T.C.

KIRIKKALE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİK ANABİLİM DALI

YÜKSEK LİSANS TEZİ

HİBRİD KARGA-ARAMA GENETİK ALGORITMASINI KULLANARAK 3 BOYUTLU KUTU PAKETLEME SORUNUNU ÇÖZME

Hazem Khudeer

Eylül 2020

ONAY SAYFASI

Bilgisayar Mühendisliği Anabilim Dalında Hazem Khudeer tarafından hazırlanan HIBRID KARGA-ARAMA GENETIK ALGORITMASINI KULLANARAK 3D KUTU

PAKETLEME SORUNUNU ÇÖZME adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Doç. Dr. Atilla ERGÜZEN
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Prof. Dr. Hasan ERBAY
Danışman

Jüri Üyeleri

Başkan	:	Dr. Öğr. Üyesi Fahrettin Horasan	_____
Üye (Danışman)	:	Prof. Dr. Hasan Erbay	_____
Üye	:	Dr. Öğr. Üyesi Ahmet Yurttakal	_____

.../.../...

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Prof. Dr. Recep ÇALIN
Fen Bilimleri Enstitüsü Müdürü

ÖZET

HİBRİD KARGA-ARAMA GENETİK ALGORITMASINI KULLANARAK 3 BOYUTLU KUTU PAKETLEME SORUNUNU ÇÖZME

Khudeer, Hazem

Kırıkkale Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi

Danışman: Prof. Dr. Hasan ERBAY

Ağustos 2020, 72 sayfa

Bu çalışmada, üç boyutlu Kutu Paketleme Problemi'ni (KPP) çözmek için Hibrit Meta-Sezgisel algoritmalar geliştirilmiştir. Algoritmalar, Karga Arama Algoritması ile Genetik algoritmanın bir kombinasyonudur. 3Boyutlu küboid Kutu paketleme problemi ilk kez Karga algoritması ve Hibrit Karga-Genetik algoritmaları kullanılarak çözülmüştür. 3Boyutlu KPP probleminin özellikleri arasında küboid şeklinde 90 derece dönebilen kutular ile birlikte genişliği ve uzunluğu sabit ve serbest yükseklikte bir konteyner bulunmaktadır. Problem için bir karşılaştırma veri seti olmadığından, rastgele bir veri seti oluşturulmuştur. Bu çalışma kapsamında Karga algoritması, Genetik algoritması, Hibrit Karga-Genetik algoritması, Hibrit Karga-Genetik algoritması tasarlanmış ve test edilmiştir. Sonuçlar Hibrit Karga-Genetik algoritmasının diğerlerinden daha iyi olduğunu göstermektedir

Anahtar kelimeler: Kombinatoryal Problemleri. Kutu Paketleme Problemi, Sezgisel, Meta-Sezgisel, Karga Arama Algoritma, Genetik algoritma, Optimizasyon, Tedarik zinciri.

ABSTRACT

HYBRID CROW-GENETIC ALGORITHM FOR SOLVING 3D BIN PACKING PROBLEM

Khudeer, Hazem

Kırıkkale University

Institute of Science and Technology

Department of Computer Engineering, Master Thesis

Supervisor: Assoc. Prof. Dr. Hasan ERBAY

August 2020, 72 pages

Throughout this study, Hybrid Meta-Heuristic algorithms developed for solving the three-dimensional Bin Packing Problem (3D-BPP). The algorithms are a combination of the Crow Search Algorithm and the Genetic algorithm. It is the first time that the bin packing problem of three dimensional-type is solved using the Crow algorithm and the hybrid Crow-Genetic algorithms. The characteristics of the problem are offline 3D-BPP, cuboid-shaped items, and a container of fixed in width and length, but free in height, in addition, the items are rotatable by 90 degrees. Since there exists no benchmark dataset for the problem, a random dataset was generated. Herein Crow algorithm, Genetic algorithm, Hybrid Crow-Genetic algorithm, Hybrid Genetic-Crow algorithm were designed and tested. The results show that the Hybrid Crow-Genetic algorithm performed better than others.

Keywords: Supply Chain, Bin Packing Problem (BPP) 3D, Combinatorial Optimization, Hybrid Meta-Heuristic, Crow Search algorithm, Genetic Algorithm.

TEŞEKKÜR

Tez danışmanım Sayın Prof. Dr. Hasan ERBAY'a, tez süreci boyunca manevi desteğinden dolayı babam, annem ve eşime teşekkür ederim.

Bu tezi, desteğini hep yanımda hissettiğim annem, babam ve eşime ithaf ediyorum.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLERİN DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Kombinatoryal problemler yapay zekanın önemli bir parçasıdır. Kutu Paketleme Problemi (KPP) bu problemlerden biridir. Tipik KPP problemi kısaca, kutuların en iyi şekilde nasıl yerleştirilebileceği, kutuların nasıl minimum miktarda alan kaplayacağını ve sonuçta kullanılan konteyner sayısının mümkün olduğunca azaltılmasını ile ilgilenir. KPP, lojistik sistem ve üretim sisteminde klasik ve önemli bir optimizasyon problemidir. KPP'nin birçok çeşidi vardır. Ancak en anlamlı ve zorlayıcı olanı, farklı boyutlardaki küboid şekilli öğelerin dörtgensel biçimde konteynerlere paketlendiği 3 boyutlu olanıdır. Meta-sezgisel algoritmalar, özellikle KPP yada seyahat eden satıcı problemi gibi NP-Zor(NP-Hard) problemler için modern sayısal optimizasyonda güçlü hale gelmektedir. Meta-sezgisel algoritmaları kullanarak kombinatoryal problemlerde daha verimli sonuçlar elde etmek [1], bu algoritmaların tercih edilmesinin sebeplerinden biridir. Ötedenberi, KPP birçok farklı yaklaşımla çözülmeye çalışılmaktadır [2,3], ancak bu çalışma kapsamında kullanılan Karga Algoritma ve Hibrit Karga Genetik Algoritma'larının kullanılması literatüre farklı yaklaşımlar sunmaktadır. Karganın yiyecek arayışındaki yüksek zekası bu yöntemi seçmenin ana nedenidir. KPP, Tedarik Zinciri sektöründe önemli bir yer almaktadır ve Araç Yönlendirme Sorunu ile yakından ilişkilidir. Kargo şirketleri tarafından da kullanılır. KPP, nakliye maliyetlerini düşürme etkisi nedeniyle taşıma sektöründe birinci önceliktedir. Bu sorunun karmaşık ve çok geniş çözüm alanı vardır. Bu tip sorunlara NP-Zor denir [4]. 3 boyutlu KPP, sıradan yöntemler veya Kaba Kuvvet (Brute-Force) algoritması kullanarak özellikle girdi boyutunun büyük olduğu durumlarda pratikte çözülememektedir. Bunun temel nedeni küboid şekilli dönebilen n adet kutunun bir konteynera (n! * 6 ^ n) farklı biçimde yerleştirilebilir olmasıdır. Öte yandan verilerin temsili ise başka bir zorluktur. Ayrıca, algoritmanın cevaplama süresi kritiktir. Bazı durumlarda birkaç saniye içinde bir karar vermek gerekir. Çünkü gecikme nakliye sürecinde aşırı maliyetlidir. Örneğin, nakliye uçağının 1 dakika gecikmesi yaklaşık 60$'dır. Bu çalışmanın katkısı, 3 boyutlu KPP çözümünde ilk kez Karga Algoritmasının[5] kullanılmasıdır. Bu çalışmada, 3boyutlu KPP Genetik Algoritma , Karga Algoritma, Hibrid Genetik-Karga Algoritma ve Karga-Genetik Algoritma gibi farklı algoritmalar kullanılarak çözülmüştür. Kullanılan yöntemlerde hem rastgele

örnekler hem de optimum çözümü bilinen bir örnek üzerinden karşılaştırma yapılmıştır. Bu çalışmanın odaklandığı yaklaşım, yüksekliği değişken olan bir konteynırın hacmini mümkün olduğunca aza indirgemektir. Değerlendirme ölçütümüz kullanılan konteynırın hacmi ile alakalıdır. Dolayısıyla bir konteynerde ne kadar az hacim kullanılarak yerleşim yapılırsa o kadar iyi sonuç elde edilmiş olur.

Tezin geriye kalan kısmında aşağıdaki konulardan bahsedilmiştir.

Tezin ikinci bölümünde, hem genel hem de hesaplamalı karmaşıklık teorisi problemlerinden bahsedilmiştir. Devamında KPP'nin çeşitleri, kriterleri ve çözüm yöntemleri anlatılmıştır. Ayrıca, kullanılan algoritmalar ve veri temsili hakkında bilgi verilmiştir. Literatür taraması da yapılmıştır. Tezin üçüncü bölümünde, KPP'nin bu çalışmada kullanılan tibi ve kriterlerinden bahsedilmiştir. Buna ek olarak bahsedilen algoritmaların KPP üzerine pratik olarak nasıl uygulanabileceği gösterilmiştir. Tezin son bölümünde ise kullanılan metotlar ile elde edilen sonuçlar sunulmuş ve analiz edilmiştir.

2. MATERYAL VE YÖNTEM

Bu kısımda, tez boyunca kullanılan materyal ve yöntemlerden, problem tiplerinden ve karmaşıklıklarından bahsedilecektir. Ayrıca KPP problemi, çeşitleri ,çözüm yöntemleri, yöntemlerin karşılaştırılması ve veri temsili türleri ele alınacaktır. Buna ek olarak literatür taraması da yapılacaktır.

2.1. Kombinatoryal problemler

Sonlu geniş bir nesne grubundan, en uygun nesneyi bulmaya kombinatoryal problem denilmektedir. Bu tür problemlerde, kapsamlı çözüm aramak izlenilebilir bir durum değildir. Geçerli bir çözüm oluşturabilecek çok sayıda olası kombinasyon nedeniyle, kombinatoryel problemlerin çözümü genellikle çok zordur. Bilgisayar bilimlerinin birçok alanında ve hesaplama yöntemlerinin uygulandığı diğer alanlarda kombinatoryal problemler ortaya çıkmaktadır. Örneğin yapay zeka, yöneylem araştırması, biyoinformatik ve elektronik ticaret gibi alanlarda kullanılır. Öne çıkan örnekler, alan kullanımını en üst düzeye çıkarmak için tüm kutuları konteynera paketleme problemi, haritalarda en kısa veya en ucuza gidiş-dönüş yolculuk bulma, önerme formüllerinin modellerini bulma veya proteinlerin 3 boyutlu yapısını belirleme gibi örneklerdir. Planlama, zamanlama, kaynak tahsisi, kod tasarımı, donanım tasarımı ve genom dizilemesinde de diğer iyi bilinen kombinatoryal problemlerle karşılaşılmaktadır. Bu problemler tipik olarak, belirli koşullara veya kısıtlamalara bakarak, sonlu bir nesne kümesinin gruplarını, düzenlerini veya atamalarını bulmayı içerir. Örneğin bir zamanlama problemi, bir atama problemi olarak görülebilir ve bu atama probleminin çözümünün bileşenleri, planlanacak olaylardır ve olaylara atanan değerler ne zaman olacağına bağlıdır. Bu şekilde, tipik olarak çok sayıda çözüm yöntemi elde edilebilir. Çoğu kombinatoryal optimizasyon problemi için potansiyel çözümlerin sayısı, üslü sayı boyutundadır[6]. Kombinatoryal problemlerin örnekleri için aday çözümler ve çözümler arasında önemli bir ayrım yapılmalıdır. Aday çözümler, verilen problem örneğini çözme girişimi sırasında karşılaşılabilecek potansiyel çözümlerdir. Ancak çözümlerden farklı olarak sorun tanımındaki tüm koşulları karşılamayabilirler. En kısa gidiş-dönüş yolculuğumuz örneğinde, uzunluktan bağımsız olarak belirli bir nokta kümesinde geçerli herhangi bir gidiş-

dönüş mesafesi, aday bir çözüm olurken, sadece minimum uzunlukta olan gidiş-dönüş mesafesi çözüm olarak nitelendirilir.

2.2. Karar Problemleri

Karar problemi, belirtilen girdi kümelerinde evet ya da hayır sorusudur. Örneğin, "iki doğal sayı x ve y verildiğinde, x, y'yi eşit olarak böler mi?" sorusu bir karar problemidir. Cevap "evet" veya "hayır" olabilir. Bu x ve y değerlerine bağlıdır. Karar problemleri, cevapları basit bir "evet" veya "hayır" dan daha karmaşık soruları olan fonksiyon problemleri ile yakından ilişkilidir. Her karar problemi eşdeğer bir fonksiyon problemine dönüştürülebilir veya her fonksiyon problemi eşdeğer bir karar problemine dönüştürülebilir. Örneğin, karar probleminin "iki x ve y sayısı verildiğinde, x, y yi eşit olarak bölermi?" ve "x'in y ile bölümü nedir?" soruları fonksiyon problemine dönüştürülebilir ve bu durumun tam tersi de olabilir. Birçok kombinatoryal problem doğal olarak karar problemleri olarak düşünülebilir. Bunlar için bir örneğin çözümleri , bir dizi mantıksal koşulla belirtilebilir. Kombinatoryal bir karar problemi örneği olarak, Grafik Renklendirme Problemi düşünülebilir. Örneğin bir grafik G ve bir dizi renk verildiğinde, G'nin düğümlerinin, bir kenara bağlanan iki düğümle asla aynı renk olmayacak şekilde bir renk ataması yapılmalıdır. Diğer kombinatoryal karar problemleri, belirli bir öneri formülü için tatmin edici doğruluk atamalarını bulmaktadır. Herhangi bir karar sorunu için iki değişken belirlenmeli: arama değişkeni, karar değişkeni. Bu değişkenler birbiriyle yakından ilişkilidir. Çünkü arama değişkenini çözen algoritmalar her zaman karar değişkenini çözmek için de kullanılabilir. İlginç bir şekilde, birçok kombinatoryal karar problemi için tersi de geçerlidir, yani algoritmalar bir sorunun karar değişkeni için ve gerçek çözümleri bulmak için de kullanılabilir.

2.3. Optimizasyon Problemleri

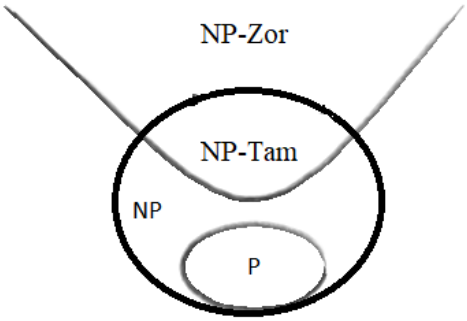
Bazı işlevleri bir kümeye göre en üst düzeye çıkarmak veya en aza indirmek, genellikle belirli bir durumda kullanılabilen bir dizi seçeneği temsil eder. Fonksiyon hangisinin "en iyi" olabileceğini belirlemek için farklı seçenekleri karşılaştırır. Yaygın kullanım örnekleri arasında minimum maliyet, maksimum kâr, minimum hata, optimum

tasarım, optimum yönetim vb gibi alanlar bulunmaktadır. Pratik olarak ilgili birçok kombinatoryal problem, karar problemlerinden ziyade optimizasyon problemleridir. Optimizasyon problemleri, çözümlerin ek olarak objektif bir fonksiyon tarafından değerlendirildiği ve hedefin optimal objektif fonksiyon değerlerine sahip çözümler bulmak olduğu karar problemlerinin, genelleştirilmesi olarak düşünülebilir. Bir çözüm adayının, objektif işlev değerine çözüm kalitesi de denir. Daha önce bahsedilen Grafik Renklendirme Problemi için, değişken sayıda renk kullanılan doğal bir optimizasyon değişkeni mevcuttur. Amaç bir grafik verildiğinde minimum renk sayısı kullanarak düğmelerinin renklerini belirlemektir. Herhangi bir kombinatoryal optimizasyon problemi, verilen objektif fonksiyonun minimize edilmesine veya maksimize edilmesine bağlı olarak bir minimizasyon problemi veya bir maksimizasyon problemi olarak ifade edilebilir. Her bir kombinatoryal optimizasyon problemi için iki değişken ayırt edilmelidir. Bunlar, arama değişkeni ve değerlendirme değişkenidir. Arama değişkeni bunlardan daha etkili olanıdır. Çünkü optimal bir çözüm bilgisi ile değerlendirme değişkeni önemsiz bir şekilde çözülebilir. Birçok kombinatoryal optimizasyon problemi, mantıksal koşulların yanı sıra objektif bir fonksiyona göre tanımlanır. Bu durumda, mantıksal koşulları karşılayan aday çözümlere uygulanabilir veya geçerli çözüm denir. Bunlar arasından optimal çözümler objektif işlev değerlerine göre ayırt edilebilir. Nesnel bir işleve ek olarak mantıksal koşulların kullanılması, genellikle bir kombinatoryal optimizasyon probleminin daha doğal formülasyonlarına yol açsa da, mantıksal koşulların her zaman nesnel işleve entegre edilebileceğine dikkat edilmelidir. Karar problemleri için birçok algoritma, ilgili optimizasyon problemlerine oldukça doğal bir şekilde uygulanabilir. Belirli karar problemlerinde iyi çalışan algoritmalar, karşılık gelen optimizasyon problemlerinin optimal veya neredeyse optimal çözümlerini bulmak için her zaman etkili değildir. Sonuç olarak, bu görev için farklı algoritmik yöntemlerin dikkate alınması gerekmektedir [6].

2.4. Hesaplamalı Karmaşıklık Teorisi (Computational complexity theory)

Karmaşıklık teorisi, teorik bilgisayar biliminde merkezi bir konudur. Hesaplanabilirlik teorisine doğrudan uygulamaları vardır ve karmaşıklığı test etmeye yardımcı olmak için soyut bir hesaplama modeli gibi hesaplama modellerini kullanır. Karmaşıklık

teorisi, bilgisayar bilimcilerinin problemleri karmaşıklık sınıflarıyla ilişkilendirmesine ve gruplandırmasına yardımcı olur. Bazen bir problem çözülebilirse, aynı karmaşıklık sınıfındaki diğer problemleri çözmenin bir yolunu açar. Karmaşıklık, genellikle belirli bir sorunu çözmek için ne kadar zaman gerektiğini ölçen bir sorunun zorluğunu belirlemeye yardımcı olur. Örneğin, bazı problemler polinom sayısı kadar çözülebilir iken, diğerleri girdi boyutuna göre üstel zaman alabilir [7]. Hesaplama karmaşıklığı teorisi, problemlerin doğasında izlenebilirlik veya takip olup olmamasına göre "kolay" veya "zor" olarak iki farklı şekilde sınıflandırılır. Bu sınıflandırma şeması iyi bilinen P ve NP sınıflarını içerir; "NP-Tam" ve "NP-Zor" terimleri NP sınıfına aittir. Karmaşıklık sınıfları, bilgisayar bilimcilerinin sorunlarını çözmek ve çözümleri doğrulamak için gereken zaman ve ne kadar alan gerektirdiklerine göre gruplanmasına yardımcı olur. Örneğin, karmaşıklık sınıfları bir Turing makinesinin A problemine karar vermesinin kaç adımı olacağını açıklamaya yardımcı olabilir. Karmaşıklık sınıflarını anlamak için Big-O notasyonunu sağlam bir şekilde kavramak gerekir. Bazı karmaşıklık sınıfları diğerlerinin bir alt kümesidir. Örneğin, deterministik polinom zamanında çözülebilen problemler sınıfı, P, belirsiz (nondeterministic) polinom zamanda NP'de çözülebilen problemler sınıfının bir alt kümesidir. Şekil 2.1 problemlerin sınıfları göstermektedir.



Şekil 2.1. Problemlerin Sınıfları; P=!NP

Yüzlerce karmaşıklık sınıfı vardır ve bu çalışmada en sık karşılaşılan sınıflardan bahsedilmektedir:

2.4.1. P Sınıfı

P sınıfı, "P" nin "polinom zamanı" anlamına geldiği bir polinom-zaman algoritmasının bulunduğu tanıma problemleri kümesi olarak tanımlanır. Bu nedenle P, resmen "kolay" kabul edilen problemleri içerir. Öte yandan birçok problem kaba kuvvet yaklaşımı kullanılarak çözülebilir, ancak bu genellikle üstel zaman alır. Bir problemin polinom zaman algoritması varsa, kaba kuvvet yöntemiyle de etkin bir şekilde çözülebilir. Aslında pratikte kullanılan algoritmaların çoğu polinom zaman algoritmalarıdır. Üstel zaman algoritmaları tipik olarak daha az kullanışlıdır ve mümkün olduğunca bu algoritmalardan kaçınılır.

2.4.2. NP Sınıfı

"NP" terimi "belirsiz (nondeterministic) polinom" anlamına gelir ve polinom zamanında NP'deki problemleri çözebilecek farklı varsayımsal bir hesaplama modelini ifade eder. Bu sınıf, polinom zamanında çözülebilen problemleri ve üstel zamanda çözülebilen problemleri de içerir. Bazen bir karar sorunun cevabını bulmanın etkili bir yolunu bilinmeyebilir, ancak cevap bilinir ve bir kanıt verilirse, geçerli bir kanıt olup olmadığını görmek için kanıtı kontrol ederek cevabın doğru olup olmadığını doğru bir şekilde doğrulayabiliriz . Bu karmaşıklık sınıfı NP'nin arkasındaki fikirdir.

2.4.3. NP-Tam Sınıfı

NP-Tam problemleri NP setindeki en zor problemlerdir. Aşağıdaki durumlarda X karar problemi, NP-Tam olarak tanımlanır:

1) X , NP problemidir. (NP-Tam problemler için verilen herhangi bir çözüm hızlı bir şekilde doğrulanabilir, ancak bu etkili bir çözüm değildir).

2) NP'deki her problem polinom zamanında X'e indirgenebilir.

Bir problem, yukarıda belirtilen ikinci özelliğine sahip ise, birinci özelliğe sahip olmadan NP-Zor sayılır. Bu nedenle, NP-Tam kümesi de NP-Zor kümesinin bir alt kümesidir.

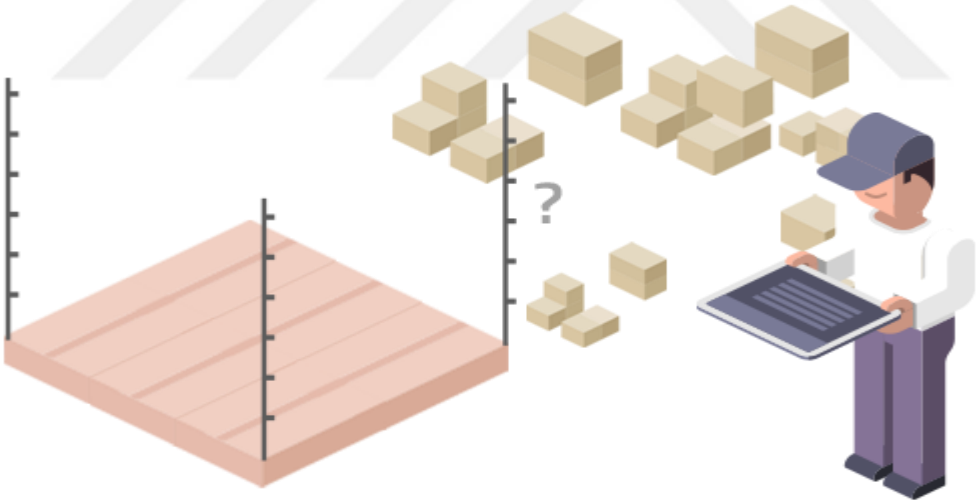
NP-Zor daki bir problem NP'deki en zor problemlerdir. NP'de herhangi bir X problemi için, X'den A'ya polinom zamanı indirgenebilir ise A problemi NP-Zor'dır.

Daha genel bir ifadeyle, problem B'yi çözmek için bir kullanılan algoritma, A problemini çözmek için de kullanılabiliyorsa, sonuç olarak bir A problemi B'ye indirgenebilir. NP-Zor problemleri, NP yada karar problemi olmak zorunda değildir. Örneğin, Durdurma(Stopping) Problemi NP-Zor bir problemdir, ancak NP problemi değildir. Bu çalışmada incelenen problem NP-Zor problemidir [4].

NP-Tam problemleri çok özeldir. Çünkü NP sınıfındaki herhangi bir problem polinom zamanında NP-Tam problemlerine dönüştürülebilir veya indirgenebilir. Bunun önemli bir sonucu, NP-Tam problemi polinom zamanında çözülebilir ise, herhangi bir NP problemide polinom zamanında çözülebilir [8-10].

2.5. Kutu Paketleme Problemi

KPP basitçe, bir yada birden fazla konteyner içine kutuların en iyi şekilde nasıl yerleştirilmesi gerektiği ile ilgilenir. Bu şekilde konteyner içinde minimum miktarda boş alan bırakarak, kullanılan konteyner sayısının mümkün olduğunca azalmasını sağlamak temel amaçtır.



Şekil 2.2. KPP Açıklayıcı Görsel

Kutu Paketleme Problemi klasik ve popüler bir optimizasyon problemidir. 1970'lerden bu yana KPP birçok araştırmacının ilgisini çekmiştir ve bu konu ile alakalı bazı başarılar elde edilmiştir [2-4]. Matematiksel olarak, KPP nin olası bir formülasyonu

aşağıdaki gibi bir tamsayı doğrusal programlama ile ifade edilebilir [11]. Kutuların sonlu kümesi I, her bir i ∈ I için $s(i) \in Z^+$ ölçüsü, pozitif bir konteyner kapasitesi B, pozitif bir değer K. Araştırma çoğunlukla, K'nin mümkün olan en küçük değerini bulmaya çalışan optimizasyon değişkeni ile ilgilenmektedir. En uygun çözüm minimum K değerini bulanmamızı sağlayan çözümdür. Bir dizi kutu için en uygun çözümün K değeri, OPT (I) veya yalnızca belli bir kümeye ait değilse OPT olarak gösterilir.

$$\min \mathbf{K} = \sum_{j=1}^{n} y_j \qquad (2.1)$$

$$Ona\ konu:$$

$$K \geq 1,$$

$$\sum_{i \in I}^{n} s(i)\, x_{ij} \leq B_{y_j}, \quad \forall j \in \{1,\ldots,n\}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in I$$

$$y_j \in \{0,1\}, \quad \forall j \in \{1,\ldots,n\}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I\ \forall j \in \{1,\ldots,n\}$$

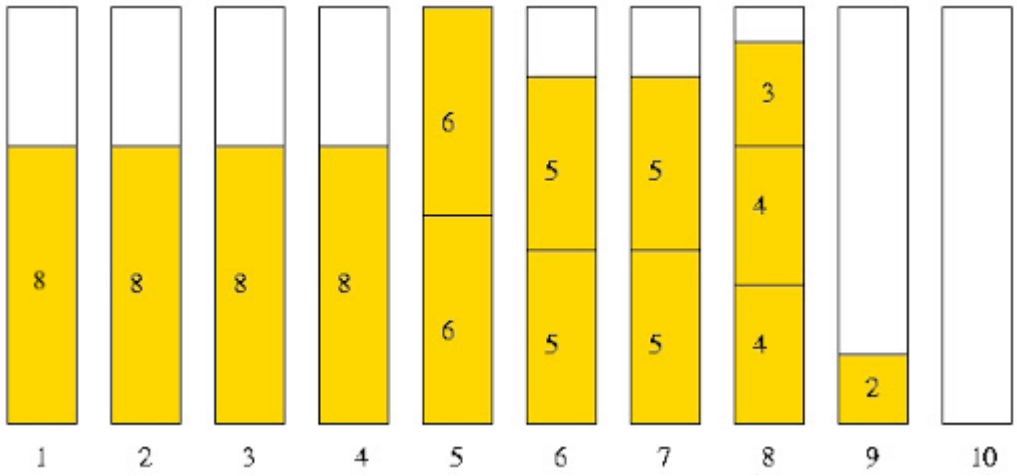
Konteyner j kullanılıyor ise y_j= 1 ve kutu i Konteyner j içinde yerleştiyse $\boldsymbol{x_{ij} = 1}$

2.5.1 Kutu Paketleme Problemi Tipleri

Kutu boyutuna göre bir gruplama yapılırsa karşımıza üç farklı tip çıkar. Bunlar, bir boyutlu kutu paketleme, iki boyutlu kutu paketme ve üç boyutlu kutu paketleme'dir.

2.5.1.1 Bir Boyutlu Kutu Paketleme

Bir Boyutlu Kutu Paketleme Problemi yalnızca bir değişken ile ilgilenir. Bir konteynere yerleşim yapılırken yalnızca bu değişkene bakılır . Örneğin yerleştirilecek öğenin bir ağırlığı ve yerleştirileceği konteynerin bir kapasitesi vardır. Bu problemde yalnızca öğenin ağırlığına bakılır. Bu, bir boyutlu kutu paketleme problem anılır. Şekil 2.3 de bir boyutlu KPP örneği gösterilmektedir.

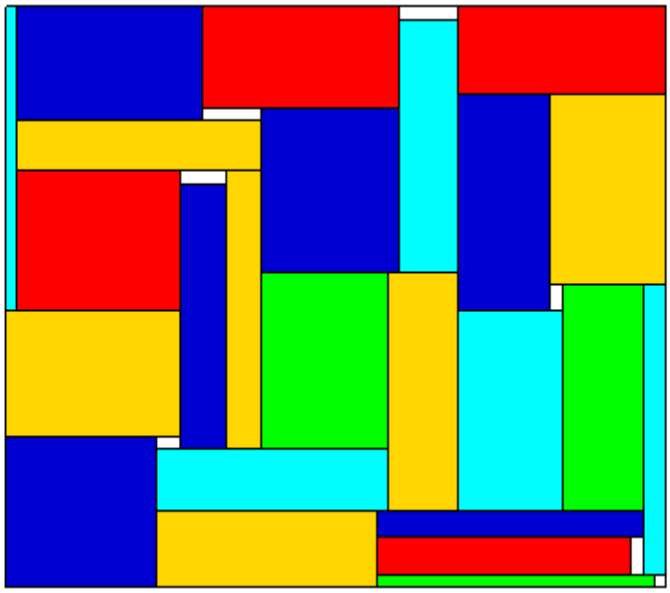


Şekil 2.3. Bir Boyutlu KPP Örneği

Basitlik açısından örneğin, 100 metrelik bir zincir en az miktarda atık ile 7 ve 9 metrelik parçalara bölmek isteniyor. En verimli bölme uzunluğuna 7 metre ile sahip oluyorken gereken 9 metrelik zincir miktarı ne kadardır gibi sorular düşünülmelidir. Bu tür sorunlara bir boyutlu problemler denir.

2.5.1.2 İki Boyutlu Kutu Paketleme

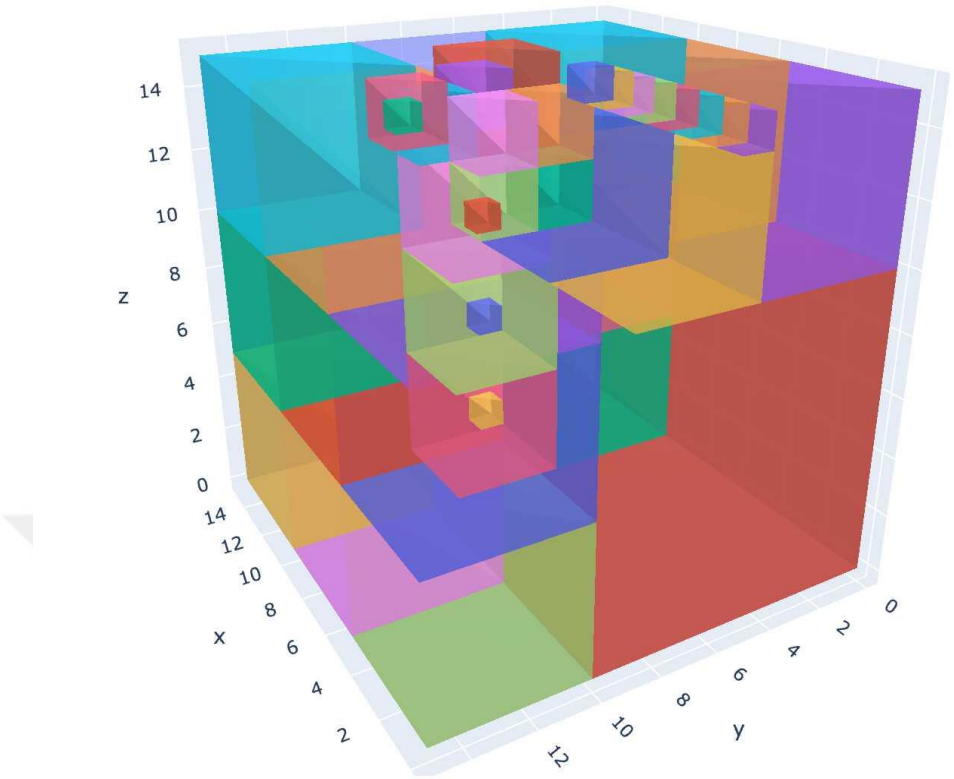
Bu tip Kutu Paketleme Problemi iki boyutlu öğelerle ilgilenir. Örneğin, kot üreten bir tekstil şirketinde, 10x10 metrekare bir kumaştan farklı boyutlarda pantolon kalıplarını minimum atıkla kesmek istenir. Bu problem iki boyutlu bir kutu paketleme problemidir (Boyutlar X ve Y). Benzer bir problem ise sınıflandırılmış reklamları en az miktarda atık içeren bir gazeteye yerleştirmedir. 2 boyutlu kutu paketleme problemi, gerçek hayatta ahşap veya cam kesimi ve Mobil WiMAX sistemleri gibi uygulamalarda da karşımıza çıkmaktadır. Şekil 2.4 İki Boyutlu KPP'ye bir örnek gösterilmiştir.



Şekil 2.4. İki Boyutlu KPP Örneği

2.5.1.3 Üç Boyutlu Kutu Paketleme

KPP problemlerinin en zor tipi olarak tanımlanır. Üç boyutlu bir konteynere yerleştirilen her şeklin x, y ve z gibi 3 farklı boyutu vardır. Örneğin, ev taşıma sırasında çıkan kutuları paketlemek, bir Üç boyutlu kutu paketleme problemi olarak düşünülebilir. Bu durum diğerler paketleme problemlerinden daha karmaşıktır. Çünkü kutular farklı boyutlarda ve şekillerdedir. Örneğin, küboid veya dikdörtgen prizma gibi kutular, birbirlerinden farklı şekillerdedirler ve bu şekiller girintili ve çıkıntılı şekillerde olabilirler. Tüm bu koşulların bir araya geldiği bir senaryo düşünülebilir. 2 ve 3 boyutlu ambalajlar, girintili olup olmamasına göre iki kısma ayrılır. Dışbükey nesneleri kutulamak nispeten daha basit bir problemdir. Ancak, içbükey neslerelde problem daha karmaşık hale gelir. Şekil 2.5.'de İki Boyutlu Kutu Paketleme Problemine örnek bir görsel verilmiştir.



Şekil 2.5. Üç Boyutlu KPP Örneği

Üç Boyutlu Kutu Paketleme Problemi, üretim ve nakliye planlamasında karşımıza sıklıkla çıkan çeşitli problemleri doğrudan modellemektedir. Bu nedenle konteyner, gemi yükleme, palet yükleme, uçak kargo yönetimi ve depo yönetimi gibi çeşitli alanlarda birçok pratik uygulamaya sahiptir. 1 ve 2 boyutlu KPP'lerde çok sayıda araştırma olmasına rağmen, 3 boyutlu KPP'lerde daha az araştırma nispeten daha düşüktür.

2.5.2. Kutu Paketleme Problemi Kriterleri

Problemi iyi anlamak için bu sorunun kriterlerini iyi anlamak gerekmektedir. Bu yüzden burada bu kriterler ile ilgili bazı kavramlar açıklanacaktır. Problemin çevrimdışı KPP tipi için eklenecek nesneler paketleme öncesinde bilinir. Bu durumda

yerleştirme sırası, sezgisel yöntemler kullanılarak yeniden düzenlenebilir. Uçak yük planlaması tipik bir çevrimdışı KPP dir. Çevrimiçi KPP tipi için kutular birer birer gelir. Bu nedenle tam giriş sırası bilinemediğinden diğer kutuları beklemeden bir bölüme yerleştirilmesi gerekir [12]. Çevrimiçi KPP'ye sabit disk bölümleme örnek olarak gösterilebilir. Buraya kadar olan bölümde birinci kriterden bahsedilmiştir. İkinci kriter ise bu kutuların 90 derece döndürülüp döndürülemeyeceğidir. Bu kriter KPP'nin çok önemli bir faktörüdür. Bu kritere göre çözüm uzayı ciddi bir şekilde büyütülür. KPP'de, üzerine hiçbir şey koyulmaması gereken kırılgan kutuları hesaba katmak, kutuların toplam ağırlığı, konteynerin tam kapasitesini aşmaması gibi başka kriterlere sahip olmalıdır.

2.5.3. Kutu Paketleme Problemi Çözüm Metodları

Kutu Paketleme Probleminin NP-Zor bir problem olması, optimum çözüm için kapsamlı bir araştırmanın hesaplamalı olarak sonuca varılamayan ve bu nedenle problem için bilinen gerçek hesaplamalı olarak uygulanabilir, optimal çözüm yönteminin bulunmadığını gösterir. Bu nedenle, bir çözüm elde etmek için başka araçlar bulunmalıdır.
Literatürdeki geçen yöntemler aşağıdaki gibi sıralanabilir.

2.5.3.1 Kesin Yaklaşımlar

KPP Problemi'ni daha küçük parametre değerleri ile çözmek ve kesin yaklaşımlar bulmak için çeşitli çalışmalar yapılmıştır. Downsland [13] bu problem için (PYP) kesin bir algoritma önermiştir. Daha sonra Bhattacharya [14]'da aynı sorun için daha derinlemesine bir başka yaklaşım önermiştir. Bunların dışında Tarnowski vd. [15] Palet Yükleme Problemi için yerleşimlerin giyotinle kesilebilir tarzda olması gerektiğiyle ilgili bir polinom algoritması sunmuştur. Ayrıca Martello vd. (2000) [16] İki Boyutlu Şerit Paketleme Problemi'ni (2BSPP) çözmek için dal ve sınır yaklaşımını önermişlerdir. Bu yaklaşımda kullanılan öğeler azalan yükseklikte sıraladıktan sonra bir indirgeme prosedürü, öğelerin bir alt kümesi için en uygun düzenlemeyi belirler ve bu şekilde örnek boyutu azaltılır. Bunun dışında, 2 boyutlu KPP Problemini çözmek için de bir yöntem önerdiler. Bu yöntem iki seviyeli, iç ve dış dal karar ağaçlarının

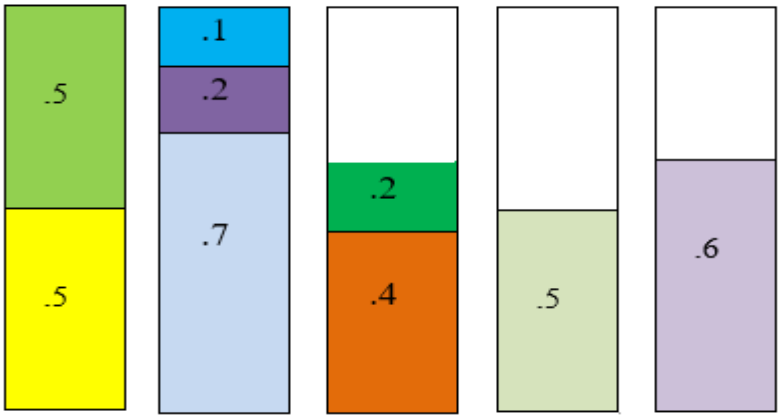
oluşturduğu bir dallanma şemasıdır. Burada kutular dış seviyede kutuların hangi konteynere gideceği belirlenir. İç seviyede ise belirlenen koyneynerler içindeki yerleşimin nasıl olacağına karar verilir.

2.5.3.2 Sezgisel Algoritmalar

KPP'lerin hesaplama karmaşıklığının yüksek olması, makul bir sürede kabul edilebilir sonuçlar elde etmek için çok sayıda sezgisel yaklaşıma yol açmıştır [3,18]. Sezgisel algoritmalarda bulunan çözümler çoğu zaman kesin olarak en uygun çözümü vermemektedir ve bulunan çözümün en uygun çözüm olup olmadığına da emin olunanamamaktadır. Ayrıca, optimum çözüme yakınlığı hesaplanamamakta veya garanti edilememektedir. Bununla birlikte, düşük hesaplama süreleri göz önüne alındığında, kesin yöntemlerle karşılaştırıldığında büyük kombinatoryal optimizasyon problemlerinin büyük örnekleri için etkili bir çözümleme stratejisi olarak kabul edilmektedir. Bu yöntemlere Yaklaşım Algoritmaları da denebilir. KPP için Yaklaşım Algoritmaları iki kategoride sınıflandırılabilir. Belirli bir sırayla öğeleri dikkate alan ve bunları kutuların içine tek tek yerleştiren yönteme çevrimiçi yöntem denir. Diğer sınıf çevrimdışı algoritmaları içerir. Öğeleri boyuta göre sıralayarak verilen öğe listesini değiştirir. Bu algoritmalar artık bu problemin çevrimiçi değişkeni için geçerli değildir. Sezgisel yaklaşımlarla ilgili literatür, KPP'de kesin yaklaşımlar kullananlardan çok daha fazladır. Ayrıca çoğunlukla meta-sezgisel yaklaşımlarla birlikte kullanılırlar. Aşağıda en sık kullanılan sezgisel algoritmalar listelenmektedir:

- **İlk Uygun Yer Algoritma (First-Fit algorithm)**

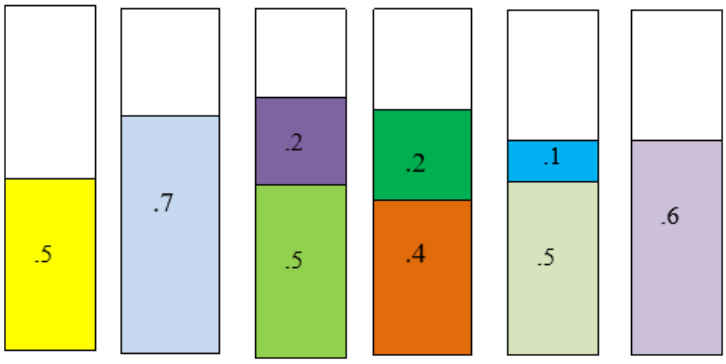
  Bu algoritma, konteynerleri(yada büyük kutu) sırayla tarar ve yeni kutuyu tutacak kadar büyük olanı ilk konteynere yerleştirir. Yeni bir konteyner yalnızca bir kutu önceki konteynerlere sığmadığında oluşturulur. Şekil 2.6'da İlk Uygun Yer Algoritma'ya örnek gösterilmiştir.
  I = (0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.1, 0.6)



Şekil 2.6. İlk Uygun Yer Algoritma Örneği

- **Sonraki Uygun Yer Algoritması (Next-Fit Algorithm)**

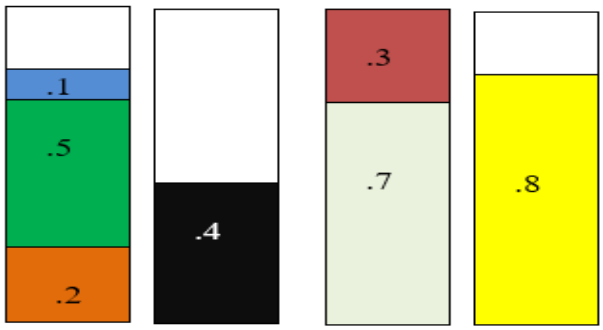
Kutu bir önceki kutu ile aynı konteynere sığarsa oraya konulur. Aksi takdirde, yeni bir konteyner açılır ve oraya konulur. Bu, son parçayı yerleştirdiğimiz konteynere bakıldıktan sonra daha önce kullandığımız bir konteynere geri dönülmeyeceği anlamına gelir. Örneğin, dördüncü konteynere sadece bir kutu koyulursa, 1-3 konteynerlere asla başka bir şey koyulmayacağı anlamına gelir. Sonraki kutu mümkünse dördüncü konteynere veya dördüncü konteynere sığmayacaksa beşinci konteynere gider. Şekil 2.7'de Sonraki Uygun Yer Algoritmasına bir örnek gösterilmiştir.



Şekil 2.7. Sonraki Uygun Yer Algoritma Örneği

● **En Uygun Yer Algoritma (Best-Fit Algorithm)**

Yeni kutu, en sıkı sığacağı yere yerleştirilir. Herhangi bir konteynere sığmıyorsa, yeni bir konteyner açılır. En Uygun Yer Algoritma kullanmanın bazı dezavantajları vardır. Daha yavaştır, çünkü her seferinde tüm listeyi tarar ve süreci karşılayabilecek en küçük boşluğu bulmaya çalışır. Tüm boyut ve işlem boyutu arasındaki farkın çok küçük olması nedeniyle, üretilen boşluklar herhangi bir işlemi yüklemek için kullanılamayacağı kadar küçük olacaktır ve bu nedenle işe yaramaz. Algoritmanın adının en uygun yer algoritma olmasına rağmen, hepsi arasında her zaman en iyi performans görtermez.
Şekil 2.8'de En Uygun Yer Algoritmasına örnek gösterilmiştir.
I = (0.2, 0.5, 0.4, 0.7, 0.1, 0.3, 0.8)



Şekil 2.8. 3 En Uygun Yer Algoritma Örneği.

● **En Kötü Uyan Algoritma(Worst-Fit Algorithm)**

Bu algoritma En İyi Yerleştir'e benzer. Kutuyu maksimum yük ile konteynere yerleştirmek yerine, minimum yük ile konteynere yerleştirilmesidir.

● **İlk Uygun Azalan Algoritma (First Fit Decreasing Algorithm)**

Bu algoritma İlk Yerleştir'e benzer çalışır. Bununla birlikte, kutuları yerleştirmeye başlamadan önce, boyutları azalan sırasına göre sıralanırlar. Bu

algoritmanın hesaplama karmaşası O(n log (n)) çalışma süresine sahip olacak şekilde uygulanabilir.

● **Değiştirilmiş İlk Uygun Azalan Algoritma (Modified First Fit Decreasing)**

Değiştirilmiş İlk Uygun Azalan Algoritma [19], boyutları yarım konteynerden büyük olanlar için, büyük, orta, küçük ve daha küçük olarak sınıflandırır.
Sonra beş aşamadan geçer.

1- Büyükten küçüğe sıralanan her büyük kutu için bir konteyner ayrılır.

2- Konteynerler üzerinden ileriye doğru ilerlenir. Her birinde, kalan orta kutulardan en küçüğü uymuyorsa, bu konteyner atlanır. Aksi takdirde, orta kutulardan en büyüğü yerleştirilir.

3- Orta boy bir madde içermeyen konteynerlerden geriye doğru devam edilir. Her birinde, küçük kutulardan kalan en küçük iki kutu uymuyorsa, bu konteyner atlanır. Aksi takdirde, kalan küçük kutulardan en küçüğünü ve en büyüğü konteynere yerleştirilir.

4- Tüm konteynerlerde ilerlenir. Herhangi bir boyut sınıfında kalan en küçük kutu uymuyorsa, bu konteyner atlanır. Aksi takdirde, en büyük kutu yerleştirilir ve bu konteynerde devam edilir.

5- Kalan kutular yeni konteynerlere paketlemek için Azalan İlk Yerleştir Algoritma kullanılır.

Bu algoritma ilk olarak 1985 yılında Johnson ve Garey [19] tarafından incelenmiştir.

● **Alt-Sol ve En Derin Alt-Sol Algoritmaları(Bottom Left and Deepest Bottom Left Algorithms)**

Alt Sol ve En Derin Alt Sol Algoritma'ları yerleştirmeye dayalı sezgisel yöntem olarak sayılmaktadır. Konumlandırma tabanlı sezgisel tarama, en kısa yol problemi literatüründeki en eskisidir ve ilk çalışmalarda en yaygın olanıdır. Oldukça esnektir ve sorunun en genel kısıtlamalarını dahil etmeyi sağlarlar. Konumlandırma tabanlı sezgisel yöntemlerin arkasındaki temel mekanizma, verilen bir kritere göre şerit üzerinde belirli bir parça için en uygun olan boş

alanın tanımlanmasıdır. Sol Alt sezgisel yönteminin arkasındaki sezgi, iki boyutlu nesnelerin her birinin kullanılabilir iki boyutlu alanın sol alt köşesine mümkün olduğunca yakın yerleştirilmesi gerektiğidir [20]. Kutular her seferinde bir tane olarak kabul edilir ve onların sırası genellikle algoritmanın etkinliğini etkileyebilir. Daha sonra her bir kutu konteynerin içerisine olabildiğince alçak ve en alt katmanda mümkün olduğunca sola yerleştirilir.

● En Derin Alt-Sol Doldur Algoritma(Deepest Bottom-Left With Fill)

En Derin Alt-Sol Doldur Algoritma, Sol Alt sezgisel taramasının üç boyutlu olana göre bir uzantısıdır. Bu da konteynerdeki en derin konumları dikkate alarak kutuyu alta ve sola mümkün olan en derin seviyede yerleştirir [20].
Bu tür yöntemlerin, elde edilen hacim yoğunluğu açısından performansı aşağıdakileri içeren bir dizi faktöre bağlıdır: Kutu sayısı, işlendikleri dizi ve tüm kutu yönelimleri [21]. En Derin Sol- Alt Algoritma'nın karmaşıklığı yüksek olduğundan, bu algoritmanın bilgisayar tarafından verimli bir şekilde uygulanması gerekmektedir. Şekil 2.9'da En Derin Alt-Sol Doldur Algoritma'nın sözde kodu verilmiştir.

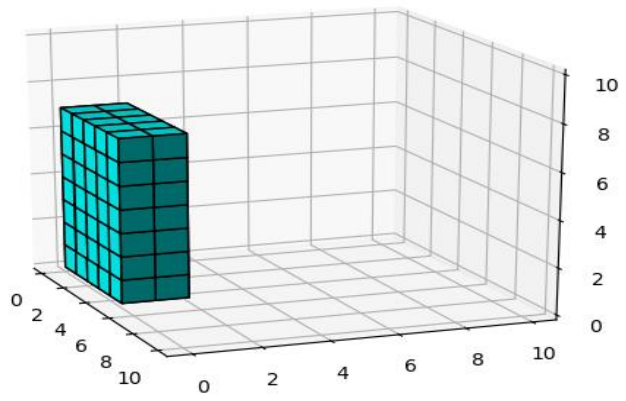
```
Tekrar et.
        Sıradaki kutuyu al.
        Tekrar et.
                Sıradaki konteynerde yeterince boş hacım olup olmadığını kontrol et.
                Yeteri kadar hacım var ise.
                        Sıradaki kutuyu en derin en aşağıda en sol müsait yere yerleştir.
        Sıradaki kutu yerleştirilene kadar yada konteynerler bitene kadar devam et.
        Sıradaki kutu yerleşmedi ise.
                Yeni boş bir konteyner ekle.
                Sıradaki kuyuyu yeni eklenen konteynerin ilk koordinatörüne yerleştir.
Kutular bitene kadar devam et.
```

Şekil 2.9. En Derin Alt-Sol Doldur Algoritma Sözde Kodu

Örnek:[(5, 2, 7), (1,2,7), (4, 2, 3), (4, 2, 5), (4, 2, 1)]

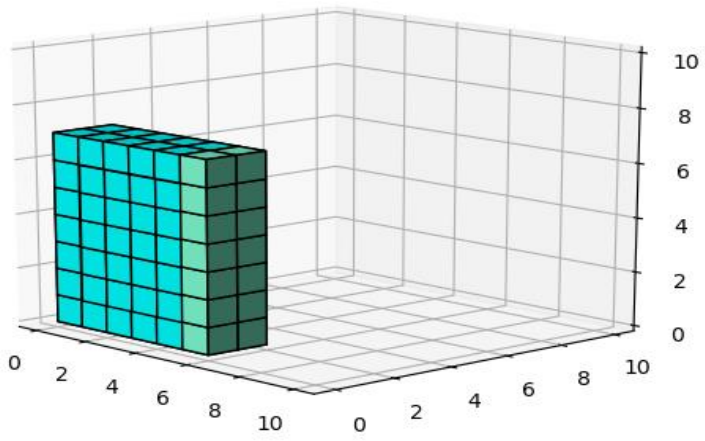
En Derin Alt-Sol Doldur Algoritma ile yukarıdaki örneğin kutularını(küboidlerini) yerleştirmek üzere aşağıdaki şekillerde (2.10, 2.11, 2.12, 2.13, 2.14) yerleştirme işlemi aşama aşama gösterilmektedir.

Aşama 1: (5, 2, 7) Küboidi konteynere yerleştirildi.



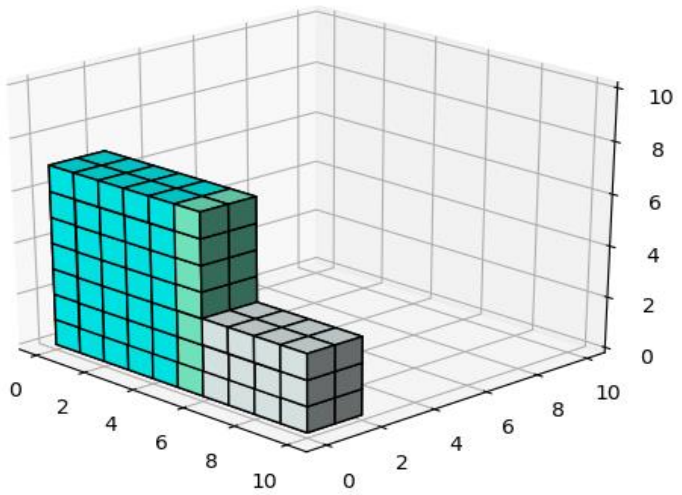
Şekil 2.10. En Derin Alt-Sol Doldur Algoritma Uygulanması 1. Aşama

Aşama 2: (5,2,7) Küboidi konteynere yerleştirildikten sonra, (1,2,7) Küboidi konteynere yerleştirildi.



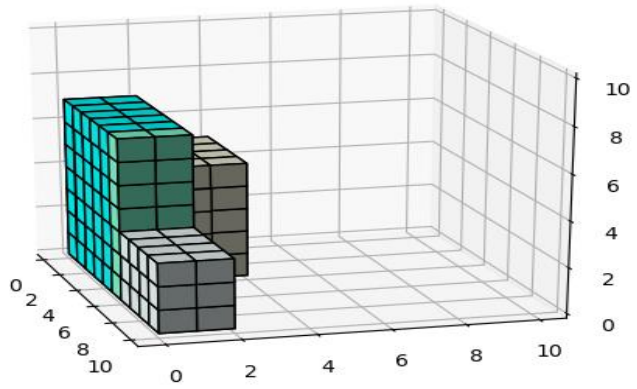
Şekil 2.11. En Derin Alt-Sol Doldur Algoritma Uygulanması 2. Aşama

Aşama 3: (1,2,7) Küboidi konteynere yerleştirildikten sonra, (4, 2, 3) Küboidi konteynere yerleştirildi.



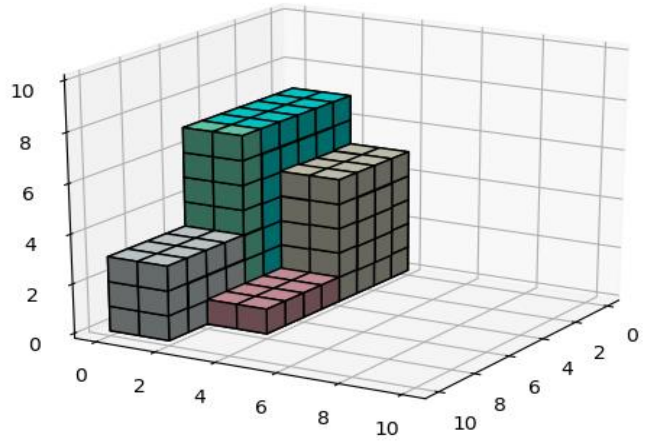
Şekil 2.12. En Derin Alt-Sol Doldur Algoritma Uygulanması 3. Aşama

Aşama 4: (4, 2, 3) Küboidi konteynere yerleştirildikten sonra konteynerin en derin alt sol tarafı dolduğu için (4, 2, 5) Küboidi konteynerin müsait olan en derin alt sol tarafına yerleştirildi.



Şekil 2.13. En Derin Alt-Sol Doldur Algoritma Uygulanması 4. Aşama

Aşama 5: (4, 2, 5) Küboidi konteynere yerleştikten sonra, (4, 2, 1) Küboidi de konteynere yerleştirildi.



Şekil 2.14. En Derin Alt-Sol Doldur Algoritma Uygulanması 5. Aşama

2.5.3.3. Meta-Sezgisel(Meta-Heuristic)

2.5.3.3.1. Meta-Sezgisel Tanımı

Farklı sezgisel çözümlerin herhangi biri tatmin edici sonuçlar üretmediği için Meta-Sezgisel algoritmalar incelenmiştir. Meta-sezgisel, sezgisel optimizasyon algoritmalarını geliştirmek için bir dizi yönerge veya strateji sağlayan, üst düzey problemden bağımsız algoritmik bir çerçevedir [22]. Bu algoritmalar gelişmiş ülkelerde sanayi devrimine neden olmuştur. Meta-Sezgisel Algoritma'ların dikkate değer örnekleri arasında Genetik / Evrimsel Algoritmalar, Tabu Arama, Benzetilmiş Tavlama, Değişken Komşuluk Arama, Karga Arama-Algoritması, Büyük Komşuluk Arama ve Karınca Kolonisi Optimizasyonu bulunur. Meta-Sezgisel bir çerçevede ifade edilen yönergelere göre, sezgisel bir optimizasyon algoritmasının probleme özgü bir uygulaması da Meta-Sezgisel olarak adlandırılır. Terim Glover [23] tarafından icat edilmiş ve Yunanca ön ek meta- (metá, ötesinde üst düzey anlamında) sezgisel (Yunan heuriskein veya euriskein'den aramak için) ile birleştirilmiştir. Meta-Sezgiler, bilim camiası tarafından, çoğu zaman dal, sınır ve dinamik programlama gibi daha geleneksel tam karma-tamsayılı optimizasyon yöntemlerinin yerine uygulanabilir üstün bir alternatif olarak gösterilmiştir. Özellikle karmaşık problemler veya büyük problem vakaları için, Meta-Sezgisel çözüm genellikle çözüm kalitesi ve hesaplama süresi arasında daha iyi bir denge sunar. Üstelik Meta-Sezgisel yöntemler kesin yöntemlerden iki önemli yönüyle daha esnektir. Birincisi, Meta-Sezgisel çerçeveler genel terimlerle tanımlandığından Meta-Sezgisel Algoritmalar, farklı problemler ve farklı durumlar arasında büyük ölçüde değişebilen, beklenen çözüm kalitesi ve izin verilen hesaplama süresi açısından çoğu gerçek yaşam optimizasyon probleminin ihtiyaçlarına uyacak şekilde uyarlanabilir. İkincisi, Meta-Sezgiselleştirme optimizasyon probleminin formülasyonuna ilişkin herhangi bir talepte bulunmaz. Bununla birlikte, bu esneklik iyi bir performans elde etmek için probleme özel adaptasyon gerektirir ve bu da ciddi maliyete sebep olur.

2.5.3.3.2. Sezgisel ve Meta-Sezgisel Karşılaştırılması

Meta-Sezgisel algoritmalarını daha net bir şekilde anlayabilmek için Sezgisel ve Meta-Sezgisel arasındaki farkı netleştirmek gerekmektedir. Sezgisel algoritmalar probleme bağımlıdır. Yani Sezgisel Algoritma belirli bir problem için uygulanır. Meta-sezgisel ise çok çeşitli sorunlara uygulanabilen, problemden bağımsız tekniklerdir. Bir Meta-Sezgisel, uygulanacağı problem hakkında hiçbir şey bilmez, işlevi kara kutular gibi ele alınabilir. Sezgisel, bir çözümün belirli bir soruna yeterince iyi bir çözüm bulmak için probleme bağımlı bilgilerden faydalandığını söylerken, meta-sezgisel yöntemler de tasarım modelleri gibi çok çeşitli sorunlara uygulanabilecek genel algoritmik fikirlerdir.

2.5.3.3.3. Genetik Algoritma

Genetik Algoritmalar, evrimsel algoritmaların büyük kısmına ait olan, uyarlanabilir sezgisel arama algoritmalarıdır. Genetik Algoritmalar doğal seleksiyon ve genetik fikirlerine dayanmaktadır. Bunlar, aramayı çözüm alanında daha iyi performans bölgesine yönlendirmek için geçmiş verilerle sağlanan rastgele aramanın akıllıca kullanılmasıdır. Genellikle optimizasyon sorunları ve arama sorunları için yüksek kaliteli çözümler üretmek için kullanılırlar. Genetik Algoritmalar ilk kez 1975'te Holland tarafından tanıtılmış [24] ve birçok araştırmacı tarafından incelenmiştir.
Genetik algoritmalar doğal seleksiyon sürecini simüle eder. Bu da çevrelerindeki değişikliklere uyum sağlayabilen türlerin hayatta kalabilmesini, üremesini ve gelecek nesillere geçmesini sağlar. Basit bir deyişle, bir problemi çözmek için ardışık nesil bireyler arasında "en uygun olanın hayatta kalmasını" simüle eder. Her nesil bir birey popülasyonundan oluşur ve her birey arama alanında bir noktayı ve olası çözümü temsil eder. Her birey bir karakter / tamsayı / kayan nokta / bit dizisi olarak temsil edilir. Bu dizi kromozoma benzer. Her birey, belirli bir sorun için arama alanında bir çözümü temsil eder. Her birey, bileşenlerin sonlu bir uzunluk vektörü olarak kodlanır. Bu değişken bileşenler genlere benzer. Böylece bir kromozom (bireysel) birkaç genden (değişken bileşenler) oluşur.

- **Genetik Algoritmaların Temelleri:**

Genetik algoritmalar, genetik yapıya ve popülasyonun kromozomunun davranışına sahip bir analojiye dayanmaktadır. Bu benzetmeye dayalı Genetik Algoritmaların temeli aşağıdaki gibidir:

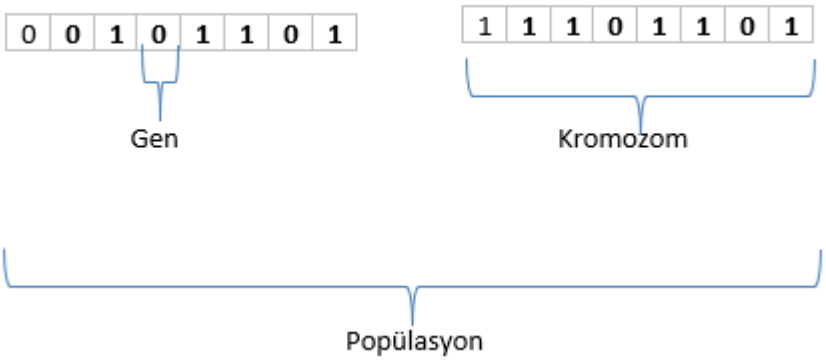
1- Nüfus içindeki birey, kaynaklar ve eş için yarışır.

2- Başarılı olan (en uygun) bireyler, diğerlerinden daha fazla döl yaratmak için çiftleşirler.

3- En uygun ebeveynden elde edilen genler nesil boyunca yayılır. Yani bazen ebeveynler her iki ebeveynden daha iyi yavrular yaratırlar.

4- Böylece birbirini takip eden her nesil çevreleri için daha uygundur.

Şekil 2.15'de Gen, Kromozom ve Popülasyon'a örnek gösterilmiştir.



Şekil 2.15. Gen, Kromozom ve Popülasyon Hakkında Açıklayıcı Resim

● Uygunluk Puanı(Fitness)

Her bireye, bireyin "rekabet etme" yeteneğini gösteren bir Fitness Puanı verilir. Optimal kondisyon skoruna (veya optimal seviyeye yakın) sahip olan birey aranır. Genetik algoritmalar, n bireylerin popülasyonunu (kromozom / çözeltiler) ve fitness skorlarını korurlar. Daha iyi fitness skorlarına sahip kişilere, diğerlerinden daha fazla üreme şansı verilir. Daha iyi fitness skorları olan bireyler şimdiki nesilden seçilir ve ebeveynlerin kromozomlarını birleştirerek çok daha iyi bireyler üretilmeye çalışılır.Nüfus büyüklüğü sabittir. Bu nedenle bazı bireyler ölür ve yerine yeni gelenler gelir. En sonunda eski nüfusun tüm çiftleşme fırsatı tükendiğinde yeni nesil oluşturur. Bu şekilde arka arkaya nesiller oluştukça daha iyi çözümlerin geleceği umulmaktadır. Her yeni

jenerasyon, önceki nesillerin bireylerinden (çözüm) ortalama olarak “daha iyi genlere” sahiptir. Böylece her yeni neslin önceki nesillere göre daha iyi “kısmi çözümleri” vardır. Üretilen yavrular, önceki popülasyonlar tarafından üretilen yavrulardan önemli bir fark göstermediğinde, popülasyon yakınsaktır.

● Genetik Algoritma Operatörleri

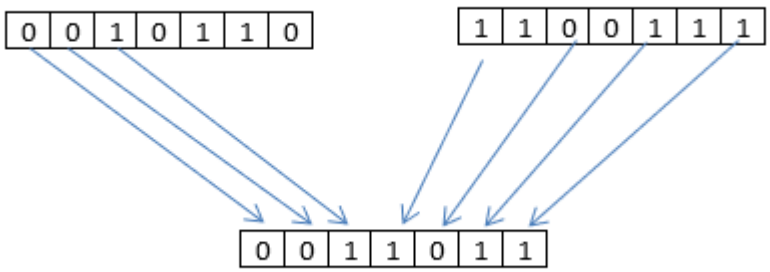
İlk nesil oluşturulduktan sonra algoritma, aşağıdaki operatörleri kullanarak nesli geliştirir:

A) Seçim Operatörü

İyi uygunluk (fitness) skorlarına sahip bireylere tercih vermek ve birbirini takip eden nesillere gen geçirmelerine izin vermektir.

B) Çaprazlama Operatörü

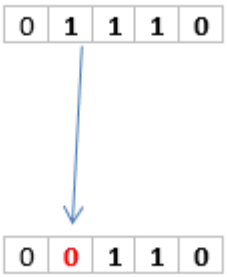
Bu, bireyler arasındaki çiftleşmeyi temsil eder. Seçim operatörü kullanılarak iki kişi seçilir ve çapraz siteler rastgele seçilir. Daha sonra bu geçiş sahalarındaki genler değiştirilir ve böylece tamamen yeni bir birey yaratılır. Şekil 2.16’da Çaprazlama İşlemine örnek verilmiştir.



Şekil 2.16. Çaprazlama İşlemi Görseli

C) Mutasyon Operatörü

Anahtar fikir, erken yakınsamayı önlemek ve popülasyondaki çeşitliliği korumak için bireye rastgele genler eklemektir. Şekil 2.17’de Mutasyon İşlemine örnek verilmiştir.



Şekil 2.17. Mutasyon İşlemi Görseli

- **Genetik Algoritma Sözde Kodu**

Şekil 2.18'de Genetik Algoritma Sözde Kodu gösterilmiştir.

```
İlk nesil çözümleri rastgele olarak yarat
Nesli fitness fonksiyonuna göre değerlendirir
Kromozomları fitness skoruna göre sırala
Nesil sayısı maksimum nesil sayısından küçüktür ise devam et(While)
        Elitizm kromozomlarını önceki nesillerden al ve Pi'ye koy
        Önceki nesil çapraz kromozomlar yapmak ve Pi'ye koymak
        Önceki nesil popülasyondan mutasyon kromozomları yapmak ve Pi'ye koy
        (Pi)'yi değerlendir
        (Pi)'yi sırala
        Durma koşulu doğruysa
                Çık
While döngüsü bitişi
```

Şekil 2.18. Genetik Algoritma Sözde Kodu Görseli

2.5.3.3.4. Karga Algoritma

● **Karga Algoritma Tanıtımı**

Karga Arama Algoritması, kargaların akıllı grup davranışlarına dayanan yeni bir meta-sezgisel yöntemdir. Karga Arama Algoritması 2016'da A.Askarzadeh tarafından önerilmiştir. Karga Arama Algoritması, kargaların ek beslenmesini özel olarak depolaması ve ona ihtiyacı olduğunda geri almasından esinlenmektedir [5]. Kargaların diğer kuşları izledikleri, diğer kuşların yiyeceklerini nerede sakladıklarını gözlemledikleri ve sahibi ayrıldıktan sonra çaldığı bilinmektedir. Eğer bir karga hırsızlık yapmışsa, saklanan yerleri gelecekteki bir kurban olmaktan kaçınmak için taşımak gibi ekstra önlemler alacaktır. Aslında bir hırsızın davranışını tahmin etmek için bir hırsız olma deneyimlerini kullanırlar ve çalınmaya karşı korumak için önbelleklerini kullanarak en güvenli rotayı belirleyebilirler [25]. Özet olarak kargalar sürü şeklinde yaşar ve sakladıkları yemeğin yerlerini ezberlerler. Kargalar hırsızlık yapmak için birbirlerini takip ederler ve önbelleklerini olası bir hırsızlığa karşı korurlar. "d"boyutlu bir ortam olduğu varsayıldığında, karga sayısına N, arama alanındaki karga i'nin t tekrarlamasında konumu bir vektörle belirtilir.

$k^{i,t}$ (i = 1, 2, …,N; t = 1, 2, ... t_{maks}) $k^{i,t}$= [$k_1^{i,t}$, $k_2^{i,t}$, . . . , $k_d^{i,t}$] ve t_{maks} maksimum tekrarlama sayısıdır. Her karga, bir hafızaya sahiptir. Bu hafiza sakladığı yemeğin yerini içermektedir. Karga i'nin hafızası $h^{i,t}$ ile gösterilir. $h^{i,t}$, şimdiye kadar elde edilen en iyi saklanma yeridir. Her karganın hafızasına en iyi deneyiminin konumu ezberlenmiştir. Kargalar çevrede hareket eder ve daha iyi yiyecek kaynakları arar. j karganın yemek sakladığı yeri ($h^{j,t}$) ziyaret etmek istediği varsayılır ise bu tekrarlamada karga i, karga j'nin yemek sakladığı yere yaklaşmak için karga j'yi takip etmeye karar verir.

Bu halde, iki durum yaşanabilir [5, 26]:

➞ Durum 1: Karga j, karga i'nin onu takip ettiğini bilmemektedir. Matematiksel olarak $r_j \geq AP^{j,t}$. Sonuç olarak, karga i karga j'nin yemek sakladığı yere yaklaşacak ve bu durumda, karga i'nin yeni pozisyonu aşağıdaki gibi elde edilir:

$$k^{i,t+1} = k^{i,t} + r_i \times fl^{i,t} \times \left(h^{j,t} - k^{i,t}\right) \qquad r_j \geq AP^{j,t} \qquad (2.2)$$

Burada r_i, 0 ile 1 arasında eşit dağılımlı rasgele bir sayıdır; $fl^{i,t}$, t tekrarında karga i'nin uçuş uzunluğunu belirtir. Küçük fl değerleri yerel aramaya yol açar ($k^{i,t}$ çevresi) ve büyük değerler global arama ile sonuçlanır ($k^{i,t}$'den uzak). $AP^{j,t}$, karga j'nin tekrarlamadaki farkındalık olasılığını gösterir.

➞ Durum 2: Karga j, karga i'nin takip ettiğini bilmektedir. Sonuç olarak karga j önbelleğini çalınmanın önlenmesi için arama alanından başka bir konuma giderek karga i'yi kandırır.

$$k^{i,t+1} = Rastgele\ Konum \qquad r_j < AP^{j,t} \qquad (2.3)$$

Karga Arama Algoritması'nda yoğunlaşma ve çeşitlendirme, esas olarak Farkındalık Olasılığı parametresi tarafından kontrol edilir. Sonuç olarak, küçük Farkındalık Olasılığı değerlerinin kullanılması yoğunlaşmayı artırır, büyük Farkındalık Olasılığı değerlerinin kullanılması çeşitliliği artırır.

● Karga Algoritma Sözde Kodu

Karga Algoritma aşamaları aşağıdaki gibi özetlenebilir:

1- Karga Algoritmanın parametreleri ve değişkenleri sağlanır, kısıtlamaları da tanımlanır. Karga Arama Algoritması'nın değişken parametreleri sürü boyutu (N), maksimum yineleme sayısı (t_{maks}), uçuş uzunluğu (fl) ve farkındalık olasılığı (AP) olarak verilebilir.

2- Kargaların konumu ve hafızası ayarlanır. N karga, sürünün üyeleri olarak rastgele bir d-boyutlu arama alanına yerleştirilir. Her karga , sorunun uygulanabilir bir çözümünü belirtir ve d , karar değişkenlerinin sayısıdır. Kargaların tecrübesi olmadığı için ilk seferde hafızası yerleştiği konuma eşit olur.

3- Çözüm kalitesi fonksiyonu değerlendirilir. Her karga için pozisyonunun kalitesi, karar değişken değerlerini objektif fonksiyona ekleyerek hesaplanır.

4- Kargalar arama alanında şu şekilde yeni bir pozisyon oluştururlar. Karganın yeni bir pozisyon oluşturmak istediği varsayılır. Bu amaç için bu karga, sürü

kargalarından birini (örneğin, karga j) rastgele seçer ve bu karga tarafından gizlenen yiyeceklerin konumunu (hj) keşfetmek için izler. Karga i'nin yeni pozisyonu bir önceki sağlanan denklemler ile belirlenir. Bu süreç tüm kargalar için tekrarlanır.

5- Her karganın yeni pozisyonunun fizibilitesi kontrol edilir. Eğer bir karganın yeni pozisyonu mümkün ise karga pozisyonunu günceller. Aksi takdirde, karga mevcut pozisyonda kalır ve üretilen yeni pozisyona hareket etmez. Bu aşama problemin şartlarına ve kısaltmalarına göre değişir.

6- Her bir karganın yeni pozisyonu için uygunluk/Çözüm kalitesi fonksiyonu değeri hesaplanır.

7- Yeni pozisyonun kalitesi, eski pozisyonun kalitesinden daha iyi ise takip eden karganın hafızası aşağıdaki gibi güncellenir:

$$h^{i,t+1} = k^{i,t+1} \tag{2.4}$$

8- (4-7) aşamaları t_{maks}'ya gelene kadar tekrarlanır.

Şekil 2.19'da Karga Arama Algoritması'nın sözde kodu gösterilmektedir.

Arama alanında N Karga sürüsünün konumunu rastgele yarat.

Kargaların konumunu değerlendir.

Her karganın hafızasını ilk konumu olarak ayarla.

Bir farkındalık olasılığı(AP) tanımla

t t_{maks} 'tan daha küçüktür ise tekrarla(While döngüsü):

(for döngüsü) i = 1: N (sürünün tüm N kargaları)

Rastgele takip edilecek kargalardan birini seç (örneğin j)

$r_j \geq AP^{j,t}$ ise

$$k^{i,t+1} = k^{i,t} + r_i \times fl^{i,t} \times (h^{j,t} - k^{i,t})$$

Yok ise

$k^{i,t+1}$ = arama alanında rastgele konum

(for sona eriyor)

Yeni pozisyonların fizibilitesini kontrol et

Kargaların yeni pozisyonunu değerlendir

Kargaların hafızasını güncelle

Tekrarlama (While döngüsü sona eriyor)

Şekil 2.19. Karga Algoritma Sözde Kodu

2.5.3.4. Hibrit Çözümler

Hibrit algoritma, aynı sorunu çözen iki veya daha fazla algoritmayı birleştiren bir algoritmadır. Ya birini seçer (verilere bağlı olarak) ya da algoritma boyunca bunlar

arasında geçiş yapar. Bu genellikle her birinin istenen özelliklerini birleştirmek için yapılır. Böylece genel algoritma tekli bileşenlerden daha iyidir. Hibrit algoritmalar, algoritmaların arama yeteneğinin geliştirilmesinde önemli bir rol oynar. Hibridizasyon, her algoritmanın avantajlarını birleştirerek hibrit bir algoritma oluşturmayı ve eşzamanlı olarak önemli dezavantajları en aza indirmeyi amaçlamaktadır. Genel olarak, hibridizasyonun sonucunda hesaplama hızı veya doğruluk açısından bazı iyileştirmeler yapabilir. Hibrit, algoritmalar birleşimi olabilir veya yöntemler birleşimi de olabilir. Bu çalışmada iki tane Meta-Sezgisel algoritmanın karışımı ve Sezgisel Algoritma kullanılmıştır.

2.5.3.5. Derin Takviye Öğrenimi

Son zamanlarda nispeten iyi sonuçlara ulaşabilen Derin Takviye Öğrenimini kullanan birçok çalışma yapılmıştır [4,27,28]. Derin Takviye Öğrenimi hala bu alanda çok yeni bir yöntem olarak kabul edilmektedir ve küçük örneklerde kullanılmıştır. Ancak incelemeye ve denemeye değmektedir.

2.6. Data Temsili

Bir optimizasyon problemini çözmek için Meta-Sezgisel algoritmalar uygulandığında, çözümün temsili önemli bir rol oynamaktadır. İyi bir temsil şeması, Meta-Sezgisel'in yüksek kaliteli çözümleri kolayca elde etmesine yardımcı olabilirken, yanlış bir temsil Meta-Sezgisel'in uygulanabilir çözümler elde etmesini zorlaştırabilir. Bazı Meta-Sezgisel algoritmalarda, bireyler Sırt Çantası Problemi için 0-1 kodlama şeması [29] ve Gezici Satıcı Problemi (GSP) için sipariş kodlama şeması [30] gibi problem çözümü ile temsil edilebilir. Bu temsil şemasının çözümü ile bireyin temsili arasındaki çeviriyi gerçekleştirmek çok kolaydır. Bununla birlikte bu düz gösterim şemasının bazı karmaşık optimizasyon problemleri için uygulanması imkansızdır. Bu çalışma 3 boyutlu dataya dayadığı için sadece 3 boyutlu data temsili ile ilgilenmiştir. 3 boyutlu veriler, nesnenin geometrisi hakkında zengin bilgiler sağlar. Bu nedenle bunların yeterli temsili büyük önem taşır. 3 boyutlu algılama teknolojilerindeki büyük ilerlemeler nedeniyle mevcut 3 boyutlu veri miktarı artmıştır. Bu veriler yukarıda

tartışıldığı gibi yapısal özelliklerine göre farklı şekillerde gelir. Öklidyen ve Öklidyen Olmayan veri aileleri olarak sınıflandırılabilirler.

2.6.1. Öklidyen Verileri

Öklid verileri 'n-boyutlu doğrusal uzayda, örneğin görüntü dosyaları olarak çizilmek için mantıklı bir şekilde modellenen verilerdir. Volumetrik verilerdeki voksel boyutları, RGB-D verilerindeki kanal türleri ve panoramik projeksiyonlarda menşe yeri gibi global parametreleme yapabilen veri yapıları Öklid veri ailesine aittir. Bunlar ayrıca aşağıdaki kategorilere ayrılabilir.

2.6.1.1. Tanımlayıcılar

Şekil tanımlayıcılar işleme, hesaplama ve karşılaştırma işlemlerine yardımcı olan 3 boyutlu nesnelerin basit gösterimleridir. Bir nesne için geometri, topoloji, doku ve yüzey gibi çeşitli özellikleri veya kombinasyonlarını temsil ederler. 3 boyutlu tanımlayıcılar ayrıca yerel ve küresel tanımlayıcılar olarak sınıflandırılır. Çeşitli el işi düşük seviyeli tanımlayıcı bir nesnenin küresel ayrımcı özelliklerini yakalayamaz ve bu nedenle genellikle hiyerarşik ayrımcı özellikleri yakalayabildiği bilinen DL yöntemleriyle birleştirilir. Denetimli yöntemler verilerin hiyerarşik soyutlamalarını sağladığından ve düşük seviye tanımlayıcıların kendi içinde bir şeklin soyutlamaları olduğundan, istenmeyen soyutlamalara yol açacağı için çoğunlukla denetimsiz DL yöntemleri bu görevler için daha uygundur.

2.6.1.2. 3 Boyutlu Veri Projeksiyonları

3 boyutlu veriler 2 boyutlu alana yansıtılır. Projeksiyonlar genellikle 3 boyutlu özelliklerin yalnızca bazılarını yakalayabilir. Korunan özellik türleri, projeksiyon tipine bağlıdır. Verileri normalleştirmek (poz normalizasyonu gibi) ve bir projeksiyon yapmak için bir ön işlem adımı kullanılır. Bundan sonra özellikleri çıkarmak için 2 boyutlu Evrişimli Sinir Ağları (Convolutional Neural Networks) kullanılır. Bu yöntemler basittir ve global tanımlayıcı tabanlı yöntemlere göre daha iyi sonuçlar vermiştir. Ancak bazı geometrik özellikler projeksiyonlar nedeniyle kaybedilmiştir.

2.6.1.3. Hacimsel Veriler

3 boyutlu verileri 3 boyutlu bir şebeke kullanarak temsil ediyoruz. Vokseller, 3 boyutlu alanda nesnenin dağılımını tanımlamak için kullanılır. Bu gösterim her bir vokselin temsil eden, görünür, tıkanmış veya kendi kendine tıkanmış bir değer atayarak bakış açısı bilgilerini de kodlayabilir. Bu soruna bir çözüm, octree tabanlı değişen boyutlu voksellerdir. Ancak bu temsillerin her ikisi de yüzeyin şekli ve düzgünlüğü ile ilgili özellikleri koruyamaz. Genel olarak 3 boyutlu tabloda özellikleri çıkarmak için 3 boyutlu-ESA'lar kullanılır. Ancak 3 boyutlu-ESA'lar hesaplama açısından çok pahalıdır. Toplu yakınlaştırma için yöntemler Açık Ağ(LightNet) [31]'de hızlı yakınsama için kullanılmıştır. Bu çalışmada hem kutu için hem de konteyner için bu yöntem kullanılmıştır.

2.6.1.4. Çoklu Görüntüleme Verileri

3 boyutlu bir nesne, her biri farklı bir görüntüleme noktasını temsil eden birden fazla 2 boyutlu görüntünün bir kombinasyonu olarak temsil edilir. Bu temsil gürültü, oklüzyon ve eksikliğe karşı dayanıklıdır. Önemli bir soru da, temsiller için kaç görüşün yeterli olduğudur. Çok azı aşırı sığmaya ve çok fazla yedekli verilere yol açacaktır. Çok görüntülü veri gösterimlerinin, başta nesne sınıflandırması olmak üzere çeşitli görevlerdeki hacimsel verilerden daha iyi performans gösterdiği gözlenmiştir. Çoklu görüntüleme verileri, çeşitli yerleşik 2 boyutlu DL paradigmalarını çok etkili bir şekilde kullanabilir. Bu nedenle mimarilerimizi 3 boyutlu veriler için uyarlamaya gerek yoktur. Etkili bir DL mimarisi, bir görünüm havuzu katmanı kavramını ortaya çıkaran MVCNN [32]'dır.

2.6.2. Öklidyen Olmayan Veriler

Bu yöntem de Geometrik veriye dayalı bir yöntemdir. Temel olarak iki alt kategoriye ayrılır: nokta bulutları ve 3 boyutlu kafesler ve grafikler. Bu temsil önemlidir. Çünkü öklidyen olmayan veriler, 1 boyutlu veya 2 boyutlu gösterime göre daha hassas olan daha karmaşık öğeleri ve kavramları temsil edebilir.

2.6.2.1. Nokta Bulutları

Nokta bulutları, 3 boyutlu nesnelerin geometrisini sırasız bir nokta kümesi olarak temsil eder. Kombinatoryal düzensizliklerden kaçınan basit yapılardır. Bu nedenle öğrenmeleri daha kolaydır. Nokta bulutlarının yakalanması çok kolay olmasına rağmen yüzey bilgileri hakkında belirsizlik gösterir.

2.6.2.2. 3 Boyutlu Kafesler Ve Grafikler

3 boyutlu alanda bir dizi köşe noktası ve her bir tepe noktasının diğerine nasıl bağlandığını açıklayan bir bağlantı listesi olarak temsil edilirler. Kafeslerin düzensiz yapısı nedeniyle DL yöntemlerinin bu temsillere genişletilmesi zor bir iştir. Bununla birlikte ağlar aynı zamanda köşeler ve aralarındaki bağlantılı kenarlar olarak, grafik veri yapıları şeklinde sunulabilir. Dolayısıyla grafikler için DL yöntemleri ağlara uygulanabilir. Grafikler üzerindeki bu evrişim yöntemleri esas olarak spektral ve uzaysal filtreleme yöntemlerine ayrılmıştır.

2.7. Literatür Taraması

Kutu Paketleme Problemi akademik camiada çok çalışılmıştır. Ancak karmaşıklığı, varyasyonları, kısıtlamaları ve endüstrinin birçok sektörünün odak noktası olduğu için akademisyenler tarafından yeni yöntemler ve algoritmalar kullanılarak sonuçları iyileştirilmeye devam edilmektedir. Meta-Sezgisel yöntemler, Kombinatoryal Problemleri çözmekle ünlü olduğu gibi KPP [33, 34], GSP [35, 36] ve Araç Yönlendirme Problemi [37, 38] gibi diğer sorunları çözmek için de kullanılmıştır. Gerçek olarak Meta-Sezgisel yöntemler Kombinatoryal Problemlerin çözümünde başarılı olmuştur. Abdel-Basset vd. 1 boyutlu KPP için doğadan ilham alan Meta-Sezgisel Algoritmayı geliştirdiler [39]. Lodia vd. Kısmi Numaralandırma Algoritmaları kullanarak giyotin kısıtlamaları ile 2 boyutlu KPP'yi çözdüler [40]. En zorlu 3 boyutlu KPP tipi için son zamanlarda birçok çözüm önerilmiştir. Bunlardan biri olan Çok Seviyeli Yerel Arama Sezgisel yöntemi Alessio Trivella ve David Pisinger tarafından sunulmuştur [41]. Başka bir çözüm de, Kang vd. tarafından yeni bir paketleme stratejisine [2] sahip Hibrit, bir Genetik Algoritmadır. Sanal Makineler yönetiminde örtüşen öğeler, önemli bir KPP sayılmaktadır. Sanal Makineler Kutu Paketleme Problemi, Sezgisel, Açgözlü(Greedy) , Açgözlü olmayan ve Genetik

Algoritmaları kullanarak zaman kısıtlamaları ve kalite gereksinimlerine göre Grange vd. tarafından bir çözüm sunulmuştur [42]. Takviye Öğrenme Yöntemleri kullanılarak yakın zamanda umut verici sonuçlar ortaya çıkmıştır. Hu vd. Derin Takviye Öğrenme tabanlı yöntemi kullanarak 3 boyutlu KPP için bir başarı elde etmeyi başarmışlardır [4]. Laterre vd. Sıralı Ödül'ü kullanarak genel Monte Carlo ağacı arama, sezgisel algoritmalar ve tamsayı programlama çözücülerinden daha iyi performans göstermiştir [27]. Duan vd. , Seçilmiş Öğrenme Yaklaşımını kullanarak iyi tasarlanmış açgözlü algoritmadan ciddi oranda maliyet azalması elde edebilmişlerdir [28].

3. ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu bölümde KPP'nin kullanılan türü ve kriterlerinden bahsedilecektir. Ayrıca bahsedilen algoritmaların KPP üzerine pratik olarak nasıl uygulanabileceği gösterilecek ve çeşitli denemelerden bahsedilecektir.

3.1. Problemin kullanılan kriterleri:

KPP'nin bir çok çeşidi bulunmaktadır. Bu yüzden problemin tipini ve kriterlerini belirtmek büyük önem taşımaktadır. Bu çalışmanın konusu detaylı olarak şu şekilde tanımlanabilir; küboid şeklinde ve 90 derecede döndürülebilen(altı farklı yön) N adet kutuyu, sabit uzunluk, sabit genişlik ve serbest (uzatılabilir) yükseklik ile üç boyutlu tek bir konteynere yerleştirmek istenmektedir. Ancak konteynerin yüksekliği en aza indirilerek yerleştirme yapılmalıdır. Bu şekilde bütün kutuları sığabilecek en minimum hacimde bir konteyner bulunmaya çalışılır.

3.2. Çözüm Uygulanması:

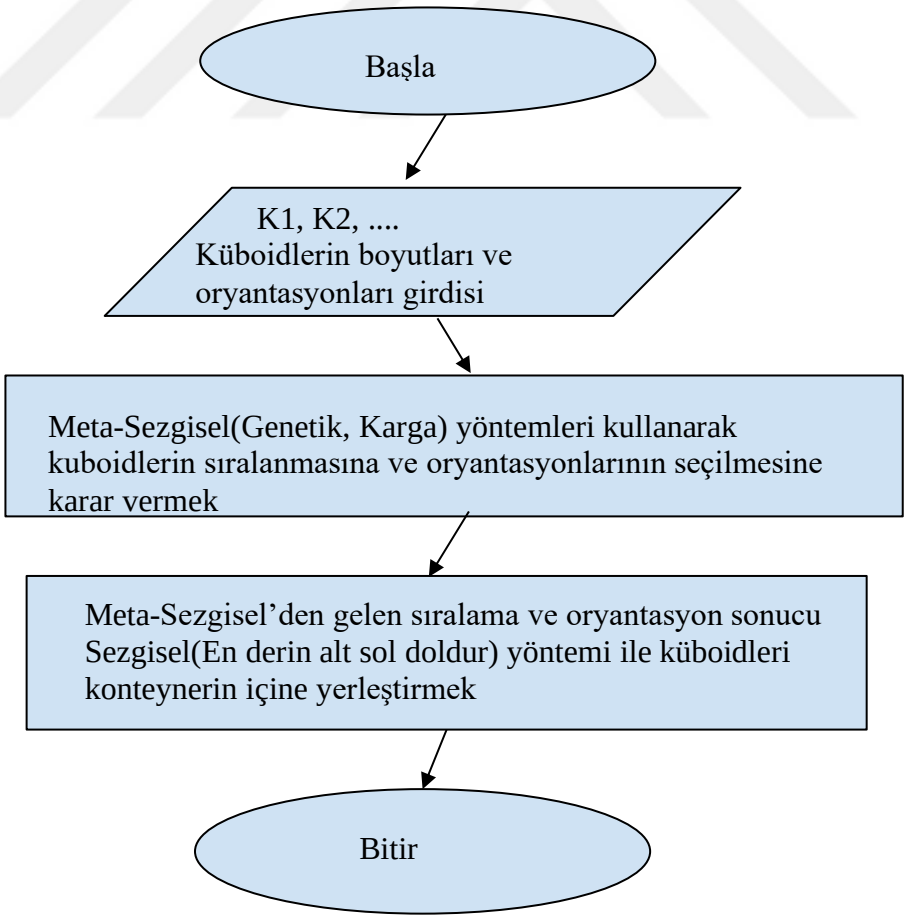
3.2.1. Metod:

Meta-Sezgisel algoritmalar güçlüdür ve diğer yöntemlerle çözülmesi zor olan çok çeşitli sorun alanlarıyla başa çıkabilir. Ancak bu algoritmalar genel yöntemler olduğundan dolayı probleme yönelik spesifik bilgi içermez. Bu şekilde problem bilgileriyle hibridizasyon, arama alanını daha akıllıca keşfederek çözümün kalitesini ve Meta-Sezgisel algoritmanın performansını önemli ölçüde arttırabilir. Bu çalışmada kullanılan yöntem bir konteynera kutular paketlenerek tek tek yerleştirilir. Bu yöntemin dezavantajı, bir kutunun nereye yerleştirileceğine ilişkin kararların yalnızca yerel ölçütlerle verilebilmesidir. Böylece algoritma zayıf bir paketleme ile sonuçlanabilir. Bu sorunun üstesinden gelmek ve farklı paketleme popülasyonunu yaratmak için Genetik ve Karga Algoritmaları kullanılmıştır.
2 boyutlu KPP'de Alt Sol ve Alt-Sol Doldur yöntemleri kullanılmıştır. Jakobs [43] tarafından sunulan Alt Sol yönteminde, her kutu mümkün olduğunca konteynerin alt tarafına ve sonra mümkün olduğu kadar sola taşınır. Alt Sol , O(n^2) karmaşıklığına sahip nispeten hızlı bir algoritmadır. Alt Sol yönteminin en büyük dezavantajı,

konteynerde boş alanlar oluşturulmasıdır. Hopper [44], bu dezavantajın üstesinden gelmek için Alt-Sol Doldur Algoritması'nı geliştirmiştir. Bu algoritma, her kutuyu konteynerin mümkün olan en düşük bölgesine tahsis eder. Böylece yerleşimdeki boş alanları doldurur. Bu algoritmanın en büyük dezavantajı O(n^3) karmaşıklığıdır. İkinci bölümde anlatılan En derin Alt-Sol Doldur Algoritma, 3 boyutlu KPP'leri kapsayan Alt-Sol Doldur yönteminin bir uzantısıdır. Özet olarak bu çalışmada hem Sezgisel hem de Meta-Sezgisel yöntemler kullanılmıştır. Sezgisel olarak En Derin Alt-Sol Doldur Algoritma, Meta Sezgisel olarak Karga-Arama Algoritma ve Genetik Algoritma birlikte kullanılmıştır.

3.2.2. Çözüm Diyagramı

Bu çalışmada KPP çözümü 3 aşamaya bölündü. Birincisi küboidlerin en iyi şekilde sıralanması. İkincisi ise küboidlerin en iyi oryantasyonda koyulması. Son aşaması küboidlerin en iyi yerleştirme şekli uygulanması. Şekil 3.1 diyagram metodun akışını göstermektedir.



Şekil 3.1. Çözüm Diyagramı

3.2.3. Kullanılan Algoritmalar Uygulanması:

Burada sunulan algoritmaların probleme entegre edilebileceği açıklanmaktadır:

3.2.3.1. Genetik Algoritma:

Sıradan Genetik Algoritma'nın bir probleme uygulanması zor değildir ama iyice tasarlanmış bir Genetik Algoritmaya ulaşmak ciddi zorluktur. Genetik algoritması ne kadar iyi tasarlanırsa o kadar optimal çözümlere ulaşılabilir.

- **Kromozom Temsili:**

Kromozom yapısının temsil edilmesi Genetik Algoritma'da büyük bir rol oynamaktadır. Bu Algoritma uygulanırken kromozom temsili ne kadar basit ve minimize olursa o kadar iyi olur. KPP'de kromozom, bir küboidlerin sırası ve oryantasyonlarından oluşmaktadır. Problemin girdisi küboidlerin uzunluk, genişlik ve yüksekliğinden oluşmaktadır. Bu şekilde bu problemin girdisi tek bir dizi olup, bütün kromozomlar aynı girdiyi paylaşmaktadır. Başka bir deyişle bu kromozom iki pozitif tamsayı dizisinden oluşmaktadır. Birisi küboidlerin sırası, öbürü küboidlerin oryantasyonlarıdır. Altı küboid yönü aşağıdaki gibi sembolize edilmiştir:

1- (uzunluk, genişlik, yükseklik) 2- (uzunluk, yükseklik, genişlik)
3- (genişlik, uzunluk, yükseklik) 4- (genişlik, yükseklik, uzunluk)
5- (yükseklik, genişlik, uzunluk) 6- (yükseklik, uzunluk, genişlik)

Bu şekilde boyutları işlerken oryantasyon koduna göre hangisi uzunluk, hangisi genişlik, hangisi yükseklik bilinmektedir.

Örnek: Verilen 5 kutunun boyutları aşağıdaki gibidir:

1- (7, 6, 2) 2- (1, 2, 3) 3- (11, 10, 12) 4- (5, 12, 11)
5- (7, 7, 7)

İlk varsayılan oryantasyon 1 kodu olan(uzunluk, genişlik, yükseklik) olandır.

Demek ki bu örnek kromozomu temsil edilmesi aşağıda gibidir:

Çizelge 3.1. Kromozom Örneği

Küboidlerin Sıralanması	1	2	3	4	5
Küboidlerin oryantasyonları	1	1	1	1	1

Bu kromozom üzerinde birkaç değişiklik aşağıdaki çizelge gibi yapıldığında,

Çizelge 3.2. Kromozom Örneği Değişikliği ile Birlikte

Küboidlerin Sıralanması	5	1	2	4	3
Küboidlerin oryantasyonları	2	1	3	5	6

Boyutları aşağıda gibi olur:

1- (7, 2, 6) 2- (1, 2, 3) 3- (10, 11, 12) 4- (11, 12, 5)

5- (7, 7, 7)

Yerleştirme, algoritmaya girdiyi verirken boyutların (uzunluk, genişlik, yükseklik) olarak geldiğini varsayılıyor.

● İlk Nesil

İlk nesil rastgele olarak yaratılmaktadır. Girdilerin oryantasyon kodu aksi belirtilmemiş ise varsayılan olarak 1. oryantasyon olarak kabul edilir. Girdilerin sıralamasına oryantasyonları rastgele bir işleme sokularak yeni bir sıralama ve oryantasyon oluşturulur. Bu işlem bir neslin kromozom sayısı kadar tekrarlandığında ilk nesil oluşturulur.

● Uygunluk Hesaplama

Bu hesaplama çözümlerin değerlendirilmesi anlamına gelir. Çözümün değerlendirmesi, kutuları konteynerin içine En Derin Alt-Sol Doldur algoritmasını kullanarak yerleştirdikten sonra hesaplanır. Konteyner küboid şeklinde olduğu için aşağıdaki gibi hacmi hesaplanabilir.

Konteyner Dolan Hacmi = (Uzunluk × Genişlik × Ulaşılan Yükseklik) (3.1)

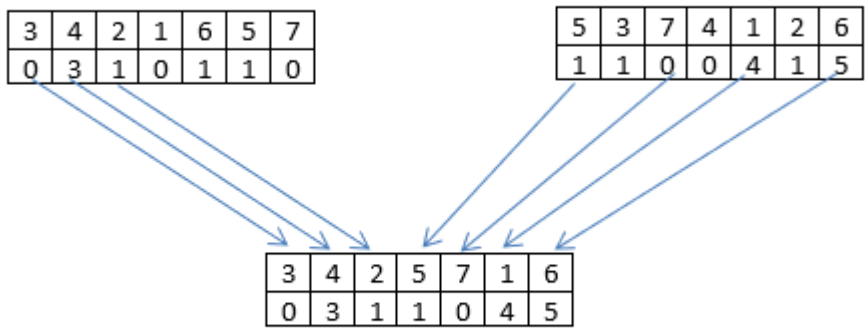
Konteynerin uzunluğu ve genişliği sabit olduğu için sadece yükseklik değerinin ne kadar değiştiği hesaplanır. Buradaki amacın, yüksekliği en aza indirmek olduğu unutulmamalıdır.

● **Seçim**

Bir nesilden başka nesile geçerken oransal olarak en iyi kromozomlar taşınır ve bu seçilen kromozomlar üzerine çaprazlama ve mutasyon uygulanıp yeni nesil bireyler oluşturulmaktadır. Seçim oranı bu çalışmada %50 olarak belirlenmiştir.

● **Çaprazlama**

Çaprazlama, farklı kromozomların çiftleşerek yeni bir kromozom ortaya çıkartmasına denir. Bu iki kromozom en iyi seçilen kromozomlardan rastgele olarak seçilir. Bu işlem çaprazlama oranına göre tekrarlanır. Bu çalışmada çaprazlama oranı %25 olarak belirlenmiştir.
Örnek: Aşağıdaki resimde iki kromozom arasında çaprazlama işlemi uygulanmıştır.



Şekil 3.2. Çaprazlama işlemi

● **Mutasyon**

Mutasyon işlemi bir kromozom üzerine uygulanır. Seçilen en iyi kromozomlardan rastgele olarak bir kromozom seçilir ve onun üzerine mutasyon uygulanır. Kromozomdan herhangi bir kutu seçilir ve bu kutunun oryantasyonu değiştirilerek mutasyon uygulanır. Mutasyon seçim ve çaprazlama gibi belli oranda uygulanır. Bu çalışmada mutasyon oranı %25 belirlenmiştir.

Örnek:

Şekil 3.3. Mutasyon işlemi

● Durdurma Koşulu

Bu algoritma tekrarlanması, nesil maksimum sayısına gelene kadar devam eder ve ulaşılan en iyi çözüm verilir. Bu çalışmada nesil maksimum sayısı değişken olarak belirlenir.

3.2.3.2. Karga Algoritma

● Algoritma Değişkenlerin Ayarlanması

Bu aşamada sürü boyutu, maksimum yineleme (yemek saklandığı yeri değiştirme) sayısı, uçuş uzunluğu ve farkındalık olasılığı gibi değişkenlerin değeri verilir. Bu çalışmada verilen değerler aşağıdaki gibidir:

1- Sürü boyutu: 50 Karga

2- Maksimum yineleme sayısı: 50 kere

3- Uçuş uzunluğu: Bu parametre, karga algoritması gerçek hayatta uygulanırken bir şeyi karşılamadığı için ihmal edilmiştir

4- Farkındalık olasılığı: Bütün kargalara 0 ve 1 arasında bir değer rastgele olarak verilmiştir.

● İlk Konum ve Hafıza Ayarlanması

Kargaların yemeğinin konumları rastgele olarak verilmiştir. Hafızalarında da ilk başta karşılaştırılacak pozisyon olmadığı için aynı pozisyon değeri verilmiştir.

● Uygunluk Değerlendirme

Genetik algoritması gibi kutuları konteynerin içine En Derin Alt-Sol Doldur algoritmasını kullanarak yerleştirdikten sonra hesaplanabilir. Konteynerin şekli küboid olduğu için aşağıdaki gibi hacmi hesaplanabilir.
Konteyner Dolan Hacmi = (Uzunluk × Genişlik × Ulaşılan Yükseklik) (3.2)
Algoritmanın amacı minimum hacime ulaşmaktır.

● Kargaların Takibi Başlatılması ve Yeni Pozisyona Geçirmesi

Sırayla her karga rastgele olarak başka bir karga seçip, onu takip edip yiyeceğini nerede sakladığını öğrenmeye çalışır. Örneğin karga i karga j'yi saklanan yiyeceklerinin konumunu keşfetmek için takip eder. Karga j'nin Farkındalık olasılığı $AP^{j,t}$ r_j'dan daha büyük ise (karga j'nin dikkatli olup takip edildiğini fark edip bilerek uzak bir yere gitmesi anlamına gelir) rastgele yeni bir pozisyon yaratılır ve karga i yemeğini oraya taşır. Bu durumda karga algoritması keşif yapılması anlamındadır. Karga j'nin Farkındalık olasılığı $AP^{j,t}$ r_j'dan daha küçük yada eşit ise (karga j'nin yeteri kadar dikkatli olmadığı anlamına gelir) aşağıdaki formülü kullanarak yeni pozisyon/konum hesaplayıp ona geçer. Bu durumda ise karga algoritması sömürü(exploitation) yapması anlamındadır. Burada 2.2 ve 2.3 eşitlikleri Pratik olarak nasıl uygulanacağı anlatılacaktır:
($h^{j,t}$ - $k^{i,t}$): karga j'nin hafızası ve karga i'nin yiyeceğinin eski pozisyonu arasındaki fark/mesafe anlamına gelir. Bunu hesaplamak için pratik olarak bir sürü yöntem vardır. Bu çalışmada Hamming Mesafesi yöntemi kullanılmıştır. Karga (çözüm) temsilinde iki değişken olduğu için iki Hamming Mesafesi oluşmaktadır. Birisi küboidlerin sıralaması Hamming Mesafesi, diğeri oryantasyon Hamming Mesafesi. Bu problem de Hamming mesafesini

hesaplamak için iki çözüm arasındaki farkın çıkartılması lazım. İki çözüm arasındaki farkın çıkartılması için sırayla kutulara bakılır. İki çözümde de birinci sırada aynı kutunun olup olmadığına bakılır. Aynı ise fark dizisinin birinci öğesi 1 değilse 0 olur. Oryantasyon için de aynı şey uygulanır. Daha sonra ortaya çıkan dizi sütunlarının toplamı alınıp, iki öğeli bir dizi ortaya çıkarılır. Aşağıdaki örnek Hamming Mesafesi'nin nasıl hesaplanacağını göstermektedir.

($h^{j,t}$ -$k^{i,t}$) = Hamming Mesafesi

Örnek: İki tamsayı dizisi arasındaki Hamming Mesafesi hesaplanmaktadır.

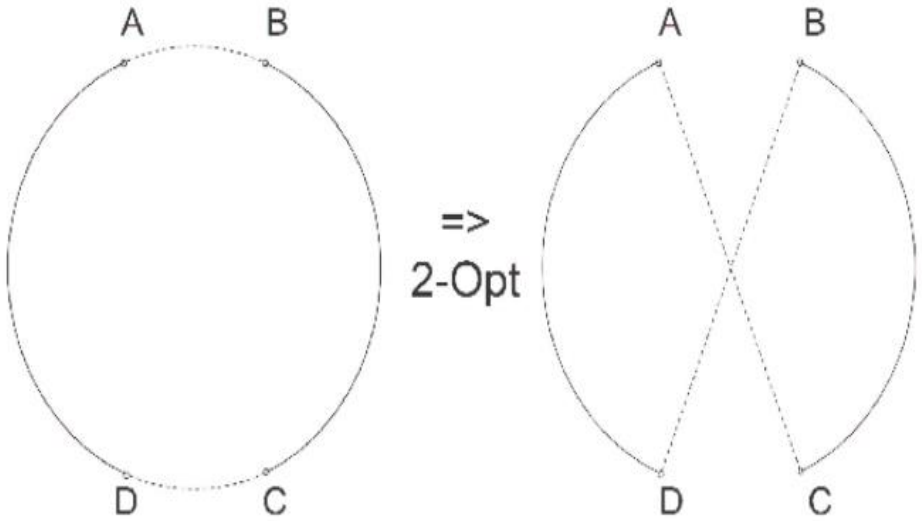
$$\text{Hamming Mesafesi} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \\ 3 & 3 \\ 1 & 3 \\ 2 & 6 \end{bmatrix} - \begin{bmatrix} 4 & 1 \\ 5 & 1 \\ 2 & 5 \\ 1 & 4 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = [4 \quad 3]$$

Şekil 3.4. Hamming Mesafesi Hesaplanması

$k^{i,t+1}$ Hamming Mesafesi: Bu ifade Karga Algoritma'sında karganın hareketi anlamındadır. Bu hareketi pratik olarak ifade edebilmek için literatürde iki çeşit vardır:

➜ **2-opt:**

Bu işlev 1965 yılında Lin tarafından tanımlanmıştır ve o zamandan beri yönlendirme sorununu çözmek için yaygın olarak kullanılmaktadır [45,46]. KPP yönlendirme sorununa benzemektedir. İkisinin de çözümü bir sıralamadan oluşmaktadır. 2-opt, çözüm dizilerini daire gibi yaptıktan sonra, rastgele iki küboidi ortadan seçip arasındaki diziyi alıp terslemektedir. Aşağıdaki örnek 2-opt nasıl yapıldığını göstermektedir, örneğin Dört küboidden oluşan bir dizi verilsin. Dizi sıra ile A, B, C, D dir. 2-opt uyguladıktan sonra A, C, B, D olarak çevrilmiştir.



Şekil 3.5. 2-OPT İşlemi

➜ 3-opt:

Lin tarafından da önerilen 3-opt işlemi 2-opt'e benzer. Bu durumda küboidler 3'e çıkarılmıştır. Bu yöntemi kullanmanın karmaşıklığı 2-opt'den daha fazladır.

Sonuç olarak bu çalışmada 2-opt yöntemi kullanılmıştır. Bu 2-opt yöntemi Hamming mesafesi kadar uygulanır. Bu şekilde karga hareketini gerçekleştirir. Algoritmanın karmaşıklığını basitleştirmek için $r_i\, vefl^{i,t}$ parametreleri dikkate alınmamıştır.

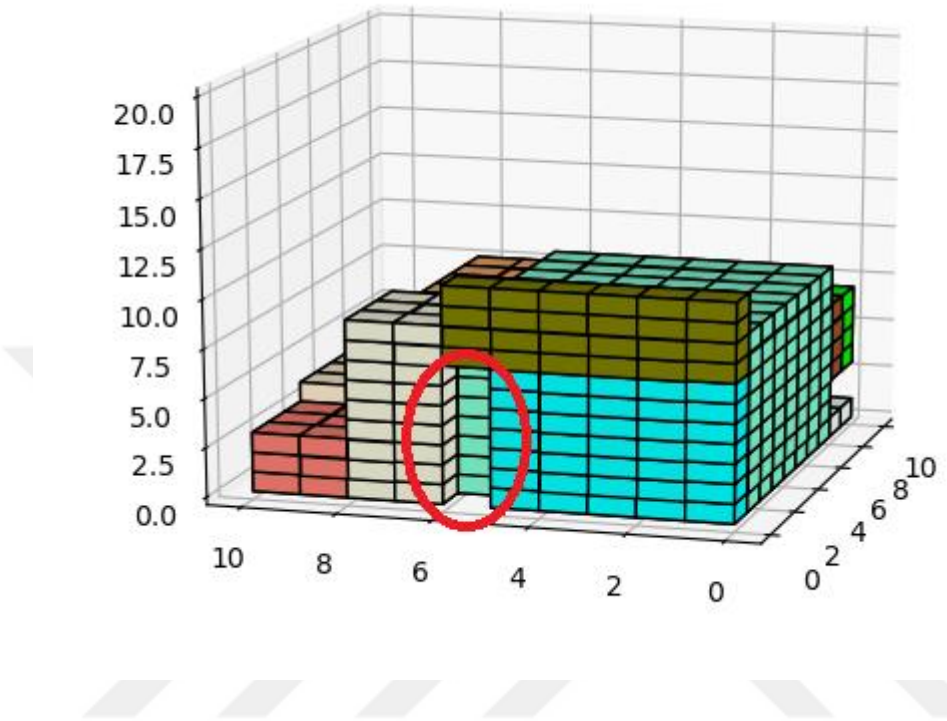
3.2.3.3. En Derin Alt-Sol Doldur Algoritma:

Bu algoritma, metodun sezgisel kısmıdır. En Derin Alt-Sol Doldur, en iyi yerleştirme algoritmalarından biridir. Önceki oluşan boşlukları doldurma özelliğinden dolayı konteynerin hacminin azaltılmasına yardımcı olur.Aşağıdaki örnek o özelliği göstermektedir.

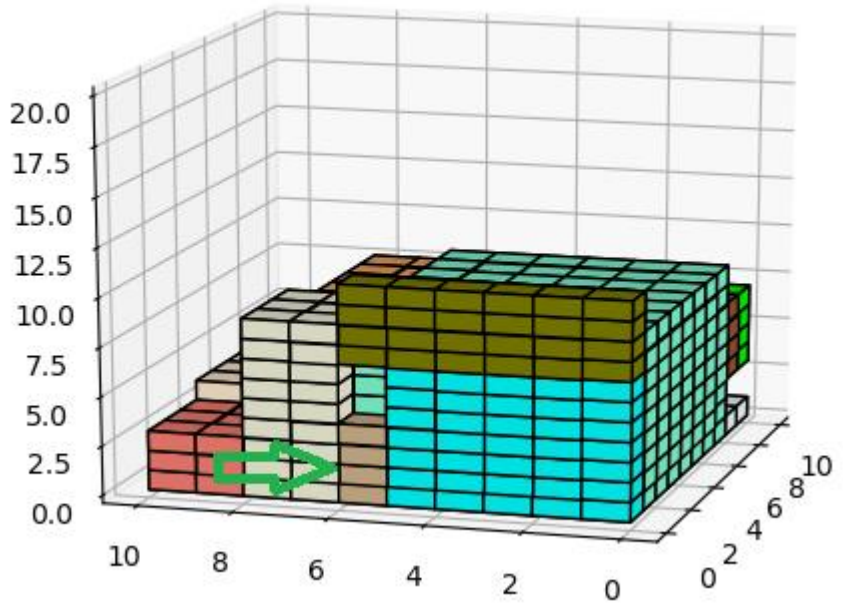
Örnek: Aşağıdaki gibi bir çok küboid En Derin Alt-Sol Doldur Algoritması ile yerleştirilmek isteniyor.

[(1, 5, 7), (7, 6, 9), (2,2,1), (3, 2, 9), (5, 2, 1), (2,2,1), (4, 4, 3), (3, 2, 3), (6, 2, 4), (1,2,3), (1,2,3), (1,6,4), (1,6,4), (1,6,4), (1,1,4)]

İlk resimde son küboid hariç bütün küboidler yerleştirilmiştir. İkinci resimde En Derin Alt-Sol Doldur Algoritması son küboidi (1,1,4) yerleştirmek için daha önceki boş alanlara gidip, uygun bir yer bulduğu için yeni bir alana gidip yerleştirmedi.



Şekil 3.6. En Derin Alt-Sol Doldur Algoritma Pratiği-1



Şekil 3.7. En Derin Alt-Sol Doldur Algoritma Pratiği-2

3.2.3.4. Hibrit Karga-Genetik Algoritma

Genetik Algoritma'da kullandığımız ilk nesli rastgele yaratmaktansa Karga Algoritma'sı kullanarak oluşturduğumuz sonuçları Genetik Algoritma ya ilk nesil olarak verilirse bulunan çözüm optimum çözüme daha yakın bir çözüm olur. Yani yeni bir problem çözülürken daha önce bu problemle alakalı Karga Algoritma'sı tarafından bulunan çözümlerden faydalanılarak çözüm uzayı daraltılmaya çalışılır. Bu şekilde probleme Hibrit Karga-Genetik Algoritma uygulanmıştır.

3.2.3.5. Hibrit Genetik-Karga Algoritma

Karga Algoritma'sı kullanılırken yiyeceklerin ilk konumlarını rastgele yaratmaktansa Genetik Algoritmadan çıkan çözümler Karga Algoritma'sına yiyeceklerinin ilk konumları olarak verilir. Bu şekilde probleme Hibrit Genetik-Karga Algoritma uygulanmıştır.

3.3. Test Verilerinin Üretimi

Bu çalışmada Kutu Paketleme Problemi varsayılan kriterlerle birlikte Benchmark veri seti olmadığı için rastgele veri seti yaratılmıştır. Yaratılan küboid veri seti boyutları 1 ve 12 arasındadır. Konteyner ise 14 genişlikli, 14 uzunluklu ve serbest yüksekliklidir. Bu rakamlar da ölçü birimi bulunmamakta ve değiştirilebilirdir. Gerçek hayatta bu rakamın ölçü birimi santimetre, milimetre yada herhangi bir metreye ait bir ölçü birimi olabilir. Bu ölçü birimi problemin gerçek hayatta uygulamasına göre belirlenebilir.

3.4. Konteyner ve Küboidlerin Temsili

Üzerinde çalışılan KPP'nin tipi 3 boyutlu olduğu için problemin karmaşıklığını arttırmaktadır. Hem küboidler hem de konteyner programsal olarak temsili önem taşımaktadır. Konteyner ve kutu da 3 boyutlu ikili(boolean) dizi olarak temsil edilmektedir. İlk başta konteynerin bütün değerleri sıfır oluyor. Kutuyu yerleştirirken konteyner ile veya(or) işlemi yapılarak yerleşmiş olur. Bu şekilde konteyner dizisinde her öğe bir koordinat hücresi sayılır. Yerleştirmeden önce de kutuların arasında kesişmeyi engellemek için En Derin Alt-Sol Doldur Algoritması uygulanarak kutunun yerleşeceği yer kontrol ediliyor. Kutunun boyutlarına göre bu yer uygun ise yerleşiyor değil ise onun için başka yer aranıyor.

3.5. Test Sonuçları

Bu çalışma, piton programlama dili kullanılarak geliştirilmiştir. Çalışmada numpy, matplotlib vb. kütüphaneleri kullanılmıştır. Bütün kullanılan algoritmalar da kod olarak hazır bir kütüphane kullanılmayıp, yazar tarafından geliştirilmiştir. Geliştirilen yöntemin uygulandığı bilgisayar özellikleri aşağıda verilmiştir.

➜ İşletim Sistemi: Windows 10 (64 bit)

➜ İşlemci: Intel Core i7

➜ 2.6 GHz. RAM: 16GB

➜ Ekran Kartı: NVIDIA GeForce

➜ HDD: 500GB

Deneme olarak 4 farklı test yapılmıştır. Metodları değerlendirmek için farklı küboid sayısında örnekler yaratılmıştır. Sonuçları ayrı ayrı analiz edilmiştir. Denemeler aşağıdaki kriterlere göre gerçekleştirilmiştir:

1- Karga Sürü boyutu: 50 Karga.

2- Karga maksimum yineleme sayısı: 50

3- Kromozom sayısı: 50 Kromozom

4- Genetik nesil sayısı: 50

5- Seçim oranı: %50

6- Çaprazlama oranı: %25

7- Mutasyon oranı: %25.

Algoritmalarda ilk aşamada rastgele çözümler yaratıldığı için bir kere denemenin adil olmadığı düşünülmüştür. Her algoritma, aynı örnek üzerinde 10 kere çalıştırılmıştır ve aşağıdaki her test sonucu bu 10 testin ortalamasıdır. Maksimum, minimum, standart sapma, ve harcanan süre saniye olarak da yazılmıştır.

3.5.1. 15 Küboidli Test:

15 küboide sahip 20 örnek üzerinden test yapılmıştır. Aşağıdaki tablo bunlardan 5 örneği gösteriyor.

Çizelge 3.3. 15 Küboidli Testin Bazı Sonuçları

No	Metod	Ort.	Maks.	Min.	Std.sapma	Süre/sa
	Karga Arama Algoritma	5390.0	5488	5292	103.30	49.89

Örnek 1	Hibrit Karga-Genetik	5174.4	5292	5096	101.21	94.08
	Genetik Algoritma	5292.0	5684	5096	226.32	45
	Hibrit Genetik-Karga	5252.8	5488	5096	180.11	95.01
Örnek 2	Karga Arama Algoritma	5272.4	5488	5096	111.25	54.27
	Hibrit Karga-Genetik	5096	5292	4900	92.39	101.34
	Genetik Algoritma	5194	5292	5096	103.30	50.93
	Hibrit Genetik-Karga	5135.2	5292	4900	123.96	103.11
Örnek 3	Karga Arama Algoritma	5272.4	5488	5096	171.61	49.55
	Hibrit Karga-Genetik	5115.6	5292	4900	171.61	92.65
	Genetik Algoritma	5292	5488	5096	130.66	44.88
	Hibrit Genetik-Karga	5233.2	5292	4900	132.28	93.45
Örnek 4	Karga Arama Algoritma	4429.6	4508	4312	101.21	50.39
	Hibrit Karga-Genetik	4272.8	4508	3920	180.11	91.60

	Genetik Algoritma	4331.6	4508	4116	111.25	44.84
	Hibrit Genetik-Karga	4331.6	4508	4116	111.25	93.94
Örnek 5	Karga Arama Algoritma	3214.4	3332	3136	101.21	28.95
	Hibrit Karga-Genetik	3057.6	3136	2940	101.21	58.33
	Genetik Algoritma	3136.0	3332	2940	160.03	27.89
	Hibrit Genetik-Karga	3096.8	3332	2940	123.96	55.94

Aşağıdaki tablo 20 tane 15'li küboid örneği testinin metodlarının karşılaştırılmasını göstermektedir:

Çizelge 3.4. 15 Küboidli Testin Yöntemlerin karşılaştırılması.

Metod 1	Metod 2	Daha iyi Performans gösteren	Oransal Başarısı(%)
Genetik Algoritma	Karga Arama Algoritma	Genetik Algoritma	1.44
Karga Arama Algoritma	Hibrit Genetik-Karga	Hibrit Genetik-Karga	2.7

Karga Arama Algoritma	Hibrit Karga-Genetik	Hibrit Karga-Genetik	3.7
Genetik Algoritma	Hibrit Genetik-Karga	Hibrit Genetik-Karga	1.2
Genetik Algoritma	Hibrit Karga-Genetik	Hibrit Karga-Genetik	2.2
Hibrit Genetik-Karga	Hibrit Karga-Genetik	Hibrit Karga-Genetik	1

Metodların karşılaştırılması tablosuna bakıldığında metodlar arasında en iyi performans gösteren Hibrit Karga-Genetik'dir.

3.5.2. 20 Küboidli Test:

20 küboide sahip 20 örnek üzerinden test yapılmıştır. Aşağıdaki tablo bunlardan 5 örneği gösteriyor.

Çizelge 3.5. 20 Küboidli Testin Bazı Sonuçları.

No	**Metod**	**Ort.**	**Maks.**	**Min.**	**Std.sapma**	**Süre/sa**
Örnek 1	Karga Arama Algoritma	6781.6	6860	6468	137.04	69.85
	Hibrit Karga-Genetik	6409.2	6860	6076	262.14	129.09

	Genetik Algoritma	6507.2	6664	6076	180.11	62.78
	Hibrit Genetik-Karga	6487.6	6664	6076	171.61	131.22
Örnek 2	Karga Arama Algoritma	5978.0	6272	5684	230.98	56.05
	Hibrit Karga-Genetik	5684.0	5880	5292	184.79	107.82
	Genetik Algoritma	6095.6	6468	5880	194.90	52.99
	Hibrit Genetik-Karga	6017.2	6272	5880	132.28	106.78
Örnek 3	Karga Arama Algoritma	4566.8	4704	4312	132.28	54.43
	Hibrit Karga-Genetik	4370.8	4508	4116	132.28	99.35
	Genetik Algoritma	4488.4	4704	4116	194.90	47.75
	Hibrit Genetik-Karga	4370.8	4508	4116	132.28	100.86
Örnek 4	Karga Arama Algoritma	9917.6	10192	9604	165.28	93.52
	Hibrit Karga-Genetik	9486.4	9800	9212	189.35	171.93
	Genetik Algoritma	9584.4	9996	9016	326.01	84.24

	Hibrit Genetik-Karga	9564.8	9996	9016	303.64	175.58
Örnek 5	Karga Arama Algoritma	5742.8	5880	5684	94.67	64.81
	Hibrit Karga-Genetik	5566.4	5880	5292	189.35	118.15
	Genetik Algoritma	5605.6	5880	5292	189.35	58.71
	Hibrit Genetik-Karga	5566.4	5880	5292	165.28	122.48

Aşağıdaki tablo 20 tane 20'li küboid örneği testinin metodlarının karşılaştırılmasını göstermektedir:

Çizelge 3.6. 20 Küboidli Testin Yöntemlerin karşılaştırılması.

Metod 1	Metod 2	Daha iyi Performans gösteren	Oransal Başarısı(%)
Genetik Algoritma	Karga Arama Algoritma	Genetik Algoritma	2
Karga Arama Algoritma	Hibrit Genetik-Karga	Hybrid Genetic-Karga	2.8
Karga Arama Algoritma	Hibrit Karga-Genetik	Hibrit Karga-Genetik	4.1

Genetik Algoritma	Hibrit Genetik-Karga	Hibrit Genetik-Karga	0.79
Genetik Algoritma	Hibrit Karga-Genetik	Hibrit Karga-Genetik	2.13
Hibrit Genetik-Karga	Hibrit Karga-Genetik	Hibrit Karga-Genetik	1.35

Metodların karşılaştırılması tablosunda bakıldığında metodlar arasında en iyi performans gösteren Hibrit Karga-Genetik'dir.

3.5.3. 25 Küboidli Test:

15 küboide sahip 20 örnek üzerinden test yapılmıştır. Aşağıdaki tablo bunlardan 5 örneği gösteriyor.

Çizelge 3.7. 25 Küboidli Testin Bazı Sonuçları.

No	**Metod**	**Ort.**	**Maks.**	**Min.**	**Std.sapma**	**Süre/sa**
Örnek 1	Karga Arama Algoritma	10407.6	10780	10192	215.69	137.09
	Hibrit Karga-Genetik	9643.2	9800	9408	154.60	268.47
	Genetik Algoritma	9878.4	10388	9408	295.08	124.96
	Hibrit Genetik-Karga	9839.2	10388	9212	343.23	259.42

Örnek 2	Karga Arama Algoritma	13818.0	14112	13328	248.78	169.90
	Hibrit Karga-Genetik	13112.4	13524	12544	326.01	318.95
	Genetik Algoritma	13484.8	14504	12152	611.48	160.85
	Hibrit Genetik-Karga	13386.8	13916	12152	506.49	327.01
Örnek 3	Karga Arama Algoritma	7212.8	7448	6860	180.11	109.02
	Hibrit Karga-Genetik	6977.6	7252	6664	230.06	207.02
	Genetik Algoritma	6997.2	7448	6468	277.95	98.51
	Hibrit Genetik-Karga	6918.8	7252	6468	245.32	205.19
Örnek 4	Karga Arama Algoritma	7879.2	8232	7644	180.11	119.67
	Hibrit Karga-Genetik	7487.2	7644	7252	180.11	229.84
	Genetik Algoritma	7781.2	8232	7448	207.63	111.94
	Hibrit Genetik-Karga	7702.8	7840	7448	161.36	227.86
Örnek 5	Karga Arama Algoritma	10172.4	10584	9800	215.69	148.23

	Hibrit Karga-Genetik	9662.8	9996	9408	227.26	285.37
	Genetik Algoritma	9996.0	10388	9604	206.60	915.60
	Hibrit Genetik-Karga	9917.6	10192	9604	165.28	1061.66

Aşağıdaki tablo 20 tane 25'li küboid örneği testinin metodlarının karşılaştırılmasını göstermektedir:

Çizelge 3.8. 25 Küboidli Testin Yöntemlerin karşılaştırılması.

Metod 1	Metod 2	Daha iyi Performans gösteren	Oransal Başarısı(%)
Genetik Algoritma	Karga Arama Algoritma	Genetik Algoritma	2.7
Karga Arama Algoritma	Hibrit Genetik-Karga	Hibrit Genetik-Karga	3.2
Karga Arama Algoritma	Hibrit Karga-Genetik	Hibrit Karga-Genetik	4.5
Genetik Algoritma	Hibrit Genetik-Karga	Hibrit Genetik-Karga	0.59

Genetik Algoritma	Hibrit Karga-Genetik	Hibrit Karga-Genetik	1.88
Hibrit Genetik-Karga	Hibrit Karga-Genetik	Hibrit Karga-Genetik	1.29

Metodların karşılaştırılması tablosunda bakıldığında metodlar arasında en iyi performans gösteren Hibrit Karga-Genetik'dir.

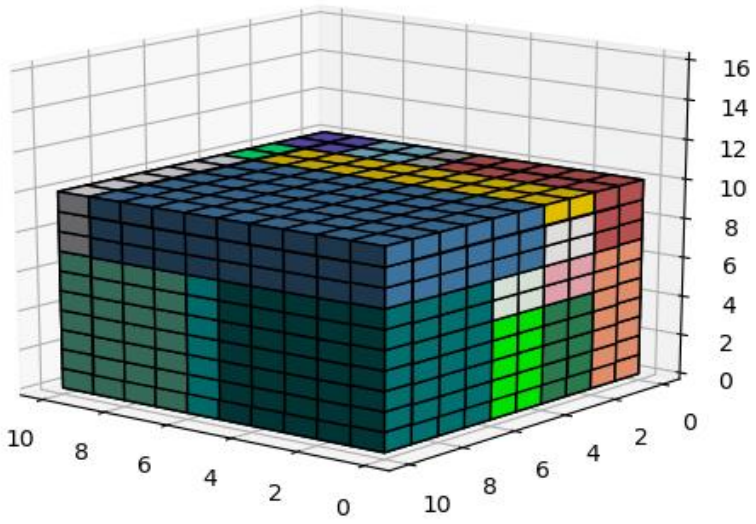
3.5.4. Optimum Çözümü Bilinen Bir Örnek

Genelde KPP'lerde ulaşılan bir çözüm küboidlerin toplam hacmine bakarak değerlendirilebilir. Başka bir deyişle ulaşılan çözüm de, konteynerin hacmi küboidlerin toplam hacmine ne kadar yakın duruyor ise bu çözümün ne kadar iyi olduğuna karar verilebilir. Ancak her örnek bu şekilde değildir. Bazı örneklerde optimum çözüm, küboidlerin toplam hacmi ya da ona yakın bir rakam olmayabilir. Bu yüzden optimum çözümü bilinen bir örnek hazırlamak gerektirmiştir. Bu örnek üzerinden değerlendirmek daha doğrudur. Örnek aşağıdaki özelliklerine sahiptir.

➜ Küboidleri şu şekilde:(uzunluk, genişlik, yükseklik)

[(5, 2, 7),(1,2,7), (4, 2, 3), (4, 2, 5), (4, 2, 1), (2,2,1)
,(3,2,5), (3, 2, 1), (4,2,3) ,(5,4,7),(1,4,7),(4,4,7)
,(6,2,4) ,(3,2,6),(10,2,2),(1,2,4) ,(3,2,4),(1,1,4)
,(4,1,4),(3,2,2),(3,1,4),(5,2,3),(1,2,3),(2,2,3),
(2,2,3), (9,2,2), (1,2,3),(9,6,3),(1,6,3),(9,2,1)]

➜ Küboidler yukarıdaki sırada (uzunluk, genişlik, yükseklik) olarak (10, 10, 10) boyutlu bir konteynerin içine boşluksuz olarak yerleşebilir. Bu yüzden yukarıdaki verilen sıralama ve oryantasyon bu örneğin en optimum çözümdür. Aşağıdaki resimlerde yerleşmesi görülebilir



Şekil 3.8. Verilen Örnek Optimum Çözümü

Bu örneği çözen algoritmalara verildiğinde aşağıdaki tablolardaki sonuçlara ulaşılmıştır.

Note: Optimum ulaşılabilecek hacım küboidlerin toplam hacmıdır=1000.

Sonuçlar ve hiperparametre ayarları aşağıda gibidir:

Deneme 1:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 50 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 50 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %60. 6- Çaprazlama oranı: %20

7- Mutasyon oranı: %20.

- Sonuçlar:

Çizelge 3.9. Optimum Çözümü Bilinen Örnek Deneme 1'in Sonuçları.

Metod	**Ort.**	**Maks.**	**Min.**	**Optimum Hacim**	**Süre/sa**
Karga Arama Algoritma	1260	1300	1200	1000	40.78
Hibrid Karga-Genetik	1220	1300	1200	1000	84.03
Genetik Algoritma	1240	1300	1200	1000	46.34
Hybrid Genetik-Karga	1220	1300	1200	1000	85.71

Deneme 2:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 100 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 100 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %70. 6- Çaprazlama oranı: %15

7- Mutasyon oranı: %15.

- Sonuçlar:

Çizelge 3.10. Optimum Çözümü Bilinen Örnek Deneme 2'in Sonuçları.

Metod	Ort.	Maks.	Min.	Optimum Hacim	Süre/sa
Karga Arama Algoritma	1220	1300	1200	1000	85.49
Hibrid Karga-Genetik	1220	1200	1200	1000	174.04
Genetik Algoritma	1240	1300	1200	1000	46.34
Hybrid Genetik-Karga	1220	1300	1200	1000	85.71

Deneme 3:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 100 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 100 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %50. 6- Çaprazlama oranı: %25

7- Mutasyon oranı: %25.

- Sonuçlar:

Çizelge 3.11. Optimum Çözümü Bilinen Örnek Deneme 3'in Sonuçları.

Metod	Ort.	Maks.	Min.	Optimum Hacim	Süre/sa
Karga Arama Algoritma	1240	1300	1200	1000	83.26

Hibrid Karga-Genetik	1200	1200	1200	1000	165.73
Genetik Algoritma	1200	1200	1200	1000	86.94
Hybrid Genetik-Karga	1200	1200	1200	1000	85.71

Deneme 4:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 100 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 100 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %10. 6- Çaprazlama oranı: %45

7- Mutasyon oranı: %45.

- Sonuçlar:

Çizelge 3.12. Optimum Çözümü Bilinen Örnek Deneme 4'in Sonuçları.

Metod	**Ort.**	**Maks.**	**Min.**	**Optimum Hacim**	**Süre/sa**
Karga Arama Algoritma	1220	1300	1200	1000	83.88
Hibrid Karga-Genetik	1200	1200	1200	1000	172.42
Genetik Algoritma	1220	1300	1200	1000	44.72

Hybrid Genetik-Karga	1200	1200	1200	1000	170.86

Deneme 5:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 100 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 100 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %10. 6- Çaprazlama oranı: %10

7- Mutasyon oranı: %80.

- Sonuçlar:

Çizelge 3.13. Optimum Çözümü Bilinen Örnek Deneme 5'in Sonuçları.

Metod	**Ort.**	**Maks.**	**Min.**	**Optimum Hacim**	**Süre/sa**
Karga Arama Algoritma	1220	1200	1200	1000	78.65
Hibrid Karga-Genetik	1200	1200	1200	1000	158.83
Genetik Algoritma	1240	1300	1200	1000	85.73
Hybrid Genetik-Karga	1220	1300	1200	1000	164.59

Deneme 6:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 100 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 100 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %10. 6- Çaprazlama oranı: %80

7- Mutasyon oranı: %10.

- Sonuçlar:

Çizelge 3.14. Optimum Çözümü Bilinen Örnek Deneme 6'in Sonuçları.

Metod	**Ort.**	**Maks.**	**Min.**	**Optimum Hacim**	**Süre/sa**
Karga Arama Algoritma	1240	1300	1200	1000	85.51
Hibrid Karga-Genetik	1200	1200	1200	1000	175.54
Genetik Algoritma	1200	1300	1200	1000	90.92
Hybrid Genetik-Karga	1200	1200	1200	1000	175.85

Deneme 7:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 100 Karga. 2- Karga maksimum yineleme sayısı: 100 kere.

3- Kromozom sayısı: 100 Kromozom. 4- Genetik nesil sayısı: 100 nesil.

5- Seçim oranı: %50. 6- Çaprazlama oranı: %25

7- Mutasyon oranı: %25.

- Sonuçlar:

Çizelge 3.15. Optimum Çözümü Bilinen Örnek Deneme 7'in Sonuçları.

Metod	**Ort.**	**Maks.**	**Min.**	**Optimum Hacim**	**Süre/sa**
Karga Arama Algoritma	1220	1300	1200	1000	165.60
Hibrid Karga-Genetik	1200	1200	1200	1000	340.25
Genetik Algoritma	1200	1200	1200	1000	177.24
Hybrid Genetik-Karga	1200	1200	1200	1000	343.42

Deneme 8:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 150 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 150 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %50. 6- Çaprazlama oranı: %25

7- Mutasyon oranı: %25.

- Sonuçlar:

Çizelge 3.16. Optimum Çözümü Bilinen Örnek Deneme 8'in Sonuçları.

Metod	Ort.	Maks.	Min.	Optimum Hacim	Süre/sa
Karga Arama Algoritma	1200	1200	1200	1000	127.88
Hibrid Karga-Genetik	1180	1200	1100	1000	263.30
Genetik Algoritma	1200	1200	1200	1000	133.45
Hybrid Genetik-Karga	1200	1200	1200	1000	256.89

Deneme 9:

- Hiperparametre ayarları:

1- Karga Sürü boyutu: 200 Karga. 2- Karga maksimum yineleme sayısı: 50 kere.

3- Kromozom sayısı: 200 Kromozom. 4- Genetik nesil sayısı: 50 nesil.

5- Seçim oranı: %50. 6- Çaprazlama oranı: %25

7- Mutasyon oranı: %25.

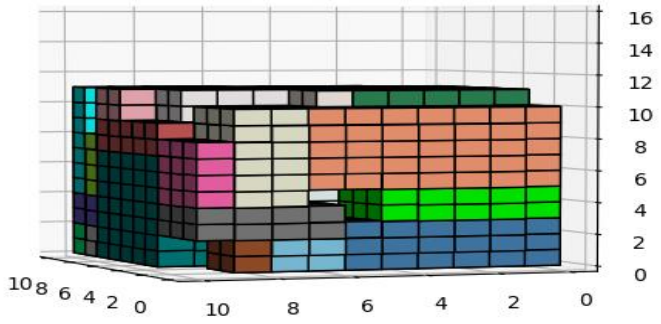
- Sonuçlar:

Çizelge 3.17. Optimum Çözümü Bilinen Örnek Deneme 9'in Sonuçları.

Metod	Ort.	Maks.	Min.	Optimum Hacim	Süre/sa
Karga Arama Algoritma	1200	1200	1200	1000	181.43

Hibrid Karga-Genetik	1160	1200	1100	1000	374.50
Genetik Algoritma	1200	1200	1200	1000	186.96
Hybrid Genetik-Karga	1200	1200	1200	1000	365.21

Gösterilen sonuçlara göre Deneme 9 en iy çözüm vermiştir, ayrıca bu çözümü veren yöntem Hibrid Karga-Genetik algoritmadır. Bir kez daha sunulan yöntemlerin en iyi çözümün Hibrid Karga-Genetik algoritma olduğunu ispatlanmıştır. Şekil 3.8'de ulaşılan çözümün uygulanması gösterilmiştir.



Şekil 3.9. Verilen Örnek En İyi Ulaşılan Çözüm

4. SONUÇ VE ÖNERILER

Bu tez çalışmasında KPP'nin bir çeşidinin üzerinde çalışılmıştır. Bu çalışmadaki temel amaç, 3 boyutlu kutuları bir konteynerin içine optimum bir şekilde yerleştirilmektir. Yöntem olarak Sezgisel ve Meta-Sezgisel yöntemler birlikte kullanılmıştır. Sezgisel olarak En Derin Alt-Sol Doldur Algoritma'sı, Meta-Sezgisel olarak dört farklı algoritma kullanılmıştır. Bunlar, Karga Arama Algoritma, Genetik Algoritma, Hibrit Genetik-Karga ve Hibrit Karga-Genetik Algoritma'dır. Sayısal sonuçlar Hibrit Karga-Genetik'in en iyi yöntem olduğunu gösteriyor. Bu çalışmanın katkısı, 3 boyutlu KPP'de Karga Arama, Hibrit Genetik-Karga ve Hibrit Karga-Genetik Algoritma'larının ilk defa uygulanmasıdır. Gelecek çalışmada kademeli çözümler ve takviyeli derin öğrenmeyi dahil ederek çözümler üretmek düşünülmektedir.

KAYNAKLAR

[1] Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P. and Carballedo, R. (2016). An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. Engineering Applications of Artificial Intelligence, 48, pp.59–71.

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

[3] Lodi, A., Martello, S. and Vigo, D. (2002). Heuristic algorithms for the three-dimensional bin packing problem. European Journal of Operational Research, 141(2), pp.410–420.

[4] Hu, H., Zhang, X., Yan, X., Wang, L. and Xu, Y. (2017). Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method. arXiv:1708.05930 [cs]. [online] Available at: https://arxiv.org/abs/1708.05930 [Accessed 4 Jun. 2020].

[5] Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures, 169, pp.1–12.

[6] Elsevier (2004). Stochastic Local Search - 1st Edition. [online] Elsevier.com. Available at: https://www.elsevier.com/books/stochastic-local-search/hoos/978-1-55860-872-6 [Accessed 30 Apr. 2019].

[7] www.cs.jhu.edu. (n.d.). Scheideler: Modern Complexity Theory. [online] Available at: http://www.cs.jhu.edu/~scheideler/courses/600.471_S05 [Accessed 4 Jun. 2020].

[8] Bovet, D. P. and Crescenzi, P. (1994). Introduction to the Theory of Complexity. Prentice-Hall, Englewood Cliffs, New Jersey.

[9] Cook, S. A. (1971). “The complexity of theorem-proving procedures,” Proc. 3rd Annual ACM Symp. Theory of Computing, 151–158.

[10] Jacobs , K. Theory of Computation. Retrieved June 26, 2016, from http://ocw.mit.edu/courses/mathematics/18-404j-theory-of-computation-fall-2006/.

[11] Michael R. Garey and David S. Johnson (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman. ISBN 0-7167-1045-5. A4.1: SR1, p. 226.

[12] Muhammed Beyaza, Tansel Dokeroglub, Ahmet Cosar Robust hyper-heuristic algorithms for the offline oriented/non-oriented 2D bin packing problems, Elsevier Applied Soft Computing 36 (2015) 236–245.

[13] K. A. Dowsland. An exact algorithm for the pallet loading problem.European Journal of Operational Research, 31(1):78–84, July 1987.

[14] S. Bhattacharya and R. Bhattacharya. An exact depth-first algorithm for the pallet loading problem.European Journal of Operational Research, 110(3):610–625, 1998.

[15] A. Tarnowski, J. Terno, and G. Scheithauer.A polynomial time algorithm for the guillotine pallet loadingproblem. INFOR, 32:275–287, 1994.

[16] S. Martello, M. Monaci, D. Vigo, An exact approachtothe strip packing problem, Technical paper OR/00/18,Dipartimento di Elettronica, Informatica e Sistemistica,Universita di Bologna, 2000.

[17] S. Martello, D. Vigo, Exact solution of the two-dimensionalfinite bin packing problem, Management Science 44(1998) 388–399.

[18] J. Egeblad. Heuristics for Multidimensional Packing Problems.PhD thesis, University of Copenhagen, Department of Computer Science, 2008.

[19] Johnson, David S; Garey, Michael R (October 1985). "A 7160 theorem for bin packing". Journal of Complexity. 1 (1): 65–106. doi:10.1016/0885-064X(85)90022-6.

[20] Art, Richard Carl. 1966. “An Approach to the Two Dimensional Irregular Cutting Stock Problem.” PhD diss., Massachusetts Institute of Technology.

[21] Bennell, J. A., and J. F. Oliveira. 2009. “A Tutorial in Irregular Shape Packing Problems.” Journal of the Operational Research Society 60: S93–S105.

[22] K. Sörensen and F. Glover. Metaheuristics. In S.I. Gass and M. Fu, editors, Encyclopedia of Operations Research and Management Science, pages 960–970. Springer, New York, 2013.

[23] F. Glover. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research, 13:533–549, 1986.

[24] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975).

[25] Clayton N, Emery N. Corvide cognition. Curr Biol 2005;15:R80–1.

[26] Díaz, P., Pérez-Cisneros, M., Cuevas, E., Avalos, O., Gálvez, J., Hinojosa, S. and Zaldivar, D. (2018). An Improved Crow Search Algorithm Applied to Energy Problems. Energies, [online] 11(3), p.571. Available at: https://www.mdpi.com/1996-1073/11/3/571 [Accessed 4 Jun. 2020].

[27] Laterre, A., Fu, Y., Jabri, M.K., Cohen, A.-S., Kas, D., Hajjar, K., Dahl, T.S., Kerkeni, A. and Beguir, K. (2018). Ranked Reward: Enabling Self-Play Reinforcement Learning for Combinatorial Optimization. arXiv:1807.01672 [cs, stat]. [online] Available at: https://arxiv.org/abs/1807.01672 [Accessed 4 Jun. 2020].

[28] Duan, L., Hu, H., Qian, Y., Gong, Y., Zhang, X., Xu, Y. and Wei, J. (2019). A Multi-task Selected Learning Approach for Solving 3D Flexible Bin Packing Problem. arXiv:1804.06896 [cs, stat]. [online] Available at: https://arxiv.org/abs/1804.06896 [Accessed 4 Jun. 2020].

[29] M. Magazine and O. Oguz. A heuristic algorithm for the multidimensional zero-one knapsack problem. European Journal of Operational Research. 16 (3) : 319 - 326, 1984.

[30] S. Arshad, S. Yang, and C. Li. A sequence based genetic algorithm with local search for the travelling salesman problem. Proceedings of the 2009 UK Workshop on Computational Intelligence, pp. 98-105, 2009.

[31] Zhi, S., Liu, Y., Li, X., Guo, Y. (2017, April). LightNet: A Lightweight 3D Convolutional Neural Network for Real-Time 3D Object Recognition. In 3DOR.

[32] Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In Proceedings of the IEEE international conference on computer vision (pp. 945–953).

[33] Laabadi, S., Naimi, M., Amri, H.E. and Achchab, B. (2020). A Binary Crow Search Algorithm for Solving Two-dimensional Bin Packing Problem with Fixed Orientation. Procedia Computer Science, [online] 167, pp.809–818. Available at: https://www.sciencedirect.com/science/article/pii/S1877050920308863 [Accessed 11 Jun. 2020].

[34] Fu, Y. and Banerjee, A. (2020). Heuristic/meta-heuristic methods for restricted bin packing problem. Journal of Heuristics.

[35] Liu, Y. and Cao, B. (2020). A Novel Ant Colony Optimization Algorithm With Levy Flight. IEEE Access, [online] 8, pp.67205–67213. Available at: https://ieeexplore.ieee.org/abstract/document/9056538 [Accessed 11 Jun. 2020].

[36] Ashraf, U., Liang, J., Akhtar, A., Yu, K., Hu, Y., Yue, C., Masood, A.M. and Kashif, M. (2020). Meta-heuristic Hybrid Algorithmic Approach for Solving Combinatorial Optimization Problem (TSP). Communications in Computer and Information Science, pp.622–633.

[37] Pitakaso, R., Sethanan, K. and Jamrus, T. (2020). Hybrid PSO and ALNS algorithm for software and mobile application for transportation in ice manufacturing industry 3.5. Computers & Industrial Engineering, [online] 144, p.106461. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0360835220301959 [Accessed 11 Jun. 2020].

[38] Kuhn, H., Schubert, D. and Holzapfel, A. (2020). Integrated order batching and vehicle routing operations in grocery retail – A General Adaptive Large Neighborhood Search algorithm. European Journal of Operational Research. [online] Available at: https://www.sciencedirect.com/science/article/abs/pii/S0377221720303088 [Accessed 11 Jun. 2020].

[39] Abdel-Basset, M., Manogaran, G., Abdel-Fatah, L. and Mirjalili, S. (2018). An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. Personal and Ubiquitous Computing, 22(5–6), pp.1117–1132.

[40] Lodi, A., Monaci, M. and Pietrobuoni, E. (2017). Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints. Discrete Applied Mathematics, 217, pp.40–47.

[41] Trivella, A. and Pisinger, D. (2016). The load-balanced multi-dimensional bin-packing problem. Computers & Operations Research, [online] 74, pp.152–164. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0305054816300909 [Accessed 11 Jun. 2020].

[42] Grange, A., Kacem, I. and Martin, S. (2018). Algorithms for the bin packing problem with overlapping items. Computers & Industrial Engineering, [online] 115, pp.331–341. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0360835217305004 [Accessed 11 Jun. 2020].

[43] Jakobs, S: On Genetic Algorithms for the Packing of Polygons. European Journal of Operational Research Vol 88 (1996) 165-181.

[44] Hopper, E: Two-dimensional Packing Utilising Evolutionary Algorithms and other MetaHeuristic Methods., A Thesis submitted to University of Wales for the Degree of Doctor of Philosophy (2000).

[45] Bianchessi, N., & Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. Computers & Operations Research, 34, 578–594.

[46] Tarantilis, C., & Kiranoudis, C. (2007). A flexible adaptive memory-based algorithm for real-life transportation operations: Two case studies from dairy and construction sector. European Journal of Operational Research, 179, 806– 822.