yaklaşımlara daha yakın veya onlardan daha iyi performans sergilediği vurgulanmıştı. ÇKYP'ye örnek olarak uygulandığı IMM veri setindeki çoğu

problemler yapısı itibariyle homojen veya homojene yakın oldukları için BHYAK algoritmasının TKYP'de olduğu gibi düşük performans göstermesi, önerilen yaklaşımının heterojen ve yüksek kompleksli problemlerin çözümü için daha uygun olduğunu göstermektedir. Öte yandan, rassal arama yerine bilinçli yönlendirmenin daha iyi çözümler sunduğu değerlendirildiğinde ise BHYAK algoritmasının kullandığı NYD'lerin yapılarının tümüyle rassal nesne sırası-nesne yönlendirmesi eşleştirmesi temelli konteyner yükleme yaklaşımı yerine daha bilinçli bir nesne yerleştirmeye yönelik tasarlanması gerektiği ortadadır.

BÖLÜM 7

SONUÇ

Konteyner yükleme problemi (KYP), yük taşıma sistemlerinin başlıca problemlerinden birisi olup, kısıtlı hacimdeki bir veya birden fazla alanın, taşınacak nesneler minimum maliyetle teslim edilecek şekilde kullanılmasını konu almaktadır. Bu çalışmada, KYP üç boyutlu düzgün dörtgen nesnelerin yine üç boyutlu düzgün dörtgen konteynerler içerisine yerleştirilmelerini kapsamaktadır. KYP'nin çözümü için literatürde matematiksel modeller yanı sıra sezgisel yaklaşımlar da önerilmektedir. KYP üzerinde uygulandığına literatürde çok rastlanmasa da Yapay Arı Kolonisi (YAK) algoritmasının benzer yapıya sahip Sırt Çantası Problemleri üzerinde uygulandığı birçok çalışma mevcuttur.

YAK algoritması daha çok sayısal optimizasyon problemleri üzerinde yaygın olarak kullanılan bir sürü optimizasyonu temelli meta-sezgisel yaklaşımdır. Bununla birlikte, bir tamsayılı problem çeşidi olan ikili (binary) optimizasyon problemlerinin çözümünde kullanıldığı örnekler literatürde mevcuttur. Bir ikili optimizasyon problemi olan SÇP çözümünde YAK algoritması tercih edilen bir yaklaşım olsa da SÇP'nin alt problemlerinden birisi olan ve taşıma sistemlerinin başta gelen problemlerinden birisi olan Konteyner Yükleme Problemi (KYP)'nin çözümü için kullanıldığı çalışmalar çok azdır. YAK algoritmasının uygulandığı SÇP'lerde problem boyutundan bahsedilirken kastedilen her bir çözümün elemanlarını oluşturan değişken sayısı iken, KYP'de türetilen çözümlerin elementleri ise üç boyutlu konteyner/konteynerlere yerleştirilecek yine üç boyutlu nesnelerin, konteyner tek ise atanıp atanmadığını birden fazla ise de yerleştirme sırasını ve nenelerin konteyner içerisinde nasıl yönlendirileceğini belirtmektedir.

Sayısal problemlerin optimizasyonu konu olduğunda, yerel araştırmada yeni çözüm türetme ve global araştırmada çözüm çeşitlendirme yönleri bakımından literatürde birçok çalışmada YAK algoritmasının güçlü ve etkili bir algoritma olduğundan bahsedilmektedir. Söz konusu KYP olduğunda ise hem literatürdeki benzer çalışmaların çıkarımları hem de bu çalışmada elde edilen test sonuçları YAK algoritmasının çözüm arama yönlerinin geliştirilmeksizin böylesi kompleks bir problemin çözümünde yetersiz kaldığını göstermektedir. Bu çalışma ise KYP gibi yüksek kompleks yapıya sahip bir problemin çözümünde istenilen verimli çıktıları elde edebilmesi için temel YAK algoritmasının çözüm arama mekanizmasının geliştirilerek, daha güçlü hale getirilmesi üzerine odaklanmaktadır. Bu çerçevede alternatif olarak, tekrarlı fakat verimsiz çözüm geçiş hareketlerinin belli bir periyot için yasaklandığı ve verimli çözüm geçiş hareketlerinin tekrar kullanılmasının teşvik edildiği bir hafıza sistemi ile YAK algoritmasının çözüm çeşitlendirme yönünü güçlendirerek, keşfedilmemiş muhtemel verimli çözümleri keşfetmesini sağlamak yolu ile çözüm uzayını genişletme potansiyeline sahip genetik operatörler takviye yaklaşım olarak önerilmektedir. Önerilen takviye yaklaşımların YAK algoritması üzerindeki etkilerinin daha iyi anlaşılması için ayrı ayrı analiz edilmişlerdir.

Sonuç olarak bu çalışmada, yerel araştırmada verimli ve verimsiz çözüm geçiş hareketlerini dikkatlice seçip sınıflandıran bir Hafıza Temelli Yapay Arı Kolonisi (HYAK) algoritması ile çözüm popülasyonu içerisinde yüksek verime sahip olanlarını, sonraki jenerasyonlarda daha verimli çözümlerin türetilebilmesi için bilinçli şekilde kullanan bir Genetik Operatör Temelli Yapay Arı Kolonisi (GYAK) algoritması önerilmektedir. Bunlara ek olarak, literatürde birçok çalışmanın önerdiği yaklaşımların yaygın biçimde test edildiği veri setleri ve literatürdeki kurallara göre türetilen veri setleri üzerinde HYAK ve GYAK algoritmalarının ayrı ayrı test edilmesi yanı sıra, her iki takviye yaklaşımın bir arada temel YAK algoritmasına entegre edildiği Bütünleşik Hibrit Yapay Arı Kolonisi (BHYAK) algoritması da ayrıca geliştirilip bu veri setleri üzerinde test edilmesi sağlanmıştır.

BHYAK algoritması, çoğu nesne yerleştirme kısıtını dikkate almadan ve nesne yerleştirme sürecinin her adımında arta kalan boşluğu hesaba katmadan sadece önceden NYD'lerle belirlenmiş nesne atama ve yönlendirme planına göre nesne

yerleştirme yaklaşımını benimseyerek hem TKYP'de hem de ÇKYP'de çözüm uzayını geliştirmede başarılı olmuş ve HYAK – GYAK algoritma ikilisine göre önemli ölçüde daha iyi bir performans göstermiştir. Tamamen rassal aramaya dayalı olan BHYAK algoritması her iki takviye çözüm aracını bir arada kullanarak, bu iki yaklaşımın temel alındığı Tabu Araştırması (TA) ve Genetik Algoritmayı (GA) yüksek heterojen yapıdaki TKYP problem setlerinde uygulayan çalışmaların elde ettiği sonuçlardan önemli ölçüde daha iyi sonuçlar elde etmiştir. Bununla birlikte, literatürdeki diğer yaklaşımlar ile karşılaştırıldığında tamamen rassal arama temelli bir yaklaşım olan BHYAK algoritması, nesneleri bir ön sınıflandırma işlemine tâbi tutan diğer yaklaşımlarınkine yakın performans göstermektedir. Bu da BHYAK algoritmasının rassal arama becerisinin ileri düzeyde olduğunu göstermektedir. Ayrıca nesne yerleştirme yaklaşımının daha da geliştirilebilir durumda olması, tez çalışmasında önerilen yaklaşımın yüksek kompleks problemlerin çözümü hususunda gelecek vaat eden bir algoritma olduğunu göstermektedir.

Gelecek çalışmalarda, önerilen BHYAK algoritması yanı sıra HYAK ve GYAK algoritmaları da çeşitli sayısal problemler ve SÇP'ler üzerinde uygulanabilir. Ayrıca KYP ile bağlantılı olarak önerilen yaklaşımlar kapasiteli araç rotalama problemlerinde uygulanabilir, bu çalışmada da yetersiz olarak görülen nesne yerleştirme yaklaşımı geliştirilerek, önerilen meta-sezgisel yaklaşımlar mevcut haliyle dahi bu tür problemlere alternatif çözüm yolu olarak sunulabilir.

KAYNAKLAR

1. Lin, E. Y.-H., "A bibliographical survey on some well-known non-standard knapsack problems", ***INFOR: Information Systems and Operational Research***, 36 (4): 274–317 (1998).

2. Dyckhoff, H., "A typology of cutting and packing problems", ***European Journal Of Operational Research***, 44 (2): 145–159 (1990).

3. Wascher, G., HauBner, H., and Schumann, H., "An improved typology of cutting and packing problems", ***European Journal of Operational Research***, 183 (3): 1109–1130 (2007).

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

5. Zhao, X. Z., Bennell, J. A., Bektas, T., and Dowsland, K., "A comparative review of 3D container loading algorithms", ***International Transactions in Operational Research***, 23 (1–2): 287–320 (2016).

6. Sundar, S., Singh, A., and Rossi, A., "An artificial bee colony algorithm for the 0–1 multidimensional knapsack problem", International Conference on Contemporary Computing, ***Springer***, (2010).

7. Sabet, S., Farokhi, F., and Shokouhifar, M., "A novel artificial bee colony algorithm for the knapsack problem", ***International Symposium on Innovations in Intelligent Systems and Applications***, ***IEEE*** (2012).

8. Cao, J., Yin, B., Lu, X., Kang, Y., and Chen, X., "A modified artificial bee colony approach for the 0-1 knapsack problem", ***Applied Intelligence***, 1–14 (2018).

9. Zeineldin, R. A. and Morsy, A. M., "A modified artificial bee colony for solving the container loading problem", ***International Journal of Computer Applications***, 114 (3): (2015).

10. Ozturk, C., Hancer, E., and Karaboga, D., "A novel binary artificial bee colony algorithm based on genetic operators", ***Information Sciences***, 297: 154–170 (2015).

11. Singh, V., Tiwari, R., Singh, D., and Shukla, A., "RGBCA-genetic bee colony algorithm for travelling salesman problem", ***World Congress on Information and Communication Technologies***, ***IEEE***, (2011).

12. Chaurasia, S. N., Sundar, S., and Singh, A., "Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release dates", ***Operational Research***, 17 (1): 275–295 (2017).

13. Panahi, V. and Navimipour, N. J., "Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators", ***Concurrency and Computation: Practice And Experience***, e5218 (2019).

14. Yin, P.-Y. and Chuang, Y.-L., "Adaptive memory artificial bee colony algorithm for green vehicle routing with cross-docking", ***Applied Mathematical Modelling***, 40 (21–22): 9302–9315 (2016).

15. Li, X. and Yang, G., "Artificial bee colony algorithm with memory", ***Applied Soft Computing***, 41: 362–372 (2016).

16. Chengli, F. A. N., Qiang, F. U., Guangzheng, L., and Qinghua, X., "Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism", ***Journal Of Systems Engineering and Electronics***, 29 (2): 405–414 (2018).

17. Bischoff, E. E. and Marriott, M. D., "A comparative-evaluation of heuristics for container loading", ***European Journal of Operational Research***, 44 (2): 267–276 (1990).

18. Wang, L., Zhang, H., Xiong, Y., Li, D. W., and Northeastern Univ, C., "Ant colony optimization algorithm based on space division for container loading problem", ***Chinese Control and Decision Conference***, Vols 1-5, 3448-+ (2010).

19. Che, C. H., Huang, W. L., Lim, A., and Zhu, W. B., "The multiple container loading cost minimization problem", ***European Journal of Operational Research***, 214 (3): 501–511 (2011).

20. Cano, I. and Torra, V., "Particle swarm optimization for container loading of nonorthogonal objects", ***Artificial Intelligence and Soft Computing***, Pt II, 403–410 (2010).

21. Egeblad, J., Garavelli, C., Lisi, S., and Pisinger, D., "Heuristics for container loading of furniture", ***European Journal of Operational Research***, 200 (3): 881–892 (2010).

22. Crainic, T. G., Gobbato, L., Perboli, G., and Rei, W., "Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic", ***European Journal of Operational Research***, 253 (2): 404–417 (2016).

23. Jamrus, T. and Chien, C. F., "Extended priority-based hybrid genetic algorithm for the less-than-container loading problem", ***Computers & Industrial Engineering***, 96: 227–236 (2016).

24. Wei, L. J., Zhu, W. B., and Lim, A., "A goal-driven prototype column generation strategy for the multiple container loading cost minimization problem", ***European Journal of Operational Research***, 241 (1): 39–49 (2015).

25. Wei, L., Hu, Q., Leung, S. C. H., and Zhang, N., "An improved skyline based heuristic for the 2D strip packing problem and its efficient implementation", ***Computers & Operations Research***, 80: 113–127 (2017).

26. Gehring, H., "A genetic algorithm for solving the container loading problem", ***International Transactions in Operational Research***, 4 (5–6): 401–418 (1997).

27. Koloch, G. and Kaminski, B., "Vehicle routing with three-dimensional container loading constraints - comparison of nested and joint algorithms", ***Iaeng Transactions on Engineering Technologies***, Vol 5, 145-+ (2010).

28. Saraiva, R. D., Nepomuceno, N., and Pinheiro, P. R., "A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company", ***IFAC-PapersOnLine***, 48 (3): 490–495 (2015).

29. Mustafee, N. and Bischoff, E. E., "Analysing trade-offs in container loading: combining load plan construction heuristics with agent-based simulation", ***International Transactions in Operational Research***, 20 (4): 471–491 (2013).

30. Kovacs, A. and Beck, J. C., "A global constraint for total weighted completion time for cumulative resources", ***Engineering Applications of Artificial Intelligence***, 21 (5): 691–697 (2008).

31. Chang, T. S. and Liao, Y. F., "Path finding with stowage planning consideration in a mixed pickup-delivery and specified-node network", ***Transportation Research Part E-Logistics and Transportation Review***, 44 (6): 970–985 (2008).

32. Zhu, W. B., Qin, H., Lim, A., and Wang, L., "A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP", ***Computers & Operations Research***, 39 (9): 2178–2195 (2012).

33. Baur, X., Budnik, L. T., Zhao, Z., Bratveit, M., Djurhuus, R., Verschoor, L., Rubino, F. M., Colosio, C., and Jepsen, J. R., "Health risks in international container and bulk cargo transport due to volatile toxic compounds", ***Journal of Occupational Medicine and Toxicology (London, England)***, 10: 19 (2015).

34. Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G., "The impact of depot location, fleet composition and routing on emissions in city logistics", ***Transportation Research Part B: Methodological***, 84: 81–102 (2016).

35. Terno, J., Scheithauer, G., Sommerweiß, U., and Riehme, J., "An efficient approach for the multi-pallet loading problem", ***European Journal of Operational Research***, 123 (2): 372–381 (2000).

36. Martello, S., Pisinger, D., and Vigo, D., "The three-dimensional bin packing problem", ***Operations Research***, 48 (2): 256–267 (2000).

37. Ceschia, S. and Schaerf, A., "Local search for a multi-drop multi-container loading problem", ***Journal of Heuristics***, 19 (2): 275–294 (2013).

38. Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M., "A GRASP/Path relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems", ***Computers & Operations Research***, 40 (12): 3081–3090 (2013).

39. Chen, C. S., Lee, S. M., and Shen, Q. S., "An analytical model for the container loading problem", ***European Journal of Operational Research***, 80 (1): 68–76 (1995).

40. de Almeida, A. and Figueiredo, M. B., "A particular approach for the Three-dimensional Packing Problem with additional constraints", ***Computers & Operations Research***, 37 (11): 1968–1976 (2010).

41. Bischoff, E. E., "Stability aspects of pallet loading", ***Operations-Research-Spektrum***, 13 (4): 189–197 (1991).

42. Bischoff, E. E., Janetz, F., and Ratcliff, M. S. W., "Loading pallets with non-identical items", ***European Journal of Operational Research***, 84 (3): 681–692 (1995).

43. Bortfeldt, A. and Gehring, H., "A hybrid genetic algorithm for the container loading problem", ***European Journal of Operational Research***, 131 (1): 143–161 (2001).

44. Hifi, M., "Approximate algorithms for the container loading problem", ***International Transactions in Operational Research***, 9 (6): 747–774 (2002).

45. Eley, M., "A bottleneck assignment approach to the multiple container loading problem", ***Or Spectrum***, 25 (1): 45–60 (2003).

46. Alonso, M. T., Alvarez-Valdes, R., Tamarit, J. M., and Parreno, F., "A reactive GRASP algorithm for the container loading problem with load-bearing constraints", ***European Journal of Industrial Engineering***, 8 (5): 669–694 (2014).

47. Iwasawa, H., Hu, Y. N., Hashimoto, H., Imahori, S., and Yagiura, M., "A heuristic algorithm for the container loading problem with complex loading constraints", ***Journal of Advanced Mechanical Design Systems and Manufacturing***, 10 (3): (2016).

48. Moura, A. and Bortfeldt, A., "A two-stage packing problem procedure", ***International Transactions in Operational Research***, 24 (1–2): 43–58 (2017).

49. Eley, M., "Solving container loading problems by block arrangement", ***European Journal of Operational Research***, 141 (2): 393–409 (2002).

50. Wei, L. J., Oon, W. C., Zhu, W. B., and Lim, A., "A reference length approach for the 3D strip packing problem", ***European Journal of Operational Research***, 220 (1): 37–47 (2012).

51. Fanslau, T. and Bortfeldt, A., "A tree search algorithm for solving the container loading problem", ***Informs Journal on Computing***, 22 (2): 222–235 (2010).

52. Lim, A., Ma, H., Xu, J., and Zhang, X. W., "An iterated construction approach with dynamic prioritization for solving the container loading problems", ***Expert Systems with Applications***, 39 (4): 4292–4305 (2012).

53. Gehring, H. and Bortfeldt, A., "A parallel genetic algorithm for solving the container loading problem", ***International Transactions in Operational Research***, 9 (4): 497–511 (2002).

54. Sheng, L., Xiuqin, S., Changjian, C., Hongxia, Z., Dayong, S., and Feiyue, W., "Heuristic algorithm for the container loading problem with multiple constraints", ***Computers & Industrial Engineering***, 108: 149–164 (2017).

55. George, J. A. and Robinson, D. F., "A heuristic for packing boxes into a container", ***Computers & Operations Research***, 7 (3): 147–156 (1980).

56. Gilmore, P. C. and Gomory, R. E., "The theory and computation of knapsack functions", ***Operations Research***, 14 (6): 1045–1074 (1966).

57. Huang, W. Q. and He, K., "A caving degree approach for the single container loading problem", ***European Journal of Operational Research***, 196 (1): 93–101 (2009).

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

59. Egeblad, J. and Pisinger, D., "Heuristic approaches for the two- and three-dimensional knapsack packing problem", ***Computers & Operations Research***, 36 (4): 1026–1049 (2009).

60. Zhu, W. B. and Lim, A., "A new iterative-doubling greedy-lookahead algorithm for the single container loading problem", ***European Journal of Operational Research***, 222 (3): 408–417 (2012).

61. Epstein, L. and van Stee, R., "On variable-sized multidimensional packing", ***European Symposium on Algorithms***, ***Springer***, (2004).

62. Bischoff, E. E., "Three-dimensional packing of items with limited load bearing strength", ***European Journal of Operational Research***, 168 (3): 952–966 (2006).

63. Ren, J. D., Tian, Y. J., and Sawaragi, T., "A tree search method for the container loading problem with shipment priority", ***European Journal of Operational Research***, 214 (3): 526–535 (2011).

64. Araya, I. and Riff, M. C., "A beam search approach to the container loading problem", ***Computers & Operations Research***, 43: 100–107 (2014).

65. Araya, I., Guerrero, K., and Nunez, E., "VCS: A new heuristic function for selecting boxes in the single container loading problem", ***Computers & Operations Research***, 82: 27–35 (2017).

66. Morabito, R. and Arenales, M., "An AND/OR-graph approach to the container loading problem", ***International Transactions in Operational Research***, 1 (1): 59–73 (1994).

67. Glover, F., "Tabu search: A tutorial", ***Interfaces***, 20 (4): 74–94 (1990).

68. Gendreau, M., Iori, M., Laporte, G., and Martello, S., "A tabu search algorithm for a routing and container loading problem", ***Transportation Science***, 40 (3): 342–350 (2006).

69. de Araujo, L. J. P. and Pinheiro, P., "Combining heuristics backtracking and genetic algorithm to solve the container loading problem with weight distribution", ***Soft Computing Models in Industrial and Environmental Applications***, 95–102 (2010).

70. Goncalves, J. F. and Resende, M. G. C., "A parallel multi-population biased random-key genetic algorithm for a container loading problem", ***Computers & Operations Research***, 39 (2): 179–190 (2012).

71. Hasni, H. and Sabri, H., "On a hybrid genetic algorithm for solving the container loading problem with no orientation constraints", ***Journal of Mathematical Modelling and Algorithms in Operations Research***, 12 (1): 67–84 (2013).

72. Feng, X., Moon, I., and Shin, J., "Hybrid genetic algorithms for the three-dimensional multiple container packing problem", ***Flexible Services and Manufacturing Journal***, 27 (2–3): 451–477 (2015).

73. Zheng, J. N., Chien, C. F., and Gen, M., "Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem", ***Computers & Industrial Engineering***, 89: 80–87 (2015).

74. Feng, X., Moon, I., and Shin, J., "Hybrid genetic algorithms for the three-dimensional multiple container packing problem", ***Flexible Services and Manufacturing Journal***, 27 (2–3): 451–477 (2015).

75. Huang, Y. H., Hwang, F. J., and Lu, H. C., "An effective placement method for the single container loading problem", ***Computers & Industrial Engineering***, 97: 212–221 (2016).

76. Bischoff, E. E. and Marriott, M. D., "A comparative evaluation of heuristics for container loading", ***European Journal of Operational Research***, 44 (2): 267–276 (1990).

77. Bischoff, E. E. and Ratcliff, M. S. W., "Issues in the development of approaches to container loading", ***Omega***, 23 (4): 377–390 (1995).

78. Gehring, H. and Bortfeldt, A., "A genetic algorithm for solving the container loading problem", ***International Transactions in Operational Research***, 4 (5–6): 401–418 (1997).

79. Raidl, G. R. and Kodydek, G., "Genetic algorithms for the multiple container packing problem", ***PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature***, 148 (2): 875–884 (1998).

80. Chien, C. F. and Wu, W. T., "A recursive computational procedure for container loading", ***Computers & Industrial Engineering***, 35 (1–2): 319–322 (1998).

81. Ratcliff, M. S. W. and Bischoff, E. E., "Allowing for weight considerations in container loading", ***Or Spektrum***, 20 (1): 65–71 (1998).

82. Raidl, G. R., "The multiple container packing problem", ***ACM SIGAPP Applied Computing Review***, 7 (2): 22–31 (1999).

83. Scheithauer, G., "LP-based bounds for the container and multi-container loading problem", ***International Transactions in Operational Research***, 6 (2): 199–213 (1999).

84. Davies, A. P. and Bischoff, E. E., "Weight distribution considerations in container loading", ***European Journal of Operational Research***, 114 (3): 509–527 (1999).

85. Bortfeldt, A. and Gehring, H., "A hybrid genetic algorithm for the container loading problem", ***European Journal of Operational Research***, 131 (1): 143–161 (2001).

86. Lins, L., Lins, S., and Morabito, R., "An n-tet graph approach for non-guillotine packings of n-dimensional boxes into an n-container", ***European Journal of Operational Research***, 141 (2): 421–439 (2002).

87. Hifi, M., "Approximate algorithms for the container loading problem", ***International Transactions in Operational Research***, 9 (6): 747–774 (2002).

88. Pisinger, D., "Heuristics for the container loading problem", ***European Journal of Operational Research,*** 141: 382–392 (2002).

89. Eley, M., "Solving container loading problems by block arrangement", ***European Journal of Operational Research***, 141 (2): 393–409 (2002).

90. Eley, M., "A bottleneck assignment approach to the multiple container loading problem", ***OR Spectrum***, 25 (1): 45–60 (2003).

91. Bortfeldt, A., Gehring, H., and Mack, D., "A parallel tabu search algorithm for solving the container loading problem", ***Parallel Computing***, 29 (5): 641–662 (2003).

92. Lins, L., Lins, S., and Morabito, R., "An L-approach for packing (l, w)-rectangles into rectangular and L-shaped pieces", ***Journal of The Operational Research Society***, 54 (7): 777–789 (2003).

93. Mau Yeh, J., Chen Lin, Y., and Yi, S., "Applying genetic algorithms and neural networks to the container loading problem", ***Journal of Information and Optimization Sciences***, 24 (3): 423–443 (2003).

94. Mack, D., Bortfeldt, A., and Gehring, H., "A parallel hybrid local search algorithm for the container loading problem", ***International Transactions in Operational Research***, 11 (5): 511–533 (2004).

95. Lau, H. C. W., Choy, K. L., Lau, P. K. H., Tsui, W. T., and Choy, L. C., "An intelligent logistics support system for enhancing the airfreight forwarding business", ***Expert Systems***, 21 (5): 253–268 (2004).

96. Pimpawat, C. and Chaiyaratana, N., "Three-dimensional container loading using a cooperative co-evolutionary genetic algorithm", ***Applied Artificial Intelligence***, 18 (7): 581–601 (2004).

97. Brunetta, L. and Grégoire, P., "A general purpose algorithm for three-dimensional packing", ***INFORMS Journal on Computing***, 17 (3): 328–338 (2005).

98. Lewis, J. E., Ragade, R. K., Kumar, A., and Biles, W. E., "A distributed chromosome genetic algorithm for bin-packing", ***Robotics and Computer-Integrated Manufacturing***, 21 (4–5): 486–495 (2005).

99. Moura, A. and Oliveira, J. F., "A GRASP approach to the container-loading problem", ***IEEE Intelligent Systems***, 20 (4): 50–57 (2005).

100. Yeung, L. H. W. and Tang, W. K. S., "A hybrid genetic approach for container loading in logistics industry", ***IEEE Transactions on Industrial Electronics***, 52 (2): 617–627 (2005).

101. Birgin, E. G., Morabito, R., and Nishihara, F. H., "A note on an L-approach for solving the manufacturer's pallet loading problem", ***Journal of The Operational Research Society***, 56 (12): 1448–1451 (2005).

102. Takahara, S. and Miyamoto, S., "An evolutionary approach for the multiple container loading problem", ***Fifth International Conference on Hybrid Intelligent Systems (HIS'05)***, ***IEEE***, (2005).

103. Tsai, J.-F. and Li, H.-L., "A global optimization method for packing problems", ***Engineering Optimization***, 38 (6): 687–700 (2006).

104. Ertek, G. and Kilic, K., "Decision support for packing in warehouses", ***International Symposium on Computer and Information Sciences***, ***Springer***, 115–124 (2006).

105. Gendreau, M., Iori, M., Laporte, G., and Martello, S., "A tabu search algorithm for a routing and container loading problem", ***Transportation Science***, 40 (3): 342–350 (2006).

106. Takahara, S., "A simple meta-heuristic approach for the multiple container loading problem", ***International Conference on Systems, Man and Cybernetics***, ***IEEE***, (2006).

107. Bortfeldt, A. and Mack, D., "A heuristic for the three-dimensional strip packing problem", ***European Journal of Operational Research***, 183 (3): 1267–1279 (2007).

108. Xue, J. and Li, Q., "Scatter search algorithm to multiple container loading problem", ***International Conference on Transportation Engineering***, 3779 – 3784 (2007).

109. Huang, W. and He, K., "An efficient algorithm for solving the container loading problem", ***Combinatorics, Algorithms, Probabilistic and Experimental Methodologies SE - 36***, 4614: 396–407 (2007).

110. Wang, Z. and Li, K., "Layer-layout-based heuristics for loading homogeneous items into a single container", ***Journal of Zhejiang University Science A***, 8 (12): 1944–1952 (2007).

111. Nepomuceno, N., Pinheiro, P., and Coelho, A. L. V, "Tackling the container loading problem: a hybrid approach based on integer linear programming and genetic algorithms", ***European Conference on Evolutionary Computation in Combinatorial Optimization***, ***Springer***, 154–165 (2007).

112. Wang, Z., Li, K. W., and Levy, J. K., "A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach", ***European Journal of Operational Research***, 191 (1): 84–97 (2008).

113. Parreno, F., Alvarez-Valdes, R., Tamarit, J. M., and Oliveira, J. F., "A maximal-space algorithm for the container loading problem", ***Informs Journal on Computing***, 20 (3): 412–422 (2008).

114. Soak, S.-M., Lee, S.-W., Yeo, G.-T., and Jeon, M.-G., "An effective evolutionary algorithm for the multiple container packing problem", ***Progress in Natural Science***, 18 (3): 337–344 (2008).

115. Leung, S. Y. S., Wong, W. K., and Mok, P. Y., "Multiple-objective genetic optimization of the spatial design for packing and distribution carton boxes", ***Computers & Industrial Engineering***, 54 (4): 889–902 (2008).

116. Yan, S., Shih, Y. L., and Shiao, F. Y., "Optimal cargo container loading plans under stochastic demands for air express carriers", ***Transportation Research Part E: Logistics and Transportation Review***, 44 (3): 555–575 (2008).

117. Chang, T. S. and Liao, Y. F., "Path finding with stowage planning consideration in a mixed pickup-delivery and specified-node network", ***Transportation Research Part E: Logistics and Transportation Review***, 44 (6): 970–985 (2008).

118. Huang, W. and He, K., "A caving degree approach for the single container loading problem", ***European Journal of Operational Research***, 196 (1): 93–101 (2009).

119. Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T., "A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem", ***IEEE Transactions on Intelligent Transportation Systems***, 10 (2): 255–271 (2009).

120. Chien, C. F., Lee, C. Y., Huang, Y. C., and Wu, W. T., "An efficient computational procedure for determining the container-loading pattern", ***Computers and Industrial Engineering***, 56 (3): 965–978 (2009).

121. Moura, A. and Oliveira, J. F., "An integrated approach to the vehicle routing and container loading problems", ***OR Spectrum***, 31 (4): 775–800 (2009).

122. Huang, W. and He, K., "A new heuristic algorithm for cuboids packing with no orientation constraints", ***Computers and Operations Research***, 36 (2): 425–432 (2009).

123. Egeblad, J. and Pisinger, D., "Heuristic approaches for the two- and three-dimensional knapsack packing problem", ***Computers and Operations Research***, 36 (4): 1026–1049 (2009).

124. Christensen, S. G. and Rousøe, D. M., "Container loading with multi-drop constraints", ***International Transactions in Operational Research***, 16 (6): 727–743 (2009).

125. He, K. and Huang, W., "A caving degree based flake arrangement approach for the container loading problem", ***Computers and Industrial Engineering***, 59 (2): 344–351 (2010).

126. Parreno, F., Alvarez-Valdes, R., Oliveira, J. F., and Tamarit, J. M., "A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing", ***Annals of Operations Research***, 179 (1): 203–220 (2010).

127. Dereli, T. and Daş, G. S., "Development of a decision support system for solving container loading problems", ***Transport***, 25 (2): 138–147 (2010).

128. He, K. and Huang, W., "A quasi-human algorithm for solving the three-dimensional rectangular packing problem", ***Science China Information Sciences***, 53 (12): 2389–2398 (2010).

129. Pires de Araujo, L. J. and Pinheiro, P., "Combining heuristics backtracking and genetic algorithm to solve the container loading problem with weight distribution", ***Soft Computing Models in Industrial and Environmental Applications, 5th International Workshop (SOCO 2010)***, ***Springer***, 73: 95–102 (2010).

130. Fanslau, T. and Bortfeldt, A., "A tree search algorithm for solving the container loading problem", ***INFORMS Journal on Computing***, 22 (2): 222–235 (2010).

131. Shiau, J. Y. and Lee, M. C., "A warehouse management system with sequential picking for multi-container deliveries", ***Computers and Industrial Engineering***, 58 (3): 382–392 (2010).

132. Parreno, F., Alvarez-Valdes, R., Oliveira, J. F., and Tamarit, J. M., "Neighborhood structures for the container loading problem: a VNS implementation", ***Journal of Heuristics***, 16 (1): 1–22 (2010).

133. Ortmann, F. G., Ntene, N., and van Vuuren, J. H., "New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems", ***European Journal of Operational Research***, 203 (2): 306–315 (2010).

134. Torra, V., Cano, I., Miyamoto, S., and Endo, Y., "Container loading for nonorthogonal objects: An approximation using local search and simulated annealing", ***Soft Computing***, 14 (5): 537–544 (2010).

135. Dereli, T. and Sena Das, G., "A hybrid simulated annealing algorithm for solving multi-objective container-loading problems", ***Applied Artificial Intelligence***, 24 (5): 463–486 (2010).

136. Egeblad, J., Garavelli, C., Lisi, S., and Pisinger, D., "Heuristics for container loading of furniture", ***European Journal of Operational Research***, 200 (3): 881–892 (2010).

137. Kang, M.-K., Jang, C.-S., and Yoon, K.-S., "Heuristics with a new block strategy for the single and multiple containers loading problems", ***Journal of the Operational Research Society***, 61 (1): 95–107 (2010).

138. Cano, I. and Torra, V., "Particle swarm optimization for container loading of nonorthogonal objects", ***Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)***, 6114 LNAI (PART 2): 403–410 (2010).

139. He, K. and Huang, W., "Solving the single-container loading problem by a fast heuristic method", ***Optimization Methods and Software***, 25 (2): 263–277 (2010).

140. Wu, Y., Li, W., Goh, M., and de Souza, R., "Three-dimensional bin packing problem with variable bin height", ***European Journal of Operational Research***, 202 (2): 347–355 (2010).

141. Koloch, G. and Kaminski, B., "Vehicle routing with three-dimensional container loading constraints - comparison of nested and joint algorithms", ***AIP Conference Proceedings***, 1285 (2010): 145–157 (2010).

142. Liu, J., Yue, Y., Dong, Z., Maple, C., and Keech, M., "A novel hybrid tabu search approach to container loading", ***Computers and Operations Research***, 38 (4): 797–807 (2011).

143. Ren, J., Tian, Y., and Sawaragi, T., "A tree search method for the container loading problem with shipment priority", ***European Journal of Operational Research***, 214 (3): 526–535 (2011).

144. He, K. and Huang, W., "An efficient placement heuristic for three-dimensional rectangular packing", ***Computers and Operations Research***, 38 (1): 227–233 (2011).

145. Dereli, T. and Das, G. S., "A hybrid "bee(s) algorithm" for solving container loading problems", ***Applied Soft Computing***, 11 (2): 2854–2862 (2011).

146. Che, C. H., Huang, W., Lim, A., and Zhu, W., "The multiple container loading cost minimization problem", ***European Journal of Operational Research***, 214 (3): 501–511 (2011).

147. Ma, H. W., Zhu, W., and Xu, S., "Research on the algorithm for 3l-cvrp with considering the utilization rate of vehicles", ***Intelligent Computing and Information Science,*** Pt I, 621–629 (2011).

148. Junqueira, L., Morabito, R., and Sato Yamashita, D., "MIP-based approaches for the container loading problem with multi-drop constraints", ***Annals of Operations Research***, 199 (1): 51–75 (2012).

149. He, Y., Wu, Y., and de Souza, R., "A global search framework for practical three-dimensional packing with variable carton orientations", ***Computers and Operations Research***, 39 (10): 2395–2414 (2012).

150. Zhang, D. F., Peng, Y., and Leung, S. C. H., "A heuristic block-loading algorithm based on multi-layer search for the container loading problem", ***Computers & Operations Research***, 39 (10): 2267–2276 (2012).

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

152. Wei, L., Oon, W. C., Zhu, W., and Lim, A., "A reference length approach for the 3D strip packing problem", ***European Journal of Operational Research***, 220 (1): 37–47 (2012).

153. Bortfeldt, A. and Jungmann, S., "A tree search algorithm for solving the multi-dimensional strip packing problem with guillotine cutting constraint", ***Annals of Operations Research***, 196 (1): 53–71 (2012).

154. Zhu, W., Qin, H., Lim, A., and Wang, L., "A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP", ***Computers and Operations Research***, 39 (9): 2178–2195 (2012).

155. Lim, A., Ma, H., Xu, J., and Zhang, X., "An iterated construction approach with dynamic prioritization for solving the container loading problems", ***Expert Systems with Applications***, 39 (4): 4292–4305 (2012).

156. Burke, E. K., Hyde, M. R., Kendall, G., and Woodward, J., "Automating the packing heuristic design process with genetic programming", ***Evolutionary Computation***, 20 (1): 63–89 (2012).

157. Yap, C. N., Lee, L. S., Majid, Z. a., and Seow, H. v., "Ant colony optimization for container loading problem", ***Journal of Mathematics and Statistics***, 8 (2): 169–175 (2012).

158. Liu, J., Zhang, X., Yue, Y., and Maple, C., "Effectively handling three-dimensional spaces for container loading", AIP Conference Proceedings, ***American Institute of Physics***, 1479 (1): 1960 – 1963 (2012).

159. Mack, D. and Bortfeldt, A., "A heuristic for solving large bin packing problems in two and three dimensions", ***Central European Journal of Operations Research***, 20 (2): 337–354 (2012).

160. Zhu, W. and Lim, A., "A new iterative-doubling greedy-lookahead algorithm for the single container loading problem", ***European Journal of Operational Research***, 222 (3): 408–417 (2012).

161. Gonçalves, J. F. and Resende, M. G. C., "A parallel multi-population biased random-key genetic algorithm for a container loading problem", ***Computers and Operations Research***, 39 (2): 179–190 (2012).

162. Zhu, W., Huang, W., and Lim, A., "A prototype column generation strategy for the multiple container loading problem", ***European Journal of Operational Research***, 223 (1): 27–39 (2012).

163. Allen, S. D., Burke, E. K., and Mareček, J., "A space-indexed formulation of packing boxes into a larger box", ***Operations Research Letters***, 40 (1): 20–24 (2012).

164. Thapatsuwan, P., Pongcharoen, P., Hicks, C., and Chainate, W., "Development of a stochastic optimization tool for solving the multiple container packing problems", ***International Journal of Production Economics***, 140 (2): 737–748 (2012).

165. Zhu, W., Zhang, Z., Oon, W. C., and Lim, A., "Space defragmentation for packing problems", ***European Journal of Operational Research***, 222 (3): 452–462 (2012).

166. Zhu, W., Oon, W. C., Lim, A., and Weng, Y., "The six elements to block-building approaches for the single container loading problem", ***Applied Intelligence***, 37 (3): 431–445 (2012).

167. Junqueira, L., Morabito, R., and Sato Yamashita, D., "Three-dimensional container loading models with cargo stability and load bearing constraints", ***Computers and Operations Research***, 39 (1): 74–85 (2012).

168. Ceschia, S. and Schaerf, A., "Local search for a multi-drop multi-container loading problem", ***Journal of Heuristics***, 19 (2): 275–294 (2013).

169. Bortfeldt, A., "A reduction approach for solving the rectangle packing area minimization problem", ***European Journal of Operational Research***, 224 (3): 486–496 (2013).

170. Ho, Z. F., Lee, L. S., Majid, Z. A., and Seow, H. V., "An improved GRMOD heuristic for container loading problem", ***AIP Conference Proceedings***, 1557: 439–443 (2013).

171. Junqueira, L., Oliveira, J. F., Carravilla, M. A., and Morabito, R., "An optimization model for the vehicle routing problem with practical three-dimensional loading constraints", ***International Transactions in Operational Research***, 20 (5): 645–666 (2013).

172. Mustafee, N. and Bischoff, E. E., "Analyzing trade-offs in container loading: Combining load plan construction heuristics with agent-based simulation", ***International Transactions in Operational Research***, 20 (4): 471–491 (2013).

173. Alvarez-Valdes, R., Parreño, F., and Tamarit, J. M., "A GRASP/Path relinking algorithm for two- and three-dimensional multiple bin-size bin packing problems", ***Computers and Operations Research***, 40 (12): 3081–3090 (2013).

174. Tian, T., Lim, A., and Zhu, W., "A heuristic to the multiple container loading problem with preference", ***Contemporary Challenges and Solutions in Applied Artificial Intelligence***, ***Springer***, 219–224 (2013).

175. Wang, N., Lim, A., and Zhu, W., "A multi-round partial beam search approach for the single container loading problem with shipment priority", ***International Journal of Production Economics***, 145 (2): 531–540 (2013).

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

177. Lim, A., Ma, H., Qiu, C., and Zhu, W., "The single container loading problem with axle weight constraints", ***International Journal of Production Economics***, 144 (1): 358–369 (2013).

178. Alonso, M. T., Valdes, R. A., Tamarit, J. M., and Parreño, F., "A reactive GRASP algorithm for the container loading problem with load-bearing constraints", ***European J. of Industrial Engineering***, 8 (5): 669 (2014).

179. Sheng, L., Tan, W., Zhiyuan, X., and Xiwei, L., "A tree search algorithm for the container loading problem", ***Computers & Industrial Engineering***, 75: 20–30 (2014).

180. Ramos, A. G., Oliveira, J. F., Gonçalves, J. F., and Lopes, M. P., "Dynamic stability metrics for the container loading problem", ***Transportation Research Part C: Emerging Technologies***, 60: 480–497 (2015).

181. Zheng, J. N., Chien, C. F., and Gen, M., "Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem", ***Computers and Industrial Engineering***, 89: 80–87 (2015).

182. Wei, L., Zhu, W., and Lim, A., "A goal-driven prototype column generation strategy for the multiple container loading cost minimization problem", ***European Journal of Operational Research***, 241 (1): 39–49 (2015).

183. Martínez, D. A., Alvarez-Valdes, R., and Parreño, F., "A grasp algorithm for the container loading problem with multi-drop constraints", ***Pesquisa Operacional***, 35 (1): 1–24 (2015).

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

185. Karoonsoontawong, A. and Heebkhoksung, K., "A modified wall-building-based compound approach for the knapsack container loading problem", ***Maejo International Journal of Science and Technology***, 9 (01): 93–107 (2015).

186. Junqueira, L. and Morabito, R., "Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier", ***Computers and Industrial Engineering***, 88: 110–130 (2015).

187. Escobar-Falcón, L. M., Álvarez-Martínez, D., Granada-Echeverri, M., Escobar, J. W., and Romero-Lázaro, R. A., "A matheuristic algorithm for the three-dimensional loading capacitated vehicle routing problem (3L-CVRP)", ***Revista Facultad de Ingeniería Universidad de Antioquia***, (78): 9–20 (2016).

188. Iwasawa, H., Hu, Y., Hashimoto, H., and Imahori, S., "A heuristic algorithm for the container loading problem with complex loading constraints", ***Journal of Advanced Mechanical Design, Systems, and Manufacturing***, 10 (3): 1–12 (2016).

189. Wang, H., Zhang, X., and Wang, S., "A joint optimization model for liner container cargo assignment problem using state-augmented shipping network framework", ***Transportation Research Part C: Emerging Technologies***, 68: 425–446 (2016).

190. Ramos, A. G., Oliveira, J. F., and Lopes, M. P., "A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions", ***International Transactions in Operational Research***, 23 (1–2): 215–238 (2016).

191. Huang, Y. H., Hwang, F. J., and Lu, H. C., "An effective placement method for the single container loading problem", ***Computers and Industrial Engineering***, 97: 212–221 (2016).

192. McDonald, C. M., "Integrating packaging and supply chain decisions: Selection of economic handling unit quantities", ***International Journal of Production Economics***, 180: 208–221 (2016).

193. Gonzalez, Y., Miranda, G., and Leon, C., "Multi-objective multi-level filling evolutionary algorithm for the 3d cutting stock problem", ***Procedia Computer Science***, 96: 355–364 (2016).

194. Brinker, J. and Gündüz, H. I., "Optimization of demand-related packaging sizes using a p-median approach", ***The International Journal of Advanced Manufacturing Technology***, 87 (5–8): 2259–2268 (2016).

195. Galrão Ramos, A., Oliveira, J. F., Gonçalves, J. F., and Lopes, M. P., "A container loading algorithm with static mechanical equilibrium stability constraints", ***Transportation Research Part B: Methodological***, 91: 565–581 (2016).

196. Jamrus, T. and Chien, C. F., "Extended priority-based hybrid genetic algorithm for the less-than-container loading problem", ***Computers and Industrial Engineering***, 96: 227–236 (2016).

197. Tian, T., Zhu, W., Lim, A., and Wei, L., "The multiple container loading problem with preference", ***European Journal of Operational Research***, 248 (1): 84–94 (2016).

198. Costa, M. da G. and Captivo, M. E., "Weight distribution in container loading: A case study", ***International Transactions in Operational Research***, 23 (1–2): 239–263 (2016).

199. Moura, A. and Bortfeldt, A., "A two-stage packing problem procedure", ***International Transactions in Operational Research***, 24 (1–2): 43–58 (2016).

200. Toffolo, T. A. M., Esprit, E., Wauters, T., and vanden Berghe, G., "A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem", ***European Journal of Operational Research***, 257 (2): 526–538 (2017).

201. Correcher, J. F., Alonso, M. T., Parreño, F., and Alvarez-Valdes, R., "Solving a large multi container loading problem in the car manufacturing industry", ***Computers and Operations Research***, 82: 139–152 (2017).

202. Araya, I., Guerrero, K., and Nuñez, E., "VCS: A new heuristic function for selecting boxes in the single container loading problem", ***Computers and Operations Research***, 82: 27–35 (2017).

203. Lee, K. Y. and El-Sharkawi, M. A., "Modern heuristic optimization techniques: theory and applications to power systems", ***John Wiley & Sons***, (2008).

204. Bayraktar, T., Aydin, M. E., and Dugenci, M., "A memory-integrated artificial bee algorithm for 1-D bin packing problems", ***Proc. CIE IMSS***, 1023 – 1034 (2014).

205. Kirchmaier, U., Hawe, S., and Diepold, K., "A swarm intelligence inspired algorithm for contour detection in images", ***Applied Soft Computing***, 13 (6): 3118–3129 (2013).

206. Karaboga, D. and Basturk, B., "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", ***Journal of Global Optimization***, 39 (3): 459–471 (2007).

207. von Frisch, K., "Decoding the language of the bee", ***Science***, 185 (4152): 663–668 (1974).

208. Pulikanti, S. and Singh, A., "An artificial bee colony algorithm for the quadratic knapsack problem", ***International Conference on Neural Information Processing***, ***Springer***, 196 – 205 (2009).

209. Sundar, S. and Singh, A., "A swarm intelligence approach to the quadratic multiple knapsack problem", ***International conference on neural information processing***, ***Springer***, 626 – 633 (2010).

210. Sabet, S., Shokouhifar, M., and Farokhi, F., "A discrete artificial bee colony for multiple knapsack problem", ***International Journal of Reasoning-Based Intelligent Systems***, 5 (2): 88–95 (2013).

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

212. Jia, D., Duan, X., and Khan, M. K., "Binary artificial bee colony optimization using bitwise operation", ***Computers & Industrial Engineering***, 76: 360–365 (2014).

213. Hancer, E., Xue, B., Karaboga, D., and Zhang, M., "A binary ABC algorithm based on advanced similarity scheme for feature selection", ***Applied Soft Computing***, 36: 334–348 (2015).

214. Ozturk, C., Hancer, E., and Karaboga, D., "Dynamic clustering with improved binary artificial bee colony algorithm", ***Applied Soft Computing***, 28: 69–80 (2015).

215. Glover, F., "Tabu search—part I", ***ORSA Journal On Computing***, 1 (3): 190–206 (1989).

216. Moayedikia, A., Jensen, R., Wiil, U. K., and Forsati, R., "Weighted bee colony algorithm for discrete optimization problems with application to feature selection", ***Engineering Applications of Artificial Intelligence***, 44: 153–167 (2015).

217. Singhal, P. K., Naresh, R., and Sharma, V., "A modified binary artificial bee colony algorithm for ramp rate constrained unit commitment problem", ***International Transactions on Electrical Energy Systems***, 25 (12): 3472–3491 (2015).

218. Luo, K., "A hybrid binary artificial bee colony algorithm for the satellite photograph scheduling problem", ***Engineering Optimization***, 52 (8): 1421–1440 (2020).

219. Elghamrawy, S. M., "Security in cognitive radio network: defense against primary user emulation attacks using genetic artificial bee colony (GABC) algorithm", ***Future Generation Computer Systems***, 109: 479–487 (2020).

220. Chu, X., Li, S., Gao, D., Zhao, W., Cui, J., and Huang, L., "A binary superior tracking artificial bee colony with dynamic cauchy mutation for feature selection", ***Complexity***, 2020: (2020).

221. Chen, M.-R., Chen, J.-H., Zeng, G.-Q., Lu, K.-D., and Jiang, X.-F., "An improved artificial bee colony algorithm combined with extremal optimization and Boltzmann Selection probability", ***Swarm and Evolutionary Computation***, 49: 158–177 (2019).

222. Bansal, J. C., Sharma, H., Arya, K. v, and Nagar, A., "Memetic search in artificial bee colony algorithm", ***Soft Computing***, 17 (10): 1911–1928 (2013).

223. Kumar, S., Sharma, V. K., and Kumari, R., "An improved memetic search in artificial bee colony algorithm", ***Int J Comput Sci Inform Technol (0975–9646)***, 5 (2): 1237–1247 (2014).

224. Jia, D., Li, T., Zhang, Y., and Wang, H., "A memetic artificial bee colony algorithm for high dimensional problems", ***International Journal of Computational Intelligence and Applications***, 19 (01): 2050008 (2020).

225. Crainic, T. G., Perboli, G., and Tadei, R., "Extreme point-based heuristics for three-dimensional bin packing", ***Informs Journal on Computing***, 20 (3): 368–384 (2008).

226. Bortfeldt, A. and Gehring, H., "Applying tabu search to container loading problems", ***Operations Research Proceedings***, ***Springer***, 533–538 (1998).

227. Zhou, Q. and Liu, X., "A swarm optimization algorithm for practical container loading problem", ***IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society***, ***Springer***, 5690 – 5695 (2017).

228. Lin, W.-C., Xu, J., Bai, D., Chung, I.-H., Liu, S.-C., and Wu, C.-C., "Artificial bee colony algorithms for the order scheduling with release dates", ***Soft Computing***, 23 (18): 8677–8688 (2019).

229. Bhambu, P., Sharma, S., and Kumar, S., "Modified g-best artificial bee colony algorithm", ***Soft Computing: Theories and Applications***, ***Springer***, 665–677 (2018).

230. Ivancic, N. J., “An integer programming based heuristic approach to the three-dimensional packing problem”, Doctoral Dissertation, ***Case Western Reserve University***, (1988).

231. Menghani, D. and Guha, A., "Packing boxes into multiple containers using genetic algorithm", ***Journal Of The Institution of Engineers (India): Series C***, 97 (3): 441–450 (2016).

EK AÇIKLAMALAR A.

DOKTORA TEZİNDEN ÜRETİLİP ULUSLARARASI HAKEMLİ DERGİDE YAYIMLANAN BİLİMSEL MAKALE

Applied Soft Computing 108 (2021) 107462

Contents lists available at ScienceDirect

Applied Soft Computing

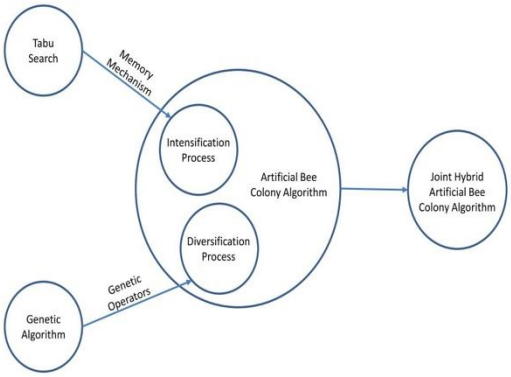
journal homepage: www.elsevier.com/locate/asoc

Effects of memory and genetic operators on Artificial Bee Colony algorithm for Single Container Loading problem

Tuğrul Bayraktar [a,*], Filiz Ersöz [a], Cemalettin Kubat [b]

[a] Karabuk University, Turkey
[b] Hasan Kalyoncu University, Turkey

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 24 July 2020
Received in revised form 21 March 2021
Accepted 21 April 2021
Available online 30 April 2021

Keywords:
Artificial Bee Colony algorithm
Tabu search
Genetic algorithm
Single Container Loading problem
Knapsack problem

ABSTRACT

The Artificial Bee Colony (ABC) algorithm is widely used to achieve optimum solution in a short time in integer-based optimization problems. However, the complexity of integer-based problems such as Knapsack Problems (KP) requires robust algorithms to avoid excessive solution search time. ABC algorithm that provides both the exploitation and the exploration approach is used as an alternative approach for various KP problems in the literature. However, it is rarely used for the Single Container Loading problem (SCLP) which is an important part of the transportation systems. In this study, the exploitation and exploration aspects of the ABC algorithm are improved by using memory mechanisms and genetic operators to develop three different hybrid ABC algorithms. The developed algorithms and the basic ABC algorithm are applied to a SCLP dataset from the literature to observe the effects of the memory mechanism and the genetic operators separately. Besides, a joint hybrid ABC algorithm using both reinforcement approaches is proposed to solve the SCLP. The results show that the joint hybrid ABC algorithm has emerged as a promising approach to solving SCLP with an average performance, and the genetic operators are more effective than the memory mechanism to develop a hybrid ABC algorithm.

1. Introduction

A container is the one of the main components of transportation systems. Widely used containers have standardized dimensions, to be loaded by several items for same or different location delivery. Single Container loading problems (SCLP) is a

* Corresponding author.
E-mail addresses: tugrulbayraktar@karabuk.edu.tr (T. Bayraktar), fersoz@karabuk.edu.tr (F. Ersöz), kubat@sakarya.edu.tr (C. Kubat).

https://doi.org/10.1016/j.asoc.2021.107462
1568-4946/

branch of knapsack problems (KP) that involves the allocation of a set of three-dimensional orthogonal items into a limited capacitated three-dimensional orthogonal domain. The main purpose of SCLP is to maximize container loading ratio under the following loading constraints; (C1) orientation constraint: the given set of items can be placed in a limited way and one or two dimensions of items cannot be oriented vertically, (C2) top placement constraint: each item can bear a limited weight and further ones cannot be allocated on top of the item that reaches its weight-bearing capacity, (C3) weight constraint: the weight of item pile may be limited to avoid overloading, (C4) stability constraint: each item should be supported by the items below with its required bottom area ratio and only the given amount of non-stable items is allowed to fall, (C5) balance constraint: items should be allocated into the container by considering the container's center of gravity [1].

SCLP has been studied by many researchers that propose different heuristic approaches to improve the container loading technique. Genetic algorithm (GA) [1–4], tree search (TrS) [5–7], and greedy randomized adaptive search procedure (GRASP) [8,9] are commonly applied heuristic approaches for solving SCLP. Additionally, tabu search (TS) [10–13], bee's algorithm (BA) [14], and swarm optimization algorithm (SOA) [15] are examples of heuristic approaches used to solve SCLP.

Gehring and Bortfeldt proposed one of the first genetic algorithms for SCLP by considering loading constraints that they stated. The container is loaded by towers in an order and the basic genetic algorithm is used to improve the order of towers which built by a set of items [1,2]. Bortfeldt and Gehring proposed a hybrid genetic algorithm that shows some differences by [1] item loading strategy and algorithmic construction. Items are grouped to form layers that load the container. Genetic operators exchange the layers between chromosomes to generate new child solutions. Besides, the mutation operator merges layers to improve the solution by filling the gaps as much as possible [2]. Araujo and Pinheiro proposed encoding the item placement options, such as layering, item stack or block building, or partitioning these item stacks. The sequences of options are improved by genetic operators to generate a new item placement procedure [3]. Yeh et al. present an artificial neural network (ANN) that learns the output values of GA-optimized container loading solutions to improve the ANN weight parameters [4].

Bortfeldt and Gehring also applied TS for SCLP to observe the different effects of basic GA and basic TS. Each allocated item generates the residual packing spaces in the container. TS minimizes the volume of the residual packing spaces by selecting the best item placement move between all possible container loading plans [10]. Liu et al. proposed a hybrid tabu search that is different from the [10] item placement approach. The item pile can be placed into the container vertically or horizontally in the initial solutions. Besides, items can be selected individually or as a group for the next move to minimize the residual packing spaces [11]. Mack et al. developed a hybrid algorithm that uses TS as a reinforcement for SA to avoid re-generating abandoned solutions in the reheating process [12]. Zhu et al. considered SCLP as a part of capacitated vehicle routing problem (CVRP) and use TS to optimize the container loading and vehicle routing process together [13].

GRASP is a type of greedy search method and also a commonly used heuristic approach to minimize the residual packing spaces by matching the best item–space pairs. Parreno et al. propose a GRASP method that allocates the items individually or as a group to utilize the residual packing spaces. After the initial solution is obtained, the residual packing spaces shrink with the item exchange between allocated and non-allocated ones [8]. Alonso et al. proposed a GRASP method that selects one of the item placement heuristics that matches item–space pairs with a varying probability throughout iterations [9].

The tree search (TrS) method is a well-performed container loading heuristic that optimizes the loading ratio by selecting one of the feasible item placement plans. After each item's allocation process called node in TrS, new item allocation candidates emerge as subsequent nodes. The objective is to optimize the loading ratio of x-y-z planes at each node. Wang et al. defined the residual packing spaces dynamically and selected among the one of three item–space pair candidates [5]. Ren et al. introduced shipment priority as an additional container loading constraint to the basic five constraints and used TrS that was based on block building strategy [6]. Sheng et al. applied the TrS method based on a wall building strategy that minimizes the residual packing space at each item allocation step and compares the method with other heuristic approaches from the literature [7].

The grasshopper optimization algorithm (GOA), recently developed by Saremi et al. is a new type of swarm based heuristic approach inspired by the behavior of grasshopper swarms. Three parametric values that affect the behavior of individuals; social interaction, gravity force and wind advection. Individuals tend to explore promising solutions and cluster around the global optima under the influence of these three factors [16]. Zhou and Liu introduced a GOA-inspired swarm optimization algorithm (SOA) for SCLP, based on block building strategy. The behavior of individuals is defined by the Hamming distance between solution pairs. The solutions move away from each other if the distance value is smaller than the value of k, or tend to get close if the distance value is bigger than the value of k. The best solution influences all others to cluster around the promising areas of the search space [15].

Bee(s) algorithm (BA) is also a swarm based heuristic approach inspired by the mating and foraging behavior of bee colonies. The bees in the colony are classified according to their fitness value and the pre-determined percentage of the colony with the highest fitness value is classified into a different group called elite bees. The colony tends to search around the solution obtained by elite bees. Dereli and Das proposed a hybrid BA for SCLP, based on layer building strategy, that represents the item type and orientation as bit string in a row. Neighborhood solutions are generated from existing ones by changing the position of bits in the string with 1-flip, swap, and k-flip operators [14].

The artificial bee colony (ABC) algorithm is a neuro-inspired meta-heuristic approach based on swarm intelligence and it has been increasing the interest in the last decade by providing solutions for various optimization problems. Many types of research have proposed a modified or hybridized ABC algorithm to improve the capability of solution exploration [17,18]. KP is an integer-based non-deterministic polynomial-time (NP)-hard problem and involves placing items into capacitated domains such as bins and containers. Solution approaches for KP vary according to the number of domains that can be single or multiple. Items in a loading sequence can be denoted by binary numbers 0–1 to see whether the item is assigned to the single knapsack [19–21]. On the other hand, the knapsack number in a loading sequence denotes the knapsack that items are assigned to when the problem involves more than one knapsack [22,23].

Previous researches [19–21,24] prove that the ABC algorithm is an effective approach to solve Knapsack Problem (KP) in which the number of dimensions is mostly regarded as a number of parameters for each item. On the other hand, most of the SCLP studies [1–13] that have developed various heuristic approaches for loading solution improvement focus on matching the residual packing space with items, blocks, and layers during each item placement step, without considering other feasible options throughout the container loading process. This means that a preloading process is necessary to evaluate the items to be classified.

In this study, the item assignment order and item orientation, represented as decision variables in the item placement sequence, was pre-determined without any classification at the beginning of the container loading process. By simply changing the elements of solutions one by one with the ABC algorithm, it is possible to provide more diverse solutions than the residual packing space placement approach offers and increase the number of feasible container loading plan options. Therefore, it can be considered as an alternative approach for solving Single Container Loading Problem (SCLP) that has not been studied in the known literature. The genetic operators that are provided by a genetic algorithm (GA) [17,25,26] and the memory mechanism provided by tabu search (TS) [24,27,28] are used as reinforcement approaches to develop hybrid ABC algorithms. However, a comparison between the effects of the genetic operators and memory mechanisms that improve different aspects of the ABC algorithm has been scarcely studied in the literature. Therefore, the SCLP can be addressed to observe the performance of the basic ABC algorithm in high complex problems. Besides, two separate hybrid ABC algorithms that are applied to the SCLP can help in understanding the effects of the memory mechanism and genetic operators that are integrated into the basic ABC algorithm. In this context, the main purpose of this study is to compare the effects of reinforcement approaches on the basic ABC algorithm by applying them to the SCLP.

GA is an evolutionary algorithm (EA) that is based on improving the individuals of the population, which are generated randomly, through iterations. GA employs crossover and mutation operators diversify the performance value of the individuals that represent the problem solutions [1]. GA is not able to exploit the search sufficiently in the local search phase despite a strong exploration aspect. The ABC algorithm operators can be used as improvement tools for GA in order not to miss the global optimum solution in the local search area. Singh et al. [18] used the ABC algorithm operators to find the fruitful ones among the solutions generated by using GA operators. First, child and grandchild (off-spring) solutions are generated from two random individuals in the population by cross-over and mutation operators respectively. Then, ABC algorithm operators, employed bees, and onlooker bees are used to generate new candidate solutions around the off-spring solutions in two phases. The best of all generated solutions replaces the current solution if it is better. The approach proposed by Singh et al. performs better than the basic GA for all cases.

GA always generates new individuals from existing ones and eliminates the worst ones among the surplus individuals in the population. On the other hand, the ABC algorithm replaces the abandoned individuals with new randomly generated ones. Thus, all improvement information about the abandoned solutions is deleted in the re-generation (re-scout) phase of the ABC algorithm and the fruitful part of abandoned solutions cannot be transferred to new generations. Ozturk et al. [17] used the genetic operators in both the onlooker bees and the employed bees phase of the ABC algorithm. First, two random neighbor solutions x_j, x_k of the current solutions x_i, x_{best} and x_{zero} are generated for each x_i solution. Then, the crossover operator is applied to the randomly generated couples among five $(x_i, x_j, x_k, x_{best}, x_{zero})$ solutions to generate child solutions. Finally, the mutation operator is applied to child solutions to generate grandchild solutions and the best of all generated solutions replaces the current solution x_i, if it is better. Panahi and Navimipour [25] used the same approach to benefit from the simplicity of the algorithm structure which is considered as easily applicable for general binary optimization problems.

The neighbor solutions x_j in local search can be generated by the genetic operator-based interaction of the current solutions x_i and the randomly selected solution x_k among employed bees, where $x_i \neq x_k$. Chaurasia et al. [26] intensified the solution search in the local search phase of the ABC algorithm by generating neighbors directly, rather than selecting from a group of genetic operator-based generated solutions. First, x_i and x_k are used for multi-point insert to generate a raw neighbor solution x_j. Then, a 3-point swap operator processes the x_j to obtain the finalized form of the neighbor solution. The approach proposed by Chaurasia et al. outperformed the hybrid GA that was proposed in the same study by using the ABC algorithm's exploitation strategy.

Local search in the ABC algorithm provides repetitive attempts to improve the obtained solution until a predetermined failure limit is exceeded. After that, the solution is erased from the memory of the bee colony. Besides, the improvement attempts are not memorized during the local search phase in the ABC algorithm, so several types of research suggest a memory-integrated ABC algorithm to expand the search area by using the TS strategy [24,27,28].

TS strategy constraints the search by classifying the moves which represent the transition between the current solution and the newly generated one. If the move does not improve the solution, it is forbidden for a short period and saved in the Short-Term Tabu List (STTL) [29]. If the move improves the solution in the exploitation phase, it is saved in the Intermediate-Term Tabu List (ITTL) and saved in the Long-Term Tabu List (LTTL) if it improves in the exploration phase [30]. The saved moves guide the search to exploit the unexplored solutions so that the convergence time to obtain the global optimum can be reduced and even the global optimum can be improved by unexplored solutions.

TS strategy provides a memory mechanism that avoids repetitive evaluation of the same solution. Li and Yang [27] integrated a memory mechanism in the ABC algorithm to save the parameter pair k and Φ of a successful move. If the saved parameter pair succeeds again the next time they are used, they remain in the memory. Otherwise, they are deleted from the memory. Chengli et al. [28] used a memory mechanism, in the same manner, to save parameter trio l_{best_i}, k_{i_j} and Φ_{i_j} for another move. The moves with memorized parameters guide the search to escape the local optimum. The approach proposed by Li and Yang is not sensitive to the size of the memory so that only the performance of memorized parameters affects the performance of the algorithm.

Yin and Chuang [24] proposed a memory integrated ABC algorithm for CVRP to optimize the vehicle routes and the shipment priority. The strategic oscillation that relaxes the problem constraints and then realizes the infeasible solutions, is used in both the employed bees and the onlooker bees phase of the ABC algorithm. The solution transitions during the strategic oscillation are considered as moves to save in tabu lists. If the move improves the current solution, it is saved in LTTL, else it is saved in STTL. The memorized moves guide the search to avoid repetitive evaluation for fruitless solutions and exploit fruitful ones more. In the previous study [31], we proposed a memory-integrated ABC (MIABC) algorithm by using STTL for a one-dimensional bin packing problem (1-D BPP) to avoid repetitive solutions for a limited number of iterations. The MIABC algorithm provides a rapid convergence to the optimum solution when compared with the basic ABC algorithm.

This study aims to improve both the exploration and exploitation aspects of the ABC algorithm by using the new hybrid ABC algorithm. The genetic operator and memory mechanism are selected by the proposed weighted local search approach, which determines the search path based on the past success rate of the approaches used. A MIABC, genetic operator-based ABC algorithm (GABC) and the new joint hybrid ABC algorithm (JHABC) with genetic operator and memory are applied to solve SCLP

respectively. The MIABC and GABC are compared to understand the effects of memory mechanism and genetic operators on SCLP implementation of the ABC algorithm.

The rest of the paper is organized as follows: the brief explanation of SCLP and the way of its implementation in this study is provided in Section 2; the proposed solution approaches are explained in Section 3; results from applied approaches are represented in Section 4, and Section 5 concludes with the future work and inferences about research undertaken.

2. Problem statement

SCLP is a branch of KP [17] that is based on placing a set of three-dimensional rectangular items with width (w_i), depth (d_i) and height (h_i) and the volume of the item (v_i) that equals to $w_i \times d_i \times h_i$, in a rectangular container with dimensions [18], where $w_i \leq W$, $d_i \leq D$ and $h_i \leq H$ and the capacity of the container (V) that equals to $W \times D \times H$.

The container loading process is restricted by various factors such as the weight-bearing capacity of items, orientation, stability, container weight balance, capacity [1], and item fragility [32]. In addition, the item placement order is determined by shipment priority in CVRP [33].

In SCLP, the aim is to maximize the utilization ratio of the container without item intersection and dimensional exceeding. If the items have values, the total value is maximized instead of the utilization ratio. The total utilization ratio (UR) is calculated as in Eq. (1) and Eq. (2), without considering any other variables but the volume of the assigned items:

$$maximize \sum_{i=1}^{n} \frac{a_i.v_i}{V} \tag{1}$$

$$subject\ to \sum_{i=1}^{n} a_i.v_i \leq V \tag{2}$$

$$where; a_i \in [0, 1], i = 1, \ldots, n.$$

UR in Eq. (1) is equal to the ratio of the total volume of the assigned items to container capacity where a_i is a decision variable indicating whether the item is assigned to be placed in the container. The amount of the total volume of the assigned items cannot be larger than the container capacity, as in Eq. (2) and the item intersection is not allowed.

SCLP can be associated with CVRP by considering vehicle routing parameters. In this case, the loading value or ratio and transportation cost are optimized together [34]. Besides, container cost is taken into account in case CLP deals with more than one container with different dimensions [35].

In this study, we deal with a single container loading problem (SCLP) to maximize the utilization ratio. *w, l, and h* dimensions of the assigned items are parallel to the W, L, and H dimensions of the container. The origin is on the deep-bottom-left (DBL) corner of the container in a three-dimensional coordinate system. The assigned items are placed on the available deeper and bottom coordinates of the container as in Fig. 1.

The assigned items form vertical layers starting from the deepest part of the container. Each item can be rotated in six orientations whose dimensions are (*w, l, h*), (*l, w, h*), (*h, l, w*), (*l, h, w*), (*w, h, l*), and (*h, w, l*) and numbered as in Fig. 2. However, the orientation type is restricted by SCLP constraints that allow items to be rotated only 90° in some cases. The problem addressed in this study includes three different orientation restrictions: (i) all orientations allowed, (ii) orientation types 0, 1, 2, 3 allowed, and (iii) orientation types 0, 1, 4, 5 allowed. In this case, the transition between the orientations is restricted while optimizing the orientation type.

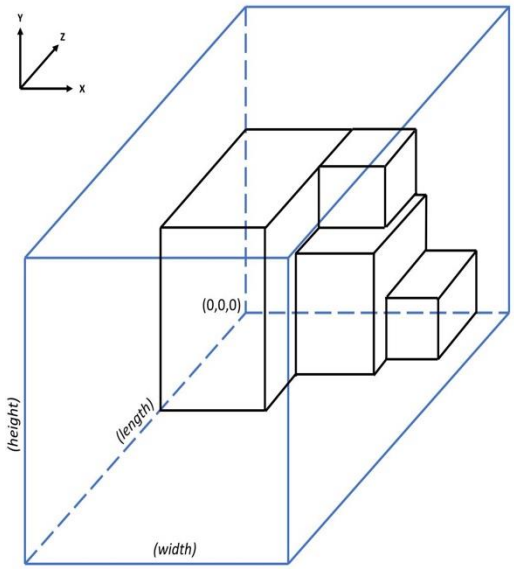


Fig. 1. Container and assigned items in three-dimensional coordinate system.

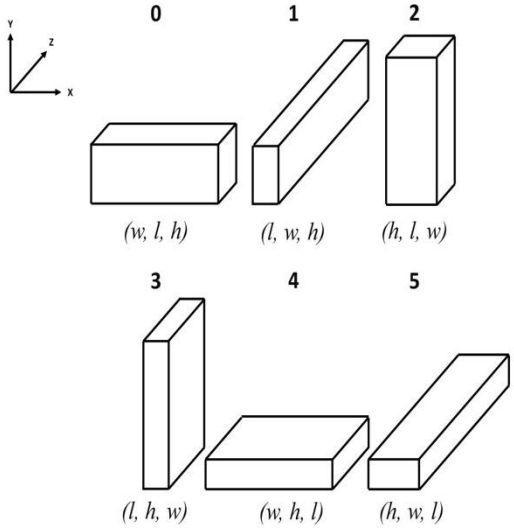


Fig. 2. Item orientations.

The orientation number $orient_i$ ($orient = 0, \ldots, 5$) indicates with which orientation the item is placed in the container. If an item is assigned to the container, the assignment variable of the item (a_i) is determined as 1 and a_i remains as 0, if the item is not assigned to the container. In this study, the SCLP constraints except dimensional and orientation constraints are ignored. In this case, the SCLP solutions include only two parameters: assignment status and orientation type. The SCLP solutions are represented as loading sequences, in which the first row indicates the assignment status of the item and the second row indicates the item orientation type as in Fig. 3.

The container is loaded by assigned items that are saved in the loading list, where some of the assigned items may not be placed if there is no available space in the container. Therefore, the dimensions of assigned items are sorted in descending order and the container is loaded until there is no place for any items in

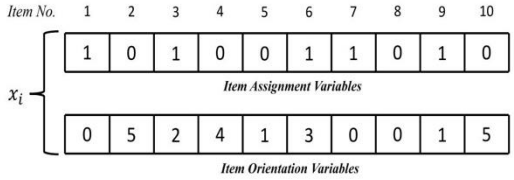


Fig. 3. Structure of container loading sequence.

the list. Item dimensions are sorted by width, height, and length to place items with larger width and height, in as deeper and bottom part of the container as possible [6].

3. Heuristic approaches for single container loading problems

3.1. Artificial bee colony algorithm

ABC algorithm is a neuro-inspired swarm intelligence-based metaheuristic approach that mimics the foraging behavior of the honeybees. ABC algorithm is developed by Karaboga [36] and easily applicable to many kinds of numerical and integer-based problems by providing search simplicity. The combination of the intensification aspect provided by neighborhood search and the diversification aspect provided by a random search mechanism creates an efficient ABC algorithm [37].

Each solution is represented by food sources in a search area and the performance of the solution is associated with the food source quality which is evaluated by honeybees. ABC algorithm classifies the honeybees into three groups: scout bees, employed bees, and onlooker bees. Scout bees explore the food sources randomly which equals the size of the bee swarm and informs the bees in the hive about the coordinates of the food sources. Randomly explored food sources represent the initial population, which is calculated with random values between the upper and the lower bound of the parameters that determine the random solution, as in Eq. (3).

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right) \tag{3}$$

In the formula, $i = 1 \ldots SN$ refers to the ith food source and SN refers to the total number of food sources in the search area. $j = 1 \ldots D$ refers to the jth dimension value of the ith food source between an upper and a lower bound and D refers to the total number of parameters of the ith food source to optimize. Initially, all $failure_i$ (the number of failures for improving ith food source) counters are set to zero.

After exploring the initial food sources, scout bees return to the hive and inform the employed bees about the coordinates of food sources. Each employed bee visits only one food source to evaluate the quality and improves it with a neighbor food source v_{ij} by using a greedy search. The neighbor food source is generated, as in Eq. (4).

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{4}$$

In the formula, v_{ij} is a neighbor food source of x_{ij} generated by changing randomly selected jth parameter of x_i. First, the difference between the jth parameter of food source x_i and the jth parameter of randomly selected food source $x_k (i \neq k \in 1, \ldots, SN)$ is calculated. Then, the difference is weighted by random value ϕ_{ij} which is between $[-1, 1]$ and added on to the current jth parameter value of x_i. If the jth parameter of v_i exceeds the upper or lower bounds, it is normalized, as in Eq. (5).

$$v_{ij} = \begin{cases} x_j^{min}, & v_{ij} < x_j^{min} \\ v_{ij}, & x_j^{min} \leq v_{ij} \leq x_j^{max} \\ x_j^{max}, & v_{ij} > x_j^{max} \end{cases} \tag{5}$$

The employed bees evaluate the newly explored neighbor food sources v_{ij} by calculating the fitness value, as in Eq. (6).

$$fitness_i = \begin{cases} 1/(1+f_i), & f_i \geq 0 \\ 1 + abs(f_i), & f_i < 0 \end{cases} \tag{6}$$

In the formula, $fitness_i$ is the quality of the food source x_i in the search area evaluated by the ith employed bee and f_i is a transition value for x_i obtained by the objective function $\left(f_i = F\left(\vec{x}_i\right)\right)$. If the quality of the neighbor food source v_i is better than x_i, v_i replaces x_i. The replaced solution is deleted from the employed bee's memory and the fail counter ($failure_i$) is reset. Otherwise, the $failure_i$ of non-improved solution is increased by one. In this study, the objective function is performed in maximization way. However, Eq. (6) is applied to solve minimization problems. For maximization problems, the container loading ratio value can be directly used as a fitness function and the solution with higher value is selected to improve the solution [38].

The employed bees return to hive and inform the onlooker bees about the coordinates and quality of the food sources to which they are assigned. The onlooker bees select one of the food sources randomly by evaluating the food quality which is associated with fitness value which represents the container loading outcome of the item placement solution. ABC algorithm uses the roulette wheel method to select a food source with a probability according to the fitness value. Each angle in the roulette wheel represents the probability of fitness value (p_i) of the food source that is calculated as in Eq. (7).

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{7}$$

A random value is generated between [0, 1] for each onlooker bee. If p_i is greater than the generated value, the onlooker bee visits the food source corresponding the angle in the roulette wheel. The onlooker bee generates a neighbor food source by using Eq. (4). If the onlooker bees improve the current food sources, they memorize the new food sources and forget the old one. Otherwise, the $failure_i$ of non-improved solution is increased by one. This process lasts until all onlooker bees are assigned to a food source.

If the maximum failure exceeds the failure limit L, the onlooker bee abandons the food source, except the food source with the best quality, and turns into the scout bee that explores new food sources randomly by using Eq. (3). In each of the iteration, only one onlooker bee is allowed to become a scout bee. The procedure of the basic ABC algorithm is given in Pseudocode 1.

3.2. Artificial bee colony algorithm for single container loading problem

The structure of the basic ABC algorithm described in Section 3.1 is suitable for numeric optimization problems. However, the ABC algorithm has been adapted for binary optimization problems in many types of research [17,19,24] and requires an adapted process of generating new solutions. In this study, the diversification process is performed by generating random solutions independently. However, the intensification process uses

Pseudocode 1: The basic ABC algorithm

```
Generate random SN food sources to form initial population by Equation (3) for maximization problem
Reset all failure counters (failure_i = 0)
Determine the maximum number of evaluations (Eval_max)
Calculate the transition values (f_i = F(x_i^*)) and fitness values (f_{v_i}) of food sources
  while Eval < Eval_max
    for i = 1 to SN do //Employed Bee Phase
      Generate a neighbor food source v_i for current food source x_i by Equation (4)
      Calculate the fitness of v_i by using v_i and f_{v_i} in Equation (6)
      if f_{v_i} > f_{x_i}
        Replace x_i by v_i
        failure_i = 0
      else
        failure_i = failure_i + 1
      end if
      if Eval ≥ Eval_max
        Memorize the best food source
        Terminate the search
      end if
    end for //End Employed Bee Phase
    Calculate probability values p_i for all food sources in roulette wheel by Equation (7)
    for t = 1 to SN do //Onlooker Bee Phase
      if random < p_i
        Generate a neighbor food source v_i for current food source x_i by Equation (4)
        Calculate the fitness of v_i by using v_i and f_{v_i} in Equation (6)
        if f_{v_i} > f_{x_i}
          Replace x_i by v_i
          failure_i = 0
        else
          failure_i = failure_i + 1
        end if
      end if
      if Eval ≥ Eval_max
        Memorize the best food source
        Terminate the search
      end if
      t = t + 1
    end for //End Onlooker Bee Phase
    Memorize the best food source
    if failure_{i_max} > L
      Replace x_i by a randomly generated food source x_j by Equation (3)
    end if
  end while
  Terminate the search
```

the xor operator proposed by Kiran and Gunduz [39], to generate new solutions as explained in Fig. 4.

Each element of the initial population is generated randomly without considering Equation (3) to get diverse solutions. In this study, solutions in the population are improved items by items to track the movement from a solution to another. In each attempt, only one item-related-element is replaced to obtain a new solution. The first loading sequence in Fig. 4 represents the current solution x_i and the second one represents another solution x_k in the population. The new solution x_j is generated by using the xor operator that adapted Eq. (4) as in Eq. (8), to interact x_i and x_k.

$$x_j = x_i \bigoplus \phi \left(x_i \bigoplus x_k \right) \tag{8}$$

The symbol $\bigoplus$ in Eq. (8) represents the xor operator that calculate the difference between the same elements in x_i and x_k. If the difference equals to zero, $x_i \bigoplus x_k$ value is set to zero, otherwise the value is set to one. The xor operator does not require extra processing to convert the item assignment state in the first column of the sequences. However, the conversion for item orientation state that is not suitable for binary conversion requires extra processing. First, the orientation element of x_i and x_k are converted to 3-digit binary code, and then generated digits are interacted by using xor operator as in Fig. 4. Then, the 3-digit binary code outcome of xor operation is converted back to the 1-digit number of orientation type.

The random value ϕ is between [0, 1] and determines whether the elements of $x_i \bigoplus x_k$ value are converted to the opposite ones. If the ϕ value is equal or greater than 0.5, the elements remain the same, otherwise they are converted to binary opposites such as zero to one or vice versa. If the ϕ is less than 0.5, the item assignment state can be converted easily to the opposite. However, the orientation type can be converted with respect to the SCLP constraints. The problem addressed in this study allows only the three categories of orientation type set, as described in Section 3.1. When three categories are considered together,

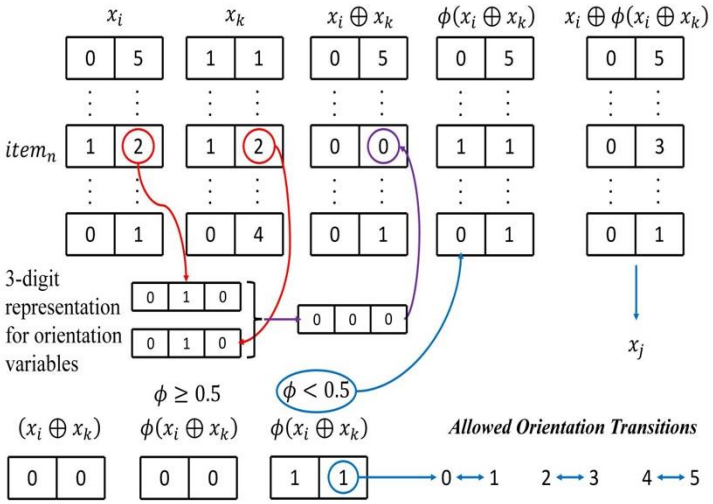


Fig. 4. XOR-based new solution generation process in ABC algorithm.

only three types of transition between orientation types can be allowed, as described in Fig. 4: (i) transition between orientation 0 and 1, (ii) transition between orientation 2 and 3, and (iii) transition between orientation 4 and 5. Finally, the current solution x_i interacts with $\phi\left(x_i \bigoplus x_k\right)$ outcome by using xor operator again to obtain the new solution x_j.

3.3. The proposed approaches

In this study, the first two hybrid algorithms based on memory mechanism and genetic operators have been developed to compare the effects of reinforcement approaches in use. The third one is a joint hybrid algorithm that combines the two separate reinforcement approaches with the basic ABC algorithm, explained as follows.

3.3.1. Memory integrated artificial bee colony algorithm

The honeybees in ABC algorithm forget all the information about the improvement process, once they abandon the food source that reaches the maximum failure limit in the population. However, the information about a succeeded or failed move from x_i to x_j can be used by other honeybees to accelerate the search process. In [27,28], the succeeded parameters are memorized to use for another search move. If the memorized parameters cannot succeed in the next use, it is deleted from the memory. On the other hand, Yin and Chuang [24] prefer to memorize the oscillation movements that are succeeded or failed to improve the solution. In this study, the search moves ($move_{ij}$) are represented as arrays in which the processed item with the related elements in x_i and x_j are saved, as shown in Fig. 5. In the $move_{ij}$ array, the first element indicates the index of item and the placement information about the item in x_i and x_j is saved respectively in the rest of the array.

The Pseudocode 2 indicates the local search phase of the MI-ABC algorithm. If the $move_{ij}$ cannot improve the current solution x_i, it is saved to the STTL to avoid using the same move in another attempt for a while, as shown in the Pseudocode 2 line 13.

The memorized moves are saved to the tabu lists temporarily and deleted after a while that is determined by the capacity of the lists. The move that is saved to the tabu list first, is deleted first, according to the first in first out (FIFO) rule. The capacities of the tabu lists are not fixed and depends on the problem dimension, which is the number of the items in the container loading sequence.

In the memory integrated ABC algorithm (MIABC), tabu lists (STTL, ITTL, and LTTL) guide the solution search by recalling the memorized moves during the process. If a generated move matches with one of the moves in STTL, the move is re-generated until a non-STTL one is generated, by doing so the STTL can restrict the local search process to find unexplored moves. In addition, the MIABC algorithm uses the memorized moves from ITTL and LTTL randomly. In this way, the MIABC algorithm decreases the convergence time for the optimum solution and can perform better than the basic ABC algorithm, with the intensification aspect.

In the MIABC algorithm, the honeybees may choose a random memorized $move_{ij}$ from tabu lists to move to a new food source or to generate a random move by using Eq. (8) without using any $move_{ij}$. In this case, there are three possible paths to select the $move_{ij}$: (i) ITTL path, (ii) LTTL path and (iii) random path. The path selection is determined by weight of the paths that is calculated as in Eq. (9).

$$iw_p = \frac{intscore_p}{\sum_{p=1}^{PI} intscore_p} \tag{9}$$

In the formula, iw_p denotes the intensification weight of the path and $intscore_p$ is the net succeeded number of attempts in the employed and onlooker bee phase that is performed by the selected one, where $p = 1, \ldots, PI$ refers to the number of possible path in exploitation process. Moayedikia et al. [40] used recruiter selection approach to guide the honeybees to search around the less visited food sources by manipulating the roulette wheel function. In MIABC algorithm, the path selection is weighted to guide honeybees to promising food sources, by the roulette wheel probability. If the $move_{ij}$ improves the solution, the score of the selected path is increased by one shown as in the Pseudocode 2 line 05, otherwise its score is reduced by one, as shown in the line 09.

Inline the 16, if any $intscore_p$ of available paths is less than 1; it is set to 1, in order to consider all available paths together to calculate the weight value. If a memorized $move_{ij}$ from ITTL or LTTL, cannot improve the solution, it is deleted from the related list, as in the line 11. In this way, the ineffective moves in the tabu lists are replaced by the possible promising moves that may intensify the solution search to improve the local optimum.

Instead listing the good solutions or moves discovered during the intensification process, LTTL lists the good solutions or moves

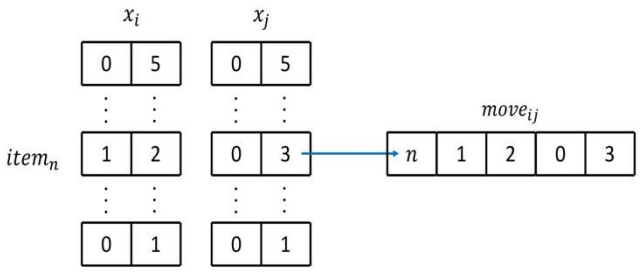


Fig. 5. Structure of the memorized move.

Pseudocode 2: Evaluation of the new food source in the employed and onlooker bee phase

```
01  Calculate the fitness of x_j by using Equation (1)
02  if f_{x_j} > f_{x_i}
03      Replace x_i by x_j
04      failure_i = 0
05      intscore_p = intscore_p + 1
06      Save the move_ij in ITTL
07  else
08      failure_i = failure_i + 1
09      intscore_p = intscore_p − 1
10      if move_ij ∈ ITTL or LTTL
11          Delete move_ij from the associated list
12      else
13          Save the move_ij in STTL
14      end if
15      if intscore_p < 1
16          intscore_p = 1
17      end if
18  end if
```

in the diversification process, guiding the search into the contrast areas of the search space for possible promising solutions [29]. The scout bee phase represents the diversification process in the basic ABC algorithm and LTTL is suitable for listing the moves that improves the current solution in this phase. In this study, if the move guides the abandoned solution to a fruitful area in the search space, it is saved in LTTL, considering that there is a possibility to guide other solutions to more fruitful areas. If the fitness value of the newly generated solution, that improves the abandoned one, is closer to the solution with the best fitness value than the average fitness value, the $move_{ij}$ is considered as a diversifying move and saved in LTTL. This type of move is rarely generated and saved into the LTTL.

In the scout bee phase, the new generated solution x_j replaces the abandoned solution x_i. If x_j meets the condition in Pseudocode 3 line 04, it is saved in LTTL as one of the most promising moves to use later.

3.3.2. Genetic operator based artificial bee colony algorithm

The main disadvantage of the basic ABC algorithm is that the solution search may not avoid premature convergence, which causes to obtain suboptimal solutions. ABC algorithm needs a diversification aspect that GA can provide, to escape from the search area that cannot converge to the global optimum.

GA algorithm diversifies the search randomly by using cross-over and mutation operators. In this section, four types of operator are used for the path selection in the employed bee and onlooker bee phase: (i) only cross-over operator, (ii) only mutation operator, (iii) cross-over and mutation operator together, and (iv) random search operator.

Cross-over operator in GA, generates two child solutions by taking the random elements of parent solutions x_i and x_k. However, we need the cross-over operator in ABC algorithm to generate one child solution for each x_i. Chaurasia et al. [26] proposed a genetic operator-based ABC algorithm, in which the multi-point insert method replaces the cross-over operator by generating one child solution from two parent solutions as shown in Fig. 6. In genetic operator-based ABC (GABC) algorithm, multi-point insert method is used as a path to generate the new solution.

The path that uses the mutation operator to swap only two item-elements of the x_i to generate the new solutions. The third path is to use multi-point insert method and mutation operator respectively to generate the new solution x_j. The reason of using genetic operators separately as options for the path selection is to increase the number of alternative paths for a more diversified solutions search. The last path is the random search same as in MIABC algorithm. The $intscore_p$ and iw_p of the paths are calculated as in MIABC to select the one of paths by roulette wheel probability.

In the GABC algorithm, the solution search in the re-scout bee phase is also weighted to enhance the diversification aspect. Singhal et al. [41] used genetic operators for an interaction between the best solution of the current iteration and the global best solution of the population. Besides, they proposed to replace

Pseudocode 3: Generating a new solution in Re-Scout Bee Phase

```
01  while move_ij ∈ STTL
02      Replace x_i by a randomly generated food source x_j by Equation (8)
03  end while
04  if |f_xbest − f_xj| < |f_xj − f_xmean| and f_xj > f_xi
05      Save the move_ij in LTTL
06  end if
```

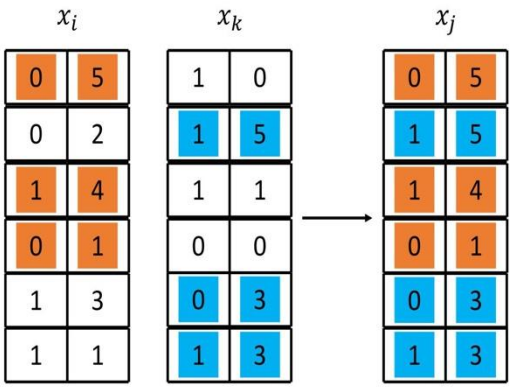


Fig. 6. Multi-point insert method.

the abandoned solution by the best solution ever obtained. Based on this idea, the abandoned solution interacts with one of the top best solutions in the search area called elite food sources *e*, using the multi-point insert method and the mutation operator, respectively. The abandoned solution can be replaced by a random solution instead of interacting with an elite one. In this way, there are two paths to replace the abandoned solutions: (i) random food source replacement *rfs* and (ii) elite food source a guided replacement *efs*, as shown in Pseudocode 4 line 02, and line 07.

If the fitness value of the generated solution that replaces the abandoned one, is closer to x_{best} than the x_{mean}, the diversification score of the path $divscore_p$ is increased by one, where $p = 1, \ldots, PD$ refers to the number of possible paths in diversification process, as shown in the line 05 and line 12. The $divscore_p$ determines the weight of the related path dw_p, as calculated in Eq. (10).

$$dw_p = \frac{divscore_p}{\sum_{p=1}^{PD} divscore_p} \tag{10}$$

The search path in the re-scout bee phase is selected by roulette wheel probability. The more diverse solutions the path can find, the more it is selected in the re-scout bee phase. The *divweight* values in the line 02 and line 07 indicate the total cumulative weight value of the path options. If the random value is less than the path's dw_p value, it is selected to generate the new solution x_j.

3.3.3. Joint hybrid artificial bee colony algorithm

In this study, we focus on the exploitation and exploration aspects of the ABC algorithm to develop a more robust algorithm. The MIABC algorithm provides a memory mechanism to find unexplored food sources in the local search area, where the GABC algorithm extends the search area to randomly find the new local search areas to avoid exhaustion of food sources. The Joint Hybrid Artificial Bee Colony (JHABC) algorithm combines the developments of intensification and diversification to benefit both aspects as shown in Pseudocode 5.

The JHABC algorithm uses a total of six different paths in the employed bee phase and onlooker bee phase which are two memory-based paths from the MIABC, three genetic operator-based paths from the GABC, and a path of random solution generation. All transformed rows of x_i are saved to ITTL or STTL as a group of $move_{ij}$'s, according to the fact that whether the path improves the solution. In this case, the memory quickly fills with moves provided by the genetic operators. If the selected path improves the solution x_i, $intscore_p$ is increased by one that affects the weight of the path iw_p, as calculated by Eq. (9). Otherwise, the $intscore_p$ is reduced by one until it is equal to one.

In the re-scout bee phase, if the selected path satisfies the condition in Pseudocode 5 line 29 and line 37, the $move_{ij}$s are saved to the LTTL and the $divscore_p$ of the path is increased by one, that affect the weight of the path dw_p, as calculated by Eq. (10).

The JHABC algorithm has been developed to solve SCLP based on binary optimization and uses evolutionary operators (crossover and mutation) individually or together with a path selection probability determined by the intensification or diversification score of the respective path. Ozturk et al. [17] generated new alternatives from a number of solutions for the solution x_i by using both crossover and mutation in a few steps and selected the best among the alternatives for a binary optimization problem. Luo [42] simplified the solution generation process proposed by Ozturk et al. and processed only the solution x_i and the neighborhood solution x_k by using one-point crossover and mutation to generate two child solutions for selection, while the JHABC algorithm only uses the multi-point insert method and mutation to generate a single solution from x_i and x_k. In this way, the JHABC algorithm provides an advantage by halving the number of evaluations in the neighborhood search, especially for an NP-hard problem such as SCLP.

Elghamrawy [43] generated new solutions for x_i in onlooker bee phase from the best solution of population *PS* and a random neighborhood solution x_k by using only the crossover operator, while Cao et al. [21] used mutation operator to obtain neighborhood solution x_k from the best solution of population with a probability, and then generated a new solution from x_i and x_k by using crossover operator. The JHABC algorithm also uses the interaction between the solution x_i, and the solution x_k selected from the elite bees in the re-scout bee phase, as a last opportunity for the abandoned solution x_i to escape the local optimum. If the solution x_i succeeds to move to a fruitful area of search space, the $move_{ij}$ is considered as one of most useful change in the search and memorized in LTTL.

The elements of solutions are improved one by one in the basic ABC algorithm for numerical optimization problems, whereas Chu et al. [44] proposed replacing multiple elements in each improvement. In addition, the x_i solution interacts with x_k selected from solutions with better fitness values, similar to what the JHABC algorithm has done in the re-scout phase by interacting the solution x_i with the x_k selected from the elite bees. Unlike

Pseudocode 4: Re-scout bee phase in GABC algorithm

```
Calculate weights of diversification operators dw_p by Equation (10)
if random < dw_{p_rfs} //Select Random Food Source
    Generate a food source xj randomly
    if |f_{x_best} − f_{x_j}| < |f_{x_j} − f_{x_mean}| and f_{x_j} > f_{x_i}
        divscore_p = divscore_p + 1
    end if
else if random < dw_{p_efs} //Interacting with an Elite Food Source
    Select a random food source x_k among the elite food sources
    Match the abandoned food source x_i and the elite food source x_k and use "multi-
    point insert" operator to generate transition food source x_z
    Generate new food source x_j by swapping two random genes of chromosome x_z
    if |f_{x_best} − f_{x_j}| < |f_{x_j} − f_{x_mean}| and f_{x_j} > f_{x_i}
        divscore_p = divscore_p + 1
    end if
end if
```

the JHABC algorithm, Chu et al. also improve the best solution in each iteration using the Cauchy mutation, a tool to escape the local optimum. Chen et al. [45] used a dynamically adjusted Boltzmann selection probability for solutions in the onlooker bee phase to give more chance to worst or worse ones in the population to avoid premature convergence at the local optimum. The JHABC algorithm can escape the local optimum by not only using genetic operators to diversify the population but also restricting the search with the memory mechanism. On the other hand, a weighted path selection approach is used to balance the use of the search strategies in both the employed bee and the onlooker bee phase. The successful strategies are selected with higher probability and the selection probability changes dynamically during the solution search.

Another way to avoid a premature convergence at the local optimum is to determine the move size that is not short enough to cause an extension of convergence time and not long enough to cause the useful parameter values to be skipped. Bansal and Kumar [46,47] used the Golden Section Search (GSS) method to determine the optimum ϕ_{ij} value in Eq. (4) as the move size. Kumar additionally used a dynamic selection probability p_{ij} for the j-th element of solution x_i in the onlooker bee phase where p_{ij} depends not only on the fitness value but also on the move size ϕ_{ij}. Jia et al. [48] also used the fitness value to determine the new solution v_{ij} in Eq. (4); here the factor λ changes with the fitness values and increase the weight of the current solutions to generate a new one as the population evolves, thus intensifying the search in the local search area. On the other hand, they diversified the search using a modification rate (MR) as the control parameter. Kiran and Gunduz [39] determined $move_{ij}$ for the binary optimization problem according to the dissimilarity between the solution x_i and the neighborhood solution x_k, and the number of 0–1 swaps between x_i and x_k was optimized with a random v value and the dissimilarity value, to generate the new solution x_j. Hancer et al. [49] similarly used the dissimilarity value to determine the solution x_j, however they also optimized the v value depends on the current iteration and the maximum number of iterations. In the JHABC algorithm, the length of $move_{ij}$ depends on the selected solution search path. If random search or mutation is selected as the search strategy, only one element of the solutions is changed. If the crossover operator used alone or as a part of genetic operator use strategy, multiple elements of the solutions are changed and saved in a tabu list for later use as a memory block, because the source of the change in fitness value cannot be detected in case the multiple elements of solutions are changed.

4. Computational results and discussion

MIABC, GABC, and JHABC algorithms that are developed for SCLP have been coded in MATLAB R2016 version software. All experimental runs are performed by the CPU that has 4 GB RAM and 3.10 GHz processors using Windows 7 operating system. Proposed algorithms are tested on commonly used BR single container problem dataset [50] which is retrieved from open source and involves 15 problems categorized by the number of item types (from 3 to 100) to be assigned, and each problem set consists of 100 sub-problems. The first 7 problems (BR01–BR07) of the BR problem set contain weakly heterogeneously distributed items, while the second part (BR08–BR15) contains strongly heterogeneously distributed ones. The objective of the problem is to maximize the container utilization ratio (CUR). The basic ABC algorithm and proposed algorithms are compared with following approaches published in literature:

- GA_GB: a genetic algorithm by Gehring and Bortfeldt (1997), Pentium/l30 MHz PC [1]
- TS_BG: a tabu search strategy by Bortfeldt and Gehring (1997), Pentium/200 MHz PC [10]
- CBGAS: a hybrid genetic algorithm by Bortfeldt and Gehring (2001), N/A [2]
- HCLPMC: a tree search method by Sheng et al. Intel Core i7 870 @2.93 GHz [7]
- GRASP: a GRASP approach by Parreno et al. (2008), Pentium IV 1.7 GHz [8]
- HTS: a hybrid tabu search strategy by Liu et al. (2011), Intel Centrino Duo CPU 1.66 GHZ and 1 GB RAM [11]
- hybrid-BA: a hybrid bee(s) algorithm by Dereli and Das (2011), N/A [14]
- SOA: a swarm optimization algorithm by Zhou and Liu (2017), N/A [15]

All compared studies testing their heuristic approach on the BR problem dataset, consider only the orientation and stability constraint, used mostly in the literature, to observe the key outcomes of the proposed approach. In this study, the same constraints have been applied to the container loading process to

Pseudocode 5: The JHABC algorithm

```
Generate random SN food sources to form initial population by Equation (3) for maximization problem
∀ failure_i = 0  //Initialization Phase
∀ intscore_p = 1 and divscore_p = 1
Determine the maximum number of evaluation ( Eval_max )
Calculate the fitness values of food sources fitness (x_i)
    while Eval < Eval_max
        for i = 1 to SN do  //Employed Bee Phase
            Path Selection;
            Pseudocode 2;
        end for  //End Employed Bee Phase
        for t = 1 to SN do  //Onlooker Bee Phase
            if random < p_i
                Path Selection;
                Pseudocode 2;
            end if
            if Eval ≥ Eval_max
                Memorize the best food source
                Terminate the search
            end if
            t = t + 1
        end for  //End Onlooker Bee Phase
        Memorize the best food source
        if failure_{i_max} > L  //Re-Scout Bee Phase
            Calculate weights of diversification operators dw_p by Equation (10)
            if random < dw_{p_rfs}  //Select Random Food Source
                while move_ij ∈ STTL
                    Replace x_i by a randomly generated food source x_j by Equation (8)
                end while
                if |f_{x_best} − f_{x_j}| < |f_{x_j} − f_{x_mean}| and f_{x_j} > f_{x_i}
                    divscore_p = divscore_p + 1
                    Save the move_ij in LTTL
                end if
            else if random < dw_{p_efs}  //Interacting with an Elite Food Source
                while move_ij ∈ STTL
                    Match the x_i with the elite solution x_k to generate x_j by using the path(iii) in GABC
                end while
                if |f_{x_best} − f_{x_j}| < |f_{x_j} − f_{x_mean}| and f_{x_j} > f_{x_i}
                    divscore_p = divscore_p + 1
                    Save the move_ij in LTTL
                end if
            end if
        end if
    end while
    Terminate the search
```

clearly observe the effects of the ABC algorithm and the proposed approaches.

None of the compared studies indicated the complexity of SCLP implementation, so we are not able to compare the computational complexity. However, the computational complexity of SCLP is $O(n^3)$ as stated in [51], and $O(2n^3)$ for the JHABC algorithm, where n denotes the number of items to be placed into the container. Among other compared studies, only Liu et al. [11] emphasized that the computational time increases with the expanding solution space, so they encoded the loading sequences with the quantity of item types rather than the total number of items. The implementation of proposed and comparative heuristic approaches requires an excessive amount of runtime, and some of compared studies have run the heuristic approaches for 3 times [14], 10 times [8] and 20 times [7] due to the excessive computational times, however the others have not specified the

number of repeats. In this study, each algorithm was executed for 20 times due to the excessive run time that is related to the number of iterations and the problem heterogeneity. [11]

4.1. Parameter settings

The basic ABC algorithm is a swarm intelligence-based heuristic approach and involves generating random solutions represented by a set of parametric values, to converge to global optima in a reasonable time. Parameter variables of ABC algorithm, the size of population SN, total number of evaluations $Eval$ or iterations Itr, and the failure limit of each bee L determine framework of the solution search process. In addition, the number of elite bees e determines how many problem solutions are capable of attracting others in the population to reduce the duration of convergence of genetic operator-based algorithms GABC and JHABC. The size of the STTL, LTTL and ITTL determines the number of search moves saved into the list that guides the search using useful moves and avoiding prohibited ones. It should be noted that the parameter values selected in comparative studies are set for different types of algorithms and each provides a different approach for solving CLP. However, swarm based comparative algorithms can provide key ideas for parameter configuration.

Gehring and Bortfeldt emphasize that the size of the population smaller than 50 causes the poorer outcomes and they determine SN as 50 [14]. Bortfeldt and Gehring also set the size of population as 50 [15]. Dereli and Das apply a factorial analysis to select optimal parameter values and they set the size of population to 20 instead of 10 to increase the solution quality. However, as the number of selected sites increase, the solution quality decreases, and they determine the optimal number of selected sites and elite bees to 4 [17–23]. Zhou and Liu proposed a different type of evaluation process that updates the position of individuals in each iteration as a result of interactions between the individual and a varying number of other individuals randomly selected. Therefore, the number of evaluations does not directly depend on the size of population. In this context, they determine the size of population as 10 [1,2]. On the other hand, the population size has been set between 20 and 100 in various ABC algorithm studies [8] and has been tested on different types of problems instead of SCLP. In this study, the size of population is determined as 50 and the failure limit is set to half of the population 25 for each bee.

The number of objective function evaluations varies for comparison algorithms as seen in Table 1, due to different search approaches. However, the proposed JHABC algorithm can be compared to comparison algorithms, at least in terms of convergence time, if provided. The comparison algorithms limit the search to a period of time or a number of iterations. Gehring and Bortfeldt determine the search limit as 500 iterations or 500 s both in [14]. They also limit the search to 120 iteration or 500 s for TS_BG. Parreno et al. propose a parametric design for the number of iterations and they run the GRASP approach for 5 times and 10 times respectively with a limit of 5000 iteration. Finally, they run the approach for once with 50 000 iteration and as a result the number of trials does not statistically change the average CUR. However, the increase in the number of iterations improves the CUR value [15]. Dereli and Das also denoted that the increase in the number of iterations improves the solution quality, and they set the parameter value as 1000 [27]. Zhou and Liu determined the number of iterations as 20. After several trials, the number of iterations is set to at least $5n^2$ for each problem set. However, to compare the parametric details of the proposed algorithms with comparative studies, assuming that an average of 100 evaluations equals to 1 iteration, the total number of evaluations is converted to the number of iterations Itr, which is equal to $n^2/20$. Besides,

Table 1
Parametric details of the algorithmic configuration.

Algorithm	SN	$Eval$	Itr	e	m	L
GA_GB	50	–	500	–	–	–
TS_BG	–	–	120	–	–	–
CBGAS	50	–	500	–	–	–
GRASP	–	–	50 000	–	–	–
Hybrid-BA	20	–	1000	4	–	–
SOA	10	–	20	–	–	–
ABC	50	$5n^2$	$n^2/20$	–	–	25
MIABC	50	$5n^2$	$n^2/20$	–	$n \times SN$	25
GABC	50	$5n^2$	$n^2/20$	10	–	25
JHABC	50	$5n^2$	$n^2/20$	10	$n \times SN$	25

the developed algorithms do not improve the loading solutions in general after 400 iteration for the first part of problem set (BR01–BR07) and 700 iteration for the second part (BR08–BR15).

The parameter values for the algorithms tested are fixed in many types of research. However, the size of the problem can affect the speed and duration of convergence when searching for the optimum solution. Li and Yang [27] set the number of iterations and the size of the memory by the size of the problem. In this study, parameter values are set by considering the number of items to be assigned to a single container. The size of the memory m is set to $n \times SN$, where n is the number of items to be assigned and SN is the size of population. Parameter design details of the algorithms are given in Table 1, where the parameters of the basic ABC algorithm, the proposed three algorithms and the compared heuristic approaches are tabulated, and the parameter values of the HCLPMC and HTS approaches are not specified.

In the re-scout bee phase, the ratio of elite bees is determined by three-level (10%, 20% and 30%) trials on BR00 problem set, as in Fig. 7. The horizontal values show the iteration number, and the vertical values show the CUR. None of ratio-level trials is significantly different ($p > .05$) from others. However, 20% ratio-level obtains the highest average CUR with 85.97% and distinguishes from other ratio levels as shown in the diversity plot, so that the ratio of elite bees is set to 20%.

4.2. Computational results

The basic ABC algorithm involves generating the first population from one individual and the same initial populations are randomly generated for sub-problems of each BR problem class and the average optimum CUR, standard deviation and runtime (seconds) of the related class presented in Table 2. It should be noted that, the CUR values are obtained after the number of iterations calculated as mentioned in the parameter settings section. The first part of the problem set (BR01–BR07) converges to the near optimum CUR at an average of 400 iterations, whereas the second part of the problem set requires an average of 700 iterations (BR08–BR15). The graphs in Figs. 8 and 9 show the average CUR values obtained by the algorithms up to the number of iterations mentioned above to reveal the difference between the performance of algorithms. The basic ABC and the proposed algorithms are statistically analyzed by using one-way ANOVA and Fisher's Least Significant Difference (LSD) post hoc test [52], and it is determined that the distribution of executed algorithms in both Figs. 8 and 9 are significantly different ($p < .05$). The diversity of algorithms is indicated by a box plot, where each algorithm is represented by a different colored box that provides information about the CUR value distribution throughout the iterations.

Fig. 8. shows the loading ratio curve of the algorithms that have different solution search characteristics. The basic ABC algorithm represented by the blue box in diversity plots, has the

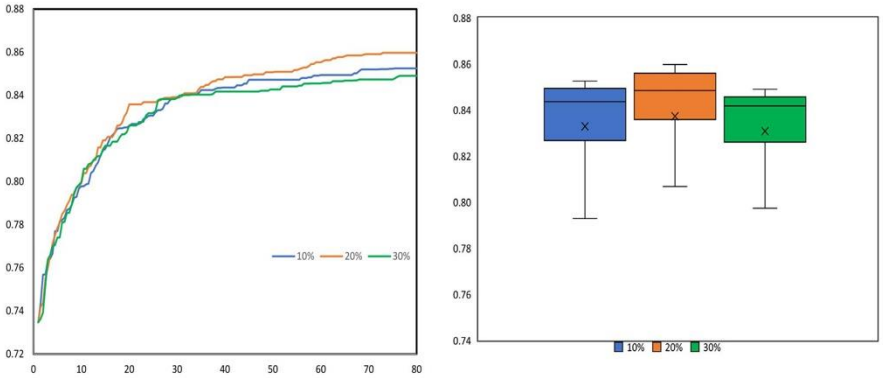


Fig. 7. The ratio levels for elite bees with diversity plot and the effects on the CUR.

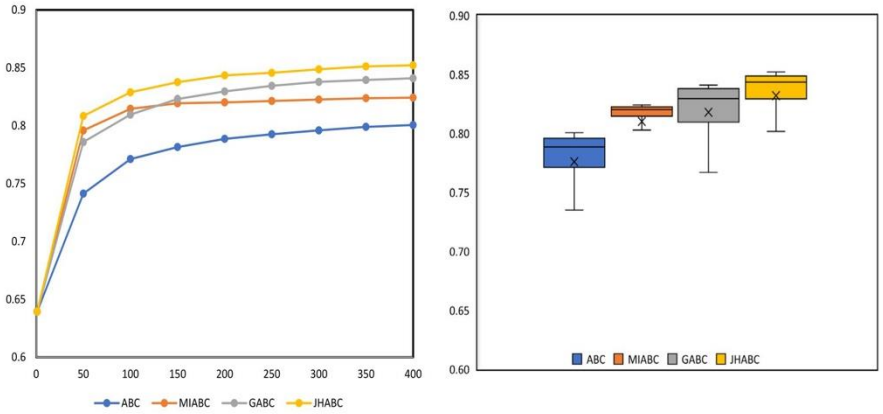


Fig. 8. BR01–BR07 average CUR values and diversity plot over search history.

significant poorest performance among others, in both Figs. 8 and 9. The MIABC algorithm represented by the orange box, and the GABC algorithm represented by the gray box, can be considered as head-on competitors until iteration 150 for the first part of the problem set as in Fig. 8 and iteration 200 for the second part as in Fig. 9 amongst themselves. At the first stage of the solution search, the GABC algorithm obtains higher CUR due to its powerful diversification aspect that spreads the honeybees over a wider search area. However, the MIABC algorithm overcomes the GABC algorithm for a short period, due to its intensification aspect that exploits the food sources rapidly in the local search area better. Then, the GABC algorithm overcomes the MIABC algorithm again and the MIABC algorithm converges prematurely by exhausting the food sources in the local search area. In this case, the MIABC algorithm can be considered as a better solution approach in the short term, whereas the GABC algorithm shows better performance in the long term. Although the curves of the MIABC and GABC algorithms are close to each other, they are significantly different ($p < .05$) for both part of the problem set. This indicates that the genetic operators emerge as a more effective reinforcement approach for the ABC algorithm in SCLP implementation.

The JHABC algorithm represented by the yellow box in Figs. 8 and 9 uses both diversification and intensification for global and local search to obtain better CUR value in a short time. The JHABC algorithm is the best one from the beginning of the search to the end, and significantly different from others ($p < .05$). However, the optimum CUR values obtained by GABC algorithm converges to the optimum values obtained by the JHABC algorithm in the long term and the GABC algorithm obtains better results than the JHABC algorithm for BR05 and BR09. This means that the effect of the memory mechanism on the JHABC algorithm is limited and reduced over time. However, the JHABC algorithm is significantly better ($p < .05$) than all other algorithms for most problem sets according to the LSD post hoc test applied to the algorithm pairs. This shows that the memory mechanism slightly contributes the results obtained by the JHABC algorithm when it works with genetic operators.

The runtime of the experiments is another indicator for observing the cost-effectiveness of the related algorithm. The runtime of the tests, which depend on the number of items to be assigned for the container and are not affected by the problem heterogeneity, obtains the highest and lowest value from the BR09 and BR02 problem sets respectively, for all algorithms. The MIABC algorithm requires the highest runtime due to the working principle of the memory mechanism. STTL is the most checked tabu list to avoid using prohibited moves after a new move is selected, and each non-profitable move makes the list longer. STTL is naturally the longest list among others as it is more likely to generate non-profitable moves in each neighborhood search process, making the control process longer over time. ITTL uses half of the designated memory size on average as the neighborhood search has a moderate probability of generating profitable moves. The newly generated moves are unlikely to be close to the best solution of population, so they are rarely saved into LTTL. All mentioned tabu lists are controlled before generating new moves, so the checking process can be excessively time consuming.

The memory mechanism cannot prevent the basic ABC algorithm from converging to the optimum solution prematurely.

Table 2
CUR values of the basic ABC algorithm and proposed algorithms.

Problem class	ABC			MIABC			GABC			JHABC		
	CUR	SD	Runtime(s)	CUR	SD	Runtime(s)	CUR	SD	Runtime(s)	CUR	SD	Runtime(s)
BR01 (3)	0.8012	0.0015	91.03	0.7882	0.0011	122.67	0.8219	0.0012	117.45	**0.8442**	0.0010	138.33
BR02 (5)	0.7811	0.0011	48.76	0.7980	0.0008	61.12	0.8139	0.0007	62.61	**0.8330**	0.0012	73.55
BR03 (8)	0.8059	0.0018	82.81	0.8424	0.0012	111.34	0.8545	0.0014	81.63	**0.8684**	0.0010	97.24
BR04 (10)	0.8239	0.0014	104.56	0.8344	0.0017	163.76	0.8398	0.0017	122.89	**0.8526**	0.0017	138.67
BR05 (12)	0.7977	0.0022	83.61	0.8218	0.0013	115.11	**0.8420**	0.0009	107.71	0.8417	0.0009	125.41
BR06 (15)	0.8248	0.0020	215.54	0.8583	0.0010	298.45	0.8784	0.0013	282.43	**0.8933**	0.0011	286.81
BR07 (20)	0.8013	0.0010	131.78	0.8354	0.0009	187.79	0.8570	0.0010	144.91	**0.8587**	0.0014	175.14
BR08 (30)	0.8241	0.0012	312.12	0.8500	0.0012	442.23	0.8731	0.0021	342.11	**0.8766**	0.0022	405.05
BR09 (40)	0.8198	0.0017	374.15	0.8525	0.0008	547.65	**0.8697**	0.0016	405.10	0.8693	0.0015	500.25
BR10 (50)	0.8199	0.0015	311.10	0.8467	0.0014	441.12	0.8670	0.0011	341.32	**0.8729**	0.0023	426.69
BR11 (60)	0.8042	0.0009	252.44	0.8246	0.0017	356.89	0.8512	0.0010	273.15	**0.8608**	0.0010	347.29
BR12 (70)	0.8108	0.0015	316.78	0.8347	0.0021	450.12	0.8461	0.0014	325.25	**0.8626**	0.0012	441.28
BR13 (80)	0.8014	0.0010	248.11	0.8248	0.0012	378.54	0.8461	0.0008	270.33	**0.8541**	0.0010	335.61
BR14 (90)	0.8089	0.0017	191.92	0.8300	0.0014	272.14	0.8517	0.0020	210.76	**0.8557**	0.0012	260.52
BR15 (100)	0.7995	0.0014	197.32	0.8266	0.0018	294.78	0.8543	0.0015	232.94	**0.8569**	0.0016	278.10
Mean BR01-07	0.8051	–	108.30	0.8255	–	151.46	0.8439	–	131.38	0.8560	–	147.88
Mean BR08-15	0.8111	–	275.49	0.8362	–	397.93	0.8574	–	300.12	0.8636	–	374.35
Mean	0.8083	–	197.47	0.8312	–	282.91	0.8511	–	221.37	0.8601	–	268.66

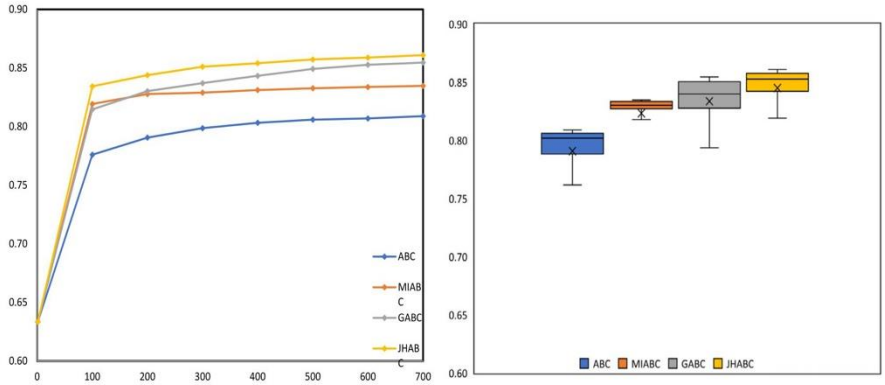


Fig. 9. BR08–BR15 average CUR values and diversity plot over search history.

As a result, the memory mechanism can be considered as an inadequate reinforcement approach for the basic ABC algorithm to solve SCLP, compared to the genetic operators, in terms of the solution quality and the required processing time. On the other hand, the memory mechanism loses its effect on the JHABC algorithm in long term. Nevertheless, it contributes to the JHABC algorithm so that it can obtain significant solutions compared to the GABC algorithm.

4.3. Comparison with other approaches

As a consequence of the results obtained in the previous section, the effects of reinforcement approaches on the basic ABC algorithm in solving SCLP, have been observed to decide which one is superior to the other one. In this section, the JHABC algorithm, which obtains significant CUR values compared to other proposed approaches, is compared with the basic and recent heuristic approaches applied on SCLP to see its overall performance. The CUR values obtained in the compared studies are shown in Table 3 without standard deviation and runtime. Most of them does not show the test runtime, but only mention a specific runtime as the limit, in the study. The compared studies except HCLPMC consider only C1 and C2 constraint to observe the performance of their approaches. HCLPMC algorithm restricts the solution search by all five basic constraints. The bold CUR values in Table 3 show the best for the related problem class. In the previous section, the basic ABC and the proposed algorithms were compared using first the one-way ANOVA and then the LSD test. However, the JHABC algorithm is compared to the algorithms from literature by using two-independent Mann–Whitney U non-parametric statistical test due to the small number of used problem-sets [53].

Approaches proposed in comparative studies, except hybrid-BA due to the lack of results for the second part of the problem set, obtain CUR values with an increasing trend for the first five problem sets in general, and then the CUR values decrease as the problem heterogeneity increases, as shown in Fig. 10. However, this trend works differently for the JHABC algorithm which obtains irregular CUR values for the first part of the problem, and then the CUR values stabilize in the second part. The compared studies provide better CUR values in the first part of the problems sets because the low problem heterogeneity enables the them to easily build item stacks, blocks and layers, while the JHABC algorithm misleads the solution search by assigning items and orientations randomly. The increasing problem heterogeneity enables the JHABC algorithm to demonstrate its search capability in the SCLP solution space, where the solution options are enormously increased, thus closing the CUR value gap and performing an average performance compared to other approaches.

The GA_GB and TS_BG approaches provide basic insights into how genetic operators and tabu lists guide the solution search for solving SCLP, and the idea is implemented in this study to enhance the basic ABC algorithm. When the GA_GB and TS_BG approaches are compared, the TS_BG is more effective than the

Table 3
CUR values of the JHABC algorithm and comparative algorithms.

Problem class	GA_GB	TS_BG	CBGAS	HCLPMC	GRASP	HTS	Hybrid-BA	SOA	JHABC
BR01 (3)	0.8677	0.9263	0.8781	0.8759	**0.9327**	0.8814	0.8341	0.9267	0.8442
BR02 (5)	0.8812	0.9270	0.8940	0.8773	**0.9338**	0.8952	0.8460	0.9319	0.8330
BR03 (8)	0.8887	0.9231	0.9048	0.8836	0.9339	0.9053	0.8542	**0.9344**	0.8684
BR04 (10)	0.8868	0.9162	0.9063	0.8820	0.9316	0.9075	0.8519	**0.9321**	0.8526
BR05 (12)	0.8878	0.9086	0.9073	0.8817	0.9289	0.9079	0.8511	**0.9294**	0.8417
BR06 (15)	0.8853	0.9004	0.9072	0.8778	0.9262	0.9074	0.8469	**0.9270**	0.8933
BR07 (20)	0.8836	0.8863	0.9065	0.8744	0.9186	0.9007	0.8399	**0.9231**	0.8587
BR08 (30)	0.8752	0.8711	0.8973	0.8655	0.9102	0.8889	–	**0.9192**	0.8766
BR09 (40)	0.8646	0.8576	0.8906	0.8593	0.9046	0.8851	–	**0.9150**	0.8693
BR10 (50)	0.8553	0.8473	0.8840	0.8510	0.8987	0.8776	–	**0.9123**	0.8729
BR11 (60)	0.8482	0.8355	0.8753	0.8474	0.8936	0.8706	–	**0.9085**	0.8608
BR12 (70)	0.8425	0.8279	0.8694	0.8418	0.8903	0.8697	–	**0.9059**	0.8626
BR13 (80)	0.8367	0.8229	0.8625	0.8388	0.8856	0.8690	–	**0.9017**	0.8541
BR14 (90)	0.8299	0.8133	0.8555	0.8347	0.8846	0.8640	–	**0.8970**	0.8557
BR15 (100)	0.8247	0.8085	0.8523	0.8328	0.8836	0.8623	–	**0.8920**	0.8569
Mean BR01-07	0.8830	0.9126	0.9006	0.8790	**0.9294**	0.9008	0.8463	0.9292	0.8560
Mean BR08-15	0.8471	0.8355	0.8734	0.8464	0.8939	0.8734	–	**0.9065**	0.8636
Mean	0.8639	0.8715	0.8861	0.8616	0.9105	0.8862	–	**0.9171**	0.8601

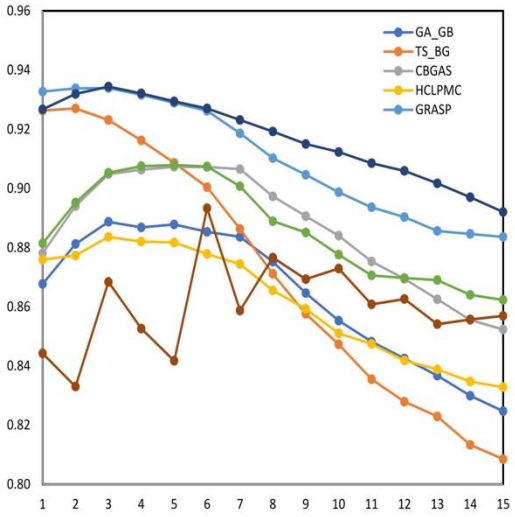


Fig. 10. CUR values for the 15 problems sets.

GA for the first part of the problem set (BR01–BR07). However, as the problem heterogeneity increases, GA starts to show its effect and obtains better than the results obtained by TS_BG. On the other hand, the GA_GB obtains the CUR values in less than 360 s on average whereas TS_BG limits the search to 500 s, so that the GA_GB can be considered a superior approach for the second part, compared to TS_BG. The JHABC algorithm overcomes the GA_GB and TS_BG by using the two approaches together as reinforcement for the basic ABC algorithm with statistically significant ($p < .05$) CUR values in the second part, and an average test runtime of 269 s. Even if the search for the JHABC algorithm is limited to an average of 500 iterations, it still obtains a better average CUR value in the second part with 0.8572. CBGAS is an advanced form of GA_GB that adapts the genetic operators to use generated item layers more effectively and obtains better results than the JHABC algorithm on average except BR14 and BR15 problem sets. The JHABC algorithm randomly searches for the loading solutions instead using a dynamic exchange of items between the solutions.

HCLPMC may obtain better CUR values than the JHABC algorithm if it only takes C1 and C2 constraints into account, but still obtains better results than GA_GB and TS_BG simply by improving the container loading heuristic. GRASP and HTS focus on the way of building suitable item piles to maximize the filling rate of the generated residual spaces in the container and also obtain better results than the JHABC algorithm.

Hybrid-BA, based on layer building, and SOA, based on block building, are swarm based heuristic approaches with different search capabilities for SCLP. The hybrid-BA CUR values for the first part of the problem sets show that the approach can be considered as a promising algorithm for solving SCLP, but still requires an improvement for the solution search strategy. CUR values of BR01–BR07 obtained by the JHABC algorithm based on the ABC algorithm similar to the hybrid-BA with the search strategy, are not sufficient to compare it and the hybrid-BA. However, the JHABC algorithm obtains better results on average compared to the hybrid-BA. On the other hand, SOA is a newly developed heuristic approach and provides significantly robust solutions for almost any problem set, with solution search strategy and the complex container loading heuristic that enables different types of items to build gigantic blocks together. In this way, the impact of problem heterogeneity on SOA is limited as shown in Fig. 10.

As a result, the JHABC algorithm has not been able to obtain the best CUR values in any problem set and can be considered as ineffective approach for solving SCLP compared to SOA, which is superior to other algorithms in most problem sets. The main reason why the JHABC algorithm fails to improve the best results in any problem set is that the proposed approach works with a random search mechanism, regardless of any item classification. Therefore, it became difficult for the JHABC algorithm to explore the correct pairs of item assignment and orientation variable for each element of the container loading sequence, which changes rapidly during the random search. However, the JHABC algorithm provides promising solutions using complete random search at a reasonable runtime compared to GA_GB, TS_BG and HCLPMC approaches used for solving SCLP. Besides, the approaches with better results than the JHABC algorithm reveal their differences with the container loading strategies which enable them to rapidly improve the loading solutions. In this context, the next focus for the JHABC algorithm should be on developing an efficient container loading heuristic to compare it equally with other approaches.

5. Conclusion

The ABC algorithm is commonly used for numeric optimization problems. However, a binary optimization-based ABC algorithm has recently been applied to the container loading problem,

which is a kind of knapsack problem and one of the main problems of transport systems. In most ABC algorithms applied knapsack problem studies, a number of dimensions are considered as the number of parameters for each item, while in this study three-dimensional items are regarded as orthogonal objects to be placed in a three-dimensional knapsack called as a container.

The ABC algorithm is a powerful and efficient algorithm for numerical optimization with a combination of intensification and diversification aspects. However, enhanced ABC algorithms are required to solve complex problems such as container loading problems. This study focuses on enhancing the search mechanisms that are used in local search and global search to develop a robust ABC algorithm for container loading problems. A memory mechanism is used to avoid repetitive item placement solutions and to benefit from the fruitful ones in local search, while the genetic operators are integrated into the basic ABC algorithm to expand the global search area in order to discover potential solutions. Reinforcement approaches that improve different aspects of the basic ABC algorithm are analyzed separately to understand their effects on the algorithm.

As a result, this study proposes a memory-integrated ABC algorithm to meticulously select useful search steps in local search, and a genetic operator-based ABC algorithm to intelligently generate the next search steps in global search inspired by efficient solutions. Moreover, a joint hybrid ABC algorithm that uses both reinforcements approaches together is developed to provide effective solutions for widely examined single container loading problem sets, regardless of many constraints. The results show that using the memory mechanism is more effective in the short run. However, the genetic operator-based ABC algorithm provides better solutions in the long run and it is more applicable to high heterogeneous single container loading problems.

The JHABC algorithm expands the SCLP search space by working with completely random solutions that assign each item in all possible orientations regardless of residual space and item matching, for the container. The complete random search behavior helps the JHABC algorithm to overcome the approaches using basic tabu search and genetic algorithm for highly heterogeneous SCLP problem sets. However, it can only perform moderately when compared to container loading focused heuristic approaches but still can be considered as a promising approach for solving highly heterogeneous SCLPs.

In future studies, the MIABC-GABC comparison can be done in different branches of knapsack problems and numerical optimization problems, where the proposed JHABC algorithm is also applicable for a variety of knapsack problems. Container loading problem-related studies such as multiple container loading problems and capacitated vehicle routing problems can use the proposed JHABC algorithm. Besides, the container usage ratio level obtained by the JHABC algorithm can be improved by developing different item allocation strategies for the single container loading problem.

CRediT authorship contribution statement

Tuğrul Bayraktar: Development of the all proposed approaches, Drafted the manuscript. **Filiz Ersöz:** Conceived of the study, Participated in its coordination and helped to draft the manuscript as co-supervisors. **Cemalettin Kubat:** Conceived of the study, Participated in its coordination and helped to draft the manuscript as co-supervisors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors read and approved the final manuscript.

References

[1] H. Gehring, A. Bortfeldt, A genetic algorithm for solving the container loading problem, Int. Trans. Oper. Res. 4 (1997) 401–418, http://dx.doi.org/10.1111/j.1475-3995.1997.tb00095.x.
[2] A. Bortfeldt, H. Gehring, A hybrid genetic algorithm for the container loading problem, European J. Oper. Res. 131 (2001) 143–161, http://dx.doi.org/10.1016/S0377-2217(00)00055-2.
[3] L.J. Pires de Araujo, P. Pinheiro, Combining heuristics backtracking and genetic algorithm to solve the container loading problem with weight distribution, 73, 2010, pp. 95–102.
[4] J. Mau Yeh, Y. Chen Lin, S. Yi, Applying genetic algorithms and neural networks to the container loading problem, J. Inf. Optim. Sci. 24 (2003) 423–443.
[5] Z. Wang, K.W. Li, J.K. Levy, A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach, European J. Oper. Res. 191 (2008) 84–97, http://dx.doi.org/10.1016/j.ejor.2007.08.017.
[6] J. Ren, Y. Tian, T. Sawaragi, A tree search method for the container loading problem with shipment priority, European J. Oper. Res. 214 (2011) 526–535, http://dx.doi.org/10.1016/j.ejor.2011.04.025.
[7] L. Sheng, S. Xiuqin, C. Changjian, Z. Hongxia, S. Dayong, W. Feiyue, Heuristic algorithm for the container loading problem with multiple constraints, Comput. Ind. Eng. 108 (2017) 149–164.
[8] F. Parreno, R. Alvarez-Valdes, J.M. Tamarit, J.F. Oliveira, A maximal-space algorithm for the container loading problem, Inform. J. Comput. 20 (2008) 412–422, http://dx.doi.org/10.1287/ijoc.1070.0254.
[9] M.T. Alonso, R.A. Valdes, J.M. Tamarit, F. Parreño, A reactive GRASP algorithm for the container loading problem with load-bearing constraints, Eur. J. Ind. Eng. 8 (2014) 669, http://dx.doi.org/10.1504/EJIE.2014.065732.
[10] A. Bortfeldt, H. Gehring, Applying tabu search to container loading problems, in: Operations Research Proceedings, Vol. 1998, Springer, 1997, pp. 533–538.
[11] J. Liu, Y. Yue, Z. Dong, C. Maple, M. Keech, A novel hybrid tabu search approach to container loading, Comput. Oper. Res. 38 (2011) 797–807, http://dx.doi.org/10.1016/j.cor.2010.09.002.
[12] D. Mack, A. Bortfeldt, H. Gehring, A parallel hybrid local search algorithm for the container loading problem, Int. Trans. Oper. Res. 11 (2004) 511–533, http://dx.doi.org/10.1111/j.1475-3995.2004.00474.x.
[13] W. Zhu, H. Qin, A. Lim, L. Wang, A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3l-CVRP, Comput. Oper. Res. 39 (2012) 2178–2195, http://dx.doi.org/10.1016/j.cor.2011.11.001.
[14] T. Dereli, G.S. Das, A hybrid 'bee(s) algorithm' for solving container loading problems, Appl. Soft Comput. 11 (2011) 2854–2862, http://dx.doi.org/10.1016/j.asoc.2010.11.017.
[15] Q. Zhou, X. Liu, A swarm optimization algorithm for practical container loading problem, in: IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2017, pp. 5690–5695.
[16] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: theory and application, Adv. Eng. Softw. 105 (2017) 30–47.
[17] C. Ozturk, E. Hancer, D. Karaboga, A novel binary artificial bee colony algorithm based on genetic operators, Inform. Sci. 297 (2015) 154–170.
[18] V. Singh, R. Tiwari, D. Singh, A. Shukla, RGBCA-Genetic Bee Colony Algorithm for Travelling Salesman Problem, IEEE, 2011, pp. 1002–1008.
[19] S. Sundar, A. Singh, A. Rossi, An Artificial Bee Colony Algorithm for the 0–1 Multidimensional Knapsack Problem, Springer, 2010, pp. 141–151.
[20] S. Sabet, F. Farokhi, M. Shokouhifar, A Novel Artificial Bee Colony Algorithm for the Knapsack Problem, IEEE, 2012, pp. 1–5.
[21] J. Cao, B. Yin, X. Lu, Y. Kang, X. Chen, A modified artificial bee colony approach for the 0-1 knapsack problem, Appl. Intell. (2018) 1–14.
[22] S. Sundar, A. Singh, A Swarm Intelligence Approach To the Quadratic Multiple Knapsack Problem, Springer, 2010, pp. 626–633.
[23] S. Sabet, M. Shokouhifar, F. Farokhi, A discrete artificial bee colony for multiple knapsack problem, Int. J. Reason.-Based Intell. Syst. 5 (2013) 88–95.
[24] P.-Y. Yin, Y.-L. Chuang, Adaptive memory artificial bee colony algorithm for green vehicle routing with cross-docking, Appl. Math. Modell. 40 (2016) 9302–9315.
[25] V. Panahi, N.J. Navimipour, Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators, Concurr. Comput.: Pract. Exper. (2019) e5218.
[26] S.N. Chaurasia, S. Sundar, A. Singh, Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release dates, Oper. Res. 17 (2017) 275–295.
[27] X. Li, G. Yang, Artificial bee colony algorithm with memory, Appl. Soft Comput. 41 (2016) 362–372.

[28] F.A.N. Chengli, F.U. Qiang, L. Guangzheng, X. Qinghua, Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism, J. Syst. Eng. Electron. 29 (2018) 405–414.
[29] F. Glover, Tabu search—part i, ORSA J. Comput. 1 (1989) 190–206.
[30] F. Glover, Tabu search: A tutorial, Interfaces 20 (1990) 74–94.
[31] T. Bayraktar, M.E. Aydin, M. Dugenci, A memory-integrated artificial bee algorithm for 1-D bin packing problems, in: 9th International Symposium on Intelligent Manufacturing and Service Systems, 2014, pp. 1023–1034.
[32] G. Koloch, B. Kaminski, Vehicle routing with three-dimensional container loading constraints - Comparison of nested and joint algorithms, AIP Conf. Proc. 1285 (2010) 145–157, http://dx.doi.org/10.1063/1.3510541.
[33] A. Kovacs, J.C. Beck, A global constraint for total weighted completion time for cumulative resources, Eng. Appl. Artif. Intell. 21 (2008) 691–697, http://dx.doi.org/10.1016/j.engappai.2008.03.004.
[34] H. wu Ma, W. Zhu, S. Xu, Research on the algorithm for 3L-CVRP with considering the utilization rate of vehicles, Commun. Comput. Inf. Sci. 134 (2011) 621–629, http://dx.doi.org/10.1007/978-3-642-18129-0_94.
[35] C.H. Che, W. Huang, A. Lim, W. Zhu, The multiple container loading cost minimization problem, European J. Oper. Res. 214 (2011) 501–511, http://dx.doi.org/10.1016/j.ejor.2011.04.017.
[36] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.
[37] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Glob. Optim. 39 (2007) 459–471.
[38] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142.
[39] M.S. Kiran, M. Gündüz, XOR-based artificial bee colony algorithm for binary optimization, Turkish J. Electr. Eng. Comput. Sci. 21 (2013) 2307–2328.
[40] A. Moayedikia, R. Jensen, U.K. Wiil, R. Forsati, Weighted bee colony algorithm for discrete optimization problems with application to feature selection, Eng. Appl. Artif. Intell. 44 (2015) 153–167.
[41] P.K. Singhal, R. Naresh, V. Sharma, A modified binary artificial bee colony algorithm for ramp rate constrained unit commitment problem, Int. Trans. Electr. Energy Syst. 25 (2015) 3472–3491.
[42] K. Luo, A hybrid binary artificial bee colony algorithm for the satellite photograph scheduling problem, Eng. Optim. 52 (2020) 1421–1440.
[43] S.M. Elghamrawy, Security in cognitive radio network: defense against primary user emulation attacks using genetic artificial bee colony (GABC) algorithm, Future Gener. Comput. Syst. 109 (2020) 479–487.
[44] X. Chu, S. Li, D. Gao, W. Zhao, J. Cui, L. Huang, A binary superior tracking artificial bee colony with dynamic Cauchy mutation for feature selection, Complexity 2020 (2020).
[45] M.-R. Chen, J.-H. Chen, G.-Q. Zeng, K.-D. Lu, X.-F. Jiang, An improved artificial bee colony algorithm combined with extremal optimization and Boltzmann Selection probability, Swarm Evol. Comput. 49 (2019) 158–177.
[46] J.C. Bansal, H. Sharma, K. v Arya, A. Nagar, Memetic search in artificial bee colony algorithm, Soft Comput. 17 (2013) 1911–1928.
[47] S. Kumar, V.K. Sharma, R. Kumari, An improved memetic search in artificial bee colony algorithm, Int. J. Comput. Sci. Inform. Technol. 5 (2014) 1237–1247, (0975–9646).
[48] D. Jia, T. Li, Y. Zhang, H. Wang, A memetic artificial bee colony algorithm for high dimensional problems, Int. J. Comput. Intell. Appl. 19 (2020) 2050008.
[49] E. Hancer, B. Xue, D. Karaboga, M. Zhang, A binary ABC algorithm based on advanced similarity scheme for feature selection, Appl. Soft Comput. 36 (2015) 334–348.
[50] E.E. Bischoff, M.S.W. Ratcliff, Issues in the development of approaches to container loading, Omega 23 (1995) 377–390, http://dx.doi.org/10.1016/0305-0483(95)00015-g.
[51] T.G. Crainic, G. Perboli, R. Tadei, Extreme point-based heuristics for three-dimensional bin packing, Inform. J. Comput. 20 (2008) 368–384.
[52] W.-C. Lin, J. Xu, D. Bai, I.-H. Chung, S.-C. Liu, C.-C. Wu, Artificial bee colony algorithms for the order scheduling with release dates, Soft Comput. 23 (2019) 8677–8688.
[53] P. Bhambu, S. Sharma, S. Kumar, Modified gbest artificial bee colony algorithm, in: Soft Computing: Theories and Applications, Springer, 2018, pp. 665–677.

Tuğrul Bayraktar is a Ph.D. student and research assistant at Karabuk University, Department of Industrial Engineering, Turkey. His current researches interests involve heuristic optimization algorithm for integer based problems with a specific focus on ABC algorithm for single container loading problem. He obtained his master's degree in Applicable Computing from University of Bedfordshire, UK, in 2014.

Filiz Ersöz is a Professor at Karabuk University, Department of Industrial Engineering, Turkey. Dr. Filiz Ersöz holds BSc. (1989) in Statistics from Anadolu University, MSc. (1992) and Ph.D. (1998) degrees in Biostatistics from Ankara University. She also received the rank of Associate Professor in Quantitative Decision-Making Methods from the Turkish Inter-University Council (UAK) in October 2011 and the rank of Professor in Industrial Engineering from Karabuk University (2017). Her current research deals with statistics, data mining, simulation and modeling, statistical quality control, multi-criteria decision-making techniques and decision support system. Dr. Ersöz has published widely in the field, with more than 119 publications and 5 books. She acted as a Project Manager and Advisor in more than 22 projects. Furthermore, she regularly works as reviewer for about 10 different scientific journals.

Cemalettin Kubat was born in 1952 in Sivas, Turkey. He graduated from Ankara University, Department of Mathematical in Turkey in 1974. He received his M.Sc. degree from Ege University in Izmir in 1980 preparing a thesis on Optimization of Inverse Quadratic Polynomials. Then he received his Ph.D. degree from the Istanbul University Department of Production Management in 1992 preparing a thesis on Goal Programming and Pattern Search Optimization Approach to the petroleum refinery unit Styren. He is currently working as a lecturer at Sakarya University in the Department of Industrial Engineering. He is interested in optimization, AI, genetic algorithms, fuzzy logic, computer simulation and modeling, multi agent systems, intelligent agents, and so on. He is a member of Turkish OR/IE Society and Soft Computing Society.

EK AÇIKLAMALAR B.

DOKTORA TEZİNDEN ÜRETİLİP ULUSLARARASI KONFERANS BİLDİRİ KITAPÇIĞINDA YAYIMLANAN TAM METİN

IMSS'21
INTERNATIONAL SYMPOSIUM ON INTELLIGENT MANUFACTURING AND SERVICE SYSTEMS

Proceedings of 11th International Symposium on Intelligent Manufacturing and Service Systems
Pages 49-58

Effects of Memory and Genetic Operators on Artificial Bee Colony Algorithm for Three-Dimensional Bin Packing Problem

Tuğrul Bayraktar[1], Filiz Ersöz[1] and Cemalettin Kubat[2]

[1] Karabuk University, Department of Industrial Engineering, Turkey
tugrulbayraktar@karabuk.edu.tr, fersoz@karabuk.edu.tr

[2] Sakarya University, Department of Industrial Engineering, Turkey
kubat@sakarya.edu.tr

KEYWORDS - Artificial Bee Colony Algorithm, Tabu Search, Genetic Algorithm, Three-Dimensional Bin Packing Problem, Knapsack Problem

ABSTRACT

The Artificial Bee Colony (ABC) algorithm is widely used to achieve optimum solution in a short time in integer-based optimization problems. However, the complexity of integer-based problems such as Knapsack Problems (KP) requires robust algorithms to avoid excessive solution search time. ABC algorithm that provides both the exploitation and the exploration approach is used as an alternative approach for various KP problems in the literature. However, it is rarely used for the Three-Dimensional Bin Packing Problem (3DBPP) which is an important part of the transportation systems. In this study, the exploitation and exploration aspects of the ABC algorithm are improved by using memory mechanisms and genetic operators to develop three different hybrid ABC algorithms. The developed algorithms and the basic ABC algorithm are applied to a generated 3DBPP dataset to observe the effects of the memory mechanism and the genetic operators separately. The results show that the genetic operators are more effective than the memory mechanism to develop a hybrid ABC algorithm, for solving heterogeneous 3DBPPs.

1 INTRODUCTION

Containers are one of the basic elements of transportation networks. Commonly used containers to be filled by many goods for distribution to the same or different locations have a variety of dimensions. Allocating items into limited spaces, is a combinatorial optimization problem and bin packing problem (BPP) is a branch of knapsack problems (KP), where a set of items is loaded into multiple capacitated bins. If the sizes of items and bins differ in all three direction, this type of problem is called as the three-dimensional bin packing problem (3D-BPP) [1].

First part of 3D-BPP researches focused on to improve the mathematical modellings [2] for better item allocation approach, the second part of the researches proposed solution approaches [3] for variable bin packing conditions and constraints, and the last part of the researches has improved

the item allocation orders with heuristic methods [4]–[6] to provide better solutions compared to the integer-based programming approaches.

Artificial Bee Colony (ABC) algorithm is a neuro-inspired meta-heuristic approach based on foraging behavior of the bee colonies and the effectiveness of the ABC algorithm has been proved by many one-dimensional KP researches [7], [8] beside the previous work [9]. However, the ABC algorithm has been scarcely used for 3D-BPP and this study aims to contribute to this branch of BPP, by using the ABC algorithm as a solution search approach.

Genetic Algorithm (GA) based on improving the randomly generated individuals of the population thorough iterations, and Tabu Search (TS) based on restricting the search moves to explore the best problem solution, are other heuristic methods commonly used for KP and can likewise be used for 3D-BPP. Gehring and Bortfeldt [10] proposed one of the first GAs for the Single Container Loading Problem (SCLP), a branch of the KP, to satisfy the loading constraints when allocating items to the container. Bortfeldt and Gehring [11] also proposed a hybrid GA to optimize the container loading plans by building layers of well-classified items. Wu et al. [12] used GA for Strip Packing Problem (SPP) to optimize the bin packing plan to determine the height of bins in use. Kang et al. [4] used GA to minimize the number of rectangular residual spaces in the bins to reduce the number of bins in use.

Bortfeldt and Gehring [13] also implemented TS for SCLP to minimize the volume of rectangular residual spaces in the container by selecting the best item in each allocation. Liu et al. [14] used a hybrid TS that selects items not only individually but also as groups in each allocation and provides alternatives for container loading solutions by allocating the assigned items vertically or horizontally. Mack et al. [15] used TS as a reinforcement to the hybrid algorithm to avoid abandoned solutions to be re-generated over a period of time. Zhu et al. [16] used TS to select one of item placement strategies (deep-bottom-left or maximum touching area) in each allocation step to obtain the optimum loading solution.

Genetic operators (mutation and crossover) in GA, and TS strategies has been used as reinforcement approaches to strengthen the exploration and the exploitation aspects of the ABC algorithm respectively. Ozturk et al. [8] and Panahi and Navimipour [17] used genetic operators to improve the exploration capability of the ABC algorithm by increasing the number of alternative solutions around existing ones in the population. The new alternative solutions were generated in both employed bee and onlooker bee phase, using the crossover operator between the current solution, two random neighbor solution, zero solution and best solution obtained. All the solutions obtained after crossover were mutated before choosing the best alternative as the new solution among the current and generated solutions. Chaurasia et al. [18] simplified the search process in the ABC algorithm by using 3-point insertion method to generate new solutions from the current one and one of its random neighbors. In this way, the number of function evaluations was reduced, compared to the approach in [8].

TS strategies avoid the repetitive search steps while generating new solutions by classifying steps as efficient or inefficient to save in different tabu lists. Chengli et al. [19] integrated a memory mechanism into the ABC algorithm to save successful parameter pairs for reuse in next iterations to increase the probability of escaping local optimum. In the previous study [9], a memory mechanism was integrated into the ABC algorithm to save inefficient solutions into the Short-Term Tabu List (STTL) to improve the exploitation ability in the neighborhood search phase.

This study aims to improve the ABC algorithm separately with genetic operators and a memory mechanism, and then observe the effects of the two reinforcement approaches on the ABC algorithm. The basic ABC algorithm, the memory integrated ABC (MIABC) algorithm and the genetic operator-based ABC (GABC) algorithm were separately applied to a generated 3D-BPP data set. The rest of

the paper organized follows: the brief explanation of 3D-BPP and the way of its implementation in this study is provided in Section 2; the proposed solution approaches are explained in Section 3; results from applied approaches are represented in Section 4, and Section 5 concludes with the future work and inferences about research undertaken.

2 3D BIN PACKING PROBLEM

3D-BPP is a branch of KP based on placing the entire set of 3D elements into as few 3D bins as possible, so the aim is to maximize the average utilization ratio (UR) of the bin in use without item intersection and dimensional exceeding. The average utilization ratio is calculated as shown in Eq. (1) and Eq. (2), without considering any other variables but the volume of the assigned items:

$$maximize \quad \frac{\sum_{b=1}^{C}\sum_{i=1}^{n} a_{ib}.v_{ib}}{\sum_{b=1}^{C} V_b} \tag{1}$$

$$subject\ to \quad \sum_{i=1}^{n} a_{ib}.v_{ib} \leq V_b \tag{2}$$

The average UR in Eq. (1) equals the sum of the volumes of all items assigned to the bins divided by the total volume of the bins used where $a_{ib} \in [0,1]$ is an item assignment variable, v_{ib} is the volume of the assigned item $i = 1, \dots, n$, and V_b is the volume of bin $b = 1, \dots, C$. In this study, V_b is identical for all bins. The total volume of the assigned items in each bin cannot be larger than the capacity of the bin as shown in Eq. (2).

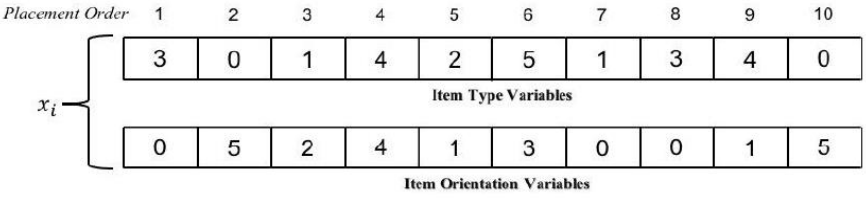


Figure 1: Bin Packing Sequence

In this study, the items are divided into five types for each problem set, and each item type can be placed in the bin in six orientation. The bin packing process begins by generating a packing sequence consisting of the item types $t = 0, \dots, 5$ in the first row and the item orientation $o = 0, \dots, 5$ in the second row in each column, as shown in Figure (1). Types and orientations are randomly placed in the packing sequence in pairs. Then, the items are placed into the first bin according to the placement order using the deep-bottom-left first (DBLF) item placement approach. Whole packing sequence is assigned to bins according to first-fit approach, that allocates the next item, starting with the first bin each time, whichever bin it can fit in first.

3 HEURISTIC APPROACHES FOR 3D BIN PACKING PROBLEM

3.1 Artificial Bee Colony Algorithm for 3D Bin Packing Problem

The ABC algorithm was developed by Karaboga and mimics the foraging behaviors of honeybees that are divided into three groups; employed bees, onlooker bees and scout bees. The ABC algorithm was originally designed for numerical problems [20] and eventually modified for integer-based problems [21] due to the easy applicability and the search simplicity.

The initial population in the basic ABC algorithm is generated from a randomly chosen solution using Eq. (3) for numerical problems, where $i = 1 \dots SN$ refers to the i-th food source and SN refers to the total number of bees and food sources in the search area. $j = 1 \dots D$ refers to the j-th dimension value of the i-th food source between an upper and a lower bound and D refers to the total number of parameters of the i-th food source to optimize. In this study, the initial population is generated from a random packing sequence by replacing one pair of type and orientation variables with a randomly selected pair, in each generation process.

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right) \tag{3}$$

In the employed bee phase, each employed bee visits only one solution to generate a new one using Eq. (4) for numerical problems, where v_{ij} is the new solution generated from the interaction between the same j elements of visited solution x_i and its neighbor solution x_k, and the difference between x_i and x_k is weighted by the ϕ_{ij}, which takes values between $[-1, 1]$. In this study, the new solution is generated according to binary optimization scheme using Eq. (5), where the $\oplus$ symbol is an xor operator [22] corresponding to the $(-)$ operator in Eq. (4), to measure the difference between x_i and x_k. If the fitness value f_{v_i} is better than the visited solution x_i's, v_i replaces it. Otherwise, the failure counter of the visited solution $failure_i$ is increased by one.

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{4}$$

$$v_{ij} = x_{ij} \oplus \phi\left(x_{ij} \oplus x_{kj}\right) \tag{5}$$

In the onlooker bee phase, the bees in the hive evaluate the fitness values of the solutions calculated in Eq. (1) and choose one of them with the probability p_i calculated as in Eq. (6). If the onlooker bees improve the current food sources, they memorize the new food sources and forget the old one. Otherwise, $failure_i$ value is increased by one again.

$$p_i = f_i \,/ \left(\sum_{i=1}^{SN} f_i\right) \tag{6}$$

If the maximum failure exceeds the failure limit L, the onlooker bee abandons the food source, except the food source with the best quality, and turns into the scout bee that explores new food sources randomly by using Eq. (3). In each of the iteration, only one onlooker bee is allowed to become a scout bee.

3.2 Memory Integrated Artificial Bee Colony Algorithm

The honeybees in ABC algorithm forget all the information about the improvement process, once they abandon the food source that reaches the maximum failure limit in the population. However, the information about a succeeded or failed move from x_i to x_j can be used by other

honeybees to accelerate the search process. In this study, the search moves ($move_{ij}$) are represented as arrays in which the processed item with the related elements in x_i and x_j are saved, as shown in Fig. (2). In the $move_{ij}$ array, the first element indicates the packing order of the item and the placement information about the item in x_i and x_j is saved respectively in the rest of the array.

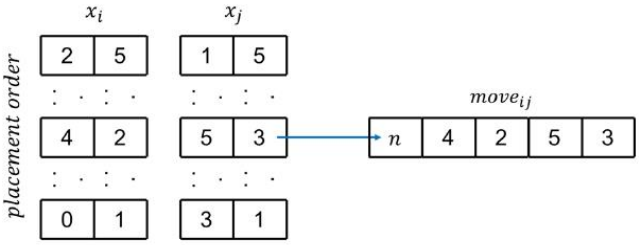


Figure 2: The Memory Mechanism for ABC Algorithm

If $move_{ij}$ improves the solution in the employed and the onlooker bee phase, it is saved in Intermediate-Term Tabu List (ITTL). Besides, $move_{ij}$ is saved in Long-Term Tabu List (LTTL) [23], if the scout bee, that abandoned a solution, meets the $\left|f_{x_{best}} - f_{x_j}\right| < \left|f_{x_j} - f_{x_{mean}}\right|$ and $f_{x_j} > f_{x_i}$ conditions, and manages to carry the solution to a fruitful search area. The next moves in the employed and onlooker bee phase are selected randomly from ITTL and LTTL list or generated randomly with a weight probability calculated in Eq. (7), where iw_p is the intensification weight of the candidate path (ITTL, LTTL or random search) and $intscore_p$ is the intensification score that must be at least one for all paths.

$$iw_p = intscore_p \Big/ \left(\sum_{p=1}^{PI} intscore_p\right) \tag{7}$$

If the selected path p improves the visited solution, $intscore_p$ is increased by one or reduced by one if the path p fails to improve the visited solution. On the other hand, if $move_{ij}$ fails to improve the visited solution, it is saved in Short-Term Tabu List (STTL) and is prohibited to be used to generate new solutions for a limited number of iterations [9].

3.3 Genetic Operator Based Artificial Bee Colony Algorithm

GA algorithm diversifies the search randomly by using cross-over and mutation operators. In this section, four types of operator are used for the path selection in the employed bee and onlooker bee phase: (i) only cross-over operator, (ii) only mutation operator, (iii) cross-over and mutation operator together, and (iv) random search operator.

Cross-over operator in GA, generates two child solutions by taking the random elements of parent solutions x_i and x_k. However, we need the cross-over operator in ABC algorithm to generate one child solution for each x_i. Chaurasia et al. [18] proposed a genetic operator-based ABC algorithm, in which the multi-point insert method replaces the cross-over operator by generating one child solution from two parent solutions as shown in Fig. (3). In genetic operator-based ABC (GABC) algorithm, multi-point insert method is used as a path to generate the new solution.

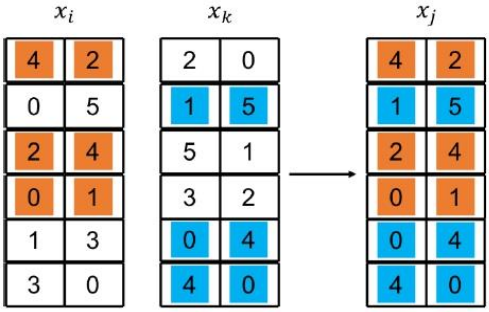


Figure 3: Multi-Point Insert Method

The path that uses the mutation operator to swap only two item-elements of the x_i to generate the new solutions. The third path is to use multi-point insert method and mutation operator respectively to generate the new solution x_j. The last path is the random search same as in MIABC algorithm. The $intscore_p$ and iw_p of the paths are calculated as in Eq. (7) to select the one of paths by roulette wheel probability.

In the GABC algorithm, the solution search paths; (i) random food source replacement rfs using random search and (ii) elite food source guided replacement efs using the interaction between the abandoned solution and the elite food sources " e ", are weighted to improve the diversification aspect in the re-scout bee phase [24].

$$dw_p = divscore_p \Big/ \left(\sum_{p=1}^{PD} divscore_p \right) \tag{8}$$

If the fitness value of the generated solution, that replaces the abandoned one, meets the $\left|f_{x_{best}} - f_{x_j}\right| < \left|f_{x_j} - f_{x_{mean}}\right|$ and $f_{x_j} > f_{x_i}$ conditions, the diversification score of the path $divscore_p$ is increased by one. The search path in the re-scout bee phase is selected by roulette wheel probability calculated as in Eq. (8), where dw_p is the diversification weight of the candidate path (rfs or efs).

4 COMPUTATIONAL RESULTS AND DISCUSSION

The basic ABC, MIABC and GABC algorithms that are developed for 3D-BPP have been coded in MATLAB R2016 version software. All experimental runs are performed by the CPU that has 4 GB RAM and 3.10 GHz processors using Windows 7 operating system. Developed algorithms are tested on a randomly generated dataset according to the random instance generator used by [1]. The generated data set includes following five types of random items to allocate into the bins with uniform dimensions where $D = W = H = 100$.

Each data set class consists of five item type $k = (1, \dots, 5)$, items of type k are chosen with probability 60%, and the rest four types are chosen with probability 10%, so the developed algorithms are tested on three classes of data sets consisting of 25 sub-problems. Each data set class considers the number of items to be placed as 20, 50, and 100, respectively.

Parameter variables of ABC algorithm, the size of population SN, total number of evaluations $Eval$, and the failure limit of each bee L determine framework of the solution search process, where

n is the number of items to be placed into bins. In addition, the number of elite bees e determines how many problem solutions are capable of attracting others in the population to reduce the duration of convergence of the GABC algorithm. The size of the STTL, LTTL and ITTL determines the number of search moves saved into the list that guides the search using useful moves and avoiding prohibited ones. The parameter values were set as seen in Table 1.

Table 1: Parametric Details of the Algorithmic Configuration

Algorithm	*SN*	*Eval*	*e*	*m*	*L*
ABC	50	$5n^2$	–	–	25
MIABC	50	$5n^2$	–	$n \times SN$	25
GABC	50	$5n^2$	10	–	25

The basic ABC algorithm involves generating the first population from one individual and the same initial populations are randomly generated for sub-problems of each data set class. The graphs in Fig. (4) shows the search history of average obtained values of 25 sub-problems for each data set class in the average bin usage ratio (BUR) obtained by the basic ABC algorithm and proposed approaches, and Table 2 shows performance of the three approaches on the generated data set. The search history of each class is statistically analyzed for the basic ABC and the proposed approaches by using one-way ANOVA and Fisher's Least Significant Difference (LSD) post hoc test.

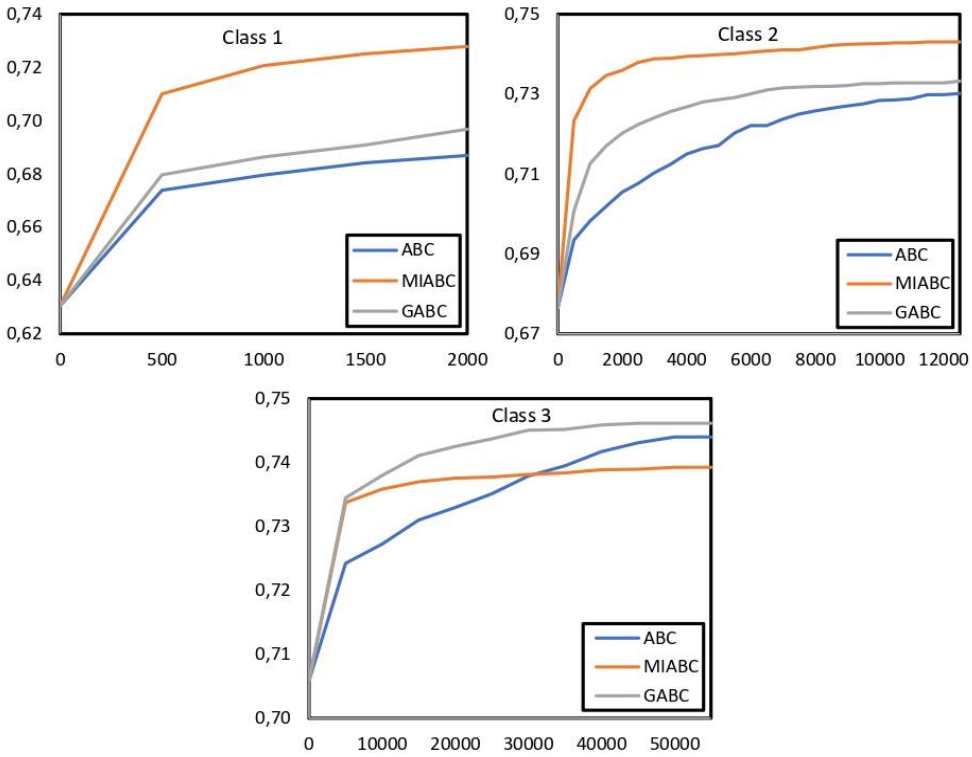


Figure 4: Search History of Three Classes of Data Sets in BUR Value

Fig. (4) depicts the change of BUR values over the evaluations. The characteristics of the three approaches vary according to the data set class they are applied to. The memory mechanism and genetic operators affect the basic ABC algorithm significantly as seen in the graphs in Fig. (4), where the three approaches are significantly different ($p < .05$) for all three data set classes. The MIABC algorithm is superior to the GABC algorithm with a difference of 3% for the Class 1 data set and 1% for the Class 2 data set, as seen in Table 2. However, as the complexity of the problem increases with the number of items to be placed, the MIABC algorithm loses its effectiveness for Class 3 data set and even lags the basic ABC algorithm. The memory mechanism shows its effectiveness in early stage of the search with significant outcomes; however, the search is limited in the long run by the memory mechanism itself. The results of the MIABC algorithm and the search process show that the memorized movements can mislead the search. As the number of prohibited moves is increased, the possibility of exploring new solutions in the local area is also restricted. On the other hand, the memorized moves in ITTL and LTTL are not as effective as expected to lead the solution search to fruitful areas in the search space.

Table 2: Average Performance of the Basic ABC Algorithm and Proposed Approaches Over Average BUR Values for Each Data Set Class

	ABC	MIABC	GABC
Class 1 (n=20)	0.6870	**0.7279**	0.6968
Class 2 (n=50)	0.7302	**0.7431**	0.7332
Class 3 (n=100)	0.7440	0.7392	**0.7461**

GABC algorithm manages to improve the basic ABC algorithm for all data set classes. However, although the GABC algorithm obtains significantly better results than the ABC algorithm in the early stage, the difference between two approaches closes over time. Unlike the MIABC algorithm, the GABC algorithm spreads the solution search through evaluations and provides a continuous improvement in the BUR value. Besides, the change in the problem complexity does not affect the GABC algorithm as much as the MIABC algorithm.

As a result, the capacity of memory mechanism is limited for three-dimensional bin packing problems and the memory integrated ABC algorithm converges prematurely when it is applied on high complex problems. However, the genetic operator integrated ABC algorithm avoids getting stuck at the local optimum with its strong diversification aspect, and the genetic operator-based reinforcement more useful than the memory mechanism in the long run.

5 CONCLUSION

The ABC algorithm is commonly used for numeric optimization problems. However, a binary optimization-based ABC algorithm has recently been applied to the three-dimensional bin packing problems, which is a kind of knapsack problem and one of the main problems of transport systems. In most ABC algorithms applied knapsack problem studies, a number of dimensions are considered as the number of parameters for each item, while in this study three-dimensional items are regarded as orthogonal objects to be placed in a three-dimensional knapsack called as a container.

The ABC algorithm is a powerful and efficient algorithm for numerical optimization with a combination of intensification and diversification aspects. However, enhanced ABC algorithms are required to solve complex problems such as 3D bin packing problems. This study focuses on enhancing the search mechanisms that are used in local search and global search to develop a robust ABC algorithm for container loading problems. A memory mechanism is used to avoid repetitive

item placement solutions and to benefit from the fruitful ones in local search, while the genetic operators are integrated into the basic ABC algorithm to expand the global search area in order to discover potential better solutions. Reinforcement approaches that improve different aspects of the basic ABC algorithm are analyzed separately to understand their effect on the algorithm.

As a result, this study proposes a memory-integrated ABC algorithm to meticulously select useful search steps in local search, and a genetic operator-based ABC algorithm to intelligently generate the next search steps in global search inspired by efficient solutions. The results show that using the memory mechanism is more effective in the short run. However, it loses its effectiveness in the long run and cannot be applied to the 3D bin packing problems with high complexity, while the genetic operator-based ABC algorithm provides better solutions in the long run and is more robust than MIABC algorithm, in high complexity.

The BUR values obtained from the proposed algorithms can be improved in future studies, focusing on bin packing heuristics, based on decoding the complete random packing sequence in this study. The packing sequences can be generated that place similar items in blocks, layers or stacks to reduce spaces in bins. On the other hand, a joint hybrid algorithm, that uses both memory mechanism and genetic operators, can be developed to observe the effects of the proposed approaches when they are used together as reinforcement approaches for the basic ABC algorithm.

REFERENCES

[1] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," Operations research, vol. 48, no. 2, pp. 256–267, 2000.

[2] N. Nepomuceno, P. Pinheiro, and A. L. v Coelho, "Tackling the Container Loading problem: A hybrid approach based on Integer Linear Programming and Genetic Algorithms," in Evolutionary Computation in Combinatorial Optimization, Proceedings, vol. 4446, C. Cotta and J. VanHemert, Eds. 2007, pp. 154-+.

[3] T. Tian, W. B. Zhu, A. Lim, and L. J. Wei, "The multiple container loading problem with preference," European Journal of Operational Research, vol. 248, no. 1, pp. 84–94, Jan. 2016, doi: 10.1016/j.ejor.2015.07.002.

[4] K. Kang, I. Moon, and H. F. Wang, "A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem," Applied Mathematics and Computation, vol. 219, no. 3, pp. 1287–1299, 2012, doi: 10.1016/j.amc.2012.07.036.

[5] L. Junqueira and R. Morabito, "Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier," Computers & Industrial Engineering, vol. 88, pp. 110–130, 2015, doi: 10.1016/j.cie.2015.06.005.

[6] A. Moura and A. Bortfeldt, "A two-stage packing problem procedure," International Transactions in Operational Research, vol. 24, no. 1–2, pp. 43–58, 2017, doi: 10.1111/itor.12251.

[7] S. Sundar and A. Singh, "A swarm intelligence approach to the quadratic multiple knapsack problem," 2010, pp. 626–633.

[8] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," Information Sciences, vol. 297, pp. 154–170, 2015.

[9] T. Bayraktar, M. E. Aydin, and M. Dugenci, "A memory-integrated artificial bee algorithm for 1-D bin packing problems," in 9th International Symposium on Intelligent Manufacturing and Service Systems, 2014, pp. 1023–1034.

[10] H. Gehring and A. Bortfeldt, "A genetic algorithm for solving the container loading problem," International transactions in operational research, vol. 4, no. 5–6, pp. 401–418, Nov. 1997, doi: 10.1111/j.1475-3995.1997.tb00095.x.

[11] A. Bortfeldt and H. Gehring, "A hybrid genetic algorithm for the container loading problem," European Journal of Operational Research, vol. 131, no. 1, pp. 143–161, 2001, doi: 10.1016/s0377-2217(00)00055-2.

[12] Y. Wu, W. K. Li, M. Goh, and R. de Souza, "Three-dimensional bin packing problem with variable bin height," European Journal of Operational Research, vol. 202, no. 2, pp. 347–355, 2010, doi: 10.1016/j.ejor.2009.05.040.

[13] A. Bortfeldt and H. Gehring, "Applying tabu search to container loading problems," in Operations Research Proceedings 1997, Springer, 1998, pp. 533–538.

[14] J. M. Liu, Y. Yue, Z. R. Dong, C. Maple, and M. Keech, "A novel hybrid tabu search approach to container loading," Computers & Operations Research, vol. 38, no. 4, pp. 797–807, 2011, doi: 10.1016/j.cor.2010.09.002.

[15] D. Mack, A. Bortfeldt, and H. Gehring, "A parallel hybrid local search algorithm for the container loading problem," International Transactions in Operational Research, vol. 11, no. 5, pp. 511–533, Sep. 2004, doi: 10.1111/j.1475-3995.2004.00474.x.

[16] W. B. Zhu, H. Qin, A. Lim, and L. Wang, "A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP," Computers & Operations Research, vol. 39, no. 9, pp. 2178–2195, 2012, doi: 10.1016/j.cor.2011.11.001.

[17] V. Panahi and N. J. Navimipour, "Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators," Concurrency and Computation: Practice and Experience, p. e5218, 2019.

[18] S. N. Chaurasia, S. Sundar, and A. Singh, "Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release dates," Operational Research, vol. 17, no. 1, pp. 275–295, 2017.

[19] F. A. N. Chengli, F. U. Qiang, L. Guangzheng, and X. Qinghua, "Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism," Journal of Systems Engineering and Electronics, vol. 29, no. 2, pp. 405–414, 2018.

[20] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[21] S. Sundar, A. Singh, and A. Rossi, "An artificial bee colony algorithm for the 0–1 multidimensional knapsack problem," 2010, pp. 141–151.

[22] M. S. Kiran and M. Gündüz, "XOR-based artificial bee colony algorithm for binary optimization," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 21, no. Sup. 2, pp. 2307–2328, 2013.

[23] F. Glover, "Tabu search—part I," ORSA Journal on computing, vol. 1, no. 3, pp. 190–206, 1989.

[24] P. K. Singhal, R. Naresh, and V. Sharma, "A modified binary artificial bee colony algorithm for ramp rate constrained unit commitment problem," International Transactions on Electrical Energy Systems, vol. 25, no. 12, pp. 3472–3491, 2015.

ÖZGEÇMİŞ

Tuğrul BAYRAKTAR; ilk öğrenimini Adapazarı'nda tamamladı. Bolu Fen Lisesi'nden mezun oldu. 2005 yılında Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nde öğrenime başlayıp 2009 yılında mezun oldu. Aynı yıl Millî Eğitim Bakanlığı bünyesinde Yurtdışı Lisansüstü Seçme Yerleştirme (YLSY) Bursu Programından faydalanmaya hak kazandı. 2010 – 2011 yılları arasında YLSY bursu kapsamında İstanbul ve Londra'da yabancı dil eğitimini tamamladıktan sonra 2013 – 2014 yılları arasında İngiltere Bedfordshire Üniversitesi Uygulamalı Bilgisayar Bilimleri bölümü araştırma temelli yüksek lisans programında öğrenimini tamamlayıp mezun oldu. 2014 yılında Karabük Üniversitesi Endüstri Mühendisliği bölümünde araştırma görevlisi olarak göreve başladı ve halen aynı yerde çalışmaya devam etmektedir.