T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ALT SINIR TEMELİNE DAYALI AĞIRLIKLI TAVLAMA YÖNTEMİ İLE KUTULAMA PROBLEMİNİN ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

NERİMAN İNAK

DENİZLİ, TEMMUZ - 2017

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ALT SINIR TEMELİNE DAYALI AĞIRLIKLI TAVLAMA YÖNTEMİ İLE KUTULAMA PROBLEMİNİN ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

NERİMAN İNAK

DENİZLİ, TEMMUZ - 2017

KABUL VE ONAY SAYFASI

NERİMAN İNAK tarafından hazırlanan "**ALT SINIR TEMELİNE DAYALI AĞIRLIKLI TAVLAMA YÖNTEMİ İLE KUTULAMA PROBLEMİNİN ÇÖZÜMÜ**" adlı tez çalışmasının savunma sınavı 11.07.2017 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / oy çokluğu ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Prof. Dr. Sezai TOKAT	.....
İkinci Danışman Yrd. Doç. Dr. Kenan KARAGÜL Pamukkale Üniversitesi	.....
Üye Doç. Dr. ÖZCAN MUTLU Pamukkale Üniversitesi	.....
Üye Yrd. Doç. Dr. Erdal AYDEMİR Süleyman Demirel Üniversitesi	.....
Üye Yrd. Doç. Dr. Yusuf ŞAHİN Mehmet Akif Ersoy Üniversitesi	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 19/07/2017.. tarih ve ..28/18.... sayılı kararıyla onaylanmıştır.

.....

Prof. Dr. Uğur YÜCEL

Fen Bilimleri Enstitüsü Müdürü

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

NERİMAN İNAK

ÖZET

ALT SINIR TEMELİNE DAYALI AĞIRLIKLI TAVLAMA YÖNTEMİ İLE KUTULAMA PROBLEMİNİN ÇÖZÜMÜ
YÜKSEK LISANS TEZI
NERİMAN İNAK
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
(TEZ DANIŞMANI:PROF. DR. SEZAİ TOKAT)
(İKİNCİ DANIŞMAN:YRD. DOÇ. DR. KENAN KARAGÜL)
DENİZLİ, TEMMUZ - 2017

Bu tezde bir boyutlu kutulama problemi için melez (hibrit) yeni bir sezgisel çözüm yöntemi sunulmuştur. Önerilen yaklaşımda, başlangıç çözümü oluşturmak için alt sınıra dayalı sezgisel bir başlangıç çözüm algoritması önerilmiştir. Önerilen sezgisel ile birlikte literatürde yer alan diğer yerleştirme algoritmaları ele alınmış, elde edilen sonuçlar literatürde ulaşılan sonuçlarla karşılaştırılmıştır. Başlangıç çözümü sonrası elde edilen çözüme ağırlıklı tavlama yöntemiyle birlikte yer değiştirme algoritmaları uygulanmış ve kullanılan kutu sayısını minimize etmek amaçlanmıştır. Literatürde yer alan test kümeleri çözülmüş, çözüm süreleri ve elde edilen sonuçlar bilinen en iyi sonuçlarla ve geliştirilen diğer yöntemlerle karşılaştırılmıştır.

ANAHTAR KELİMELER: kutulama problemi, ağırlıklı tavlama, sezgisel algoritma

ABSTRACT

SOLUTION OF BIN PACKING PROBLEM WITH WEIGHTED ANNEALING METHOD BASED ON LOWER BOUND

MSC THESIS
NERİMAN İNAK
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
COMPUTER ENGINEERING
(SUPERVISOR:PROF. DR. SEZAİ TOKAT)
(CO-SUPERVISOR:ASSIST. PROF. DR. KENAN KARAGÜL)
DENİZLİ, JULY 2017

In this thesis, a heuristic solution method is presented for one dimensional bin packing problem. A heuristic initial solution algorithm based on the lower bound is proposed to create the initial solution. In addition to the proposed heuristics, other placement algorithms in the literature are discussed, and the results obtained are compared with the results obtained in the literature. Swap algorithms together with weighted annealing method are applied to the results of the initial solutions, and the number of bins used are minimized. The test sets in the literature are solved, the resolution times and the results obtained are compared with the best known solution in the literature and other developed methods.

KEYWORDS: bin packing problem, weighted annealing, heuristic algorithm

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

KISALTMA LİSTESİ

BPP : Bin Packing Problem
WA1BP : Weight Annealing Algorithm for the One-Dimensional Bin Packing Problem
1-D BPP : 1-Dimensional Bin Packing Problem
2-D BPP : 2-Dimensional Bin Packing Problem
3-D BPP : 3-Dimensional Bin Packing Problem
ISAA : İlk Sığan Azalan Algoritması
ESAA : En İyi Sığan Azalan Algoritması
MTP : Martello ve Toth Prosedürü
HGGA : Hibrid Gruplama Genetik Algoritma
H-SGGA: Hibrid Sezgisel Gruplama Genetik Algoritması
HCT : Hospital Community Territory
MAK : Minimum Aylak Kapasite
SMAK : Sarsımlı Minimum Aylak Kapasite
DKA : Değişken Komşuluk Araması
GMAK : Gevşetilmiş Minimum Aylak Kapasite
ÖMAK : Örneklemeli Minimum Aylak Kapasite
ISA : İlk Sığan Algoritması
ESA : En İyi Sığan Algoritması
EKSA : En Kötü Sığan Algoritması
AS_SBC : Alt Sınıra Dayalı Sezgisel Başlangıç Çözümü
EIBC : En İyi Bilinen Çözüm
BKS : Best Known Solution
ORT : Ortalama
MAKS : Maksimum
Sn : Saniye
LB : Lower Bound
VNS : Variable Neighborhood Search
PMBS : Perturbation Minimum Bin Slack
HI_BP : Hybrid Improvement Bin Packing

İNGİLİZCE - TÜRKÇE TERİM KARŞILIKLARI TABLOSU

Bin Packing Problem	**:** Kutulama Problemi
1-Dimensional Bin Packing Problem	**:** 1 Boyutlu Kutulama Problemi
2-Dimensional Bin Packing Problem	**:** 2 Boyutlu Kutulama Problemi
3-Dimensional Bin Packing Problem	**:** 3 Boyutlu Kutulama Problemi
First Fit Algorithm	: İlk Sığan Algoritması
Best Fit Algorithm	: En İyi Sığan Algoritması
Worst Fit Algorithm	: En Kötü Sığan Algoritması
First Fit Decreasing Algorithm	: İlk Sığan Azalan Algoritması
Best Fit Decreasing Algorithm	: En İyi Sığan Azalan Algoritması
First Fit Algorithm	: İlk Sığan Algoritması
Best Known Solution	: En İyi Bilinen Çözüm
Lower Bound	**:** Alt Sınır
Weighted Annealing	: Ağırlıklı Tavlama
Minimum Bin Slack (MBS)	**:** Minimum Aylak Kapasite (MAK)
Relaxed MBS	: Gevşetilmiş MAK
Sampling MBS	: Örneklemeli MAK
Perturbation MBS	: Sarsımlı MAK
Branch-and-Bound Algorithm	**:** Dal-ve-Sınır Algoritması
Branch-and-Price Algorithm	**:** Dal-ve-Fiyat Algortiması
Column Generation	: Sütun Oluşturma
Maximum Satisfaction Problem	**:** Maksimum Doyum Problemi
Knapsack Problem	: Sırt Çantası Problemi
Travelling Salesman Problem	: Gezgin Satıcı Problemi
Swap	: Yer Değiştirme
Variable Neighborhood Search	: Değişken Komşuluk Araması

SEMBOL LİSTESİ

$\mathbf{N_{LB}}$	**:**	Listedeki elemanların alt sınır değeri
S	**:**	Kutulanacak parçalar kümesi
C	**:**	Kutu kapasitesi
N	**:**	Test kümesinde bulunan örnek sayısı
$\boldsymbol{LB}$	**:**	Lower Bound
$\boldsymbol{l_i}$	**:**	i.kutudaki öğelerin boyutlarının toplamı
K	**:**	sabit
$\mathbf{r_i}$	**:**	i. kutunun kalan kapasitesi
$\mathbf{t_{ij}}$	**:**	i. kutudaki j. öğenin boyutu
$\mathbf{q_i}$	**:**	i. kutudaki öğelerin sayısı
T	**:**	Kontrol Parametresi (sıcaklık)
$\mathbf{w_i^T}$	**:**	Ağırlık

ÖNSÖZ

Bu çalışmada yardımlarını, desteklerini ve bilgi birikimlerini benden esirgemeyen danışmanlarım **Prof. Dr. Sezai TOKAT** ve **Yrd. Doç. Dr. Kenan KARAGÜL** hocalarıma, bu zorlu süreçte her zaman yanımda olan ve manevi desteğini hiçbir zaman esirgemeyen canım Aileme ve büyük sabırla çalışmalarıma katlanan sevgili arkadaşlarıma çok teşekkür ederim.

1. GİRİŞ

Kutulama problemi, çözümü polinom zamanlı olmayan NP-zor bir problemdir (Garey ve Johnson 1979). Bu problem gerçek hayatta lojistik, tekstil, bilgisayar bilimleri gibi pek çok alanda karşımıza çıkmaktadır. Günlük hayattaki amaç her zaman kârı maksimuma ulaştırmaktır. Bu nedenle literatürde pek çok matematiksel model ve sezgisel yöntem önerilmiştir. Küçük ölçekli problemler için kesin çözümler bulunabilirken büyük ölçekli problemler için kesin çözüm bulmak mümkün olmamaktadır. Büyük ölçekli problemler için en iyi çözümü bulmak amacıyla genetik algoritma, tavlama benzetimi, ağırlıklı tavlama, tabu araması gibi sezgisel yöntemler yaygın olarak kullanılmıştır.

Kutulama problemi boyutlarına göre bir boyutlu, iki boyutlu ve üç boyutlu olmak üzere üç gruba ayrılmaktadır. Kutulama problemi çözülürken boyutları ve diğer kriterler de göz önünde bulundurularak çözüm yöntemi seçilir. Boyutlarına göre sınıflandırıldığında her bir problem tipi için farklı sezgisel algoritmalar geliştirilmiştir. Bu çalışmada bir boyutlu kutulama problemi için ağırlıklı tavlama yaklaşımı önerilmiştir. Ağırlıklı tavlama yaklaşımı bir boyutlu kutulama problemi gibi kombinatoryal bir optimizasyon problemine uygulandığında arama alanının farklı noktalarında bozulmalar yaratarak arama alanını genişletmeyi amaçlamaktadır. Bozulmalar bir sonraki iterasyonda kazanılan analizlere dayalı ağırlık atamaları ile kontrol edilir. Ağırlık atama işlemleriyle birlikte yerel arama uygulanarak optimum çözüm elde edilmeye çalışılır.

İkinci bölümde kutulama problemi tanımlanmış, bu problem türlerinden olan bir boyutlu, iki boyutlu ve üç boyutlu kutulama problemleri açıklanmıştır. Literatürde kutulama problemi ile ilgili sık kullanılan çözüm yöntemleri ve geliştirilen sezgisel algoritmalar taranmış ve bu bölümde bu çalışmalara yer verilmiştir.

Üçüncü bölümde kutulama probleminde başlangıç çözümü için kullanılan yerleştirme algoritmaları yer almaktadır. Bu yöntemler kesin ve sezgisel algoritmalar şeklinde 2 grupta incelenmiş, sezgisel algoritmalarda ise 6 farklı yerleştirme

algoritmaları detaylı bir şekilde açıklanmıştır. Ayrıca bu yerleştirme algoritmaları farklı test kümelerine uygulanmış ve elde edilen sonuçlar grafikler ve şekillerle gösterilmiştir.

Dördüncü bölümde bu çalışmanın temelini oluşturan ağırlıklı tavlama yöntemi, tavlama süreci, amaç fonksiyonu ve yer değiştirme işlemlerine detaylı bir şekilde yer verilmiştir. Algoritmanın işleyişinin daha iyi anlaşılabilmesi için bu algoritma örnek bir test kümesine uygulanmış ve başlangıç çözümü, ağırlıklı tavlama, yer değiştirme işlemleri sırasıyla test kümesine uygulanmıştır.

Beşinci bölümde ise bu tezde gerek başlangıç çözümleri algoritmalarında gerekse ağırlıklı tavlama yöntemiyle birlikte oluşturulan sezgisel çözümde kullanılan test kümeleri tablo halinde gösterilmektedir.

Altıncı bölümde ise önerilen kutulama problemi için sezgisel yaklaşım ile Kok-Hua Loh'un (2006) çalışmasında yer alan WA1BP (The Weight Annealing Algorithm for the One-Dimensional Bin Packing Problem) sezgiseli karşılaştırılmış, bunun sonucunda elde edilen çalışma süreleri ve çözüm sonuçları tablolarda gösterilmiştir.

Son bölümde ise bu çalışmada elde edilen sonuçlarla ilgili yorumlara ve önerilere yer verilmiştir.

2. KUTULAMA PROBLEMİ

Kutulama problemi (BPP: bin packing problem), klasik sırt çantası problemleriyle bazı benzerlikleri olan kombinatoryal bir optimizasyon problemidir. Geleneksel sırt çantası probleminde, mevcut kapasitenin mümkün olduğu kadar verimli kullanılabilmesi için verilen boyutlarda nesneler yine verilen kapasitedeki bir kutu setine yerleştirilmelidir. Kutulama problemi ise basitçe, kutuların içerisine en az boş alan bırakarak, nesnelerin en iyi şekilde nasıl yerleştirileceği olarak düşünülebilir. Kutulama problemi bilgisayar bilimi ve mühendisliği, ulaşım, lojistik ve iletişim gibi birçok alanda kullanılmıştır (Hashim ve diğ. 2014). Konteynır yükleme, birden çok diske veri yerleştirme, iş planlaması, reklamları sabit uzunlukta radyo veya televizyon istasyonlarında paketleme, CD'lere büyük bir müzik koleksiyonu saklama gibi örneklendirilebilir.

Kutulama problemini boyutlarına göre incelendiğinde bir boyutlu, iki boyutlu ve üç boyutlu olmak üzere sınıflandırılır.

2.1 Bir Boyutlu Kutulama Problemi

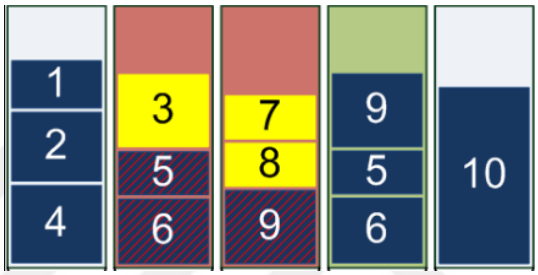
Bir boyutlu kutulama problemlerinde (1-D BPP : 1-dimensional bin packing problem); yüklemesi yapılacak olan nesnelerin sadece tek boyutları ele alınmakta ve buna bağlı olarak optimum yükleme yapılmaya çalışılmaktadır (Ensari 2007). Bu problemde amaç bir çizgi veya hat gibi görülebilecek yapının içerisine farklı boyutlardaki çizgileri yerleştirmek olarak düşünülebilir. Bir boyutlu kutulama probleminde, farklı hacimlerdeki nesneler kullanılan kutuların sayısını en aza indirecek şekilde C kapasiteli kutulara yerleştirilmelidir. Bu problemde yapılan yüklemelerin maksimize edilmesinin yanı sıra kullanılan kutu sayısını azaltmak amaçlanmıştır (Hashim ve diğ. 2014)

Örneğin ders programına derslerin verimli bir şekilde yerleştirilmesi bir boyutlu problem olarak değerlendirilebilir. Bu problemde her dersin farklı miktarda

zaman gerektirdiğini ve her güne ait sınırlı sayıda çalışma saati olduğunu düşünürsek en fazla dersin en az zamanda yapılması gerekmektedir.

Bilgisayar bilimlerinde hafızanın daha verimli kullanılması için yani bellek yönetimi için programları hafızanın içerisine en verimli şekilde yerleştirmek gerekmektedir. Bu tür problemler de bir boyutlu kutulama problemine örnek gösterilebilir.

Aşağıdaki Şekil 2.1 belirli bir yüksekliğe sahip olan nesnelerin kutulara yerleştirimesine örnek olarak verilebilir.



Şekil 2.1: Bir boyutlu kutulama problemi örneği [internet 3]

2.1.1 Bir Boyutlu Kutulama Problemi İçin Matematiksel Model

Bir boyutlu kutulama problemi aşağıdaki gibi tanımlanmıştır. Nesneler kümesi $L = \{1, \ldots, n\}$ ve bu nesnelerin ağırlıkları $W_i \in (0,1)$, $i \in L$ olarak verilsin. L kümesini minimum sayıda(m) $B_1, B_1, \ldots, B_m$ alt kümelerine aşağıdaki gibi bölmek istiyoruz.

$$\sum_{i \in B_{1j}} W_i \leq 1, \qquad 1 \leq j \leq m. \tag{2.1}$$

Buradaki amaç tüm öğeleri en az sayıda kutuya yerleştirmektir. Kutulama problemi için matematiksel model aşağıdaki gibidir (Delorme ve diğ. 2016).

Karar Değişkenleri :

$$y_j = \begin{cases} 1 & eğer\ kutu\ j\ kullanıldıysa \\ 0 & diğer\ durumlarda \end{cases}; \tag{2.2}$$

$$x_{ij} = \begin{cases} 1 & eğer\ i.nesne\ \ j.kutudaysa \\ 0 & diğer\ durumlarda \end{cases}; \tag{2.3}$$

$$min \sum_{j=1}^{n} y_j \tag{2.4}$$

$$\sum_{i=1}^{n} W_i x_{ij} \ \leq y_j, \qquad j = 1, \dots, n; \tag{2.5}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad i = 1, \dots, n; \tag{2.6}$$

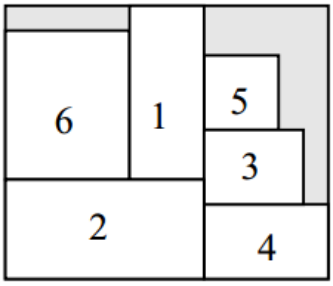
$$y_j x_{ij} \in \{0,1\}, \quad \mathrm{i,j} = 1, \dots, \mathrm{n}. \tag{2.7}$$

2.2 İki Boyutlu Kutulama Problemi

İki boyutlu kutulama probleminde (2-D BPP: 2-dimensional bin packing problem) cismin iki boyutu (en ve boy) ele alınmaktadır. Bu problemlerde yine amaç en az sayıda kutu kullanmak ve boş alanları minimum seviyeye indirmektir.

En yaygın olarak bilinen kesme ve yükleme problemleri buna örnektir (Ensari 2007). Depolama alanından tasarruf etmek için dikdörtgen şekilli ürünler raflara verimli bir şekilde yerleştirilmelidir. Yayıncılık endüstrilerinde makale ve reklamların asgari sayıda sayfaya bölünmesi gerekir (Loh ve diğ. 2006). Haberlerin gazeteye yerleştirilmesi örneğini ele alırsak haberin en ve boyundan bahsedilir ve problem iki boyutlu kutulama problemine örnek gösterilebilir.

Depoya ürün yerleştirme problemine örnek olarak verilen kutuların enine ve boyuna göre kutuya nasıl yerleştirildiğini gösteren şekil aşağıdaki gibidir (Şekil 2.2).



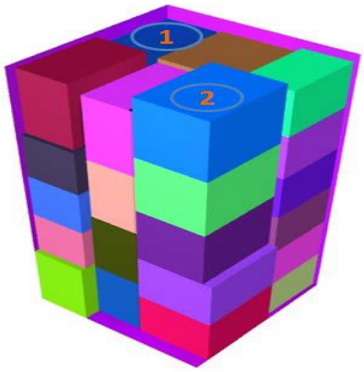
Şekil 2.2: İki boyutlu kutulama problemi örneği (Imahori ve diğ. 2007)

2.3 Üç Boyutlu Kutulama Problemi

Üç boyutlu kutulama problemi (3-D BPP : 3-dimensional bin packing problem) en zor kutulama problemi olarak tanımlanır. Bu problemde şeklin x, y ve z boyutları ele alınır. Genellikle konteynır yükleme problemi olarak karşımıza çıkmaktadır. 3 boyutlu kutulama problemi tüm kutuları ortogonal olarak en az sayıda kutuya yerleştirmekten oluşur (Martello ve diğ. 2000)

Lojistik uygulamalarında kullanılan konteynıra palet yerleştirme problemi üç boyutlu kutulama problemine örnek verilebilir. Burada cismin genişlik, yükseklik ve derinliği baz alınmaktadır. Bu tip problemlerde cismin döndürülüp döndürülemeyeceği de kriter olarak belirlenmektedir. Ayrıca cisimlerin girintili olup olmamasına göre de problemin zorluk derecesi artmaktadır.

Şekil 2.3'te üç boyutlu kutuların yerleştirilmesi örnek olarak verilmiştir.



Şekil 2.3: Üç boyutlu kutulama problemi örneği (Küçük 2010)

2.4 Literatür Taraması

Bir boyutlu kutulama problemi kombinatoryal optimizasyonda en ünlü problemlerden biridir. Yapılan araştırmalar ve geliştirilen yöntemler Tablo 2.1'de literatür haritası olarak gösterilmektedir.

Tablo 2.1'de görüldüğü gibi kutulama probleminin yapısı ve uygulamaları 1930'lu yıllardan beri araştırılmıştır. Bu problemin çözümü için ilk olarak matematiksel yöntemler önerilmiştir (Kantorovich 1960). Gilmore ve Gomory (1961), bu problem sınıfı için, Dantzig-Wolfe (1960) ve Ford-Fulkerson (1961) tarafından geliştirilen yöntemlerden yararlanarak, sütun oluşturma kavramını yöntemini geliştirmişlerdir. Optimal/optimale yakın sonuca ulaşmak için birçok sezgisel algoritma geliştirilmiştir. Coffman, Garey, ve Johnson (1997) tarafından yapılan bir literatür araştırmasında yaklaşım (approximation) algoritmalarının kapsamlı bir incelemesi yapılmıştır. İlk sığan azalan (first fit decreasing) algoritması (ISAA) (Eilon ve Christofides 1971; Johnson ve diğ. 1974) ve en iyi sığan azalan (best fit decreasing) algoritması (ESAA) (Johnson ve diğ. 1974) başlangıç çözümleri elde etmek için en sık kullanılan iki algoritmadır.

Martello ve Toth Prosedürü (MTP), dal-ve–sınır prosedürü bir boyutlu kutulama problemi için kullanılan en eski yöntemlerden biridir (Martello ve Toth 1987). Bu yöntem pek çok çalışmada karşılaştırma yapmak için referans olarak kullanılmıştır.

Hübscher ve Glover (1994) tarafından eşdeğer m işlemcide n görevin çizelgelenmesi probleminde boş zamanların minimizasyonu için bir Tabu Arama algoritması önerilmiştir. Bu çizelgeleme problemi çoklu kutulama probleminin bir çeşidine eşbiçimlidir. Araştırmacılar tarafından etkin bir çeşitlendirme yöntemine dayanan genişletilmiş Tabu Arama algoritması ile kutulama problemine oldukça kaliteli çözümler üretmişlerdir. Falkenauer (1996) kutulama problemi için hibrid gruplama genetik algoritması (HGGA) tanımlamıştır. Scholl ve diğ. (1997), çeşitli sınırları, indirgeme prosedürlerini, sezgisel yöntemleri ve yeni bir dal şemasını kullanarak bir dal ve sınır prosedürünü kullanan kesin bir yöntem (BISON) önermiştir.

Tablo 2.1: Literatür haritası

	Ana Akım	Gerçekleyen	Yöntem	Referans	Yıl
Kesin Yöntemler	Matemaiksel Model	L.V. Kantorovich	Matematiksel Model	(Kantorovich 1960)	1930
		M. Delorme, M.Iori, S.Martello	Matematiksel Model ve Kesin Algoritmalar	(Delorme ve diğ. 2016)	2016
	Sütun Oluşturma Algoritması (column generation algorithm)	P.C. Gilmore, R.E. Gomory	Doğrusal Programlama ve Sütun Oluşturma Algoritması	(Gilmore ve Gomory 1961)	1961
		J.M.V. Carvelho, F. Vanderbeck	Sütun Oluşturma Algoritması	(Carvelho 1999) (Vanderbeck 1999)	1999
	Martello ve Toth Prosedürü (MTP)	S. Martello, P. Toth	MTP ile Dal-ve-Sınır (Branch and Bound) Prosedürü	(Martello ve Toth 1987)	1987
		P. Schwerin, G. Wascher	Alt Sınıra Dayalı Yeni MTP	(Schwerin ve Wäscher 1999)	1999
Sezgisel Yöntemler	Tabu Arama	R. Hübscher, F. Glover	Tabu Arama Algoritması	(Hübscher ve Glover 1994)	1994
	Genetik Algoritma	E. Falkenauer	Hibrid Gruplama Genetik Algoritması (HGGA)	(Falkenauer 1996)	1996
		A.K.Bhatia, S.K.Basu	Çok Kromozomlu Genetik Gösterim Yaklaşımı	(Bhatia ve Basu 2004)	2004
		A. Singh, A.Gupta	Hibrid Kararlı Durum Gruplamalı Genetik Algoritma (Hybrid Steady-State Grouping Genetic Algorithm (H-SGGA)	(Singh ve Gupta 2007)	2007
		R. Poli, J. Woodward, E.K. Burke	Doğrusal Kayıt Tabanlı Genetik Algoritma	(Poli ve diğ. 2007)	2007
		P. Rohlfshagen, J. Bullinaria	Sezgisel Genetik Algoritma	(Rohlfshagen ve Bullinaria 2007)	2007

Tablo 2.1: Literatür haritası (devam)

Sezgisel Yöntemler	Hibrid Sezgiseller	A.Scholl, R.Klein, C.Jürgens	BISON: Kesin Hibrid Çözüm Prosedürü	(Scholl ve diğ. 1997)	1997
		A.Alvim, C.Ribeiro, F.Glover, D.J. Aloise	Hibrid İyileştirme Sezgiseli	(Alvim ve diğ. 2001,2004)	2001,2004
		S. Fekete, J. Schepers	Yeni Alt Sınır Sınıfı Sezgiseli	(Fekete ve Schepers 2001)	2001
		M. Gourgand, N.Grangeon, N.Klement	HCT Problemi (Hospital Community of Territory)	(Gourgand ve diğ. 2014)	2014
		S. Ozcan, T. Dokeroglu, A. Cosar, A. Yazici	Grafik İşlemci Birimi üzerinde Yeni Bir Gruplama Genetik Algoritması	(Ozcan ve diğ. 2016)	2016
	Minimum Aylak Kapasite (minimum bin slack) (MAK)	J. Ho, J. Gupta	MAK	(Ho ve Gupta 1999)	1999
		K. Fleszar, K.Hindi	Sarsımlı Minimum Aylak Kapasite (SMAK) (perturbation MBS) + Değişken Komşuluk Araması (variable neighborhood search) (DKA)	(Fleszar ve Hindi 2002)	2002
		A.Caprara, U.Pferschy	MAK	(Caprara ve Pferschy 2004,2005)	2004, 2005
	Ağırlıklı Tavlama Algoritması	K. Loh, B. Golden, E. Wasil	Ağırlıklı Tavlama Algoritması (Weighted Annealing Algorithm)	(Loh ve diğ. 2006)	2006
Genel Değerlendirme	Kesin Algoritmalar ve Sezgisel Algoritmalar	N. Hashim, F.Zuklipli, S.Januri, S.Shariff	Kesin Algoritma ve Sezgisel Algoritmaların Karşılaştırılması	(Hashim ve diğ. 2014)	2014

Carvelho (1999) ve Vanderbeck (1999) sütun oluşturma ve dal-ve–sınıra dayalı kesin algoritmalar sunmuşlardır. Daha sonra Schwerin ve Wäscher (1999), kullanılacak kutu sayısının stok kesme probleminden türetilmiş yeni bir alt sınırı kullanarak MTP'nin çok daha iyi sonuçlar verdiğini göstermiştir. Ho ve Gupta (1999) yaptığı bir çalışmada ise kutularda kalan toplam ağırlığı en aza indirmek için minimum aylak kapasite (MAK) algoritmasını kullanmıştır. Çalışmada ISAA, ESAA ve MAK algoritmaları karşılaştırılmıştır.

Fekete ve Schepers (2001), kutulama problemi için eşlenik uygunluk fonksiyonlarına (dual feasible functions) dayanan bir alt sınır sınıfı sunmuştur. Bu çalışmada, eşlenik uygunluk fonksiyonlarına dayalı olarak alt sınırlar elde etmek için basit ve hızlı bir genel yaklaşım sunulmaktadır. Fleszar ve Hindi (2002), kutulama problemi için birkaç yeni sezgisel yöntem önermiştir; bunlardan en etkili olanı değişken komşuluk araması (VNS : Variable neigborhood search) (DKA) (Hansen ve Mladenović 1999) meta-sezgiseli ile Fekete ve Schepers (2001) tarafından önerilen yeni alt sınırların kullanılmasına dayanan bir yöntemdir. Önerilen bu sezgiselde sarsımlı minimum aylak kapasite (SMAK) ve DKA bir arada kullanılmaktadır. Elde edilen sonuçlar standart MAK, yeni MAK tabanlı sezgisel, gevşetilmiş (relaxed) MAK (GMAK), SMAK ve örneklemeli (sampling) MAK (ÖMAK) algoritmalarıyla karşılaştırılmıştır. MAK ve sezgisel MAK gibi yaklaşımların en kötü performansı ise Caprara ve Pferschy (2004 ,2005) tarafından incelenmiştir.

Alvim, Ribeiro, Vanderbeck ve Alosie (2001) alt sınır tekniklerini içeren sofistike bir prosedür olan hibrid iyileştirme sezgiseli önermiştir. Bu çalışmaya ek olarak yine Alvim ve arkadaşları 2004 yılında yaptığı bir çalışmada önerdikleri hibrid iyileştirme sezgiselini eşlenik kutulama problemi için kullanmışlardır. (Alvim ve diğ. 2004). Bhatia ve Basu (2004), çok kromozomlu genetik gösterim yaklaşımı ve daha iyi sığan bir sezgisel yöntem sunmuştur. Genetik algoritmayı kullanarak kutulama problemini çözmek için çok kromozomlu bir genetik kodlama ve küme tabanlı genetik operatörler önermişlerdir. Daha iyi sığan sezgisel yöntem önerilmiştir; burada, kutuyu daha iyi doldurabiliyorsa, soldan çıkmış bir nesne, bir kutudan mevcut bir cisimle yer değiştirir. Daha iyi sığan sezgisel yöntemle zenginleştirilmiş genetik algoritmanın performansı HGGA ile karşılaştırılmıştır.

Loh ve diğ. (2006) ise ağırlıklı tavlama yöntemini kullanarak yeni bir sezgisel önermişlerdir. Bu çalışmada başlangıç çözümü olarak ISAA kullanılmış, ağırlıklı tavlama kullanılarak kutularda yer değiştirme yönteminin uygulanmasıyla optimum sonuç bulunmaya çalışılmıştır. Bu yöntem hibrid iyileştirme sezgiseli ile karşılaştırılmıştır (Loh ve diğ. 2006). Singh ve Gupta yaptıkları çalışmada genetik algoritmayı ve sezgisel yaklaşımı birleştiren hibrid kararlı durum gruplamalı genetik algoritma (hybrid steady-state grouping genetic algorithm) H-SGGA + SMAK yöntemi ile diğer MAK algoritmalarını karşılaştırmıştır Singh ve Gupta 2007). 2007 yılında evrimsel algoritmalar Poli (2007) tarafından incelenmiştir. Yapılan çalışmada nesne-boyut histogramıyla kutu-boşluk histogramını eşleştirme kavramına dayanan yeni bir algoritma sunulmuştur. Yaklaşım, histogramlar arasındaki farkı en aza indirmek için öğelerin nasıl önceliklendirileceğine karar veren yapıcı bir sezgisel işlevle kontrol edilir. Böyle bir fonksiyon doğrusal kayıt tabanlı genetik programlama sistemi kullanarak geliştirilmiştir. Geliştirilen bu yöntem ESAA ile karşılaştırılmıştır. Yine evrimsel algoritmalarla ilgili Rohlfshagen ve Bullinaria (2007) de çalışma yapmıştır.

Hashim ve diğ. yaptığı çalışmada da bir boyutlu kutulama probleminin çözümü için kesin (exact) algoritma, rastgele (random) algoritma, ilk sığan (first fit) algoritması (ISA) , en iyi sığan (best fit) algoritması (ESA), ISAA ve ESAA karşılaştırılmıştır (Hashim ve diğ. 2014).

Gourgand ve diğ. (2014) tarafından yapılan bir çalışmada HCT (Hospital community of Terriority) problemi olarak adlandırılan eylem planlaması ve kaynak ataması sorununu modellemek için kutulama problemi çözüm yaklaşımı kullanılmıştır. HCT, Fransa'da kurulan bir topluluktur ve farklı yerlerdeki kaynakların organize edilmesi ve planlanması ile kaynak kullanım verimliliğini artırmayı amaçlamaktadır. Burada geliştirilen düşünce, tıp alanındaki sınavların farklı coğrafi bölgelere ve dönemlere atanması sağlanarak kaynakların küresel etkinliğinin geliştirilmesini sağlamaktır. Bu problemde amaç her bir eylemi bir kaynağa atamak ve bir periyoda atamaktır. Bu problemde amaç eylemlerin mümkün olan en kısa sürede yapılmasının sağlanması ve bu sayede tahsis edilen sürelerin en küçük olmasını gerçeklemektir. Bu çalışmada çözüm yöntemi olarak meta-sezgisel ve sığan algoritmalarının (sonraki sığan yer algoritması (next fit algorithm), ISA,

ESA) birleşiminden oluşan hibrid bir yöntem kullanılmıştır (Gourgand ve diğ. 2014). HCT problemi ile kutulama problemleri arasındaki benzerlikler Tablo 2.2'de gösterilmiştir (Gourgand ve diğ. 2014).

Tablo 2.2: HCT problemi ile kutulama problemi arasındaki benzerlikler (Gourgand ve diğ. 2014)

	Kutulama Problemi	***Eylem Planlaması ve kaynak Ataması Problemi***
Veri	Nesne	Eylem
	Kutu	Çift (kaynak, periyot)
	Nesne boyutu	Eylemin işlem süresi
Amaç	Kutulara nesneleri atamak	Periyotlara ve kaynaklara eylem atamak
Kısıtlamalar	Kutu kapasitesi	Kaynak açık zamanı
	-	Uyumsuzluk kısıtı
Kriter	Kullanılan kutu sayısını en aza indirgemek	Kullanılan çift (kaynak, periyot) sayısını en aza indirgemek

2016 yılında Delorme, Iori ve Martello (2016) kutulama probleminin tam olarak çözümü için geliştirilen en önemli matematiksel modelleri ve algoritmaları gözden geçirmek ve mevcut en yeni yazılım araçlarının performansını en son teknoloji bilgisayarlarda deneysel olarak değerlendirmek amacıyla bir çalışma yapmıştır. Yine 2016'da Ozcan ve arkadaşları yaptıkları bir çalışmada CUDA kullanarak grafik işlem biriminin gücünden yararlanan gruplama genetik algoritmasını önermişlerdir.

3. BAŞLANGIÇ ÇÖZÜMÜ İÇİN KULLANILAN ÇÖZÜM YÖNTEMLERİ

Bir boyutlu kutulama probleminde başlangıç çözümü oluşturmak için kullanılan kesin yöntemler ve sezgisel algoritmalar aşağıdaki gibi ele alınmıştır.

3.1 Kesin Yöntemler

Kesin yöntemler çeşitli problemlerin çözümünde kullanılan optimal çözümü bulmayı garanti eden yöntemlerdir. Ancak kesin yöntemler hesaplama zamanı ve sarf edilen çabanın (zaman ya da maliyet) fazla olması nedeniyle her zaman tercih edilmez (Güler 2008). Kesin yöntemler aşağıdaki gibi sayılabilir.

- Dal-ve-sınır Algoritması
- Sütun Oluşturma
- Dal-ve-fiyat (branch-and-price) Algoritması

3.1.1 Dal-ve-Sınır Algoritması

Dal-ve-sınır algoritması (Land ve Doig 1960) en iyi tamsayılı çözümün bulunması için kullanılan genel amaçlı bir ağaç arama yöntemidir (Kabak 2012). Dal-ve-sınır algoritması, basitçe verilen bir fonksiyon için optimum çözümü bulmayı amaçlar. Dal-ve-sınır algoritması tamsayılı değişkenlerin alabileceği tamsayılar kümesindeki en küçük ve en büyük değerler arasında herhangi bir reel sayıyı alabilecek şekilde gevşetilmesiyle elde edilen problemlerin çözümü esasına dayanır. Her problemin çözümü sonucunda en iyi çözüm için bir sınır elde edilir. Ayrıca, tamsayılı değer alması gereken değişkenler sistematik bir şekilde her problemin sonucuna göre değerlendirilerek yeni düğümler oluşturulur. Dal-ve-sınır algoritmasında her dal aşağıdaki durumlarda sonlandırılır:

1. Tamsayılı değer alması gereken değişkenlerin tümü tamamen veya kısmen gevşetilmiş problemde tamsayılı değer alır ve bu çözüme tamsayılı çözüm adı verilerek en iyi amaç fonksiyonu değeri gerekli olursa güncellenir,
2. Bir düğümdeki amaç fonksiyonu değeri güncel olan en iyi amaç fonksiyonu değerinden daha kötüdür,
3. Bir noddaki problemin çözümü kısıtlardan bazılarını sağlamamaktadır. Bu kriterlere uyularak problem için dal-sınır ağacı oluşturulur ve tüm dallar sonlandırıldıktan sonra en iyi çözüme ulaşılır (Türkay 2006).

Dal-ve-sınır algoritmasının kullanıldığı bazı problem tipleri aşağıda sıralanmıştır:

- Maksimum Doyum Problemi (Maximum Satisfaction Problem)
- Sırt Çantası Problemi (Knapsack Problem)
- Gezgin Satıcı Problemi (Travelling Salesman Problem)

3.1.2 Sütun Oluşturma

Sütun oluşturma, büyük doğrusal ve tamsayılı programlama problemlerini çözmek için kullanılan bir tekniktir. Genel olarak, tüm sütun oluşturma teknikleri, ana problemler olarak adlandırılan doğrusal programlama modelleri ile bunlara karşılık gelen fiyatlandırma alt problemleri arasındaki ilişkilerin kurulmasını gerektirir. Ana sorunun yapısı ve buna karşılık gelen fiyatlandırma alt problemi ne olursa olsun ana sorun her zaman yeni modellerin üretilmesi için fiyatlandırma alt sorununa dayanır. Fiyatlandırma alt problemi belirli bir sonlandırma şartı doldurulunca kadar çağrılır (Ataei 2015).

Sütun oluşturma, kısıtlamalı ana problem olarak adlandırılan modele basit uygun bir çözüm içeren bir problemin tanımlanmasıyla başlar. Daha sonra kısıtlanmış ana sorun, en iyi duruma getirilmesi için gözden geçirilmiş bir simpleks yöntemi ile çözülür ve eşlenik değişkenler, tek boyutlu sırt çantası sorununun maliyet

fonksiyonu olarak kabul edilir. Oluşturulan sütun daha sonra sınırlı ana soruna eklenir ve gözden geçirilmiş tek yönlü yöntemin bilgileri güncellenir (Ataei 2015).

3.1.3 Dal-ve-Fiyat Algoritması

Dal-ve-fiyat terimi sütun oluşturmadan uyarlanmış olup dal-ve-sınır kategorisine aittir. Dal-ve-fiyat algoritması sütun oluşturmayla kök düğümdeki kutulama probleminin formülasyonunun tamsayılı olmayan gevşetmesini çözerek başlar ve daha sonra optimal çözümü aramak için dal-ve-sınır prosedürü çağrılır. Ağacın diğer tüm düğümlerinde, düğümlerin alt sınırını bulmak için sütun oluşturma yöntemi tekrar kullanılır. Dal-ve-fiyat işleminin performansı, her düğümde bulunan üst ve alt sınırların kalitesine bağlıdır ve daha önemlisi verimliliği dallanma ve arama stratejilerine dayanır (Ataei 2015).

3.2 Sezgisel Algoritmalar

Sezgisel algoritmalar bir problemin çözümünde kullanılan, optimal/optimale yakın çözüm yolları elde etmeyi amaçlayan teknikdir. Sezgisel algoritmalar genellikle optimal/optimale yakın çözüme hızlı ve kolay şekilde ulaşırlar. Kutulama problemi için de başlangıç çözümü oluşturmakta çeşitli sezgisel algoritmalar kullanılır. Kutulama probleminde kullanılan en yaygın sezgisel algoritmalar aşağıda verilmiştir.

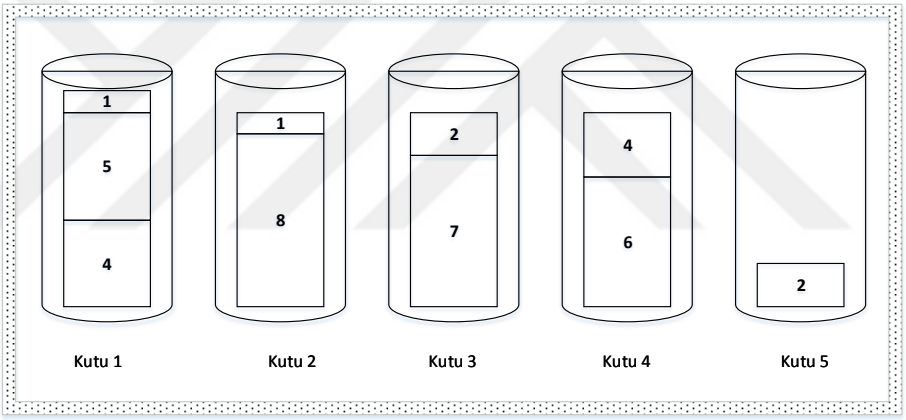
3.2.1 İlk Sığan Algoritması

ISA'da listedeki elemanlar varolan sıraya göre baştan sona taranır. Listedeki ilk eleman ilk kutuya yerleştirilir. Listedeki her eleman yerleştirilirken her zaman ilk kutudan başlayarak yerleştirme yapılır, eğer sığmıyorsa bir sonraki kutuya geçilir. Eğer mevcut kutulardaki boş alan elemanın boyutundan daha küçük ise yani eleman o kutuya sığmıyorsa yeni bir kutu açılır ve eleman o kutuya yerleştirilir. Algoritma 1'de ISA'nın sözde kodu verilmiştir.

Örnek olarak kutulanacak parçalar kümesi *S = {4, 8, 5, 7, 6, 1, 1,4, 2, 2 }* ve kutu kapasitesi *C = 10* alındığı durum için Algoritma 1'de verilen ISA'ya göre çözüm adımları Şekil 3.1'de gösterilmektedir.

Algoritma 1 : ISA Sözde Kodu

```
1: for Tüm nesneler i = 1,2...,n do
2:       for Tüm kutular j = 1,2,… do
3:       if i elemanı j kutusuna sığıyorsa then
4:                i elemanını j kutusuna yerleştir.
5:                Döngüden çık ve diğer elemanı yerleştir.
6:       end if
7:     end for
8:       if i elemanı hiçbir kutuya sığmıyorsa
9:                yeni kutu aç ve i elemanını yerleştir.
10:      end if
11:  end for
```

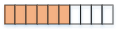
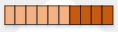


Şekil 3.1: ISA çözüm sonucu

ISA için adım adım çözüm safhaları Tablo 3.1'de gösterilmektedir. Bu tabloda ilk sütunda; listede yer alan parçalar doğal veriliş sırasında gösterilmiştir. En son sütunda işlem adımları kodlanmıştır. Arada kalan sütunlarda ise kutular için algoritmanın yerleştirme süreçleri gösterilmiştir. Örneğin; ilk sütunda yer alan ilk eleman "4" parçası ilk açılan kutuya yerleştirilmektedir. Bu işlem en sonda yer alan işlemler sütununda T1 olarak gösterilmiştir. Tablodaki akış sürecini gösteren kutu simgeleri kutu kapasitelerini ifade edecek şekilde birimlere bölünmüştür. Bu birim bölümleri kutuya yerleştirilen birimleri ifade edecek şekilde renklendirilmiştir. Renksiz olan birimler boş kalan kapasitenin ölçüsünü göstermektedir. "Son Durum" satırında yer alan kutular ise algoritmanın sonunda bütün kutuların durumunu ifade etmektedir. Tablonun son satırında bulunan "Kalan Kapasite" kısmı ise yerleştirme

işlemi sonucunda her bir kutuda kaç birim boş kapasite kaldığını göstermektedir. Aşağıda verilen diğer başlangıç algoritmalarında yer alan tablolar da aynı şekilde yapılandırılmıştır.

Tablo 3.1: ISA çözüm safhaları gösterimi

Elemanlar	Kutu 1	Kutu 2	Kutu 3	Kutu 4	Kutu 5	Sıra
4						T1
8						T2
5						T3
7						T4
6						T5
1						T6
1						T7
4						T8
2						T9
2						T10
Son Durum						
Kalan Kapasite	0	1	1	0	8	

Tablo 3.2'de ise ISA'nın kutulara yerleştirme adımları gösterilmektedir.

Tablo 3.2: ISA çözüm adımları

T1	Listenin ilk sırasında yer alan 4 birimlik eleman ilk kutuya yerleştirilir. Bu işlem sonrasında ilk kutunun kalan kapasitesi 6'dır.
T2	8 birimlik eleman ilk kutuya sığmadığı için 2. kutu açılır ve buraya yerleştirilir.
T3	3.sıradaki 5 birimlik eleman yerleştirilirken ilk kutudan itibaren mevcut kutular kontrol edilir. 5 birimlik eleman ilk kutuya sığabildiği için buraya yerleştirilir.
T4	7 birimlik eleman 1.ve 2.kutuya sığmadığı için yeni bir kutu açılarak 3.kutuya yerleştirilir.
T5	6 birimlik eleman mevcut kutulara sığmadığı için yeni bir kutu açılarak 4.kutuya yerleştirilir.
T6	1 birimlik eleman yine baştan itibaren kutular kontrol edilerek ilk kutuya sığdığı için buraya yerleştirilir.
T7	Sıradaki 1 birimlik eleman ilk kutu dolduğu için 2.kutuya yerleştirilir.
T8	4 birimlik eleman ilk 3 kutuya sığmadığı için 4.kutuya yerleştirilir.
T9	2 birimlik eleman 3.kutuya yerleştirilir.
T10	Son adımda ise listede kalan 2 birimlik eleman mevcut kutulara sığmadığı için yeni kutu açılarak yerleştirilir.

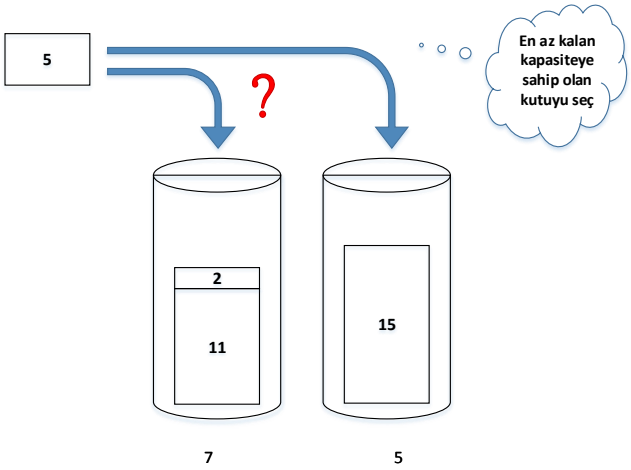
3.2.2 En İyi Sığan Algoritması

Bu algoritmaya göre elemanlar listedeki sıraya göre baştan sona taranır. Listedeki eleman için gerekli olan, boyutuna en uygun ve en küçük boyutlu boş kutuyu bulur. Bulmuş olduğu kutuya bu elemanı yerleştirir. Burada amaç yerleştirme yapılırken minimum aylak kapasite kalacak şekilde yerleştirme yapılmasıdır. Eğer listedeki eleman hiçbir kutuya sığmıyorsa yeni kutu açılır ve eleman o kutuya yerleştirilir. Aşağıdaki Algoritma 2'de ESA'nın sözde kodu verilmiştir.

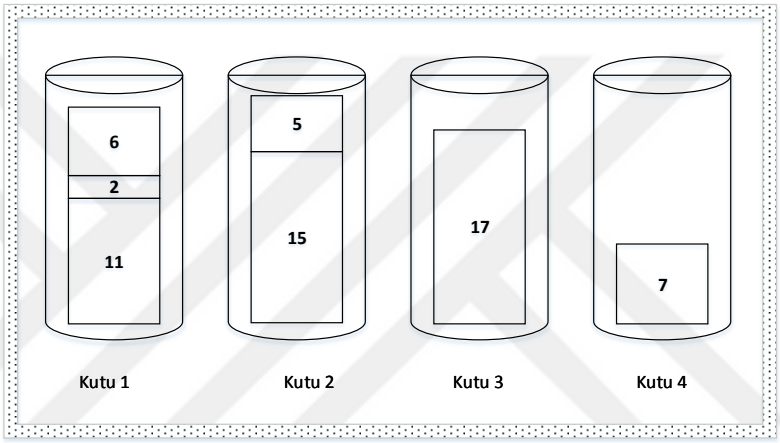
Algoritma 2: ESA Sözde Kodu

```
1: for Tüm nesneler i = 1,2...,n do
2:       for Tüm kutular j = 1,2,… do
3:       if i elemanı j kutusuna sığıyorsa then
4:               Nesne eklendikten sonra kalan kapasiteyi hesapla
5:       end if
6:    end for
7:       Nesne ekledikten sonra j kutusu minimum kalan kapasiteye sahipse i
         elemanını j kutusuna ekle.
8:       Eğer böyle bir kutu yoksa yeni kutu aç ve nesneyi yeni kutuya ekle.
9: end for
```

Örnek olarak kutulanacak parçalar kümesi $S = \{11, 2, 15, 5, 6, 17, 7\}$ ve kutu kapasitesi $C = 20$ varsayılsın. Bu durumda Algoritma 2'de verilen ESA'ya göre çözüm sonucu oluşan kutulardaki yerleşim durumu Şekil 3.2 ve Şekil 3.3'te; algoritmanın çözüm adımları Tablo 3.3 ve çözüm adımlarının açıklamaları Tablo 3.4'te gösterilmektedir.



Şekil 3.2: ESA çözümü(1)



Şekil 3.3: ESA çözümü(2)

Tablo 3.3: ESA çözümü tablo gösterimi

Elemanlar	Kutu 1	Kutu 2	Kutu 3	Kutu 4	Sıra
11					T1
2					T2
15					T3
5					T4
6					T5
17					T6
7					T7
Son Durum					
Kalan Kapasite	1	0	3	13	

Tablo 3.4: ESA çözüm adımları

T1	İlk adımda 11 birimlik eleman ilk kutuya yerleştirilir.
T2	2 birimlik eleman ilk kutuya sığdığı için buraya yerleştirilir. 2.kutunun kalan kapasitesi 7'dir.
T3	3.adımda 15 birimlik eleman ilk kutunun kalan kapasitesini aştığı için yeni bir kutu açılarak 2.kutuya yerleştirilir.
T4	5 birimlik eleman yerleştirilirken mevcut kutular baştan itibaren kontrol edilir. Bu eleman ilk kutuya sığabilir ancak ESA'nın amacı en az artık kapasiteye sahip olacak kutuya yerleştirme yapmaktır. 5 birimlik eleman ilk kutuya yerleştirilirse ilk kutunun kalan kapasitesi 2 olur. 2.kutuya yerleştirilirse 2.kutunun kalan kapasitesi 0 olur, 2.kutunun kalan kapasitesi daha az olduğu için eleman 2.kutuya yerleştirilir.
T5	6 birimlik eleman ilk kutuya sığdığı için ve 2.kutu dolu olduğu için ilk kutuya yerleştirilir.
T6	17 birimlik eleman mevcut kutulara sığamadığı için yeni bir kutu açılarak 3.kutuya yerleştirilir.
T7	7 birimlik son eleman mevcut kutulara sığamadığı için yeni bir kutu açılarak 4.kutuya yerleştirilir. Listedeki elemanlar toplamda 4 kutuya yerleştirilmiş olur.

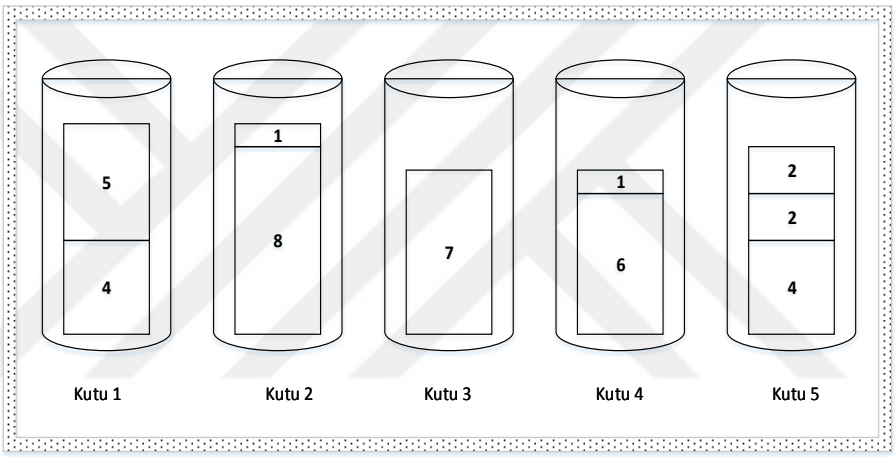
3.2.3 En Kötü Sığan Algoritması

En kötü sığan (worst fit) algoritması (EKSA) ESA'nın tam tersi olarak nitelendirilebilir. EKSA'ya göre elemanlar listedeki sıraya göre baştan sona taranır. Listedeki eleman için gerekli olan, boyutuna en uygun ve en büyük boyutlu boş kutuyu bulur. Bulmuş olduğu kutuya bu elemanı yerleştirir. Burada amaç yerleştirme yapılırken maksimum aylak kapasite kalacak şekilde yerleştirme yapılmasıdır. Eğer listedeki eleman hiçbir kutuya sığmıyorsa yeni kutu açılır ve eleman o kutuya yerleştirilir. EKSA'da öğeler en boş bölmeye yerleştirilir. Bu algoritma yaklaşık olarak aynı ağırlığa sahip paket ya da aynı değere sahip ürün ile doldurulmak için arzu edilen bir durumdur. Algoritma 3'te EKSA'nın sözde kodu verilmiştir.

Örnek olarak kutulanacak parçalar kümesi $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ ve kutu kapasitesi $C = 10$ varsayılsın. Algoritma 3'te verilen EKSA'ya göre parçaların kutulara yerleşimi Şekil 3.4'te; algoritmanın çözüm adımları Tablo 3.5 ve çözüm adımlarının açıklamaları Tablo 3.6'da gösterilmektedir.

Algoritma 3: EKSA Sözde Kodu

1: **for** Tüm nesneler i = 1,2...,**n do**
2: **for** Tüm kutular j = 1,2,… **do**
3: **if** i elemanı j kutusuna sığıyorsa **then**
4: Nesne eklendikten sonra kalan kapasiteyi hesapla
5: **end if**
6: **end for**
7: Nesne ekledikten sonra j kutusu maksimum kalan kapasiteye sahipse i elemanını j kutusuna ekle.
8: Eğer böyle bir kutu yoksa yeni kutu aç ve nesneyi yeni kutuya ekle.
9: **end for**



Şekil 3.4: EKSA çözümü

Tablo 3.5: EKSA çözümü tablo gösterimi

Elemanlar	Kutu 1	Kutu 2	Kutu 3	Kutu 4	Kutu 5	Sıra
4						T1
8						T2
5						T3
1						T4
7						T5
6						T6
1						T7
4						T8
2						T9
2						T10
Son Durum						
Kalan Kapasite	1	1	3	3	2	

Tablo 3.6: EKSA çözüm adımları

T1	Listenin ilk sırasında yer alan 4 birimlik eleman ilk kutuya yerleştirilir. Bu işlem sonrasında ilk kutunun kalan kapasitesi 6'dır.
T2	8 birimlik eleman ilk kutuya sığmadığı için 2. kutu açılır ve buraya yerleştirilir.
T3	3.sıradaki 5 birimlik eleman yerleştirilirken ilk kutudan itibaren mevcut kutular kontrol edilir. 5 birimlik eleman ilk kutuya sığabildiği için buraya yerleştirilir.
T4	1 birimlik elaman için mevcut kutular kontrol edilir. Bu eleman 2 kutuya da sığmaktadır ancak EKSA'nın amacı en fazla artık kapasite kalacak kutuya elemanı yerleştirmektir. İlk kutuya yerleştirilirse 0, 2.kutuya yerleştirilirse 1 birimlik kapasite kalacağı için 2.kutuya yerleştirilir.
T5	7 birimlik eleman mevcut kutulara sığmadığı için yeni bir kutu açılarak 4.kutuya yerleştirilir.
T6	6 birimlik eleman mevcut kutulara sığmadığı için yeni bir kutu açılarak 5.kutuya yerleştirilir.
T7	Sıradaki 1 birimlik eleman 4.kutuya yerleştirildiğinde daha fazla artık kapasite kaldığı için buraya yerleştirilir.
T8	4 birimlik eleman mevcut kutulara sığmadığı için yeni bir kutu açılarak 5.kutuya yerleştirilir.
T9	Sıradaki 2 birimlik eleman 5.kutuya yerleştirildiğinde daha fazla artık kapasite kaldığı için buraya yerleştirilir.
T10	Aynı şekilde kalan 2 birimlik eleman artık kapasitesi en fazla olan son kutuya yerleştirilir.

3.2.4 İlk Sığan Azalan Algoritması

Bu algoritmada listedeki elemanlar azalan sırada sıralanır ve yeni liste oluşturulur. Yeni oluşturulan listedeki elemanlara "ISA" uygulanarak çözüme ulaşılır. ISAA'nın sözde kodu Algoritma 4'te verilmiştir.

Algortima 4: ISAA Sözde Kodu

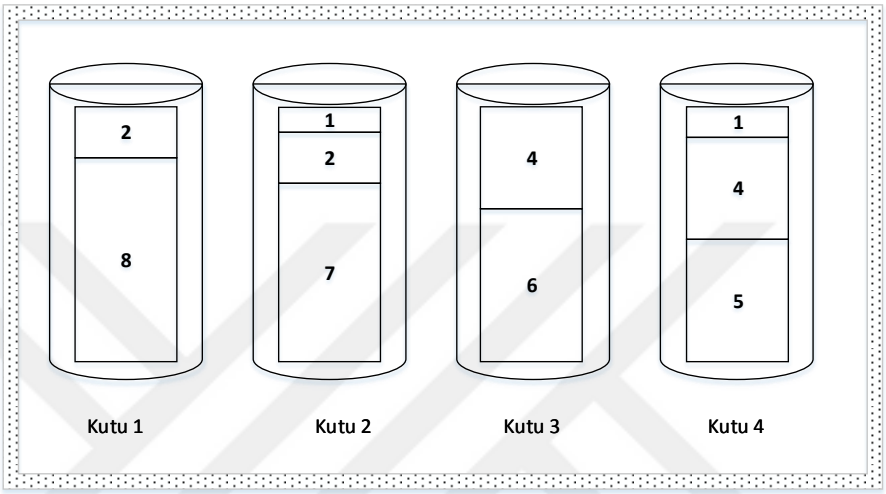
```
1: Listedeki elemanlar azalan sırada sıralanır
2: Sıralanmış listeye ISA uygulanır.
```

Örnek olarak kutulanacak parçalar kümesi $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ ve kutu kapasitesi $C = 10$ varsayılsın. Algoritma 4'te verilen ISAA'ya göre parçaların

kutulara yerleşimi Şekil 3.5'te; algoritmanın çözüm adımları Tablo 3.7 ve çözüm adımlarının açıklamaları Tablo 3.8'de gösterilmektedir.

ISAA algoritmasına göre, ilk adımda *S* kümesindeki elemanlar azalan sıraya göre dizilir.

S' = {8, 7, 6, 5, 4, 4, 2, 2, 1, 1}, *S'* kümesine ISA uygulanır (Şekil 3.5).



Şekil 3.5: ISAA çözümü

Tablo 3.7: ISAA çözümü tablo gösterimi

Elemanlar	Kutu 1	Kutu 2	Kutu 3	Kutu 4	Sıra
8					T1
7					T2
6					T3
5					T4
4					T5
4					T6
2					T7
2					T8
1					T9
1					T10
Son Durum					
Kalan Kapasite	0	0	0	0	

Tablo 3.8: ISAA çözüm adımları

T1	ISAA'ya göre listedeki elemanlar büyükten küçüğe sıralanır ve bu listedeki elemanlar ISA'ya göre yerleştirilir. Buna göre ilk olarak listedeki 8 birimlik elaman ilk kutuya yerleştirilir.
T2	2.sıradaki 7 birimlik eleman ilk kutuya sığmadığı için 2. Kutu açılır ve buraya yerleştirilir.
T3	6 birimlik eleman mevcut kutulara sığmadığı için 3. Kutu açılır ve bu kutuya yerleştirilir.
T4	5 birimlik eleman mevcut kutulara sığmadığı için yeni bir kutu açılarak 4.kutuya yerleştirilir.
T5	4 birimlik elemanın yerleştirilmesi için kutular baştan itibaren taranır 3.kutuda yeterli kapasite olduğu için buraya yerleştirilir.
T6	4 birimlik eleman 4.kutuda yeterli kapasite olduğu için buraya yerleştirilir.
T7	Sıradaki 2 birimlik eleman ilk kutuya sığdığı için buraya yerleştirilir.
T8	2 birimlik eleman ilk kutu dolduğu için ve 2.kutuda yeterli alan olduğu için 2.kutuya yerleştirilir.
T9	1 birimlik eleman yine 2.kutuya yerleştirilir.
T10	Kalan 1 birimlik eleman sadece son kutuda yer kaldığı için 4.kutuya yerleştirilir. Böylece listedeki elemanlar hiç artık kapasite kalmayacak şekilde toplamda 4 kutuya yerleştirilmiş olur.

3.2.5 En İyi Sığan Azalan Algoritması

ESAA'da listedeki elemanlar azalan sıraya göre sıralanır. Sıralanan liste ESA'ya göre çözülür. Algoritma 5'te ESAA'nın sözde kodu verilmiştir.

Algortima 5: ESAA Sözde Kodu

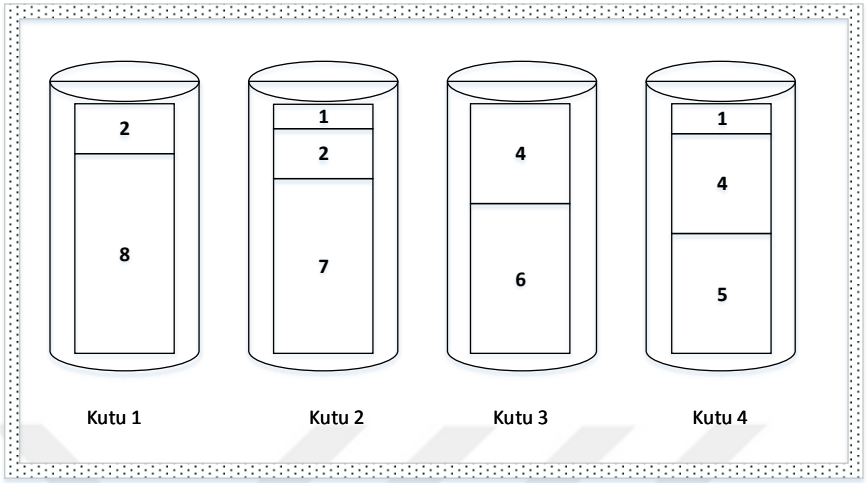
1: Listedeki elemanlar azalan sırada sıralanır
2: Sıralanmış listeye **ESA** uygulanır.

Örnek olarak kutulanacak parçalar kümesi $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ ve kutu kapasitesi $C = 10$ varsayılsın. Algoritma 5'te verilen ESAA'ya göre parçaların kutulara yerleşimi Şekil 3.6'da; algoritmanın çözüm adımları Tablo 3.9 ve çözüm adımlarının açıklamaları Tablo 3.10'da gösterilmektedir.

ESAA'ya göre öncelikle S kümesindeki elemanlar azalan sıraya göre dizilir.

S = {8, 7, 6, 5, 4, 4, 2, 2, 1, 1}

Yeni kümeye ESA uygulanır (Şekil 3.6).



Şekil 3.6: ESAA çözümü

Tablo 3.9: ESAA çözümü tablo gösterimi

Elemanlar	Kutu 1	Kutu 2	Kutu 3	Kutu 4	Sıra
8					T1
7					T2
6					T3
5					T4
4					T5
4					T6
2					T7
2					T8
1					T9
1					T10
Son Durum					
Kalan Kapasite	0	0	0	0	

Tablo 3.10: ESAA çözüm adımları

T1	ESAA'ya göre listedeki elemanlar büyükten küçüğe sıralanır ve bu listedeki elemanlar ESA'ya göre yerleştirilir. Buna göre ilk olarak listedeki 8 birimlik elaman ilk kutuya yerleştirilir.
T2	2.sıradaki 7 birimlik eleman ilk kutuya sığmadığı için 2. Kutu açılır ve buraya yerleştirilir.
T3	6 birimlik eleman mevcut kutulara sığmadığı için 3. Kutu açılır ve bu kutuya yerleştirilir.
T4	5 birimlik eleman mevcut kutulara sığmadığı için yeni bir kutu açılarak 4.kutuya yerleştirilir.
T5	4 birimlik elemanın yerleştirilmesi için kutular baştan itibaren taranır 3.kutuda yeterli kapasite olduğu için buraya yerleştirilir. Eğer yeterli kapasiteye sahip birden fazla kutu olsaydı, en az artık kapasite kalacak olan kutuya yerleştirilirdi.
T6	4 birimlik eleman 4.kutuda yeterli kapasite olduğu için ve yeterli kapasiteye sahip başka kutu olmadığı için buraya yerleştirilir.
T7	Sıradaki 2 birimlik eleman ilk kutuya sığdığı için buraya yerleştirilir.
T8	2 birimlik eleman ilk kutu dolduğu için ve 2.kutuda yeterli alan olduğu için 2.kutuya yerleştirilir.
T9	1 birimlik eleman yine 2.kutuya yerleştirilir.
T10	Kalan 1 birimlik eleman sadece son kutuda yer kaldığı için 4.kutuya yerleştirilir. Böylece listedeki elemanlar hiç artık kapasite kalmayacak şekilde toplamda 4 kutuya yerleştirilmiş olur.

3.2.6 Alt Sınıra Dayalı Sezgisel Başlangıç Çözümü

Alt sınıra dayalı sezgisel başlangıç çözümünde kullanılmak üzere "alt sınır değeri" aşağıdaki gibi açıklanmaktadır.

3.2.6.1 Alt Sınır

Kutulama problemi için belirgin bir alt sınır *LB* şu şekilde hesaplanabilir.

S_i : Boyut,

C: Kapasite olmak üzere,

$$LB = \left\lceil \sum_{i \in I} S_i / C \right\rceil \quad (3.1)$$

LB'nin, boyut S_i'nin kapasite C'ye göre yeterince küçük olduğu problemler için "iyi" davranışa sahip olması beklenebilir. *LB* aslında, öğelerin farklı kutular arasında bölünebileceğini varsayarak elde edilir, Dolayısıyla getirdiği hata veri genellikle süreklilik eğilimi gösterirse azalır (Loh ve diğ. 2006).

3.2.6.2 Alt sınıra Dayalı Sezgisel Başlangıç Çözümü Algoritması

Başlangıç çözümü oluşturmak için kullanılan alt sınıra dayalı sezgisel başlangıç çözümünde (AS_SBC) öncelikle kutudaki elamanlar için Formül (3.1)'deki gibi bir *LB* değeri bulunur ve bulunan *LB* kadar kutu açılır. Daha sonra kutudaki elemanlar büyükten küçüğe (artmayan sırada) sıralanır ve açılan kutulara birinci eleman birinci kutuya ikinci eleman ikinci kutuya şeklinde son kutuya kadar yerleştirilir. Son kutuya gelindiğinde sıradaki eleman yine sonuncu kutuya, bir sonraki eleman ise sonuncudan bir önceki kutuya yani (*LB* – 1)'nci kutuya yerleştirilir. Algoritma eleman kalmayıncaya kadar bu şekilde devam eder (Beisiegel ve diğ. 2005). Eğer kutular doluysa ve elimizde hala eleman varsa kalan elemanlarla algoritma tekrarlanır. Algoritmanın işleyişi Algoritma 6'da verilmiştir.

Algortima: 6 AS_SBC Sözde Kodu

---

1) Elemanların bulunduğu liste *S*'yi artmayan sırada sırala.
2) Kullanılacak tüm kutuları kapat.
3) Liste *S* için en uygun *LB* bul.
4) Kullanılmak için *LB* kadar kutu aç.
5) Listedeki elemanlar öngörülen sırayla açık kutulara yerleştir.
6) Yerleştirilen elemanı listeden kaldır.
7) Liste boşsa dur.
8) Eğer kapalı kutular seti boş değilse adım 3'e git, aksi takdirde kalan kutuları yerleştir (Beisiegel ve diğ. 2005)

Örnek olarak kutulanacak parçalar kümesi *S = {20, 60, 30, 20, 50, 20}* ve kutu kapasitesi *C = 100* varsayılsın. Algoritma 6'da verilen AS_SBC'ye göre parçaların kutulara yerleşimi aşağıdaki gibi adım adım gösterilmektedir.

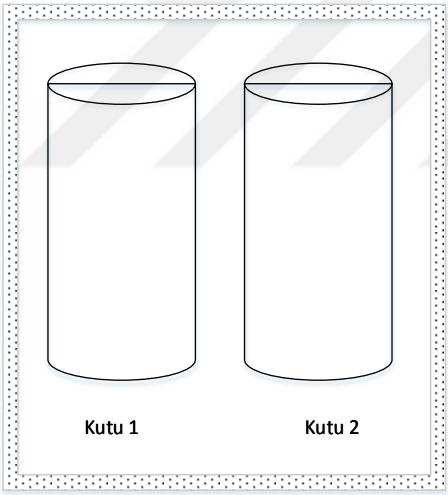
1) *S* kümesindeki elemanlar azalan sıraya göre dizilir.

S = {60, 50, 30, 20, 20, 20}

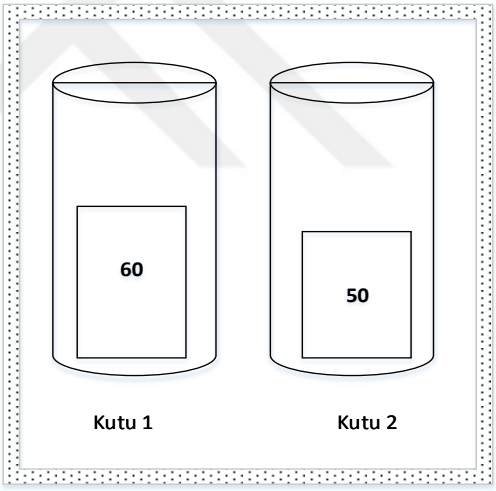
2) *S* kümesindeki elemanlar için *LB* bulunur.

LB = (60+50+30+20+20+20) / 100 = 2

3) *LB* sayısı kadar kutu açılır (Şekil 3.7). *S* kümesindeki 1.eleman 1.kutuya, 2.eleman 2.kutuya yerleştirilir (Şekil 3.8).



Şekil 3.7: AS_SBC çözüm adımları (1)

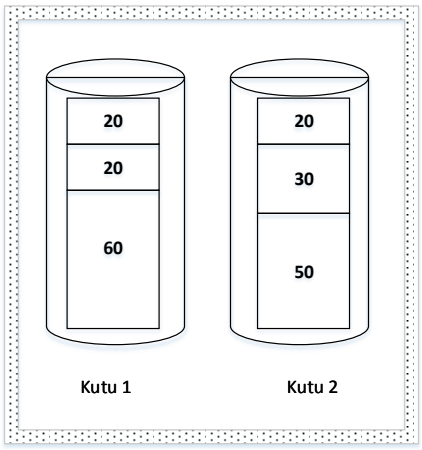


Şekil 3.8: AS_SBC çözüm adımları (2)

4) 3. eleman başka kutu olmadığı için yine 2. kutuya 4. eleman 1. kutuya yerleşecek şekilde baştan sona - sondan başa doğru bütün elemanlar kutuların kapasitesini aşmayacak şekilde yerleştirilir (Şekil 3.9).

5) Eğer bu işlemler sonucunda kümede eleman kalıyor yani kutulara sığmıyorsa kalan elemanlar bütün elemanlar yerleşinceye kadar algoritma tekrarlanır.

Algoritmaya ait çözümün tablo şeklindeki gösterimi Tablo 3.11'deki gibidir. Adım adım çözüm süreci ise Tablo 3.12'de gösterilmektedir.



Şekil 3.9: AS_SBC çözümü

Tablo 3.11: AS_SBC tablo gösterimi

Elemanlar	Kutu 1	Kutu 2	Sıra
60			T1
50			T2
30			T3
20			T4
20			T5
20			T6
Son Durum			
Kalan Kapasite	0	0	

Tablo 3.12: AS_SBC çözüm adımları

T0	Listedeki elemanlar azalan sıraya göre sıralanır. LB değeri bulunur ve LB değeri kadar kutu açılır. Bu örnekte LB değeri 2 olarak bulunur ve başlangıçta 2 adet kutu açılır.
T1	60 birimlik eleman ilk kutuya yerleştirilir.
T2	Algoritmaya göre ilk eleman ilk kutuya 2.eleman 2.kutuya yerleştirileceği için 50 birimlik eleman 2.kutuya yerleştirilir.
T3	Mevcut başka kutu olmadığı için 30 birimlik eleman yine 2.kutuya yerleştirilir. AS_SBC'de yerleştirme baştan sona- sondan başa doğru yapılmaktadır.
T4	Algoritmaya göre T4 anında sondan başa doğru gidildiği için 20 birimlik eleman 1.kutuya yerleştirilir.
T5	İlk kutuya gelindiği için ve baştan sona gidileceği için 20 birimlik eleman yine ilk kutuya yerleştirilir.
T6	Kalan 20 birimlik eleman ise 2.kutuya yerleştirilir. Böylece elemanlar hiç boş alan kalmayacak şekilde toplamda 2 kutuya yerleştirilir. Eğer listede eleman kalmış olsaydı tekrardan LB değeri hesaplanarak LB değeri kadar kutu açılacak ve aynı şekilde yerleştirme işlemi yapılacaktı.

3.3 Başlangıç Çözümlerinin Analizi

Bu tez çalışması kapsamında algoritmaların daha iyi anlaşılabilmesi ve sonuçlarının incelenmesi amacıyla sezgisel algoritmalar farklı test kümeleri için çalıştırılarak sonuçlar elde edilmiştir. Bu analizde 3 farklı test sınıfı kullanılmış, bu test kümeleri OR Library'den alınmıştır [internet 1].

İlk test sınıfı Falkenauer (1996) tarafından geliştirilen U serisi; düzgün dağılımlı, 20 örneğe sahip, kapasitesi 150'dir. Bu seride U120, U250, U500 ve U1000 olmak üzere 4 farklı örnek kümesi bulunmaktadır ve bunlar içerdikleri eleman sayısına göre isimlendirilmiştir. Örneğin U120 test kümesi 120 elemana sahiptir. İkincisi Wäscher ve Gau (1996)'dan alınan 17 probleme sahip Gau serisi test kümesidir. Her problem farklı eleman sayısına sahiptir. Son olarak da J. Schoenfield'ın 28 örnekten oluşan hard28 test kümesi sezgisel algoritmalarla test edilmiştir.

Algoritmalar Visual Studio ortamında C++ programlama dili kullanılarak yazılmış ve testler Windows 7 Enterprise işletim sisteminde yapılmıştır. Testler için kullanılan bilgisayar, 4 GB RAM ile 2.26 GHz Intel ® Core™ 2 Duo CPU'ya sahiptir. Tek CPU kullanılmıştır. Test sonuçları her algoritma için ayrı ayrı tablolarda gösterilmiş olup her test kümesi referans çözümlerle karşılaştırılmıştır.

Test kümesinde bulunan örnek sayısı N ile ifade edilmiştir. Referans çözümlere ulaşılıp ulaşılmadığını gösteren değerler "başarı oranı" sütununda yer almaktadır. Bu sütun "optimum sonuca ulaşılan örnek sayısı / örnek sayısı"'nı göstermektedir. Örneğin u120 kümesinin ilk sığan algoritmasıyla çözümünde hiçbir örnekte optimum sonuca ulaşılamamıştır. Bu yüzden başarı oranı 0/20'dir.

Yine elde edilen sonuçlara göre mutlak sapma ve bağıl sapma hesaplanmıştır. Mutlak sapma; literatürdeki en iyi bilinen çözüm (best known solution) (EIBC) ile elde edilen çözüm arasındaki farkın mutlak değerinin alınmasıyla elde edilir. Her bir test kümesindeki yine her bir örnek için mutlak sapma Formül 3.2'deki gibi hesaplanır. Bunların ortalamasının alınmasıyla ise ortalama mutlak sapma bulunur, bütün örneklerdeki en yüksek değere sahip olan sapma da maksimum sapmayı verir. Bağıl sapma ise mutlak sapmaların literatürdeki bilinen çözüme bölümünden elde edilir (Hashim ve diğ. 2014) (Formül 3.3). Bağıl sapma için de ortalama ve maksimum değerler tablolarda gösterilmektedir. Mutlak sapma ve bağıl sapma sırasıyla 3.2 ve 3.3 formüllerine göre hesaplanır:

$$\text{Mutlak Sapma} = \quad \Delta x = |x_0 - x| \tag{3.2}$$

$$\text{Bağıl Sapma} = \text{BS} = \frac{\Delta x}{x_0} \tag{3.3}$$

Formül (3.2) ve (3.3) 'te;

Δx : Mutlak sapmayı,

x_0 : EIBC değerini,

x : Ölçülen değeri ifade etmektedir.

Örneğin U120 test kümesindeki U120_00 örneği için;

EIBC = 48,

ISA sonucu elde edilen değer = 50 ‘dir.

Bu durumda mutlak sapma ve bağıl sapma aşağıdaki gibi hesaplanır.

$$\Delta x = |x_0 - x| = |48 - 50| = 2$$

$$BS = \frac{\Delta x}{x_0} = \frac{2}{48} = 0{,}042$$

Son sütunda ise ortalama çalışma süreleri saniye cinsinden hesaplanmıştır. Tablo 3.19’da ise U120 dizisindeki 20 örnek için bahsedilen bütün algoritma sonuçları tek bir tabloda gösterilmiştir.

3.3.1 ISA Çözümü

Test kümelerinin ISA’ya göre çözümü Tablo 3.13’te verilmiştir.

Tablo 3.13: ISA’ya göre sonuçlar

			MUTLAK SAPMA		BAĞIL SAPMA		
TEST KÜMESİ	**N**	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
U120	20	0/20	3,05	5	0,06	0,104	0,001
U250	20	0/20	6,4	9	0,06	0,086	0,003
U500	20	0/20	11,6	15	0,06	0,074	0,005
U1000	20	0/20	20,9	24	0,05	0,058	0,013
Gau1	12	**4/12**	0,67	1	0,04	0,083	0,002
Gau2	5	**1/5**	0,8	1	0,05	0,09	0,001
Hard28	28	0/28	13	17	0,18	0,204	0,002
TOPLAM	125	5/125	56,42	72	0,5	0,699	0,028

3.3.2 ESA Çözümü

Test kümelerinin ESA'ya göre çözümü Tablo 3.14'te verilmiştir.

Tablo 3.14: ESA'ya göre sonuçlar

			MUTLAK SAPMA		BAĞIL SAPMA		
TEST KÜMESİ	**N**	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
U120	20	0/20	2,75	4	0,06	0,083	0,002
U250	20	0/20	6	8	0,06	0,08	0,003
U500	20	0/20	10,8	13	0,05	0,066	0,005
U1000	20	0/20	19,8	23	0,05	0,056	0,013
Gau1	12	**7/12**	0,58	1	0,04	0,083	0,002
Gau2	5	**2/5**	0,6	1	0,04	0,09	0,002
Hard28	28	0/28	11,79	17	0,17	0,212	0,002
TOPLAM	125	9/125	52,32	67	0,47	0,67	0,027

3.3.3 EKSA Çözümü

Test kümelerinin EKSA'ya göre çözümü Tablo 3.15'te verilmiştir.

Tablo 3.15: EKSA'ya göre sonuçlar

			MUTLAK SAPMA		BAĞIL SAPMA		
TEST KÜMESİ	**N**	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
U120	20	0/20	7,85	13	0,16	0,208	0,001
U250	20	0/20	16,3	22	0,16	0,211	0,003
U500	20	0/20	31,65	39	0,16	0,191	0,006
U1000	20	0/20	60,8	71	0,15	0,174	0,014
Gau1	12	**2/12**	1	2	0,06	0,125	0,002
Gau2	5	0/5	1,6	3	0,08	0,12	0,002
Hard28	28	0/28	15,5	21	0,22	0,253	0,003
TOPLAM	125	2/125	134,7	171	0,99	1,282	0,031

3.3.4 ISAA Çözümü

Test kümelerinin ISAA'ya göre çözümü Tablo 3.16'da verilmiştir.

Tablo 3.16: ISAA'ya göre sonuçlar

			MUTLAK SAPMA		BAĞIL SAPMA		
TEST KÜMESİ	**N**	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
U120	20	**8/20**	0,6	1	0,01	0,022	0,001
U250	20	0/20	1,4	3	0,01	0,03	0,003
U500	20	0/20	2,65	3	0,01	0,015	0,008
U1000	20	0/20	4,9	7	0,01	0,018	0,022
Gau1	12	**5/12**	0,58	1	0,04	0,083	0,002
Gau2	5	**3/5**	0,4	1	0,03	0,09	0,001
Hard28	28	**5/28**	0,82	1	0,01	0,0166	0,003
TOPLAM	125	21/125	11,35	17	0,12	0,2746	0,041

3.3.5 ESAA Çözümü

ESAA çözümleri Tablo 3.17'de verilmiştir.

Tablo 3.17: ESAA'ya göre sonuçlar

			MUTLAK SAPMA		BAĞIL SAPMA		
TEST KÜMESİ	**N**	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
U120	20	**8/20**	0,6	1	0,01	0,022	0,002
U250	20	0/20	1,4	3	0,01	0,03	0,003
U500	20	0/20	2,95	8	0,02	0,041	0,009
U1000	20	0/20	4,85	7	0,01	0,018	0,023
Gau1	12	**5/12**	0,58	1	0,04	0,083	0,002
Gau2	5	**3/5**	0,4	1	0,03	0,09	0,002
Hard28	28	**5/28**	0,82	1	0,01	0,0166	0,003
TOPLAM	125	21/125	11,6	22	0,13	0,3006	0,042

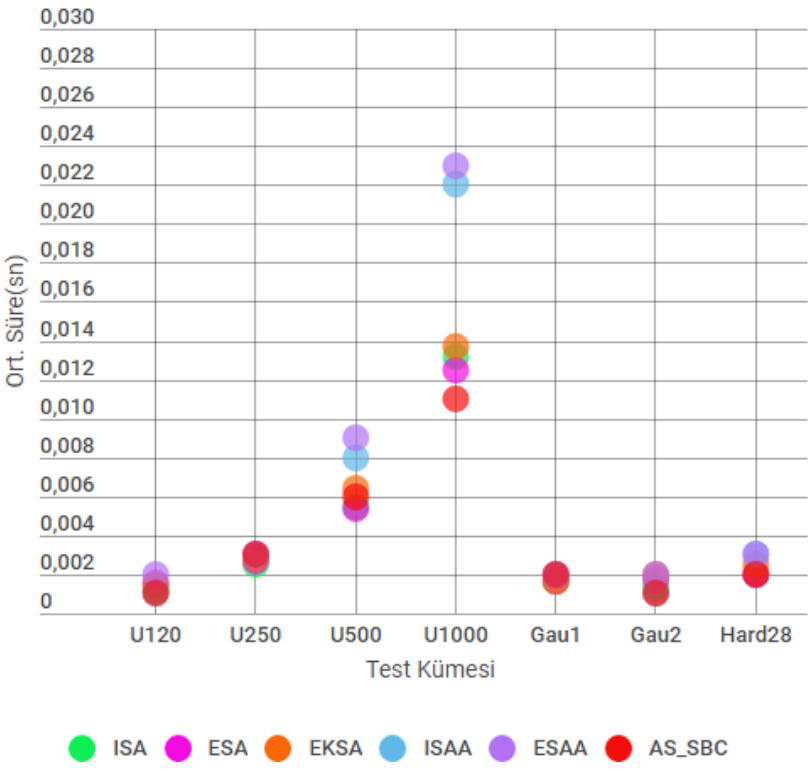
3.3.6 AS_SBC

AS_SBC'ye göre çözümler Tablo 3.18'de verilmiştir.

Tablo 3.18: AS_SBC'ye göre sonuçlar

			MUTLAK SAPMA		BAĞIL SAPMA		
TEST KÜMESİ	**N**	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
U120	20	0/20	4,55	6	0,09	0,13	0,001
U250	20	0/20	9,4	10	0,092	0,1	0,003
U500	20	0/20	19,6	22	0,097	0,112	0,006
U1000	20	0/20	36,45	42	0,098	0,106	0,011
Gau1	12	**5/12**	0,58	1	0,04	0,833	0,002
Gau2	5	**2/5**	0,6	1	0,04	0,909	0,001
Hard28	28	0/28	3	5	0,04	0,074	0,002
TOPLAM	125	7/125	74,18	87	0,497	2,264	0,026

Aşağıdaki grafikte ise test edilen bütün kümelerin 6 farklı başlangıç çözümüne göre ortalama çözüm süreleri(sn.) gösterilmektedir (Grafik 3.1).



Grafik 3.1: Tüm test kümelerinin tüm başlangıç algoritmalarına göre ortalama çözüm süreleri

3.3.7 U120 İçin Tüm Algoritmaların Çözümleri

Tablo 3.19'da U120 test kümesi için 6 farklı algoritmadan elde edilen optimum sonuçlar ve *LB* değeri ile EIBC değerleri de yer almaktadır. Buna göre EIBC'ye en çok yaklaşan algoritmalar ISAA ve ESAA'dır. ISAA ve ESAA algoritmaları 8 örnekte de EIBC değerine ulaşmıştır.

Tablo 3.19: U120 test kümesi için başlangıç algoritmalarının sonuçları

TEST KÜMESİ	**EIBC**	**LB**	**ISA**	**ESA**	**EKSA**	**ISAA**	**ESAA**	**AS_SBC**
u120_00	48	48	50	50	55	49	49	53
u120_01	49	49	51	51	57	**49**	**49**	53
u120_02	46	46	48	48	51	47	47	51
u120_03	49	49	52	53	57	50	50	54
u120_04	50	50	52	52	57	**50**	**50**	54
u120_05	48	48	52	52	54	49	49	53
u120_06	48	48	51	52	56	49	49	52
u120_07	49	49	52	52	56	50	50	54
u120_08	51	50	54	53	59	**51**	**51**	54
u120_09	46	46	49	48	55	47	47	52
u120_10	52	52	56	55	61	**52**	**52**	56
u120_11	49	49	52	51	57	50	50	54
u120_12	48	48	52	51	58	49	49	53
u120_13	49	49	51	51	55	**49**	**49**	54
u120_14	50	50	53	53	59	**50**	**50**	54
u120_15	48	48	53	52	57	49	49	53
u120_16	52	52	56	55	65	**52**	**52**	56
u120_17	52	52	56	56	60	53	53	56
u120_18	49	49	52	51	55	50	50	54
u120_19	50	49	52	52	56	**50**	**50**	54
TOPLAM	983	981	1044	1038	1140	**995**	**995**	1074

Yukarıda her algoritma için verilen sonuç tablolarına ve aşağıda Tablo 3.20'de verilen tüm algoritmaların başlangıç çözümünden elde edilen sonuçlara bakıldığında EIBC'ye yaklaşan, en iyi sonucu veren algoritmalar ISAA ve ESAA'dır. Ortalama çözüm sürelerine bakıldığı zaman ise AS_SBC'nin en kısa sürede çözüme ulaştığı görülmektedir. Sonuç olarak bu çalışmada kutulama

probleminin çözümünde kullanılmak üzere başlangıç çözüm kalitesi kötü olmasına rağmen hızlı olduğu için başlangıç çözüm algoritması olarak AS_SBC tercih edilmiştir.

Tablo 3.20: Tüm algoritmalara göre başlangıç çözümleri

		MUTLAK SAPMA		**BAĞIL SAPMA**		
ALGORİTMA	**BAŞARI ORANI**	**ORT.**	**MAKS.**	**ORT.**	**MAKS.**	**ORTALAMA SÜRE(sn.)**
ISA	5/125	56,42	72	0,5	0,699	0,028
ESA	9/125	52,32	67	0,47	0,67	0,027
EKSA	2/125	134,7	171	0,99	1,282	0,031
ISAA	**21/125**	11,35	17	0,12	0,2746	0,041
ESAA	**21/125**	11,6	22	0,13	0,3006	0,042
AS_SBC	7/125	74,18	87	0,497	2,264	**0,026**
TOPLAM		340,57	436	2,71	5,49	0,20

4. KUTULAMA PROBLEMİ ÇÖZÜMÜNDE AĞIRLIKLI TAVLAMA SEZGİSELİ

Bir boyutlu kutulama probleminde n tane öğe özdeş kutulara yerleştirilmeye çalışılacaktır. Her bir kutunun kapasitesi C'dir. Amaç, kapasite kısıtlarını ihlal etmeden kutu sayısını en aza indirmektir. Bu amacı sağlamak için ağırlıklı tavlama (weighted annealing) algoritması kullanılmıştır.

4.1 Ağırlıklı Tavlama Kavramı

Ağırlıklı tavlama kombinasyonel optimizasyon problemlerinin çözümünde kullanılmak üzere Elidan, Ninio ve Schneider (2002) tarafından geliştirilmiştir. İlk olarak 2002 yılında Elidan ve Ninio (2002) genel makine öğrenme problemlerinde geliştirilmiş hipotezleri keşfetmek için küresel aramayı (global search) yerel optimizasyon ile birleştiren bir yaklaşım sunmuştur. Bu çalışmada eğitim verilerini bozan yerel maksimumlardan kaçma stratejileri üzerinde durulmuş, eğitim örneklerini yeniden ağırlıklandırmak için basit stratejiler sunulmuştur. Bu bağlamda eğitim örneklerinde W göz önüne alınacak şekilde puanlar arttırılmıştır (Elidan ve diğ. 2002). 2004 yılında Ninio ve Schneider (2004) tarafından yapılan bir çalışmada gezgin satıcı problemi ele alınmış, sistemin her bir bölümüne bir ağırlık (Wi) atanarak yeni bir ağırlıklı maliyet fonksiyonu tanımlanmıştır. Ağırlıklı tavlamada elde edilen sonuçlar açgözlü algoritması (greedy algorithm) ve benzetimli tavlama ile karşılaştırılmış, ağırlıklı tavlamanın benzetimli tavlamaya göre daha kısa sürede daha iyi sonuçlar verdiği gözlemlenmiştir (Ninio 2004)

Ağırlıklı tavlama; benzetimli tavlama ve deterministik tavlama gibi diğer meta-sezgisel algoritmalardan faklı olarak birçok özellik sağlar. Özellikle benzetimli tavlama ve deterministik tavlama bir başlangıç çözümüyle başlar ve kötüleşen hareket içeren lokal aramada bir dizi işlem gerçekleştirir. Bozulan bir hareket amaç fonksiyonu değeri belirli bir eşiğin ötesinde daha da kötü olmadığı sürece izin verilebilir. Ağırlıklı tavlama sadece amaç fonksiyonunu dikkate almaz aynı zamanda bir optimizasyon çalışmasının her aşamasında çözülmüş arama alanının ne kadar iyi

kullanıldığına bakar. Bir boyutlu kutulama problemi gibi kombinasyonel optimizasyon problemlerine uygulandığında, ağırlıklı tavlama yaklaşımı arama alanının farklı noktalarında bozulmalar oluşturarak komşu aramayı hızlandırmayı ve arama alanını genişletmeyi amaçlar. Bozulmalar bir sonraki iterasyonda kazanılan analizlere dayalı ağırlık atamaları ile kontrol edilir. Yerel optimum için arama hızlandırılmış ve çözüm kalitesi ağırlıklı tavlama ile geliştirilmiş olmalıdır (Loh ve diğ. 2006).

4.1.1 Ağırlıklı Tavlama Algoritması

Ağırlıklı tavlama algoritması gezgin satıcı problemi, kutulama problemi gibi çeşitli problem türlerinde kullanılan bir algoritmadır. Bu algoritmanın çözüm adımları Algoritma 7'de verilmiştir.

Algoritma 7: Ağırlıklı Tavlama Algoritması (Ninio ve Schneider 2004).

1) İlk olarak başlangıç çözümüyle elde edilen bir başlangıç konfigürasyonuyla algoritmaya başlanır.

2) Daha sonra, önceki başlangıç çözümünün sonucuna dayanılarak yeni bir W_i ağırlık seti belirlenir.

3) Bir önceki optimizasyon çalışmasının sonuç konfigürasyonundan başlayarak ve yeni ağırlık değerlerini kullanarak yeni açgözlü optimizasyon çalıştırması gerçekleştirir.

4.1.2 Kutulama Problemi İçin Ağırlıklı Tavlama Algoritması

Kutulama problemi için ağırlıklı tavlama algoritması bir boyutlu, iki boyutlu ve üç boyutlu kutulama problemleri için kullanılabilir. Aşağıda bir boyutlu ve iki boyutlu kutulama problemleri için algoritma adımları verilmiştir.

4.1.2.1 Bir Boyutlu Kutulama Problemi İçin Ağırlıklı Tavlama Algoritması

Bir boyutlu kutulama problemi için ağırlıklı tavlama algoritması aşağıdaki gibi 5 adımda tanımlanmıştır.

Algoritma 8: Bir Boyutlu Kutulama Problemi İçin Ağırlıklı Tavlama Algoritması (Loh ve diğ. 2006).

---

1) ISAA algoritması kullanarak başlangıç çözümü oluşturulur.

2) Başlangıç çözüme dayanarak kutular için bir dizi ağırlık hesaplaması ve ataması yapılır, her bir kutu için kalan artık kapasiteler hesaplanır.

3) Kutuların tüm çiftleri arasında yer değiştirme (swap) işlemi yapılarak yerel arama yapılır. Amaç fonksiyonu kutuya yüklenenlerin kapasitelerinin karelerinin toplamını maksimum yapmaktır.

4) Bir önceki optimizasyon çalışmasının sonuçlarına göre yeni ağırlık dizisi hesaplanarak tekrar ağırlıklandırma yapılır.

5) Durma kriterlerine ulaşıncaya kadar 3. adıma geri dönülür.

Amaç Fonksiyonu

Bir boyutlu kutulama problemi için amaç fonksiyonu kutuya yüklenenlerin kapasitelerinin karelerinin toplamını maksimize etmektir (Loh ve diğ. 2006). Maksimize fonksiyonu Formül 4.1'de gösterilmiştir. Amaç fonksiyonu ile ilgili örnek Şekil 4.1'de verilmiştir.

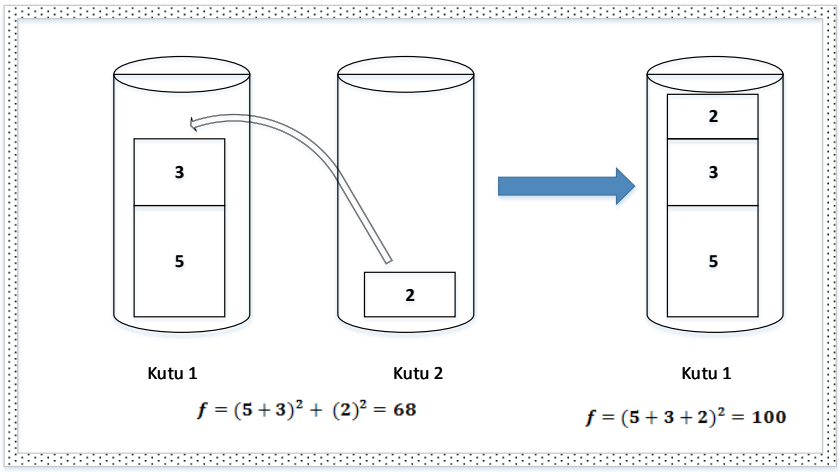
$$\text{Maksimize } f = \sum_{i=1}^{p} (l_i)^2 \tag{4.1}$$

$$l_i = \sum_{j=1}^{q_i} t_{ij} \tag{4.2}$$

l_i: i.kutudaki öğelerin boyutlarının toplamını

t_{ij} : i. kutudaki j. öğesinin boyutunu

q_i : i. kutudaki öğelerin sayısını temsil etmektedir.



Şekil 4.1: Kutuların karelerinin toplamını maksimuma çıkarmak

Ağırlık Atama

Öğelerin görünür boyutlarda değişimler farklı kutulara farklı ağırlıklar verilerek elde edilir ve bunlar her iterasyonda tekrar hesaplanır. i kutusuna atanan ağırlık W_i ile gösterilir ve (4.3) ve (4.4)'teki formülle hesaplanır (Loh ve diğ. 2006). Ağırlık hesaplama Şekil 4.2'deki örnekte gösterilmiştir.

$$w_i = 1 + Kr_i \tag{4.3}$$

$$r_i = \left(\frac{C - l_i}{C}\right) \tag{4.4}$$

l_i: i. kutudaki öğelerin boyutlarının toplamını

r_i : i. kutunun kalan kapasite oranını

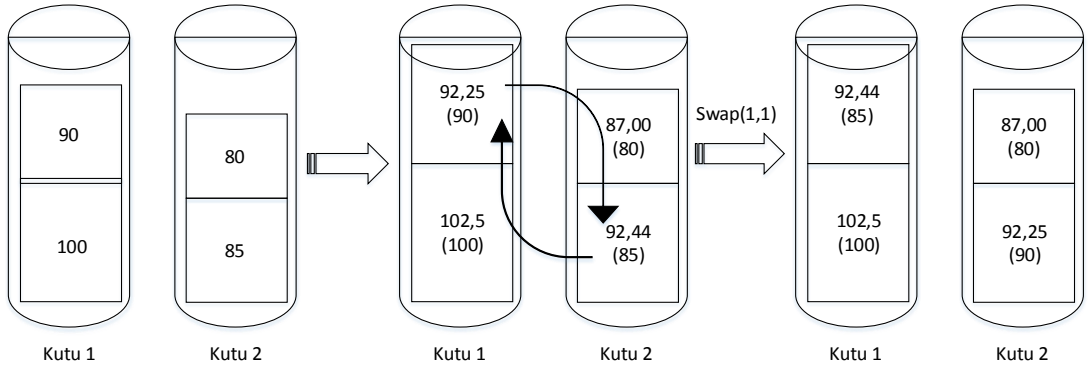
C = Kapasiteyi

K = Sabit değeri temsil etmektedir.

C = 200 K = 0.5

$$W_i = 1 + 0.5\left(\frac{200 - l_i}{200}\right)$$

$W_1 = 1.0250$ $W_2 = 1.0875$



$f' = (102.50 + 92.25)^2 + (87.00 + 92.44)^2 = 70126.3$ $f'_{yeni} = (102.50 + 92.44)^2 + (87.00 + 92.25)^2 = 70132.2$

Şekil 4.2: Ağırlık hesaplama (Loh ve diğ. 2006)

Tavlama Benzetimi

Tavlama; malzemeyi belirli bir süre (tavlama sıcaklığına kadar) ısıttıktan sonra, yavaş yavaş soğutmaktır. Tavlama malzemeyi rahatlatmak, yumuşatmak ve iç yapıyı daha kullanılabilir hale getirmek için yapılan ısıl işlemlerin geneline verilen addır. Isıl işlem, bir katının sıcaklığının belirli bir maksimum dereceye kadar artırılarak tekrar azaltılması işlemidir. Maksimum sıcaklıkta kristalin tüm molekülleri, kendilerini rasgele olarak sıvı faza ayarlar. Sonra, erimiş kristalin sıcaklığı kristal yapı soğutuluncaya kadar düşürülür. Soğuma uygun şekilde yapılırsa kristal yapı çok düzenli olur (Kirkpatrick, 1983).

Tavlama benzetimi katıların fiziksel tavlanma süreci ile olan benzerlikten ileri gelmektedir. Katıların ısıtılması ve sonra yavaş yavaş soğutulması esasına dayanır. Kirkpatrick ve arkadaşları (1983) tarafından önerilmiştir. Tavlama benzetiminde, amaç fonksiyonunda artışa neden olabilecek komşu hareketler bazen kabul edilerek yerel eniyi noktalarından kurtulmak mümkündür. Amaç fonksiyonunda artışa neden olabilecek bu komşu hareketin kabul edilip edilmemesi, rassal olarak belirlenmektedir. Sıcaklık yüksek olduğunda, amaç fonksiyonunda artışa neden olabilecek hareketlerin kabul edilme olasılığı çok yüksek olacak, sıcaklık düştükçe bu olasılık da azalacaktır. Bu sebeple, aramaya yeteri kadar yüksek bir sıcaklık değeri ile başlamak gereklidir. Algoritmada, sıcaklık yavaş yavaş azaltılırken, her

sıcaklık değerinde belli sayıda hareket deneyerek arama işlemi sürdürülür (Aydın 2013).

Ağırlıklı tavlama yaklaşımında da benzer yöntem kullanılmaktadır. Sürecin başında yüksek sıcaklığa yer verilerek arama alanı genişletilir ve sıcaklık her iteresyonda yavaş yavaş azaltılır. Ağırlıklı tavlamada farklı olarak bir dizi ağırlık hesaplaması ve ataması yapılarak yerel arama yapılır.

Ağırlıklı Tavlama Yönteminde Tavlama Süreci

Tavlama süreci için miktarı yöneten bir T kontrol parametresi kullanılmıştır. Ağırlık W_i^T olarak belirtilmiştir ve T parametresi aşağıdaki gibi süreci kontrol eder.

- Sürecin başında daha fazla yer değiştirme sağlamak için ağırlık değişikliklerinde yüksek sıcaklığa izin verilir.
- Sıcaklık soğurken her bir öğenin bozulma miktarı azalır.
- Eğer T 0'a ulaşırsa, algoritma bütün ağırlıkların 1'e eşit olmasıyla problemi çözer (Loh ve diğ. 2006).

4.1.2.2 İki Boyutlu Kutulama Problemi İçin Ağırlıklı Tavlama Algoritması

İki boyutlu kutulama problemi için ağırlıklı tavlama algoritması adımları aşağıdaki gibi tanımlanmıştır.

Algoritma 9: İki Boyutlu Kutulama Problemi İçin Ağırlıklı Tavlama Algoritması

Adım 1	Değiştirilmiş hibrid ilk sığan algoritması kullanılarak başlangıç çözümü oluşturulur.
Adım 2	Başlangıç çözüme dayanarak kutuların yükseklik değerleri için bir dizi ağırlık hesaplaması ve ataması yapılır, her bir kutu için kalan artık kapasiteler hesaplanır.
Adım 3	Kullanılmayan alanları doldurmak için kutular arasında yer değiştirme (swap) işlemi yapılır (Loh ve diğ. 2006).

4.1.2.3 Bir Boyutlu Kutulama Problemi İçin Önerilen Ağırlıklı Tavlama Algoritması

Önerilen algoritmanın Loh'un çalışmasındaki ağırlıklı tavlama algoritmasından temel farkı başlangıç çözüm yöntemi olarak ISAA yerine AS_SBC'yi kullanmasıdır. Bir boyutlu kutulama problemi için önerilen algoritma süreci Algoritma 10'da tanımlanmıştır:

Algoritma 10: Bir Boyutlu Kutulama Problemi İçin Önerilen Ağırlıklı Tavlama Algoritması

---

1) AS_SBC algoritması kullanarak başlangıç çözümü oluşturulur.

2) Başlangıç çözüme dayanarak kutular için bir dizi ağırlık hesaplaması ve ataması yapılır, her bir kutu için kalan artık kapasiteler hesaplanır.

3) Kutuların tüm çiftleri arasında yer değiştirme (swap) işlemi yapılarak yerel arama yapılır. Amaç fonksiyonu kutuya yüklenenlerin karelerini toplamını maksimum yapmaktır.

4) Bir önceki optimizasyon çalışmasının sonuçlarına göre yeni ağırlık dizisi hesaplanarak tekrar ağırlıklandırma yapılır.

5) Durma kriterlerine ulaşıncaya kadar 3. adıma geri dönülür.

Başlangıç çözümü oluşturulduktan sonra ağırlıklı tavlama ile entegre bir şekilde çalışan yer değiştirme işlemleri uygulanır. Bu işlemler "çözüm kalitesini arttıran operatörler" başlığı altında açıklanmıştır.

4.2 Çözüm Kalitesini Arttıran Operatörler

Çözüm kalitesini arttırmak, daha iyi sonuçlar elde etmek amacıyla yer değiştirme operatörleri kullanılır. Yer değiştirme işlemlerinde amaç kutudaki elamanların yerlerini değiştirerek kullanılan kutu sayısını minimuma düşürmek ve kutuların doluluk oranını maksimuma çıkarmaktır. Kutulama probleminde ağırlıklı tavlama algoritmasıyla birlikte kullanılan bu operatörler Swap(1.0), Swap(1,1), Swap(1,2) ve Swap(2,2) olarak aşağıda açıklanmıştır.

4.2.1 Swap(1,0)

Swap (1,0) işleminde α kutusundaki bir öğe β kutusuna taşınır ve listedeki her eleman için amaç fonksiyonunun değişimi hesaplanır. Bu uygulamada yer değiştirme işleminden önce ve sonra amaç fonksiyonunu tekrar hesaplamaya gerek yoktur, j öğesinin α kutusundan β kutusuna taşınmadaki değişimi hesaplamak yeterli olacaktır. Bu da (4.5)'deki formülle hesaplanabilir (Loh ve diğ. 2006). Swap(1,0)'a ait örnek Şekil 4.3'te verilmiştir.

$$\Delta f = (l_\alpha - t_{\alpha j})^2 + (l_\beta + t_{\alpha j})^2 - {l_\alpha}^2 - {l_\beta}^2 \qquad (4.5)$$

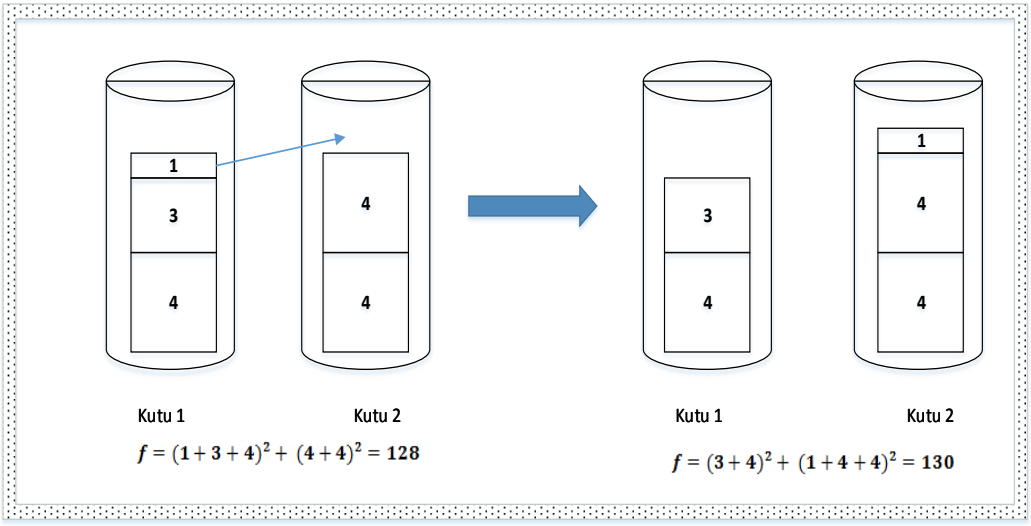
Aşağıdaki örnek için amaç fonksiyonundaki değişimi önceki yer değiştirme ve sonraki yer değiştirme işlemlerindeki farka bakarak hesaplamak yerine (4.5) numaralı formüle göre hesaplarsak;

$l_\alpha = (1 + 3 + 4) = 8$

$t_{aj} = 1$

$l_\beta = (4 + 4) = 8$

$\Delta f = (8 - 1)^2 + (8 + 1)^2 - (8)^2 - (8)^2 = 2$

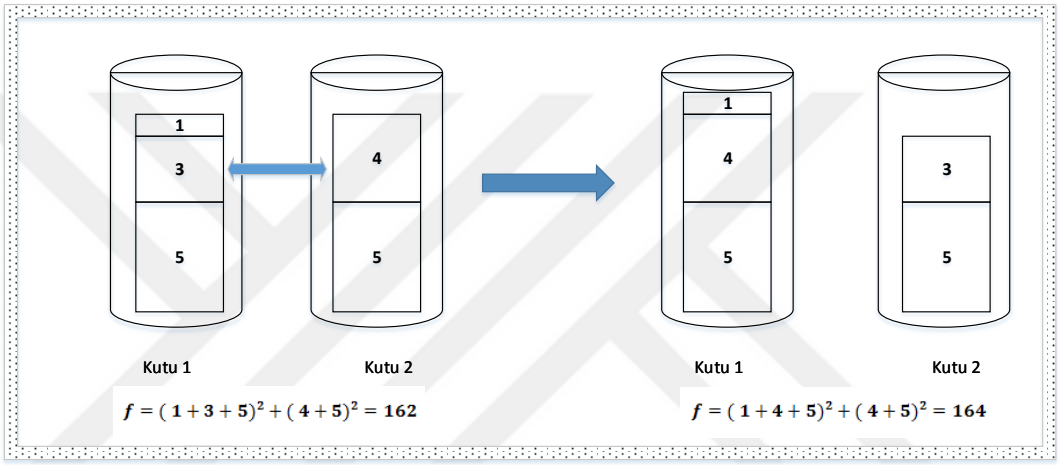


Şekil 4.3: Swap (1,0) örneği

4.2.2 Swap (1,1)

Swap (1,1) işleminde α kutusundaki i öğesi β kutusundaki j öğesiyle yer değiştirir. Eğer bu işlem amaç fonksiyonunda bir artışa neden oluyorsa o zaman yapılır, eğer artışa neden olmuyorsa bu işlemi yapmaya gerek yoktur. Değişim aşağıdaki formülle hesaplanır (4.6) (Loh ve diğ. 2006). Swap (1,1) için örnek Şekil 4.4'te verilmiştir.

$$\Delta f = (l_\alpha - t_{\alpha i} + t_{\beta j})^2 + (l_\beta - t_{\beta j} + t_{\alpha i})^2 - {l_\alpha}^2 - {l_\beta}^2 \qquad (4.6)$$



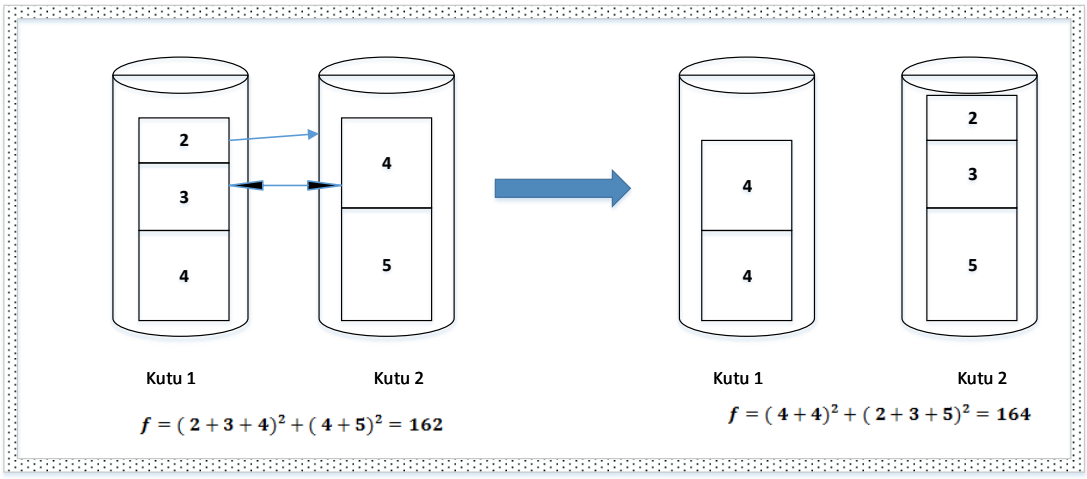
Şekil 4.4: Swap (1,1) örneği

4.2.3 Swap (1,2)

Swap(1,2)'de α kutusundaki i öğesi ile β kutusundaki j ve k öğesi yer değiştirir. Değişim Formül 4.7'deki gibi hesaplanır (Loh ve diğ. 2006).

$$\Delta f = (l_\alpha - t_{\alpha i} + t_{\beta j} + t_{\beta k})^2 + (l_\beta - t_{\beta j} - t_{\beta k} + t_{\alpha i})^2 - {l_\alpha}^2 - {l_\beta}^2 \qquad (4.7)$$

Swap (1,2) için örnek Şekil 4.5'te verilmiştir.



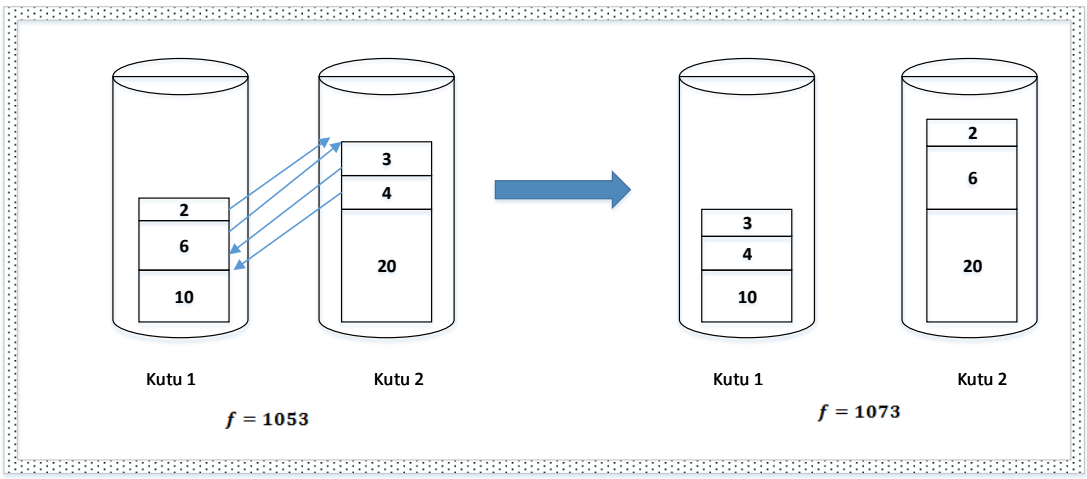
Şekil 4.5: Swap (1,2) örneği

4.2.4 Swap (2,2)

Swap(2,2)'de α kutusundaki i ve j öğeleri ile β kutusundaki k ve l öğeleri yer değiştirir. Değişim aşağıdaki gibi hesaplanır (4.8) (Loh ve diğ. 2006).

$$\Delta f = (l_\alpha - t_{\alpha i} - t_{\alpha j} + t_{\beta k} + t_{\beta l})^2 + (l_\beta - t_{\beta k} - t_{\beta l} + t_{\alpha i} + t_{\alpha j})^2 - {l_\alpha}^2 - {l_\beta}^2 \quad (4.8)$$

Aşağıdaki şekilde Swap (2,2) için örnek verilmiştir (Şekil 4.6).



Şekil 4.6: Swap (2,2) örneği

4.3 Ağırlıklı Tavlama Algoritması İçin Çözüm Kalitesini Arttıran Operatörler

Kutulama probleminde kullanılan yer değiştirme işlemleri ağırlıklı tavlama yaklaşımı ile birlikte uygulanır. Yer değiştirme işlemlerine ağırlıklı tavlama yöntemi ile hesaplanan ağırlık değerleri entegre edilerek değişim fonksiyonları aşağıdaki gibi tekrar düzenlenir (Loh ve diğ. 2006).

Swap (1,0)

$$\Delta f = (l_\alpha * w_\alpha - t_{\alpha i} * w_\alpha)^2 + (l_\beta * w_\beta + t_{\alpha i} * w_\alpha)^2 - (l_\alpha * w_\alpha)^2 - (l_\beta * w_\beta)^2 \quad (4.9)$$

Swap (1,1)

$$\Delta f = (l_\alpha * w_\alpha - t_{\alpha i} * w_\alpha + t_{\beta j} * w_\beta)^2 + (l_\beta * w_\beta - t_{\beta j} * w_\beta + t_{\alpha i} * w_\alpha)^2 - (l_\alpha * w_\alpha)^2 - (l_\beta * w_\beta)^2 \quad (4.10)$$

Swap (1,2)

$$\Delta f = (l_\alpha * w_\alpha - t_{\alpha i} * w_\alpha + t_{\beta j} * w_\beta + t_{\beta k} * w_\beta)^2 + (l_\beta * w_\beta - t_{\beta j} * w_\beta - t_{\beta k} * w_\beta + t_{\alpha i} * w_\alpha)^2 - (l_\alpha * w_\alpha)^2 - (l_\beta * w_\beta)^2 \quad (4.11)$$

Swap(2,2)

$$\Delta f = \left(l_\alpha * w_\alpha - t_{\alpha i} * w_\alpha - t_{\alpha j} * w_\alpha + t_{\beta k} * w_\beta + t_{\beta l} * w_\beta\right)^2 + \left(l_\beta * w_\beta - t_{\beta k} * w_\beta - t_{\beta l} * w_\beta + t_{\alpha i} * w_\alpha + t_{\alpha j} * w_\alpha\right)^2 - (l_\alpha * w_\alpha)^2 - (l_\beta * w_\beta)^2 \quad (4.12)$$

4.4 Bir Boyutlu Kutulama Problemi İçin Önerilen Ağırlıklı Tavlama Algoritması Süreci

Bir boyutlu kutulama problemi için önerilen yöntemde öncelikle kullanılacak en az kutu sayısını belirlemek için *LB* değeri belirlenir. *LB* hesaplama için kullanılan formül 4.13 de verilmiştir:

$$LB = \left\lceil \sum_{i \in I} S_i / C \right\rceil \tag{4.13}$$

Başlangıç çözümü AS_SBC algoritması kullanılarak oluşturulur. Mevcut çözümü iyileştirmek için ağırlıklı tavlama ile yer değiştirme işlemleri uygulanır. Başta T = 1, K = 0,05 olarak atanır. T sıcaklık değişkeni kullanılarak ağırlık hesabı aşağıdaki gibi revize edilir.

$$w_i{}^T = (1 + Kr_i)^T \tag{4.14}$$

Başlangıç çözümü belirlendikten sonra sırayla Swap(1,0), Swap(1,1), Swap(1,2) yer değiştirme işlemleri yapılır. Birinci iterasyon bittikten sonra T sıcaklığı Tred değişkeni oranında değişir. Tred = 0,95 olarak belirlenmiştir. Her iterasyon için T değişkeni tekrar hesaplanır (Loh ve diğ. 2006).

$$T = T \; x \; Tred \tag{4.15}$$

Ağırlıklı tavlama yaklaşımıyla gerçekleştirilen algoritmanın işleyişi aşağıdaki gibidir.

Algoritma 11: Bir Boyutlu Kutulama Problemi İçin Önerilen Ağırlıklı Tavlama Algoritması

Adım 0. **Başlangıç**

Parametreler K , $nloop$, T, $Tred$

K = 0.05, $nloop$ = 50, T = 1, $Tred$ = 0.95

Girdiler : Eleman sayısı, elemanların boyutu, kutu kapasitesi ve alt sınır

Adım 1. **AS_SBC kullanılarak başlangıç çözümünü oluştur.**

Öğe boyutlarına göre artan olmayan sırayla öğeleri sırala

Do while (eleman listesi boş değilse)

{

Elemanların bulunduğu liste L artmayan sırada sırala.

Kullanılacak tüm kutular kapat.

Liste L için en uygun alt sınır (*NLB*) bul.

Kullanılmak için alt sınır kadar kutu aç.

Listedeki elemanlar öngörülen sırayla açık kutulara yerleştir.

Yerleştirilen eleman listeden kaldır.

Liste boşsa dur.

Eğer kapalı kutular seti boş değilse adım 3'e git, aksi takdirde kalan kutuları yerleştir.

}

Artık kapasiteyi hesapla ri

Adım 2. **Mevcut çözümü iyileştir**

For i =1 to $nloop$

Ağırlıkları hesapla: $w = (1\text{-}K\,ri)T$

Tüm kutu çiftleri için yap

{

Swap (1,0) uygula

Δf değerlerini değerlendir

if $\Delta f \geq 0$

Öğeyi taşı

Swap(1,0) 'dan çık,

Eğer alt sınıra ulaşıldıysa döngüden çık

Swap (1,1) uygula

Δf değerlerini değerlendir

if $\Delta f \geq 0$

Yer değiştir

Swap(1,1) 'den çık,

Eğer alt sınıra ulaşıldıysa döngüden çık

Swap (1,2) uygula

Δf değerlerini değerlendir

if $\Delta f \geq 0$

Yer değiştir

Swap(1,2) 'den çık,

Eğer alt sınıra ulaşıldıysa döngüden çık

}

$T := T$ x $Tred$

Döngü sonu

Adım 3.

Çıktılar; kullanılan kutu sayısı, elemanların nihai dağılımı verilmiştir

4.5 Örnek Veri Kümesi İçin Algoritma Süreci

Aşağıdaki örnek test kümesi için program çalıştırılmış ve algoritma süreci Loh'un çalışmasındaki yönteme göre ve önerilen sezgisel yönteme göre adım adım incelenmiştir.

S= {3, 4, 3, 4, 5, 3, 4, 3, 7, 3}

C = 13

4.5.1 Önerilen Yaklaşım: AS_SBC ve Ağırlıklı Tavlama Yöntemi ile Çözüm

4.5.1.1 Başlangıç Çözümü

Adım 1: Listedeki elemanlar artmayan sırada sıralanır.

S = {7, 5, 4, 4, 4, 3, 3, 3, 3, 3}

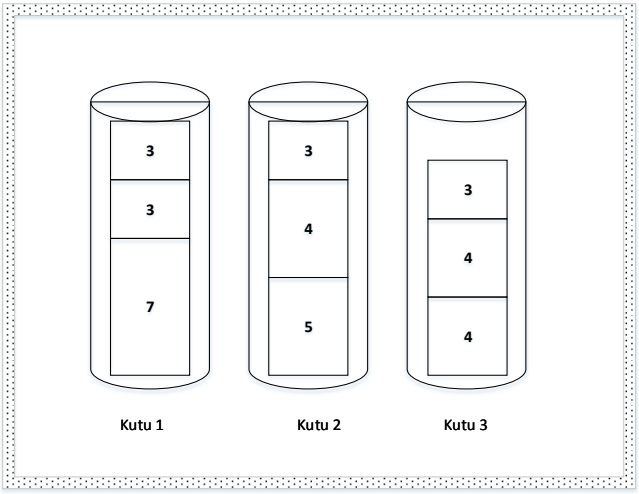
Adım 3: Kullanılacak tüm kutular kapatılır.

Adım 3: *LB* hesaplanır.

$$LB = (7 + 5 + 4 + 4 + 4 + 4 + 3 + 3 + 3 + 3 + 3) / 13 = 3$$

Adım 3: *LB* kadar yani 3 adet boş kutu açılır.

Adım 4: Listedeki elemanlar kutu kapasiteleri göz önünde bulundurularak AS_SBC'ye göre yerleştirilir (Şekil 4.7).



Şekil 4.7: AS_SBC'ye göre çözüm (1)

Adım 5: Yerleştirilen elemanlar listeden kaldırılır. S kümesinde kalan elemanlar aşağıdaki gibidir.

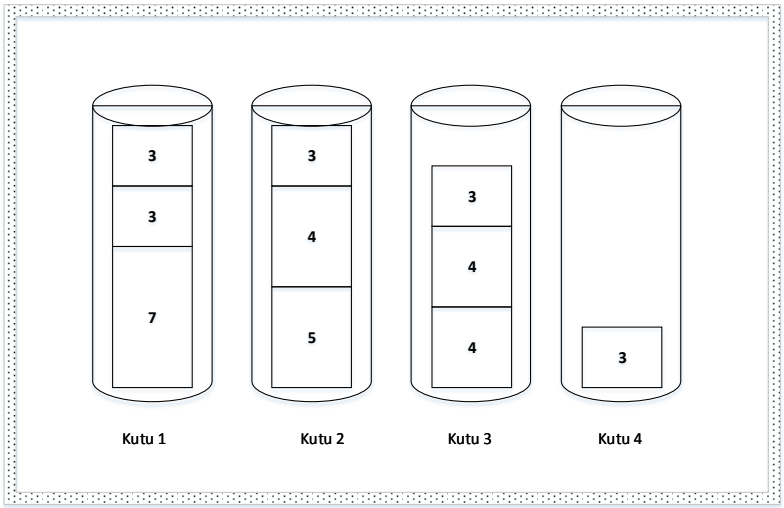
$$S = \{3\}$$

Adım 6: Eğer liste boş değilse adım 3'e dönülür. Liste boş olmadığı için kalan elemanlar için tekrar *LB* hesaplanır.

$$LB = 1$$

LB kadar yani 1 tane daha kutu açılır ve kalan eleman bu kutuya yerleştirilir. Listede eleman kalmadığı için algoritma sonlanır.

Başlangıç çözümü sonucunda listedeki elemanlar aşağıdaki gibi toplamda 4 kutuya yerleştirilir (Şekil 4.8). Başlangıç çözümüne göre ağırlıklı tavlama yaklaşımı kullanılarak yer değiştirme işlemleri uygulanır.



Şekil 4.8: AS_SBC'ye göre çözüm (2)

4.5.1.2 Ağırlık Tavlama Yaklaşımı ile Yer Değiştirme İşlemleri

Başlangıç çözümü belirlendikten sonra her kutu için aşağıdaki formüllere göre ağırlık ataması yapılır. Başlangıç çözümüne göre kutuların ağırlık değerleri aşağıdaki gibi hesaplanmıştır.

$$w_i = 1 + Kr_i$$

r_i = i. kutunun kalan kapasitesi

$$r_i = \left(\frac{C - l_i}{C}\right)$$

C =10

l_i : i.kutudaki öğelerin boyutlarının toplamı

t_{ij} : i. kutudaki j. öğenin boyutunu

K = 0.05

$T = T \; x \; Tred$

$Tred$ = 0,95

T = 1

$$l_i = \sum_{j=1}^{q_i} t_{ij}$$

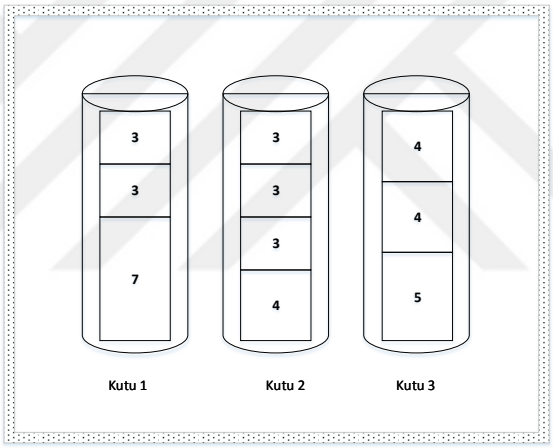
$$W[1] = [1 + 0{,}05 * \left(\frac{13-(7+3+3)}{13}\right)] = 1$$

$$W[2] = [1 + 0{,}05 * \left(\frac{13-(5+4+3)}{13}\right)] = 1$$

$$W[3] = [1 + 0{,}05 * \left(\frac{13-(4+4+3)}{13}\right)] = 1$$

$$W[4] = [1 + 0{,}05 * \left(\frac{13-(3)}{13}\right)] = 1$$

Ağırlık atamaları yapıldıktan sonra yer değiştirme işlemleri yapılır. Bu çalışmada sırasıyla Swap (1,0), Swap(1,1) ve Swap (1,2) işlemleri kullanılmıştır. Ağırlıklı tavlama ve yer değiştirme işlemlerinin ardından elemanlar kutulara aşağıdaki gibi yerleştirilmiştir (Şekil 4.9).



Şekil 4.9: AS_SBC, ağırlıklı tavlama ve yer değiştirme sonucu kutuların durumu

4.5.2 ISAA Çözümü ve Ağırlıklı Tavlama Yöntemi ile Çözüm

Bu aşamada Loh'un çalışmasında bahsedildiği şekilde başlangıç çözümü ISAA olarak belirlenmiştir ve buna göre çözüm geliştirilir.

4.5.2.1 Başlangıç Çözümü

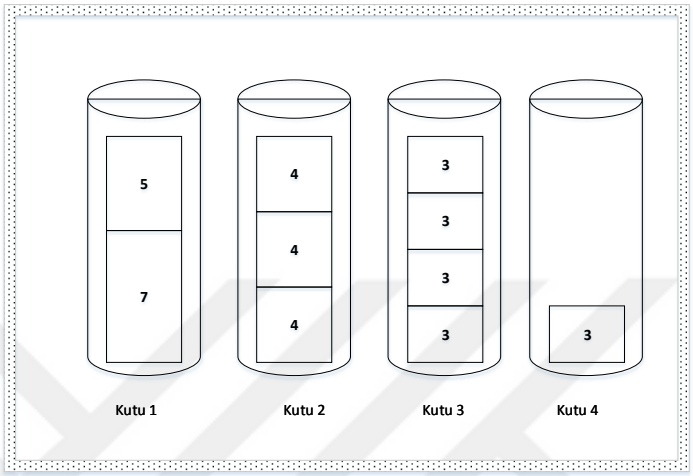
Adım 1: Listedeki elemanlar artmayan sırada sıralanır.

S = {7, 5, 4, 4, 4, 3, 3, 3, 3, 3}

Adım 2: Yeterli kalan kapasiteye göre en düşük numaralı kutuya listedeki ilk öğe yerleştirilir.

Adım 3: Eğer öğe o kutuya sığmıyorsa yeni kutu açılır.

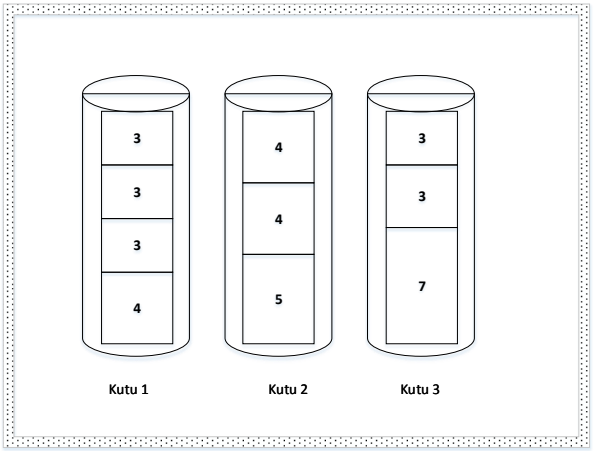
ISAA'ya göre aşağıdaki sonuç elde edilir (Şekil 4.10).



Şekil 4.10: ISAA'ya göre başlangıç çözümü

4.5.2.2 Ağırlıklı Tavlama Yaklaşımı ile Yer Değiştirme İşlemleri

ISAA ile yerleştirilen kutuların ağırlık değerleri hesaplandıktan sonra sırasıyla Swap(1,0), Swap(1,1), Swap(1,2) ve Swap(2,2) işlemleriyle elemanlar yer değiştirilmiştir. Burada önerilen sezgisel yöntemden farklı olarak Swap(2,2) yöntemi de kullanılmıştır. Yer değiştirme işlemleri sonucunda oluşturulan çözüm aşağıdaki gibidir (Şekil 4.11).



Şekil 4.11: ISAA, ağırlıklı tavlama ve yer değiştirme sonucu kutuların durumu

Tablo 4.1'de görüldüğü gibi önerilen sezgisel ile Loh'un çalışması arasında farklılıklar vardır. Önerilen sezgiselde başlangıç çözümü olarak AS_SBC kullanılırken Loh'un yaptığı çalışmada ISAA kullanılmıştır. Ayrıca yer değiştirme operatörlerinde de farklılık vardır. Önerilen yöntemde Swap(2,2) işlemine gerek kalmadan optimum sonuca ulaşıldığı görülmüştür. Bu sayede çözüm süresinin kısalması açısından da avantaj sağlanmıştır. Ancak Loh'un çalışmasındaki yöntem aynı bilgisayar ortamında test edildiğinde Swap(2,2) işlemi olmadan bazı test kümelerinde optimum çözüme ulaşılamadığı gözlemlenmiştir. Her iki algoritmadaki testler için kullanılan bilgisayar, 4 GB RAM ile 2.26 GHz Intel ® Core™ 2 Duo CPU'ya sahiptir. Tek CPU kullanılmıştır.

Tablo 4.1: Önerilen sezgisel ile Loh'un çalışması arasındaki işlem farkı

İşlem	**Loh'un Çalışması**	**Önerilen Sezgisel**
Başlangıç Çözümü	ISAA	AS_SBC
Uygulanan Algoritma	Ağırlıklı Tavlama	Ağırlıklı Tavlama
Yer Değiştirme İşlemleri	• Swap(1,0) • Swap(1,1) • Swap(1,2) • Swap(2,2)	• Swap(1,0) • Swap(1,1) • Swap(1,2)

5. KULLANILAN TEST KÜMELERİ

Aşağıdaki Tablo 5.1'de problemlerin çözümünde kullanılan test kümeleri ve özellikleri verilmiştir. Bu çalışmada U, T, Set, Gau, Was, Hard28 olmak üzere 6 farklı test serisinde 16 farklı test kümesine sahip toplamda 1615 tane örnek test edilmiştir. Her test kümesi farklı eleman sayısına, farklı kapasiteye ve farklı zorluk derecesine sahiptir.

Tablo 5.1: Test kümeleri tanımı (Loh ve diğ. 2006)

Adı	Gösterim	Açıklama
Tekdüze(Uniform)	U120,U250,U 500, 1000	• Kutu kapasitesi C = 150'dir. • Elemanlar tam sayı olup 20 ile 150 arasında tekdüze dağılım göstermektedir. • U120 n = 120 eleman anlamına gelmektedir. Her n = 120, 250, 500 ve 1000 değeri için 20 örneği vardır. • Bu problemler Falkneuer tarafından geliştirilmiştir (1996) ve Carvalho tarafından optimal olarak çözülmüştür (1999). • Bu problemlere [internet 1] kaynağından ulaşabilirsiniz.
Üçlü (Triplet)	T60, T120, T249, T501	• Kutu kapasitesi *C* = 1000'dir. • Elemanlar tam sayı olup 250 ile 500 arasında tekdüze dağılım göstermektedir. • T60 *n* = 60 eleman anlamına gelmektedir. Her *n* = 60, 120, 249 ve 501 değeri için 20 örneğe sahiptir. • Bilinen optimal sonuçlar n/3'e eşit olduğu için bu problem üçlü olarak adlandırılmıştır. • Bu problemler Falkneuer tarafından geliştirilmiştir (1996). • Bu problemlere [internet 1] kaynağından ulaşabilirsiniz.
Set	Set1, Set 2, Set3	• Tekdüze dağılım gösteren Set 1 720 örneğe sahiptir. • Kutu kapasitesi *C* = 100, 120 ve 150 'dir, *n* = 50, 100, 200 ve 500 'dir. • Set 2 480 örneğe sahip olup kutu kapasitesi *C* = 1000'dir ve *n* = 50, 100, 200, 500'dir. Her kutunun ortalama 3 – 9 elemanı vardır. • Set 3 10 örneğe sahip olup kutu kapasitesi C = 100000'dir. • n = 200 ve elemanlar 20000 ile 35000 arasında tekdüze dağılım göstermektedir. Set 3 set kümesinin en zoru olarak kabul edilir. • Bu problemler Scholl, Klein ve Jürgens tarafından geliştirilmiştir (1997) ve 1184 problemin en iyi duruma getirildiğini bildirmişlerdir. • Alvim, Ribeiro, F. Glover, ve D. Aloise (2004) kalan 26 problemler için en iyi sonuçları bildirmişlerdir. • Problem kümelerine [internet 2] kaynağından ulaşabilirsiniz.

Tablo 5.1 : Test kümeleri tanımı (devam) (Loh ve diğ. 2006)

Adı	Gösterim	Açıklama
Was	Was1, Was2	• Was1 100 örneğe sahiptir, kutu kapasitesi C = 1000 ve eleman sayısı n = 100'dür. • En küçük eleman boyutu 150, en büyük eleman boyutu 200'dür. • Was2 100 örneğe sahip, kutu kapasitesi C = 1000 ve eleman sayısı n = 120'dir. • En küçük eleman boyutu 150, en büyük eleman boyutu 200'dür. • Bu problemler Schwerin ve Wäscher tarafından geliştirilmiştir(1997, 1999). Bütün problemler optimal olarak çözülmüştür. • Bu problem kümelerine [internet 4] kaynağından erişebilirsiniz.
Gau	Gau1, Gau2	• Gau1 12 örneğe, Gau2 ise 5 örneğe sahiptir. • Bu 17 problem Wäscher ve Gau 'dan alınmıştır (1996). • Yazarlar tarafından zorlu sorunlar olarak bildirilmektedirler. Bu sorunlardan bazıları optimal olarak çözülmüştür. • Problem kümelerine [internet 4] kaynağından ulaşabilirsiniz.
Hard28	Hard28	• Bu problem kutulama problemi için çok zor 28 örnek içerir. • J. Schoenfield tarafından geliştirilmiştir. • Test kümesine [internet 4] kaynağından ulaşabilirsiniz.

6. ÖNERİLEN SEZGİSEL YÖNTEMLE ELDE EDİLEN SONUÇLAR

Aşağıdaki Tablo 6.1'de U120 test kümesine ait sonuçlar yer almaktadır. Bu tabloda her bir örnek için; kapasite (C), örneklere ait hesaplanan LB değerleri, literatürdeki EIBC, baz alınan Loh 'un çalışmasından elde edilen çözümler (WA1BP : Weight Annealing Algorithm for the One-Dimensional Bin Packing Problem), AS_SBC ve ağırlık tavlama sezgiseli kullanılarak elde edilen çözümler (Önerilen Sezgisel), bu iki çözüme ait sapma ve bu algoritmaların çözüm süreleri gösterilmektedir.

Tablo 6.1'de de görüldüğü gibi u120_08 örneğinde sezgisel algoritmalar kullanılarak elde edilen EIBC'de toplamda 51 kutuya yerleştirilirken önerilen sezgisel yöntemiyle 50 kutuya yerleştirilmiştir. Aynı şekilde u120_19 örneğinde de EIBC sonucu 50 iken önerilen sezgisele göre 49 sonucuna ulaşılmaktadır. Buna göre U120 serisi için EIBC'de toplamda 983 kutu kullanılması gerekirken önerilen algoritmaya göre 981 kutu kullanılması gerekmektedir ve bunlara göre toplamda 0,20 sapma meydana gelmektedir. Böylece önerilen sezgisel sayesinde literatürde yer alan sonuçlardan daha iyi bir sonuç elde edilmiştir. Sürelere bakıldığında ise önerilen sezgisel algoritmanın ortalama 0,08 saniyede WA1BP 'de ise 0,21 saniyede çözüme ulaştığı gözlenmektedir. Buna göre hem toplamda yerleştirilen kutu sayısı hem de çözüm süreleri bakımından önerilen sezgisel algoritmada daha iyi sonuçlar elde edildiği görülmektedir.

Tablo 6.2'de U, T, Set, Was, Gau ve Hard problem kümelerine ait sonuçlar gösterilmektedir. Tabloda bu problem kümelerinde kaç tane örnek olduğunu gösteren "Örnek Sayısı" sütunu yer almaktadır. Test kümelerindeki örneklerin kaç tanesinde optimum sonuca ulaşıldığını gösteren 2 farklı algoritmanın optimum sonuç sütunları yer almaktadır. Örneğin U120 kümesinde 20 örnek yer almakta ve 2 algoritma da bu 20 örnekte optimum sonuca ulaşmıştır. Algoritmalar için ortalama çözüm süresi de Tablo 6.2'de yer almaktadır. WA1BP yönteminde ortalama 0,47 saniyede, önerilen sezgiselde ise ortalama 0,25 saniyede optimum çözümlere ulaşılmıştır. Bu sonuçlara

göre optimum çözüm sayıları aynı olmasına rağmen önerilen sezgisel daha hızlı çözüme ulaşmıştır.

Tablo 6.1: U120 test kümesi için sonuçlar

				OPTİMUM SONUÇLAR			ALGORİTMA SÜRELERİ (ms.)	
Test Kümesi	**C**	**LB**	**EIBC**	**WA1BP**	**Önerilen Sezgisel**	**Sapma**	**WA1BP**	**Önerilen Sezgisel**
u120_00	150	48	48	48	48	0,00	5	5
u120_01	150	49	49	49	49	0,00	**2**	3
u120_02	150	46	46	46	46	0,00	5	**4**
u120_03	150	49	49	49	49	0,00	10	**7**
u120_04	150	50	50	50	50	0,00	**2**	4
u120_05	150	48	48	48	48	0,00	14	**5**
u120_06	150	48	48	48	48	0,00	30	**4**
u120_07	150	49	49	49	49	0,00	17	**5**
u120_08	150	50	51	51	**50**	**1,96**	97	**17**
u120_09	150	46	46	46	46	0,00	89	**29**
u120_10	150	52	52	52	52	0,00	**2**	4
u120_11	150	49	49	49	49	0,00	13	**6**
u120_12	150	48	48	48	48	0,00	16	**8**
u120_13	150	49	49	49	49	0,00	**2**	5
u120_14	150	50	50	50	50	0,00	**2**	3
u120_15	150	48	48	48	48	0,00	8	**4**
u120_16	150	52	52	52	52	0,00	**2**	3
u120_17	150	52	52	52	52	0,00	26	**11**
u120_18	150	49	49	49	49	0,00	9	**4**
u120_19	150	49	50	50	**49**	**2,00**	64	**22**
Toplam		981	983	983	**981**	0,20	415	**153**

Önerilen sezgiseldeki başlangıç çözümünde elemanlar AS_SBC'ye göre daha fazla kutuya yerleştirilmiştir ancak yer değiştirme işlemleri bu çözümde daha kısa sürede yapılabilmektedir. Ayrıca önerilen sezgiselde yer değiştirme işlemlerinde

Swap(1,0) ,Swap(1,1) ,Swap(1,2) kullanılmakta, WA1BP çözümünde Swap(1,0) ,Swap(1,1) ,Swap(1,2) ve Swap(2,2) kullanılmaktadır. Önerilen sezgiselde her iterasyon için diğer algoritmaya göre yer değiştirme daha az yaptığı için çözüm süresi azalmaktadır. Ayrıca önerilen sezgiselde başlangıç çözümünden sonraki yer değiştirme işlemlerini daha iyi yaptığı için çözüm süresi daha iyidir. WA1BP çözümünde Swap(2,2) işlemini kaldırarak algoritmayı çalıştırdığımızda bazı örneklerde optimum sonuca ulaşılmadığı gözlemlenmiştir.

Tablo 6.2: U, T, Set, Was, Gau ve Hard test kümeleri için WA1BP ve Önerilen Sezgisel çözümleri

		WA1BP		**ÖNERİLEN SEZGİSEL**	
Test Kümesi	**Örnek Sayısı**	**Optimum Sonuç**	**Ortalama süre (sn.)**	**Optimum Sonuç**	**Ortalama süre (sn.)**
U120	20	20	0,02	20	**0,01**
U250	20	20	0,16	20	**0,09**
U500	20	20	0,74	20	**0,36**
U1000	20	20	2,32	20	**2,18**
T60	20	20	0,02	20	**0,01**
T120	20	20	0,07	20	**0,03**
T249	20	20	0,31	20	**0,17**
T501	20	20	1,27	20	**0,44**
SET1	720	720	1,85	720	**0,44**
SET2	480	480	0,14	480	**0,03**
SET3	10	10	0,19	10	**0,13**
WAS1	100	100	0,00	100	0,00
WAS2	100	100	0,00	100	0,00
GAU1	12	12	0,03	12	**0,01**
GAU2	5	5	0,16	5	**0,01**
HARD28	28	28	0,21	28	**0,10**
Toplam	1615	1615		1615	
Ortalama			0,47		**0,25**

Tablo 6.3'te ise Loh'un makalesinde yer alan algoritmalar ve çözüm süreleri gösterilmiştir. Yukarıdaki tablolarda yer alan çözüm süreleri Loh'un WA1BP algoritmasına ait kodlar alınarak test edilen sürelerdir. İki algoritma da aynı bilgisayarda çalıştırılıp test edilmiştir.

Tablo 6.3: Loh'un WA1BP algoritmasına göre çözümü (Loh ve diğ. 2006).

		HI_BP		PMBS' + VNS		WA1BP	
Test Kümesi	**Örnek Sayısı**	**Optimum Sonuç**	**Ortalama süre (sn.)**	**Optimum Sonuç**	**Ortalama süre (sn.)**	**Optimum Sonuç**	**Ortalama süre (sn.)**
U120	20	20	0,00	20	0,02	20	0,00
U250	20	20	0.12	19	0,03	20	0,03
U500	20	20	0,00	20	0,04	20	0,18
U100	20	20	0,01	20	0,07	20	1,24
T60	20	20	0,37	20	0,01	20	0,00
T120	20	20	0,85	20	0,02	20	0,00
T249	20	20	0,22	20	0,02	20	0,01
T501	20	20	2,49	20	0,06	20	0,03
SET1	720	720	0,19	694	0,15	720	0,17
SET2	480	480	0,01	474	0,10	480	0,19
SET3	10	10	4,60	2	3,74	10	0,13
Toplam	1370	**1370**		1329		**1370**	
Ortalama			0,87		0,39		**0,18**

Aşağıdaki Tablo 6.4'te ise Gau'ya ait örneklerde elde edilen sonuçlar gösterilmektedir. WA1BP çözümünde 3 örnekte (TEST0058, TEST0068 ve TEST0082) literatürdeki bilinen en iyi sonuçlardan daha iyi sonuçlar elde edilmiştir. Önerilen sezgisel çözümünde ise 1 örnekte (TEST0068) daha iyi sonuç elde edilmiştir. Literatürde 17 örnek için toplamda 290 kutu kullanılmış, WA1BP çözümünde 287, önerilen sezgisel çözümünde de 289 kutu kullanılmıştır.

Tablo 6.4: Gau test kümesine ait WA1BP ve Önerilen Sezgisel sonuçları

Test Kümesi	**EIBC**	**WA1BP**	**Önerilen Sezgisel**
TEST0005	29	29	29
TEST0014	24	24	24
TEST0022	15	15	15
TEST0030	28	28	28
TEST0058	21	**20**	21
TEST0065	16	16	16
TEST0068	13	**12**	**12**
TEST0082	25	**24**	25
TEST0044	14	14	14
TEST0049	11	11	11
TEST0054	14	14	14
TEST0055	15	15	15
TEST0055	20	20	20
TEST0075	13	13	13
TEST0084	16	16	16
TEST0095	16	16	16
Toplam		**287**	289

Literatürde bahsedilen Fleszar ve Hindi (2002) ‘nin çalışmasında yer alan MAK algoritmalarından elde edilen sonuçlar ile başlık 2.4’te bahsedilen H-SGGA yöntemi ile H-SGGA ve DKA yönteminin birlikte kullanıldığı sezgisel bir algoritmaya ait sonuçlar ve bu çalışmada yer alan önerilen sezgisel Tablo 6.5’te gösterilmektedir. Fleszar ve Hindi’nin çalışmasında MAK, GMAK, SMAK, ÖMAK yöntemleri ve SMAK ile DKA algoritmalarını birlikte kullanılan yöntemden elde edilen sonuçlar karşılaştırılmıştır. Her bir yöntemden elde edilen optimum sonuç “Opt.” sütununda, bu yöntemlere ait ortalama çalışma süreleri ise “Ort. Süre(s)” sütununda gösterilmiştir (Tablo 6.5). Sonuçlarda da görüleceği gibi en iyi sonuca önerilen sezgiselde ulaşılmıştır.

Tablo 6.5: Önerilen Sezgisel ile MAK (Fleszar ve Hindi 2002), H-SGGA, H-SGGA + SMAK (Singh ve Gupta 2007) yöntemlerinin sonuçları

		Önerilen Sezgisel		**H-SGGA**		**H-SGGA + SMAK**		**MAK**		**GMAK**		**SMAK**		**ÖMAK**		**SMAK + DKA**	
Test Kümesi	**Örnek Sayısı**	**Opt.**	**Ort. süre (sn.)**	**Opt.**	**Ort. süre (sn.)**	**Opt.**	**Ort. süre (sn.)**	**Opt.**	**Ort. Süre (sn.)**	**Opt.**	**Ort. Süre (sn.)**	**Opt.**	**Ort. Süre (sn.)**	**Opt.**	**Ort. Süre (sn.)**	**Opt.**	**Ort. Süre (sn.)**
U120	20	20	0,01	20	0,01	20	0,01	12	0	19	0	11	0,04	19	0,03	20	0,04
U250	20	20	0,09	18	0,63	19	0,42	10	0	16	0,01	13	0,06	16	0,13	19	0,16
U500	20	20	0,36	20	0,27	20	0,05	11	0,01	16	0,03	14	0,14	17	0,33	20	0,14
U100	20	20	2,18	20	0,89	20	0,25	7	0,02	13	0,19	16	0,25	19	1,29	20	0,27
T60	20	20	0,01	6	0,2	20	0	0	0	0	0,01	20	0,02	0	0,02	20	0,01
T120	20	20	0,03	0	0,82	20	0,01	0	0	0	0,02	20	0,04	0	0,06	20	0,04
T249	20	20	0,17	1	2,83	20	0,01	0	0,01	0	0,08	20	0,04	0	0,14	20	0,04
T501	20	20	0,44	2	9,13	20	0,02	0	0,02	0	0,32	20	0,1	0	0,37	20	0,1
SET1	720	720	0,44	717	0,34	719	0,31	-	-	-	-	-	-	-	-	-	-
SET2	480	480	0,03	479	0,05	480	0,04	-	-	-	-	-	-	-	-	-	-
SET3	10	10	0,13	10	0,42	10	0,92	-	-	-	-	-	-	-	-	-	-
Toplam	1370	**1370**	3,89	1293	15,59	1368	2,04	40	0,06	64	0,66	134	0,69	71	2,37	159	0,8
Ortalama			0,35		0,47		0,19		0,01		0,08		0,09		0,30		0,1

7. SONUÇ VE ÖNERİLER

Bu çalışmada bir boyutlu kutulama problemi ele alınmış, AS_SBC ile birlikte ağırlıklı tavlama ve yer değiştirme algoritmaları kullanılarak sezgisel bir yöntem geliştirilmiştir.

Kutulama problemi için geliştirilen bu yöntem iki aşamadan oluşmaktadır. İlk aşamada elemanların kutulara ilk yerleştirilmesini sağlayan başlangıç sezgiseli önerilmiştir. Her test kümesi Bölüm 3.2 başlığı altında bahsedilen bütün algoritmalar kullanılarak başlangıç çözümler elde edilmiştir. Alt sınır yaklaşımını temel alan AS_SBC, literatürde yer alan diğer yerleştirme algoritmalarıyla karşılaştırılmış ve sonuçlar tablolar halinde sunulmuştur.

Problem çözümünün ikinci aşaması olarak, ağırlıklı tavlama yöntemiyle birlikte elemanlara ağırlık ataması yapılıp başlangıç çözümüyle oluşturulan kutular yer değiştirme işlemleri sayesinde tekrar düzenlenmiştir. Ağırlıklı tavlama ve yer değiştirme algoritmalarıyla birlikte, başlangıç çözümünde elde edilen toplam kutu sayısından daha az sayıda kutuya yerleştirme yapılmıştır. Bu çalışmada yer değiştirme işlemlerinde Swap (1,0), Swap (1,1) ve Swap (1,2) algoritmaları kullanılmıştır.

Elde edilen sonuçlar Loh'un çalışmasında yer alan WA1BP yöntemiyle karşılaştırılmıştır. Önerilen sezgisel algoritmanın WA1BP yönteminden farkı; başlangıç çözümü olarak farklı bir sezgisel başlangıç algoritması kullanılması ve yer değiştirme işlemlerinde WA1BP yönteminin aksine Swap(2,2) yer değiştirme algoritmasını kullanmadan optimum sonuçlara ulaşabilmesidir. Bu çalışmanın sonucunda iki yöntemin çözüm süreleri karşılaştırılarak önerilen sezgisel yöntemin daha kısa sürede çözüme ulaştığı gözlemlenmiştir. Ayrıca U120 test kümesindeki 2 örnekte literatürdeki EIBC'den daha iyi bir çözüm elde edildiği gözlemlenmiştir.

Önerilen sezgisel yaklaşım sayesinde bir boyutlu kutulama problemi için çözüm süresi kısaltılmış, 2 örnekte de literatürden daha iyi sonuçlar elde edilmiştir. Bu yöntem iki boyutlu ve üç boyutlu kutulama probleminde de kullanılarak sonuçlar

değerlendirilebilir. Ayrıca bu yöntem gezgin satıcı problemi, sırt çantası problemi, maksimum doyum problemi ve araç rotalama problemi gibi problemlerde de kullanılarak yeni bakış açıları oluşturulabilir.

8. KAYNAKLAR

Alvim, A. C. E., Ribeiro, C. C., Glover, F., & Aloise, D., "A hybrid improvement heuristic for the one-dimensional bin packing problem", In Extended Abstracts of the 4th Metaheuristics International Conference, 63-68, (2001).

Alvim, A. C. F., Glover, F., & Aloise, J. J., "A hybrid improvement heuristic for the one-dimensional bin packing problem", *Journal of Heuristics*, 10, 205–229, (2004).

Alvim, A. C. F., Glover, F., Ribeiro, C. C., & Aloise, J. J., "Local search for the bin packing problem", In Extended Abstracts of the 3rd Metaheuristics International Conference, 7–12, (1999).

Ataei, M., "A branch-and-price algorithm for bin packing problem", Msc Thesis, York University, (2015).

Aydın, İ. "Tavlama Benzetimi Algoritması (Simulated Annealing)", Fırat Üniversitesi, *Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı* (2013).

[internet 1] : Beasley, J., "OR-Library[online]", (6 Nisan 2016), http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html, (1990).

Beisiegel, B., Kallrath, J., Kochetov, Y., & Rudnev, A., "Simulated annealing based algorithm for the 2D bin packing problem with impurities", *Operations Research Proceedings 2005*, 309-314, (2005).

Bhatia, A. K., & Basu, S. K., "Packing bins using multi-chromosomal genetic representation and better fit heuristic", *Lecture Notes in Computer Science*, 181–186, (2004).

[internet 2] : Brandão, F., "Arc-flow Formulation (w/ graph compression) Results",(10 Mayıs 2016), http://www.dcc.fc.up.pt/~fdabrandao/research/vpsolver/results/

Caprara, A.,and Pferschy, U., "Modified subset sum heuristics for bin packing", *Information Processing Letters*, 96(1), 18–23, (2005).

Caprara, A., and Pferschy, U., "Worst-case analysis of the subset sum algorithm for bin packing", *Operations Research Letters*, 32(2), 159–166, (2004).

Carvelho, J. M. V., "Exact solutions of bin-packing problems using column generation and branch and bound", *Annals of Operations Research*, 86, 629 – 659, (1999).

Coffman, Garey, M. and Johnson, D., "Approximation algorithms for bin packing problems: A survey", *Analysis and Design of Algorithms in Combinatorial Optimization*, 147-172, (1997).

Dantzig, G. B., and Philip, W., "Decomposition principle for linear programs", *Operations Research*, *8*(1), 101–111, (1960).

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

Eilon, S., and Christofides, N., "The loading problem", *Management Science*, 259–268, (1971).

Elidan, G., Ninio, M., and Friedman, N., "Data perturbation for escaping local maxima in learning", 132–139, (2002).

Ensari, M., "Konteynır yükleme problemine sezgisel bir yaklaşım", Yüksek Lisans Tezi, Gazi Üniversitesi, *Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabili Dalı* (2007).

Falkenauer, E., “A hybrid grouping genetic algorithm for bin packing”, *Journal of Heuristics*, 2, 5–30, (1996).

Fekete, S. P., and Schepers, J., “New classes of fast lower bounds for bin packing problems”, *Mathematical Programming*, 91(1), 11–31, (2001).

Fleszar, K., and Hindi, K. S., “New heuristics for one-dimensional bin-packing”, *Computers & Operations Research*, 29(7), 821–839, (2002).

Ford, L. R. J., and Fulkerson, D. R., “A linear programming approach to the cutting-stock problem”, *Operations Research*, *9*(6), 849–859, (1961).

Ford, L., and Fulkerson, D., “A suggested computation for maximal multicommodity network flows”, *Management Science*, *5*(1), 97–101, (1958).

Garey, M. R., and Johnson, D. S., “A guide to the theory of NP-completeness”, *Computers and Intractability*. W.H. Freeman and Company, San Francisco, California, (1979).

Gilmore, P. C., and Gomory, R. E., “A linear programming approach to the cutting-stock problem”, *Operations Research*, *9*(6), 849–859, (1961).

Gourgand, M., Grangeon, N., and Klement, N., “Activities planning and resource assignmenton multi-place hospital system: Exact and approach methods adapted from the bin packing problem”, 7th International Conference on Health Informatics, Angers, France, 117–124, (2014).

Gourgand, M., Grangeon, N., and Klement, N., “An analogy between bin packing problem and permutation problem: A new encoding scheme”, *IFIP International Conference on Advances in Production Management Systems*, 572–579, (2014).

Güler, A., “Tamsayılı programlama problemleri için garanti değerli algoritmalar”, Yüksek Lisans Tezi, Ege Üniversitesi, *Fen Bilimleri Enstitüsü, Matematik Anabilim Dalı* (2008).

Hansen, P., and Mladenović, N., "An introduction to variable neighborhood search", *Meta-heuristic*, 433-458, (1999).

Hashim, N. A., Zulkipli, F., Januri, S. S., and Shariff, S. S. R., "An alternative heuristics for bin packing problem", *Proceedings of the International Conference on Industrial Engine* , 1560, (2014).

Ho, J. C., and Gupta, J. N. D., "A new heuristic algorithm for the one-dimensional bin-packing problem", *Production Planning and Control*, 10(6), 598–603, (1999).

Hübscher, R., and Glover, F., "Applying tabu search with influential diversification to multiprocessor scheduling", *Computers & Operations Research*, *21*(8), 877–884, (1994).

Imahori, S., Yagiura, M., and Nagamochi, H., "Practical algorithms for two-dimensional packing", *Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/CRC Computer & Information Science Series*, 13, (2007).

[internet 3] : Janković, M.,"Genetic algorithm for bin packing problem",(17 Mayıs 2017) http://www.codeproject.com/Articles/633133/ga-bin-packing, (2013).

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L., "Worst-case performance bounds for simple one-dimensional packing algorithm", *SIAM J Ournal of Computing*, *3*, 299–326, (1974).

Kabak, Ö. , "TP Sorunlarının Çözümleri", İstanbul Teknik Üniversitesi, *Endüstri Mühendisliği Bölümü*, (2012).

Kantorovich, L. V., "Mathematical methods of organizing and planning production", *Management Science, (English Translation of a 1939 Paper Written in Russian)*, *6*(4), 366–422, (1960).

Kirkpatrick, S., L., Gelatt, C.D., Vecchi, M.P. "Optimization by Simulated Annealing", *Science, New Series, 220*, 671-680, (1983).

Küçük, M., "Konteyner yükleme probleminin karma evrimsel algoritmalar ile çözümü", Yüksek Lisans Tezi, *Hava Harp Okulu Havacılık ve Uzay Teknolojileri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı* (2010).

Land, A.H. and Doig, A. G., "An automatic method for solving discrete programming problems"*, Econometrica*, 28, 497-520, (1960).

Lodi, A., "Algorithms for two-dimensional bin packing and assignment problems", Doktorarbeit, DEIS, Universita di Bologna, (1999).

Loh, K.-H., Golden, B., and Wasil, E., "Weighted annealing heuristics For solving bin packing and other combinatorial optimization problems: Concepts, algorithms, and computational results", PhD Thesis, University of Maryland, (2006).

Loh, K.-H., Golden, B., and Wasil, E., "Solving the one-dimensional bin packing problem with a weight annealing heuristic", *Computers & Operations Research*, 35(7), 2283–2291, (2008).

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

Martello, S., and Toth, P., "Algorithms for knapsack problems", *North-Holland Mathematics Studies*, *132*, 213–257, (1987).

Ninio, M., and Schneider, J. J., "Weight annealing", *Physica A: Statistical Mechanics and its Applications*, 349(3), 649-666, (2004).

[internet 4] : Oliveira, J., "ESICUP- Datasets", (20 Nisan 2016), https://paginas.fe.up.pt/~esicup/datasets, (2003).

Ozcan, S. O., Dokeroglu, T., Cosar, A., and Yazici, A., "A novel grouping genetic algorithm for the one-dimensional bin packing problem on GPU", *Computer and Information Sciences*, 52–60, (2016).

Peeters, M., and Degraeve, Z., "Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem", *Operational Research*, 170, 416–439, (2004).

Poli, R., Woodward, J., and Burke, E. K., "A histogram-matching approach to the evolution of bin-packing strategies", *IEEE Congress on Evolutionary Computation 2007 (CEC 2007),* 3500–3507, (2007).

Rohlfshagen, P., and Bullinaria, J., "A genetic algorithm with exon shuffling crossover for hard bin packing problems", *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* 1365–1371, (2007).

Sadykov, R., and Vanderbeck, F., "A branch-and-price algorithm for the bin packing problem with conflicts", *INFORMS Journal on Computing*, *23*(3), (2008).

Scholl, A., Klein, R., and Jürgens, C., "BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem", *Computers & Operations Research*, 24(7), 627–645, (1997).

Schwerin, P., and Wäscher, G., "A new lower bound for the bin packing problem and its integration into MTP", *Pesquisa Operational*, *19*, 111–129, (1999).

Schwerin, P., and Wäscher, G., "The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP", *International Transactions in Operational Research*, *4*(5–6), 377–389, (1997).

Singh, A., and Gupta, A. K., "Two heuristics for the one-dimensional bin-packing problem", *OR Spectrum*, *29*(4), 765-781, (2007).

Türkay, M., "Optimizasyon Modelleri ve Çözüm Metodları", Koç Üniversitesi, *Endüstri Mühendisliği Bölümü*, (2006).

Vanderbeck, F., “Computational study of a column generation algorithm for bin packing and cutting stock problems”, *Maths Programming*, *86*, 565 – 594, (1999).

Wäscher, G., and Gau, T., “Heuristics for the integer one-dimensional cutting stock problem: a computational study”, *OR Spektrum*, *18*, 131–144, (1996).

9. ÖZGEÇMİŞ

1991 yılında Denizli'nin Tavas ilçesinde doğan Neriman İnak ilköğretim ve lise eğitimini Tavas-Karahisar'da tamamlamıştır. Akademik eğitimini 2009-2014 yılları arasında Pamukkale Üniversitesi Bilgisayar Mühendisliği bölümünde tamamlamış, 2014'te yine Pamukkale Üniversitesi Bilgisayar Mühendisliği Anabilim dalında yüksek lisans eğitimine başlamıştır. Yüksek lisans eğitimi sırasında 2014-2016 yılları arasında Denizli'de bulunan Er-Bakır A.Ş'de yazılım uzmanı olarak görev yapmış, 2016 yılından itibaren ise Denizli Büyükşehir Belediyesi'nde yazılım geliştirme birim sorumlusu olarak çalışmaya devam etmektedir.

Adı Soyadı	: Neriman İNAK
Doğum Yeri ve Tarihi	: Tavas / DENİZLİ - 30.10.1991
Lisans Üniversite	: Pamukkale Üniversitesi
Elektronik posta	: nerimann_91@hotmail.com
	ninak@denizli.bel.tr
İletişim Adresi	: Denizli Büyükşehir Belediyesi,
	Altıntop Mah. Lise Cad. No:2
	Merkezefendi / DENİZLİ