T.C.
Bahçeşehir Üniversitesi

# İKİ BOYUTLU KESME PROBLEMİ İÇİN SEZGİSEL YAKLAŞIM İLE BİR UYGULAMA

Yüksek Lisans Tezi

Yakup Alper ERDOĞAN

İSTANBUL, 2010

T.C
Bahçeşehir Üniversitesi
Fen Bilimleri Enstitüsü
Bilgi Teknolojileri

# İKİ BOYUTLU KESME PROBLEMİ İÇİN SEZGİSEL YAKLAŞIM İLE BİR UYGULAMA

Yüksek Lisans Tezi

Yakup Alper ERDOĞAN

Tez Danışmanı: Yrd.Doç.Dr. Orhan GÖKÇOL

İstanbul, 2010

**T.C.**
**BAHÇEŞEHİR ÜNİVERSİTERSİ**
**Fen Bilimleri Enstitüsü**
**Bilgi Teknolojileri**

| | |
|---|---|
| Tezin Başlığı | : İki Boyutlu Kesme Problemi İçin Sezgisel Yaklaşım ile Bir Uygulama |
| Öğrencinin Adı Soyadı | : Yakup Alper ERDOĞAN |
| Tez Savunma Tarihi | : 18 OCAK 2010 |

Bu yüksek lisans tezi Fen Bilimleri Enstitüsü tarafından onaylamıştır.

İmza

Y. Doç. Dr. F. Tunç BOZBURA
Enstitü Müdürü Vekili

Bu tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Tez Sınav Jürisi Üyeleri :

Y. Doç. Dr. Orhan GÖKÇÖL (Tez Danışmanı)

Y. Doç. Dr. F. Tunç BOZBURA (Yardımcı Danışman)

Doç. Dr. Erkan BAYRAKTAR

Y. Doç. Dr. Ahmet BEŞKESE

# ÖNSÖZ

Teknoloji, insanın doğayı taklit etmeye çalışması ile başladı ve ulaşdığı en büyük adım ise bilgisayar olmuştur. Günümüzde bir çok işi insan faktörünü ortadan kaldırarak yapılacak işleri kısa zamanda çözmek için kullanılan bilgisayar her alanda önemini korumaktadır. Optimizasyon yöntemlerinin bilgisayar tarafından çözülebiliyor olması bir çok önemli firmanın stoklarını verimli kullanımı alanında büyük fayda sağlamıştır. Bu sebeple polinomial zamanda istenilen işi yapan algoritmalar değer kazanmaktadır.

Bu çalışmada endüstriyel alanda en sık karşılaşılan kesme ve yerleştirme problemlerine çözüm aranmaktadır. İzlenilen yol, geliştirilen yöntemlerin süresel karmaşıklığının polinomial olmasıdır. Optimal çözüme ulaşmak için bilinen tek yöntem ise çok boyutlu uzayda kombinasyonel eniyilemedir. Optimal çözüm için polinomail zamanda çözüm üreten algoritmalar 2 boyutlu kesme problemi için günümüzde sonuç bulunamamış sorulardan bir tanesidir. Çalışmanın amacı farklı algoritmaların bir arada kullanarak sonuca olan katkılarının karşılaştırılmasıdır.

Bu çalışmayı hazırlarken değerli yardım, fikir ve bilgilerini benden esirgemeyen
Sayın Yrd. Doc. Dr. Orhan Gökçöl ‘ a ve Sayın Yrd. Doc. Dr. F. Tunç Bozbura‘ ya, sonsuz manevi desteği ve emeği için eşim ve aileme teşekkürü bir borç bilirim.

OCAK - 2010 Yakup Alper ERDOĞAN

# ÖZET

İKİ BOYUTLU KESME PROBLEMİ İÇİN
SEZGİSEL YAKLAŞIM İLE BİR UYGULAMA

ERDOĞAN, Yakup Alper

Bilgi Teknolojileri
Tez Danışmanı: Yrd.Doç.Dr. Orhan GÖKÇÖL

Ocak, 2010, 114 Sayfa

Pek çok değişik endüstride farklı koşullar ve amaçlarla karşımıza çıkan kesme problemlerinin her biri birer eniyileme problemidir. Bu problemler NP-tam sınıfında olduğundan çözümü bulmak için kombinasyonel eniyileme yapılır. Kesme problemlerinin zorluğu, büyük parça üzerine yerleştirilecek parçaların geometrisine ve problemin doğasından gelen kısıtlamalarına (çok farklı çözümlerin varlığına) bağlıdır. Problemlerin çözüm yaklaşımları çeşitli sayıdaki bu kısıtlamalara bağlı olarak geliştirilir.

İlk sığan algoritmasının en kötü durumda %22 optimal sonuca yaklaştığı bilinmektedir. Bu sebepten ötürü, optimal çözümlere yaklaşmak için çeşitli sezgisel yöntemler kullanılmıştır.

Bu çalışmada iki boyutlu kesme problemi için genetik algoritma ve dinamik programlama yardımı ile oluşturulmuş sezgisel bir yerleştirme algoritması geliştirilmiştir. Ayrıca seçim ve yerleşim algoritmaları olarak çalışma iki bölüme ayrılmış ve yerleşim algoritması olarak ilk sığan (first fit) ve dinamik programlama, seçim algoritması olarak genetik algoritma kullanılmıştır. Yerleşim algoritmalarının problem çözümüne etkisi incelenmiştir.

**Anahtar Kelimeler**: Genetik Algoritma, Sezgisel Yöntem, Sırt Çantası Problemi, İki Boyutlu Kesme Problemi, Dinamik Programlama, İki Boyutlu Bidon Paketleme Proble

## ABSTRACT

A HEURISTIC APPROACH FOR
TWO DIMENSIONAL CUTTING PROBLEM

ERDOĞAN, Yakup Alper

Information Technologies
Supervisor : Yrd.Doç.Dr. Orhan GÖKÇÖL

Ocak, 2010, 114 page

Each cutting problem is an optimization problem where it occures in many different industries with different conditions and objectives. This problem is member of NP-complete class so solution needs to be found with combinational optimization. Difficulty of cutting problems are related with geometry of the small parts which will be placed on major part and limitations (the existence of different solutions). A variety of approaches to the solution of problems are developed depending on the number of these restrictions.

It is known that the first-fit algorithm may reach to 22% optimal results in the worst case. For this reason, various heuristic methods are used to find more optimal solutions.

In this tesis, placement algorithm has been devoloped with genetic algorithms and dynamic programming for two-dimensional cutting problem. In addition, selection and placement algorithms work as divided into two parts: the first fit algorithm and dynamic programming as placement algorithm, genetic algorithms are used as the selection algorithm. The effect of placement algorithms to the problem solutions were examined.



**Keywords:** Genetic Algorithm, Heuristic, Knapsack Problem, Two Dimensional Cutting Problem, Dynamic Programming, Two Dimesional Bin Packing Problem

# İÇİNDEKİLER

## TABLOLAR

# ŞEKİLLER

# KISALTMALAR

Kesilecek ya da yerleşimin yapılacağı büyük dikdörtgenin yüksekliği .....L

Kesilecek ya da yerleşimin yapılacağı büyük dikdörtgenin genişliği .....W

Kesilecek ya da yerleşim yapılacak küçük dikdörtgenin yüksekliği .....l

Kesilecek ya da yerleşim yapılacak küçük dikdörtgenin genişliği.....w

İki Boyutlu Kesme Problemi.....2BKP

Genetik Algoritma.....GA

# BÖLÜM 1 : GİRİŞ

İki boyutlu kesme problemi (2BKP) geçtiğimiz kırk yıldan daha uzun süredir kombinasyonal optimizasyon alanında en çok çalışılan konulardan bir tanesi olmuştur. Bunun sebebi hem formülasyonunun çok basit olması hemde kullanım alanının genişliğidir. 2BKP tekstil, cam, mobilya ve çelik gibi birçok endüstriyel alanda kullanılmaktadır.

Genel bir tanım yapmamız gerekirse; 2BKP çeşitli genişlik ve yüksekliklere sahip dikdörtgenlerin kendilerinden daha büyük bir dikdörtgenin içine en iyi şekilde yerleştirme (büyük diktörtgeni kendinden daha küçük diktörtgenlere kesme) problemidir.

Bu problemin çözümü seri üretim yapan sektörlerde, maliyetleri düşürmek ve verimi arttırmak için çok değerlidir. Stok kesme probleminin hedefi müşteri taleplerine göre en optimum planı yaparak 2 boyutlu bir parçadan talebe göre malzeme çıkarmaktır.

Pek çok değişik endüstride farklı koşullar ve amaçlarla karşımıza çıkan kesme problemlerinin her biri birer eniyileme problemidir. Bu problemlerin çözümü için belirli bir matematiksel model gösterilemediğinden, çözümü bulmak için çok boyutlu uzayda kombinasyonel eniyileme yapılır. Kesme problemlerinin zorluğu, büyük parça üzerine yerleştirilecek parçaların geometrisine ve problemin doğasından gelen kısıtlamalarına (çok farklı çözümlerin varlığına) bağlıdır. Problemlerin çözüm yaklaşımları çeşitli sayıdaki bu kısıtlamalara bağlı olarak geliştirilir.

Problem ilk bakışta kullanım alanı geniş görünmeyebilir fakat kesme ve yerleştirme problemleri, zaman sıralaması problemleri, iletişim ile ilgili karışık ağ yapıları, rotalama problemleri, gezgin satıcı gibi bir çok problem 2BKP nin alt problemleri olarak alınabilir. Bu nedenle bu konuda bir çok algoritma yazılmıştır fakat optimal sonucu polinomial zamanda verememektedirler. Bunun en büyük sebebi algoritmaların büyük bir kısmının heuristic (sezgisel) yöntemlerle oluşturulmuş olmasıdır. Deterministik yöntemler

uygulanarak hazırlanmış algoritmalar ise süre ve kaynak sorunu ile karşı karşıya gelmektedirler. Aynı zamanda problemin bütün olasıklarının denendiği kombinasyonel eniyileme algoritmaları ile çözülmesi çok büyük bir bellek alanı kaplıyacağı gibi aynı zamanda programın hızınıda olumsuz etkileyeceği için bu tarz algoritmalarında uygulanması bir yarar getiremez. Arama uzayının kısıtlı olduğu durumlar haricinde optimal sonuca polinamial zamanda ulaşılması neredeyse imkansızdır. Probleme farklı en iyi çözümlerim bulunması için, büyük arama uzayı içinde düzenli bir arama gereklidir (Söke, 2003). Kesme ya da yerleştirme problemleri tam bir matematiksel modele sahip olmayan NP-TAM sınıfına girmektedir. Bu sebeple yönlendirilmemiş arama oldukça verimsiz sonuçlar üretmektedir (Callaghan et al. 1999)

Paketleme (kesme) problemleri üzerindeki araştırmaların büyük bir kısmını kesin çözüm üretmek için kullanılan doğrusal programlama teknikleri oluşturmaktadır. Kesin çözümler için geliştirilen algoritmalar sadece ufak kümeler için etkili çalışabilmektedir. Bu sebeple daha büyük kümeler için başka teknikler uygulanmaktadır.

Bellman(1950) dinamik programlama modelini ve çözümünü geliştirmesi yöneylem araştırmaları için önemli bir adım olmuştur. Lineer programların çözümü için Simpleks yöntem Dantzig (1947) tarafından geliştirildi. Bu iki çalışma, optimizasyon dünyasında gerçekten bir devrim sayılmaktadır. Bir çok optimizasyon problemi bu iki yöntem yardımıyla kesin çözümlere ulaşılmıştır.

Steudel (1979) çalısmasında kutulara yerleştirilecek olan parçaların belirli ve eşit ölçülerde oldugu, ancak “giyotin kesme” biçiminde bir yerlesimin şart olmadığı bir probleme, sezgisel bir algoritma geliştirmiştir. Amaç, yerleştirme sırasında kullanılmayan alanın tüm alana oranını minimum yapacak yerlestirme biçimini seçmektir.

Hodgson (1982) yaptığı çalışmada degisik boyutlardaki ufak parçaları büyük parçanın üzerine yerleştirerek maksimum şekilde alana yerleşim sağlayacak bir sezgisel algoritma geliştirmiştir.

Bischoff ve Dowsland (1982) üretim dizaynı ve dagıtımı ile ilgili problemlerin mikro bilgisayar ortamındaki uygulamaları üzerinde çalısmıslardır. 2-boyutlu kesme problemlerinde sezgisel çözümlerin yardımı ile büyük ilerlemeler kaydedilmistir. Bu çalısmaların yanı sıra, analitik çözümlerle yardımı ile yapılan çalısmalar bulunmaktadır.

Dowsland (1987) tarafından geliştirilen analitik teknik, kargo yükleme ile ilgili her türlü probleme uygulanabilmekte ve sonuçları bilgisayardan birkaç dakikada alınabilmektedir ancak optimal sonucu garanti etmemektedir.

Analitik çözümlerle ilgili önemli bir çalışma Beasley (1985) tarafından yapılmıştır. Bu çalışmada kesme kaybı problemlerini tamsayılı programlayarak yapmıştır. Beasley, büyük bir dikdörtgen ana parçadan küçük parçaların kesildigi durumda kesilen küçük parçaların toplam degerini maksimum yapan bir algoritma gelistirmistir. Aynı yıl Beasley (1985) iki boyutlu kesme problem için dal ve sınır algoritması ve Lagrange gevşetmesi yöntemlerini kullanarak yeni bir algoritma geliştirmiştir.

Chen, Sarin ve Ram (1991) iki boyutlu kesme problemlerinde optimal sonuç veren dogrusal programlama modeli kurmuslardır. Chen, Sarin ve Ram ise birden fazla sayıdaki yerleştirelecek alan koşulunu ekleyen bir model gelistirmislerdir.

Ayrıca Lodi, Marcello ve Monaci (2002) matematiksel modeller, sınır algoritmaları, yaklaşım algoritmaları, sezgisel ve sezgi üstü (meta-heuristic) yöntemlerin iki boyutlu kesme problemlerinde ki uygulamalarının incelendiği kapsamlı bir araştırma yapmıştır.

Genetik algoritma tekniği, Michigan Üniversitesinde yer alan John Holland tarafından 1970 li yıllarda ortaya çıkmıştır. Smith (1985), ilk kez paketleme problemlerine çözüm bulmak için genetik algoritmayı kullanmıştır. Jain ve Gea (1998) yaptığı çalışmada genetik algoritma kullanarak çokgen parçaların yerleşimi ile ilgili bir çalışma hazırlamışlardır. Genetik algritma işlemcileri geliştirilerek hybrid bir yaklaşımda bulunulmuştur. Ayrıca

Hupper ve Turton (1999) tarafından sezgisel bir yerleşim yöntem geliştirerek ve genetik algoritma kullanarak önemli bir çalışma yapmışlardır.

Sonuç olarak günümüzde en çok çalışılan problemler arasına 2BKP de dâhil olmaktadır ve bu çalışmada 2 BKP'ne seçme ve yerleştirme algoritmalarının üzerinde durularak iki farklı algoritma olarak düşünülerek sezgisel yöntemler ile çözüm getirme yolları aranmaktadır.

Yapılan çalışma iki nokta üzerinde algoritma farklılıkları ve sonuçları üzerinde durmaktadır.

1- Seçme Algoritması
2- Yerleştirme Algoritması

Seçme işlemi genetik algoritma işlemcileri ile oluşturulmuştur. Yerleştirme algoritması için ise iki farklı algoritma seçilmiş ve sonuca olan katkıları karşılaştırılmıştır.

# BÖLÜM 2 : PROBLEMİN TANIMI VE FORMÜLASYONU

## 2.1 – PROBLEMİN TANIMI

Kesme ve yerleştirme problemleri, bir parça üzerinden birden çok sayıda küçük parçaların ayrılmasının ya da yerleştirmenin bulunması ile ilgilenen eniyileme problemleridir (Leung et al. 2001). Kesilecek olan ana parçanın boyutu küçük parçaların alanlarına eşit olabileceği gibi sınırsız da olabilir. Kesim için çeşitli sayıda kısıt aşağıdaki şekilde tanımlanabilmektedir.

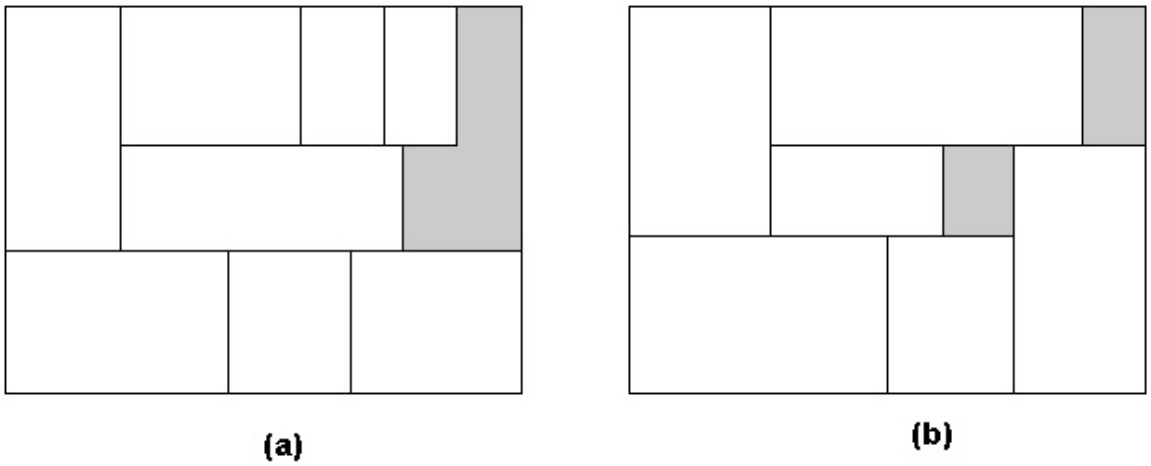
- Küçük parçalar yatay veya dikey döndürülebiliyor mu,
- Büyük parça üzerinde herhangi bir hatalı kısım var mı,
- Kesilecek cisimler dikdörtgen mi yoksa çok köşeli cisimler mi,
- Kesilecek küçük parçalar düzgün köşelimi yoksa değilmi,
- Küçük parçaların ağırlık (öncelik) değeri var mı,

Ayrıca kesim biçimine bağlı olarak probleme giyotinli ve giyotinsiz kesim olarak iki sınıflama daha getirilmiştir (Haessler 1991).

Giyotinli (guillotineable) yerleşim için giyotinli problemler özel bir kısıtlama içerirler. Bu kısıtlama yerleştirilmiş her parçanın kesilmesinden sonra kesimin kalan parçanın bütün uzunluğu boyunca bir ucundan diğer ucuna kadar yapılmasına izin veren bir yerleşimi gerektirir. Bu tip problemlere çoğunlukla cam ve kağıt endüstrisinde yapılan çalışmalar örnek gösterilebilir. Şekil 2.1 (a)'da giyotinli yerleşim planına örnek bir yerleşim verilmiştir.

Giyotinsiz (non-guillotineable) yerleşim için giyotinsiz problemler kısıtlamalarla sınırlandırılmazlar. Herhangi bir parça, yerleşim planında çakışmaya imkan vermeyen

mümkün olan her konuma yerleştirilebilir. Şekil 2.1 (b)'de giyotinli yerleşim planına örnek bir yerleşim verilmiştir.




**Şekil 2.1 : Giyotinli ve giyotinsiz kesme için yerleşim örneği (Söke, 2003)**

Yapılan çalışmada küçük parçaların döndürülebilir, yine küçük parçalarda ve yerleşimin veya kesim yapılacak büyk parçada kusur (fire) olmadığı, kesilecek büyük alanın ve yerleşicek küçük parçaların düzgün dikdörtgen olduğu ve ağırlık değerlerinin olmadığı kabul edilerek giyotinsiz kesme uygulanarak algoritmalar geliştirilmiştir. Kısıtların uygulanış amacı endüstiride en genel biçimde karşılaşılan koşulların gerçeklenmesidir.

## 2.2 – PROBLEMİN FORMÜLASYONU

Günlük hayatta bir çok kullanım alanı olan iki boyutlu kesme problemi (2BKP) için keseceğimiz büyük dikdörtgenin uzunluğunu L, genişliğini ise W ile büyük dikdörtgenden keseceğimiz küçük dikdörtgenlerin uzunluğunu $l_i$, genişliklerini $w_i$ ve herbirinin adedini $k_i$ ve her bir paça için belirlenmiş König sayısını da $b_i$ ile gösterelim. (i = 1 , ... , n)

Bu verilere göre $l_i < L$ ve $w_i < W$ ve de her ufak dikdörtgenden $k_i \geq 1$ tane olacak şekilde problemin amacı büyük dikdörtgende en az boş yer kalacak şekilde küçük dikdörtgenlerin

yerleştirilmesidir. Küçük dikdörtgen parçanın büyük parça üzerine yerleşiminin yapılıp yapılmadığı koşulu formül 2.1 de verilen $x_{ipq}$ fonksiyonu ile 0 veya 1 şeklinde belirlenir.

$$x_{ipq} = \begin{cases} 1, & eğer\ i\ inci\ kutu\ (p,q) koordinatına\ yerleştirilmiş\ ise \\ 0, & aksi\ halde \end{cases} \tag{2.1}$$

p ve q noktaları aşağıda verilen ayrık kümelere bağlı olacağını varsayalım.

$$P = \left\{ p \mid p = \sum_{i=1}^{m} \alpha_i l_i, p \leq L - \min\{l_i, i = 1, \ldots, m\}, \alpha_i \geq 0, \alpha_i \in Z \right\} \tag{2.2}$$

$$Q = \left\{ q \mid q = \sum_{i=1}^{m} \beta_i w, q \leq W - \min\{q_i, i = 1, \ldots, m\}, \beta_i \geq 0, \beta_i \in Z \right\} \tag{2.3}$$

Parçaların üst üste gelmesinin engellemesi için formül 2.4 deki şekilde $a_{ipqrs}$ fonksiyonu tanımlanır.

$$a_{ipqrs} = \begin{cases} 1, & eğer\ p \leq r \leq p + l_i\ ve\ q \leq s \leq q + w_i \\ 0, & aksi\ halde \end{cases} \tag{2.4}$$

$a_{ipqrs}$ fonksiyonu her bir i parçası için p ve q noktasından başlayarak r ve s noktaları için hesaplanmak zorundadır.

$$P = \{p \mid p \in P, p \leq L - l_i\} \tag{2.5}$$

$$Q = \{q \mid q \in Q, q \leq W - w_i\} \tag{2.6}$$

Aynı zamanda iki önemli kümenin tanımlaması 2.5 ve 2.6 da verildiği şekilde ortaya çıkmaktadır.

$$\sum_{i=1}^{m} \sum_{p \in P_i} \sum_{q \in Q_i} x_{ipq} a_{ipqrs} \leq 1, \forall r \in P, s \in Q \tag{2.7}$$

$$\sum_{p \in P_i} \sum_{q \in Q_i} x_{ipq} \leq b_i, i = 1, \ldots, m \tag{2.8}$$

$$x_{ipq} \in \{0,1\}, i = 1, \ldots, m, \forall p \in P, q \in Q \tag{2.9}$$

2.7, 2.8 ve 2.9 kısıtları altında

$$\min\left\{W.L \quad - \quad \sum_{i=1}^{n} (w_i.l_i).x_i\right\} \qquad (x_i \in Z_0) \quad , \quad (Z_0 = \{0\} \cup Z^+) \tag{2.10}$$

Formül 2.10 da verilen şekilde amaç fonksiyonu tanımlanır. (Nepomuceno et al. 2008)

Formül 2.10 de verildiği üzere, amaç fonksiyonumuz, k¡ adet bulunan ufak dikdörtgenlerin alanlarının toplamını büyük olan dikdörtgenin alanını en fazla dolduracak şekilde en iyi l¡, w¡ uzunluklarının seçilmesidir. Ufak dikdörtgenlerin alanlarının toplamı büyük dikdörtgeninkine eşit ya da fazla olmalıdır. Ancak yerleştirme (seçme) işleminde küçük dikdörtgenler büyük dikdörtgeninin alanına eşit ya da daha küçük olmalıdır.

# BÖLÜM 3: YERLEŞİM VE SEÇİM ALGORİTMALARI

## 3.1 – YERLEŞİM ALGORİTMALARI

Problemi 2 farklı algoritma yerleştirme yöntemiyle çözümünü inceleyeceğiz. Bunlardan ilki ilk sığan algoritması literatürde bilinen ismi ile aşağı sol algoritması ile çözümdür.

İkinci bölümde ise dinamik programlamadan yararlanarak oluşturulan sezgisel bir yöntem ile yerleştirme yapılacaktır.

### 3.1.1 - İlk Sığan Yöntemi (Aşağı Sol Dolgu Algoritması)

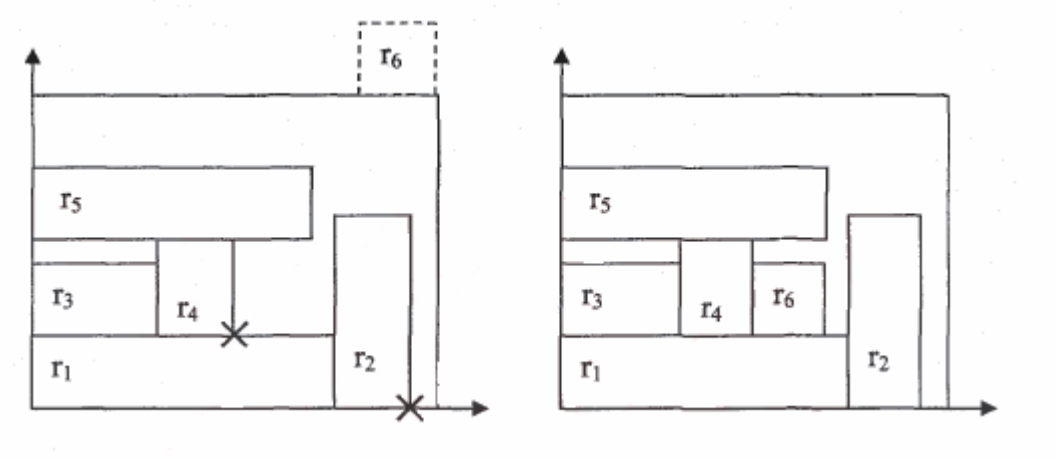
Bu yöntem 2BKP için en sık kullanılan greedy yaklaşım algoritmasıdır. Algoritmanın polinomial zamanda çözülmesi hız açısından olumludur. Her ufak dikdörgen için, bu parçayı yerleştirebildiği ilk boşluğa yerleştirmek için çalışır. Eğer hiçbir boşluk bulunamaz ise, bu parça dışarıda bırakılır. Algoritma işleyisi aşağıdaki şekilde verilmiştir:

**Adım 1:** Yüksekliği L ve genişliği W olan büyük dikdörtgen ve genişliği $w_i$ ve yüksekliği $l_i$ ve âdeti $k_i$ olan küçük dikdörtgenleri belirle.
**Adım 2:** Küçük dikdörtgenlerin alanlarını ( $w_i * l_i$ ) azalan şekilde sırala.
**Adım 3 :** i = 1
**Adım 4 :** Büyük dikdörtgendeki ilk boş yeri al ve boş alanın uzunluğu $\geq w_i$ ya da $l_i$ ve genişliği $\geq w_i$ ya da $l_i$ ise i inci dikdörtgenin oraya yerleşip yerleşmediğini kontrol et yerleşiyor ise alana yerleştir.
**Adım 5 :** Eğer boş alana yerleşicek durumda ve dikdörtgen boş alana yerleşmedi ise parçayı 90º çevir ve aynı boş alana yerleştir.
**Adım 6 :** Eğer i inci kutunun yerleşiceği boş alan yok ise i = i + 1 .
**Adım 7 :** Eğer i ≤ kutu sayısı ise adım 4 e git değil ise adım 8 e git.

**Adım 8 :** Bitir.

Literatürde aşağı sol dolgu algoritması olarakta bilinen bu yerleşim algoritması, yerleştirilecek olan küçük parçanın kısmi yerleşim planı içerisinde mümkün olan en alt seviyedeki boş alana yerleştirmesiyle gerçeleşir. Bu yerleşim planı diğer aşağı sol algoritmalarına göre daha yoğun bir yerleşim sağlamaktadır ancak en büyük dezavantajı $O(n^3)$ olan hesapsal yüküdür.(Chazelle, 1983)

Şekil 3.1'de aşağı sol dolgu algortiması için bir yerleşim planı dösterilmektedir.




**Şekil 3.1 : Aşağı Sol Dolgu Algortiması gösterimi (Söke, 2003)**

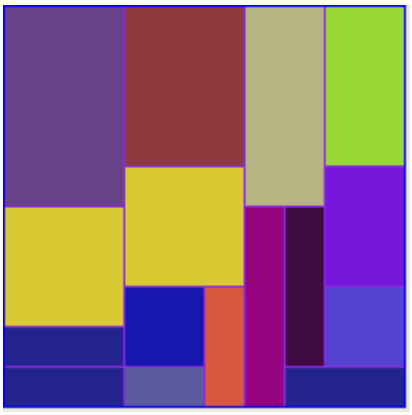
Ufak dikdörtgenlerin boyutları ve adetleri Tablo 3.1 de verilen veriler için ilk sığan algoritmasının uygulanışı aşağıda gösterilmiştir.

**Tablo 3.1 : İlk Sığan algoritması uygulaması için verilen küçük dikdörtgen değerleri**

| GENİŞLİK | YÜKSEKLİK | ADET |
|---|---|---|
| 50 | 30 | 1 |
| 40 | 30 | 1 |
| 50 | 20 | 1 |
| 30 | 30 | 2 |
| 40 | 20 | 1 |
| 30 | 20 | 1 |
| 50 | 10 | 1 |
| 20 | 20 | 2 |
| 40 | 10 | 1 |
| 30 | 10 | 4 |
| 20 | 10 | 1 |

Kesilecek alanın boyutları L=100 ve W=100 olarak kabul edilmiştir.

İlk sığan algoritması bu veriler için uygulandığı zaman Şekil 2.1.2 de verilen yerleşim planı sonucu elde edilir.



**Şekil 3.2 : İlk sığan algoritmasına göre yerleşim planı**

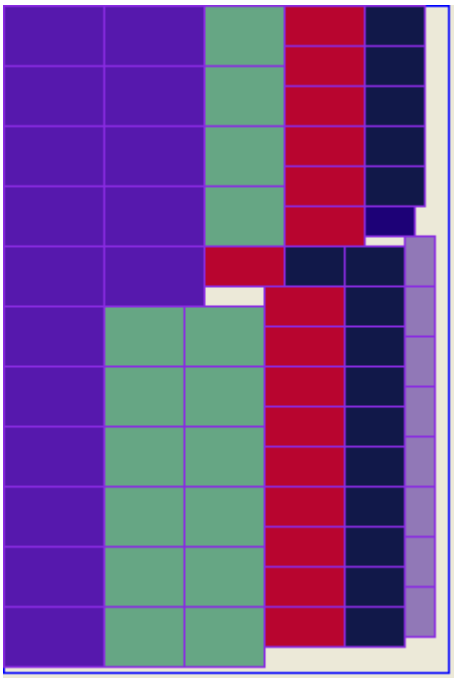
Şekil 3.2' ye bağlı olarak bütün dikdörtgenler alana yerleştirilmiştir.

Diğer bir veri kümesi Tablo 3.2'de verilen şekildedir.

**Tablo 3. 2 : İlk Sığan algoritması uygulaması için verilen küçük dikdörtgen değerleri**

| GENİŞLİK | YÜKSEKLİK | ADET |
|---|---|---|
| 50 | 30 | 16 |
| 40 | 30 | 16 |
| 40 | 20 | 16 |
| 30 | 20 | 16 |
| 25 | 15 | 16 |

Kesilecek alnımızın boyutları ise L=333 ve W=222 olsun.



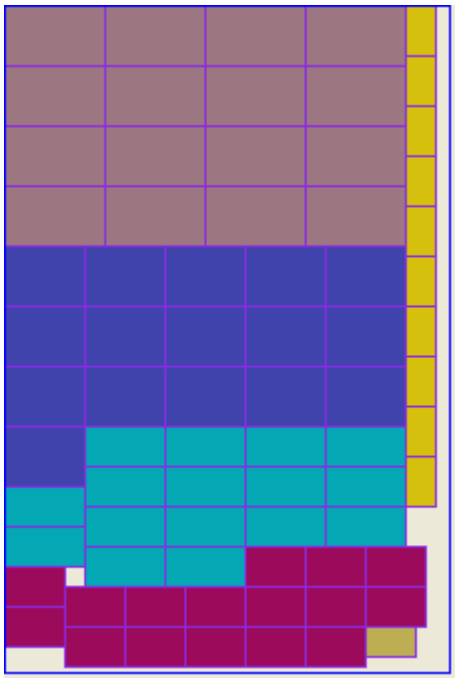
**Şekil 3. 3 : Tablo 3.2 verilerin ilk sığan algortmasına göre oluşan yerleşim planı**

w = 25 ve l = 15 olan dikdörtgenlerden 7 tanesi alana yerleştirilemedi.

İlk sığan algoritmasının optimal sonuca ulaşabileceği örnekler bulunmuştur. Ancak ikinci örnekte olduğu gibi kutuların tümü alana yerleştirilebilecek bir yerleşim planı olmasına rağmen ilk sığan algoritması bu yerleşimi gerçekleştirememiştir.

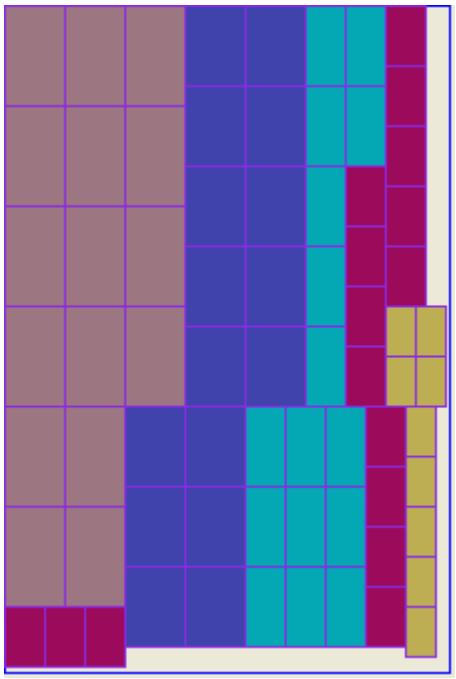
İlk sığan yönteminde dikat edilmesi gereken bir diğer nokta ise yerleştirme şekli önceliği ve yönüdür. Ufak dikdörtgenlerin büyük dikdörtgene yerleştirmeden önce yatay ya da dikey öncelikli olarak almamız ya da koordinat sistemine göre y düzlemi yönünde ya da x düzlemi yönünde yerleştirmemiz yerleşim değerlerini değiştirmektedir.

Bir önceki örneği bu kısıtlara göre dört farklı şekilde yerleşimi incelenmiştir.



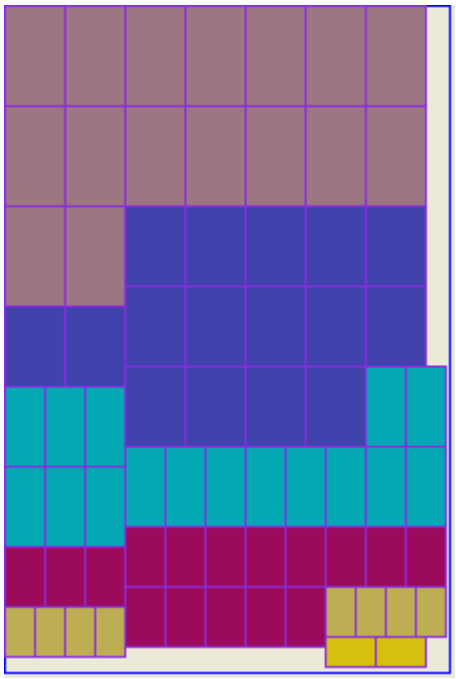
**Şekil 3. 4 : İlk sığan algoritması x koordinatı doğrultusunda, ufak diktörtgenler dik öncelikli yerleşim planı**

Şekil 3.4'de sağdan sola yani x doğrultusunda yerleştirilmiş ve kutular bu duruma göre dik olarak alınmıştır. Kullanılan alan 69725 birim² dir yani alanın %94 ünü doldurmuştur.



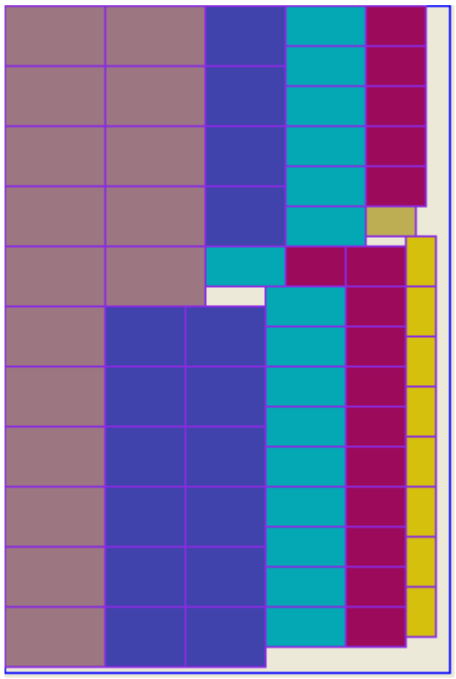
**Şekil 3. 5 : İlk sığan algoritması y koordinatı doğrultusunda, ufak diktörtgenler dik öncelikli yerleşim planı**

Şekil 3.5’ de y doğrultusunda yerleştirilme yapılmış ve parçalar dik öncelikli olarak kullanılmıştır. Kullanılan alan 68925 birim² dir, yani alanın %93 ünü doldurmuştur.



**Şekil 3. 6 : İlk sığan algoritması x koordinatı doğrultusunda, ufak diktörtgenler yatay öncelikli yerleşim planı**

Şekil 3.6'da ise x doğrultusunda yani sağdan sola ve kutular yatay öncelikle yerleştirilmiştir. Kullanılan alan 69350 birim² dir, yani alanın %94 ünü doldurmuştur.



**Şekil 3. 7 : İlk sığan algoritması y koordinatı doğrultusunda, ufak diktörtgenler yatay öncelikli yerleşim planı**

Şekil 3.7’ de ise y doğrultusunda yani yukarıdan aşağıya ve kutular yatay öncelikle yerleştirilmiştir. Kullanılan alan 68975 birim² dir, yani alanın %93 ünü doldurmuştur.

Tablo 3.2’de verilen veriler için uygulanan yerleşim planlarının sonucu olarak yerleştirme yönü ve dikdörtgenlerin önceliği elde edilen sonuca etkisi önemlidir.

Her deneme bize farklı sonuçlar verebilir. Ancak her bir küme için sürenin dört katına çıkması algoritmanın çalışma süresini olumsuz olarak etkiler. Bu nedenle yerleştirme yönü bu çalışmada sol ilk sığan şeklinde seçilmiştir diğer bir söylemle y doğrultusunda alınmış, kutu öncelikleri keyfidir.

Problemi tek boyuta indirgememiz problem için yeni sezgisel yaklaşımlar getirerek çözüme ulaşmayı kolaylaştıracaktır. Bir boyutlu kesme ve yerleştirme problemleri için kesin sonuç üreten birçok çalışma yapılmıştır. Lineer programlama ve dinamik programlama en

bilinenler arasında yer almaktadır. Örneğin elimizde büyüklükleri 10 , 9 , 8 , 8 , 6 , 6 , 5 , 5 , 3 olan paketleri kapasitesi 20 olan bidonlara doldurmaya çalışalım.

İlk sığan yöntemi ile kutuları bidonlara yerleştirirsek şu sonucu elde ederiz:

1.bidon . . . . . 10 , 9

2.bidon . . . . . 8 , 8 , 3

3.bidon . . . . . 6 , 6 , 5

4.bidon . . . . . 5

Bu dizilimden daha iyi bir sonuç elde etmemiz mümkündür:

1.bidon . . . . . 10 , 5 , 5

2.bidon . . . . . 9 , 8 , 3

3.bidon . . . . . 8 , 6 , 6

İlk sığan yönteminde elde edilen sonuçtan daha iyi bir çözüm bulundu.

1BKP için birçok çalışma yapılmış olmasına rağmen hala NP-zor sınıfındadır.Yani 2BKP zorluk durumu 1BKP içinde aynıdır. (Lines, 1990)

Bu çalışmada problemin bir boyutlu çözümü için Knapsack probleminden yararlanılmıştır.

### 3.1.2 - Knapsack Problemi ve Dinamik Programlama

0-1 Sırt çantası problemi (SÇP-*Knapsack problemi*), üzerinde en çok çalışılmış tamsayılı problemlerinden birisidir. 50'li yıllardan günümüze değin, çözülebilen problem boyutlarının artmasını ve çözüm süresinin azalmasını sağlayan pek çok gelişme kaydedilmiştir. Ancak halen zor SÇP için makul sürelerde çözüm verebilecek algoritmalara

gereksinim duyulmaktadır. SÇP kısıtları ve amaç fonksiyonu doğrusal olmasına rağmen, karar değişkeninin kesikli olması nedeniyle dışbükey değildir. (Gasımov ve diğerleri, 2004)

Problem bir sırt çantasının kapasitesini aşmadan en değerli eşyalarla doldurmayı amaçlar.

Formülasyonda kullanılan değişkenler; $a_j$ j inci parçanın ağırlığı, $c_j$ j inci parçanın değeri, $x_j$ j inci parçanın âdeti ve b de taşıma kapasitesi olarak verilmiştir.

$$\sum_{j=1}^{n} a_j . x_j \leq b \qquad (a_j, b \in Z^+ \; ; j = 1, \ldots, n) \qquad (3.1)$$

$$x_j \geq 0 \qquad (x_j \in Z^+) \qquad (3.2)$$

Kısıtları altında;

$$\max \sum_{j=1}^{n} c_j . x_j \qquad (c_j \geq 0 \; ; j = 1, \ldots, n) \qquad (3.3)$$

Amaç fonksiyonu olarak formül 3.3 elde edilir. Problemin çözümüne dinamik programlama kullanarak ulaşmaya çalışacağız. Bu sebeple yeni bir fonksiyon tanımlama zorunluluğu ortaya çıkar.

$$\sum_{j=1}^{k} a_j . x_j \leq y \qquad (0 \leq y \leq b) \qquad (3.4)$$

Kısıtı altında;

$$\Psi_k(y) = \max \sum_{j=1}^{k} c_j . x_j \qquad (0 \le k \le n) \tag{3.5}$$

Burada eğer hiçbir parça seçilmez ise fonksiyon 3.5 de belirtilen $\Psi_0(0) = 0$ değerini alır , eğer taşıma kapasitesi 0 olursa yine $\Psi_0(0) = 0$ değerini alır. $\Psi_1(y)$ için taşıma kapasitesi aşılmadan olabildiğince birnci parça $\Psi_1(y) = \lfloor y / a_1 \rfloor c_1$ ile doldururuz. $\Psi_k(y)$ fonksiyonu rekürsif halde genişletirsek;

$$\Psi_k(y) = \max \{ \Psi_{k-1}(y) , \Psi_k(y - a_k) + c_k \} \tag{3.6}$$

fonksiyon 3.6 yı elde ederiz.

Fonksiyon 3.6 yı kullanarak birinci tablo oluşturulur. Daha sonra bu tablodaki değerleri elde eden parçaları seçmek için ikinci bir tabloya ihtiyaç duyarız.

$$i(k, y) = \begin{cases} i(k-1, y) & eger \quad \psi_{k-1}(y) > \psi_k(y - a) + c_k \\ k & eger \quad \psi_{k-1}(y) \le \psi_k(y - a) + c_k \end{cases} \tag{3.7}$$

İkinci tabloyu oluşturmak için fonksiyon 3.7 de belirtilen i( k , y ) fonksiyonunu kullanırız. (Bellman, 1957)

Taşıma limiti 10, parçaların ağırlıkları $a_1 = 2$, $a_2 = 3$, $a_3 = 4$ ve $a_4 = 7$, parçaların değerleri ise $c_1 = 1$, $c_2 = 3$, $c_3 = 5$ ve $c_4 = 9$ olan bir test verisinin uygulanışı aşağıdaki şekilde olacaktır.

**Tablo 3. 3 : $\Psi_k$ ( y ) tablosu değerleri:**

| k \ y | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 |
| 2 | 0 | 1 | 3 | 3 | 4 | 6 | 6 | 7 | 9 | 9 |
| 3 | 0 | 1 | 3 | 5 | 5 | 6 | 8 | 10 | 10 | 11 |
| 4 | 0 | 1 | 3 | 5 | 5 | 6 | 9 | 10 | 10 | 12 |

**Tablo 3. 4 : i( k , y ) tablosu değerleri :**

| k \ y | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 4 | 0 | 1 | 2 | 3 | 3 | 3 | 4 | 3 | 4 | 4 |

Tablo 3.3'deki $\Psi_k$ ( y ) tablosunda ulaşılan en büyük değer $\Psi_4$ ( 10 ) = 12 dir. Tablo 3.4'deki i( k , y ) tablosunda buna karşılık gelen değer ise i( 4 , 10 ) = 4 dir.

$$i( 4 , 10 - a_4 ) = i( 4 , 3 ) = 2$$

bu değer hangi parçanın çantaya alınacağını belirtir.

$$i( 4 , 3 - a_3 ) = i( 4 , 0 ) = 0$$

elde edilir. Yani $a_4$ ağırlıklı parçadan 1 adet ve $a_3$ ağırlıklı parçadan 1 adet alarak çantanın içindeki ağırlık toplamı 10 ve değerlerinin toplamı 12 dir.

### 3.1.2.1 - Knapsack probleminin 1bkp uygulanması

Knapsack probleminde elimizdeki parçaların ağırlıkları, değerleri ve de taşıma kapasitesi verileri ile problemin çözüm kümesine ulaşılmaktadır. 1BKP ise elimizde sadece kesilecek parçanın uzunluğu ve ufak parçaların uzunlukları bulunmaktadır. Kesilecek büyük parçanın uzunluğu ile taşıma kapasitesini, ufak parçaların uzunlukları ile de ağırlıkları eşleştirebiliriz. Her parçanın uzunluğu aynı zamanda değeri anlamına gelir, yani parçalar uzunlukları kadar değerli olacaktır.
Büyük parçanın uzunluğunu L ile, küçük parçaların uzunluklarınıda l ile gösterilmiştir.

Bu durumda Knapsack probleminin formülasyonu;

$$\sum_{j=1}^{n} l_j \cdot x_j \leq L \qquad (l_j, L \in Z^+ \; ; j = 1, \ldots, n) \tag{3.8}$$

$$x_j \geq 0 \qquad (x_j \in Z^+) \tag{3.9}$$

kısıtları altında;

$$\max \sum_{j=1}^{n} l_j \cdot x_j \qquad (l_j \geq 0 \; ; j = 1, \ldots, n) \tag{3.10}$$

fonksiyon 3.10 da verilen şekilde amaç fonksiyonu tanımlanır.

$\Psi_k(y)$ fonksiyonu ise;

$$\sum_{j=1}^{k} l_j \cdot x_j \leq y \qquad (0 \leq y \leq L) \tag{3.11}$$

kısıtı altında;

$$\Psi_k ( y ) = \max \sum_{j=1}^{k} l_j . x_j \qquad ( 0 \le k \le n ) \tag{3.12}$$

fonksiyon 3.12 de verilen şekle dönüşür.

Aynı şekilde eğer hiçbir parça seçilmez ise $\Psi_0 ( 0 ) = 0$ değerini alır , eğer kesilecek parçanın uzunluğu 0 olursa yine $\Psi_0 ( 0 ) = 0$ değerini alır. $\Psi_1 ( y )$ için taşıma kapasitesi aşılmadan olabildiğince birinci parça $\Psi_1 ( y ) = \lfloor y / l_1 \rfloor l_1$ ile doldururuz. $\Psi_k ( y )$ fonksiyonu rekürsif halde genişletirsek

$$\Psi_k ( y ) = \max \{ \Psi_{k-1} ( y ) , \Psi_k ( y - l_k ) + l_k \} \tag{3.13}$$

Fonksiyon 3.13 ü elde ederiz.

Kesilecek olan parçanın uzunluğu L = 10, parçaların uzunlukları ise $l_1 = 2, l_2 = 3, l_3 = 4$ ve $l_4 = 7$ olan her bir uzunluğun sadece bir kez kullanılacak olması kısıtları altında yöntemin uygulanması aşağıdaki şekilde olacaktır. Her bir parçanın yalnızca bir kez kullanılması kısıtı $\Psi_k ( y )$ tablosunun değişmesine yol açacaktır. Çünkü eğer k indisli parçadan aynı satırda birden fazla sayıda kullanılamayacaktır.

$\Psi_k ( y ) = \max \{ \Psi_{k-1} ( y ) , \Psi_k ( y - l_k ) + l_k \}$ fonksiyonuna göre eğer bir k,y bölümü $\Psi_k ( y - l_k ) + l_k$ ile oluşturulmuş ise k ıncı parçadan kullanılmış anlamına gelir. Bu durumda aynı satırda birden fazla kez kullanamayız ve k, y bölümü $\Psi_{k-1} ( y )$ ile oluşturulur.

**Tablo 3. 5 : $\Psi_k$ ( y ) tablosu değerleri**

| k \ y | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 0 | 2 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 |
| 3 | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 9 | 9 |
| 4 | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 9 | 10 |

Tablo 3.5'deki $\Psi_k$ ( y ) tablosunda görüldüğü üzere aldığı en büyük değer $\Psi_4$ ( 10 ) = 10 dur. $\Psi_k$ ( y ) tablosundaki değişiklik, her bir parçanın değeri yerine uzunluklarını tabloya yazışımız olmuştur. Yani $\Psi_4$ ( 10 ) = 10 değeri bize verdiğimiz sınır değer olan L = 10 a ulaşıldığını göstermektedir.

$\Psi_4$( 10 ) - $l_4$ = $\Psi_4$( 10 ) – 7 = 3

3 değerini $\Psi_k$ ( y ) tablosunda arıyoruz ve bulduğumuzda $\Psi_k$ ( y ) = $\Psi_{k-1}$ ( y ) olduğu sürece k yı 1 azaltıyoruz. Bunun sebebi 3 değerine hangi parça ile ulaştığını bulmaktır.

$\Psi_4$( 3 ) = 3 dür. Fakat $\Psi_3$( 3 ) de 3 değerine sahiptir. 3 değerini ilk aldığı satır ikinci satırdır. O halde;

$\Psi_3$( 10 ) - $l_3$ = $\Psi_3$( 10 ) – 3 = 0

değerini elde ederiz.

Yani dördüncü parçadan bir adet ve ikinci parçadan 1 adet keserek optimal sonucu buluruz.

Bu problemden de görüldüğü üzere i( k , y ) tablosu kullanılmamıştır. Bunun sebebi artık tablomuz parçaların değerlerini değil uzunluklarını kullanmış olmamız bize $\Psi_k$( y ) tablosunun son adımında sınır değere ulaşılıp ulaşılmadığı görülebilmektedir. Ancak i( k , y ) tablosunu oluşturmak bize bellek alanı açmaktadır. $\Psi_k$( y ) tablosunun hepsini hafızada tutmak yerine i( k , y ) tablosunun sadece son satırını hafızada tutmak yeterli olucaktır fakat işlem sayısını arttırmaktadır. Buda bize işlem zamanı ve bellek alanı arasında tercih yapma zorunluluğunu sunmaktadır.

### 3.1.2.2 – 1Bkp'den 2bkp'e geçiş

1BKP de aldığımız değerler sadece uzunlukla sınırlı idi. Ancak 2BKP de parçalarımızın yükseklik ve genişlik değerleri vardır. Bu noktadan yola çıkarak 1BKP için yazdığımız formülasyonu 2BKP için gösterimi aşağıdaki şekilde olacaktır.

$$\sum_{j=1}^{n} (l_j \,.\, x_j \text{ ya da } w_j \,.\, x_j) \leq L \qquad (l_j, L \in Z^+ \; ; j = 1, \ldots, n) \qquad (3.14)$$

$$x_j \geq 0 \qquad (x_j \in Z^+) \qquad (3.15)$$

kısıtları altında

$$\max \sum_{j=1}^{n} (l_j \,.\, x_j \text{ ya da } w_j \,.\, x_j) \qquad (l_j \geq 0 \; ; j = 1, \ldots, n) \qquad (3.16)$$

fonksiyon 3.16 daki halini alır.

Ancak $\Psi_k(\ y\ )$ fonksiyonu knapsack problemi çözümü için sadece tek değişken hesaplamaktadır. Bu sorunu ise $\Psi_k(\ y\ )$ tablosunu iki parçaya ayırmakla çözebiliriz. İlk k satırını yükseklikliklerin katkıları ile oluşumunu sağlayarak ve k dan k * 2 satırı da genişliklerin eklenmesi ile oluşturabiliriz. Böylece uzunluk ve genişlikleri tek tabloda yazabiliriz. Fonksiyon 3.17 verildiği şekilde $g_i$ fonksiyonu hangi parçanın uzunluğu ya da genişliğinin kullanılıcağını, bu şekilde aynı parçayı iki kez kullanmayı ortadan kaldırır. Problemin formülasyonunda verilen kısıtlar bu model içinde geçerli olacaktır. formül 2.1 ve 2.4 de verilen $x_{ipq}$ ve $a_{ipqrs}$ fonksiyonlarının tanımlarına bağlı olarak $\Psi_k(\ y\ )$ fonksiyonu

$$g_i = \begin{cases} 1, & \textit{eğer i inci kutu l uzunluğu kullanılmış ise} \\ 0, & \textit{aksi halde} \end{cases} \tag{3.17}$$

$$\sum_{j=1}^{n} (l_j \ . x_j) g_i \leq y \qquad (0 \leq y \leq L) \tag{3.18}$$

$$\sum_{j=1}^{n} (w_j \ . x_j)(1-g_i) \leq y \qquad (0 \leq y \leq L) \tag{3.19}$$

$$\sum_{i=1}^{m} \sum_{p \in P_i} \sum_{q \in Q_i} x_{ipq} a_{ipqrs} \leq 1, \forall r \in P, s \in Q \tag{3.20}$$

$$\sum_{p \in P_i} \sum_{q \in Q_i} x_{ipq} \leq b_i, i = 1, \dots, m \tag{3.21}$$

$$x_{ipq} \in \{0,1\}, i = 1, \dots, m, \forall p \in P, q \in Q \tag{3.22}$$

kısıtları altında

$$\Psi_k(y) = \max \sum_{j=1}^{k} l_j . x_j \qquad (0 \le k \le n) \qquad (3.23)$$

ve

$$\Psi_{m+k}(y) = \max \sum_{j=1}^{k} w_j . x_j \qquad (0 \le k \le n, m = k) \qquad (3.24)$$

veya

$$\Psi_k(y) = \max \sum_{j=1}^{k} w_j . x_j \qquad (0 \le k \le n) \qquad (3.25)$$

ve

$$\Psi_{m+k}(y) = \max \sum_{j=1}^{k} l_j . x_j \qquad (0 \le k \le n, m = k) \qquad (3.26)$$

şekline dönüşür.

Aynı şekilde eğer hiçbir parça seçilmez ise $\Psi_0(0) = 0$ değerini alır, eğer kesilecek parçanın uzunluğu 0 olursa yine $\Psi_0(0) = 0$ değerini alır. $\Psi_1(y)$ için taşıma kapasitesi aşılmadan olabildiğince birinci parça $\Psi_1(y) = \lfloor y / l_1 \rfloor l_1$ (eğer genişlikleri ilk olarak tabloya yerleştiricek isek $\Psi_1(y) = \lfloor y / w_1 \rfloor w_1$ uygulanır) ile doldururuz.

Formül 3.21 ve 3.22 de verilen $\Psi_k$ ( y ) fonksiyonunu rekürsif halde genişletirsek

$$\Psi_k ( y ) = \max \{ \Psi_{k-1} ( y ) , \Psi_k ( y - l_k ) + l_k \} \quad (3.27)$$

ve

$$\Psi_{m+k} ( y ) = \max \{ \Psi_{m+k-1} ( y ) , \Psi_{m+k} ( y - l_{m+k} ) + l_{m+k} \} \quad (3.28)$$

Fonksiyon 3.27 ve 3.28 i elde ederiz.

Yani $\Psi_k$ ( y ) tablosu k satırdan m + k satıra yükselmiştir. Bu problemin doğası itibariyle yükseklikler ve genişlikler birbirlerinden bağımsızdırlar. İlk k satırı yükseklikler için yazıyorsak k + 1 den m + k satıra kadar da genişlikler yazılır.

Burada dikkat edilmesi gereken nokta yükseklik ilk k satıra yazılıyor ise genişlikler hesaplanmaya başlarken k + 1 inci satırdan itibaren yazılmaya başlamasıdır. Yani;
$\Psi_1 ( y ) = \lfloor y / l_1 \rfloor l_1$ fonksiyonu ilk genişlik satırı için uygulanamaz. Bunun sebebi o satıra kadar olan tüm yüksekliklerin katkısı ile son değere ulaşılmıştır, bu adımdan sonra
$\Psi_1 ( y ) = \lfloor y / l_1 \rfloor l_1$ uygulanır ise yüksekliklerin katkısı yok sayılmış olur, genişlikleın katkıları yüksekliklerin katkılarına eklenerek gitmelidir.

Ancak önümüzde hala bir sorun vardır. Buda eğer k ıncı kutunun genişliğini kullanıyor ise genişliğini de kullanabileceğidir. Diğer bir değişle elimizde yalnız bir adet bulunan bir parçanın hem genişliğini hem de uzunluğunu kullanarak alanı doldurabilir.
Bu sorunun önüne geçmek için algoritmada değişiklik yapmamız gerekmektedir.

Eğer $\Psi_k$ ( y ) tablosunun ilk k satırına yükseklikleri yazıyorsak;

Eğer $\Psi_{m+k-1}(y) < \Psi_{m+k}(y - l_{m+k}) + l_{m+k}$ ise bu m + k ıncı satırın y inci sütununda k ıncı parçanın genişliğini kullandığımız anlamına geliyordu. $\Psi_{m+k}(y - l_{m+k})$ i oluşturmak için k nın yüksekliğinin kullanılıp kullanılmadığını kontrol etmemiz gerekmektedir. Eğer;

$\Psi_{m+k}(y)$ da k nın yüksekliği kullanılmamış ya da k ıncı parçanın adedi aşılmamış ise genişlik kullanılabilir aksi halde k ıncı parçanın adedine bakılır ve hepsi kullanılmış ise kullanılamaz ve satırda elde edilmiş en büyük değer geçerlidir.

Eğer $\Psi_k(y)$ tablosunun ilk k satırına genişlikleri yazıyorsak;

Eğer $\Psi_{m+k-1}(y) < \Psi_{m+k}(y - l_{m+k}) + l_{m+k}$ ise bu m + k ıncı satırın y inci sütununda k ıncı parçanın yüksekliğini kullandığımız anlamına geliyordu, $\Psi_{m+k}(y - l_{m+k})$ i oluşturmak için k nın genişliğinin kullanılıp kullanılmadığını kontrol etmemiz gerekmektedir.

Eğer $\Psi_{m+k}(y - l_{m+k})$ da k nın genişliği kullanılmamış ya da k ıncı parçanın adedi aşılmamış ise yükseklik kullanılabilir aksi halde k ıncı parçanın adedine bakılır ve hepsi kullanılmış ise kullanılamaz ve satırda elde edilmiş en büyük değer geçerlidir.

$\Psi_k(y)$ tablosu oluşturulduktan sonra seçme işlemi 1BKP tanımlandığı şekilde uygulanır.

$\Psi_k(y)$ tablosunun hesaplanışı tablo 3.6'da verilen değerler için gösterilmiştir.

**Tablo 3. 6 : 2BKP ve dinamik programlama algoritması uygulaması için verilen küçük dikdörtgen değerleri**

| GENİŞLİK | YÜKSEKLİK | ADET |
|---|---|---|
| 5 | 3 | 2 |
| 4 | 3 | 2 |
| 4 | 2 | 1 |
| 3 | 2 | 2 |
| 2 | 1 | 1 |

Kesilecek büyük parçanın uzunluğu L = 25 dir.

Bu verilere göre $\Psi_k$ ( y ) tablosunu tablo 3.7'de ki şekilde oluşacaktır.

**Tablo 3. 7 : Tablo 3.6'deki verilere bağlı oluşturulmuş $\Psi_k$ ( y ) tablosu değerleri**

| | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 |
|---|---|
| G1 | 0 0 0 0 **5** 5 5 5 5 **10** 10 10 10 10 10 10 10 10 10 10 10 10 10 10 |
| G2 | 0 0 0 4 5 5 5 8 9 **10** 10 10 13 **14** 14 14 14 **18** 18 18 18 18 18 18 |
| G3 | 0 0 0 4 5 5 5 8 9 10 10 12 13 14 14 12 17 **18** 18 18 17 **22** 22 22 22 |
| G4 | 0 0 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 **22** 23 24 **25** |
| G5 | 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 **25** |
| Y1 | 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 **25** |
| Y2 | 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 **25** |
| Y3 | 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 **25** |
| Y4 | 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 **25** |
| Y5 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 **25** |

Seçim işlemi tablo 3.5 de verilen $\Psi_k$ ( y ) tablosuna bağlı olarak aşağıda belirtilen şekilde olacaktır:

3 birim genislik 4 nolu parçadan

4 birim genislik 3 nolu parçadan

4 birim genislik 2 nolu parçadan

4 birim genislik 2 nolu parçadan

5 birim genislik 1 nolu parçadan

5 birim genislik 1 nolu parçadan kullanılarak L = 25 tam olarak doldurulmuştur.

### 3.1.2.3 – Dinamik programlama yönteminin test verisi ile incelenmesi

Bu bölümde dinamik programlama kullanılarak, tablo 3.6 da verilen bir örnek stok problemi için uygulanışı her bir adım için incelenmiştir.

Tablo 3.8'de verilen küçük parçaların büyük dikdörtgene (L = 240 ve W = 480)
yerleşicek parçalar olarak atanarak, bu parçaların yerleşim planının oluşumunu verilen adımlarda gösterilmiştir.

**Tablo 3. 8 : 2BKP ve dinamik programlama algoritması uygulaması için verilen küçük dikdörtgen değerleri**

| GENİŞLİK | YÜKSEKLİK | ADET |
|---|---|---|
| 90 | 90 | 4 |
| 100 | 80 | 4 |
| 90 | 75 | 4 |
| 80 | 65 | 6 |
| 75 | 50 | 6 |
| 60 | 50 | 6 |
| 55 | 50 | 10 |
| 50 | 30 | 10 |
| 40 | 30 | 10 |
| 40 | 20 | 20 |
| 30 | 20 | 20 |
| 25 | 15 | 20 |

**ADIM 1 :**

Bu adımda L = 240 uzunluğunu en iyi şekilde dolduracak olan parçalar $\Psi_k(y)$ fonksiyonu yardımı ile seçilir. Daha sonra seçim sırasına göre ( 1 , 1 ) noktasından itibaren büyük alana yerleştirilir.

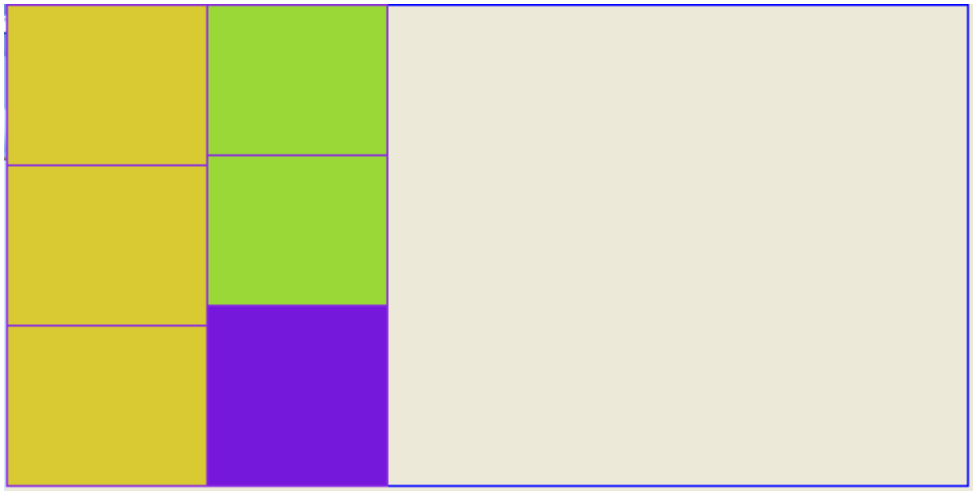
**Şekil 3. 8 : Tablo 3.8 verilerine göre birinci adımda oluşan yerleşim planı**

Şekil 3.8'de görüldüğü üzere l = 80 ve w = 100 olan kutulardan 3 tane kullanarak L = 240 ı tam olarak doldurmuştur. Diğer adıma geçmeden önce boyutları 100 ve 80 olan dikdörtgenlerin üç tanesi listeden çıkartılır.

**ADIM 2 :**

İkinci adımda ise yeni sınır değer belirlenmek zorundadır. Bunuda adım 1 de elde ettiğimiz sonuçtan yararlanarak bulacağız. (1, 1) noktasından itibaren L yönünde ilk boş alanı bulmaya çalışacağız, eğer W = 1 iken L yönünde hiç boş yer yok ise W değerini bir arttırırız. Bu arama sonunda ilk boş yerin (1, 101) noktası olduğu bulunmaktadır. Bu noktada doldurulacak olan alanın uzunluğunu, bir başka deyişle sınır değerini bulmamız gerekmektedir. Yine aynı yöntemle L yönünde boş alanın uzunluğu bulunmaktadır. Çıkacak sonuç 240 yani L nin değerine eşittir. $\Psi_k(y)$ fonksiyonuna verileri göndermeden

önce başlangıç noktası ile parçaların genişliği ya da uzunluğu alanın genişliğinden büyük olup olmadıklarının kontrolü zorunludur. Eğer başlangıç noktası ile parçanın genişliğinin toplamı büyük parçanın genişliğinden büyük ise o zaman genişliğin $\Psi_k(y)$ fonksiyonuna katkısı 0 olarak alınır, parçanın yüksekliği alanın genişliğinden büyük ise parçanın yüksekliğinin $\Psi_k(y)$ fonksiyonuna katkısı 0 alınır. Daha sonra kalan parçaların herbirini $\Psi_k(y)$ fonksiyonuna göndeririz



**Şekil 3. 9 : Tablo 3.8 verilerine göre ikinci adımda oluşan yerleşim planı**

ve şekil 3.9'da ki gibi boyutları l = 75 w = 90 olan parçalardan iki adet ve l = 90 w = 90 olan parçadan bir adet kullanarak sınır değerimizi yani L değerini doldurdurarak sonucu elde ederiz.

**ADIM 3 :**

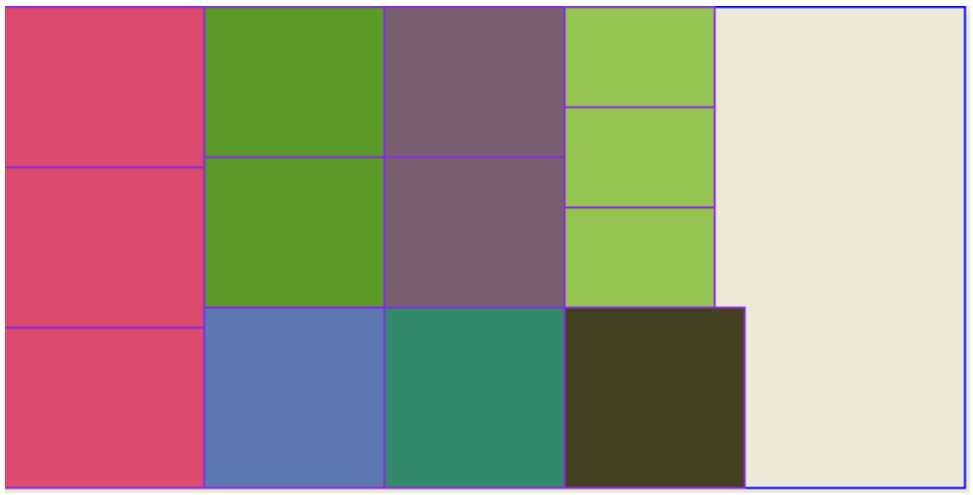
Adım 2'de uyguladığımız işlemlere benzer olarak başlangıç noktası ve sınır değer bulunur, kullanılan parçalar listeden çıkartılarak $\Psi_k(y)$ fonksiyonuna uygulanır.

**Şekil 3. 10 : Tablo 3.8 verilerine göre üçüncü adımda oluşan yerleşim planı**

3. Adımda şekil 3.10 daki sonuç elde edilir. Bu adımda Adım 2'de kullanılan parçaların boyutlarının aynısı olan parçalar kullanılarak sınır değere ulaşılmıştır.

**ADIM 4 :**

Bu adımda da daha önceki işlemler devam eder, sınır değer yine 240 bulunur ve $\Psi_k(y)$ fonksiyonuna değerler yerleştirilip seçim aşamasına geçilir.

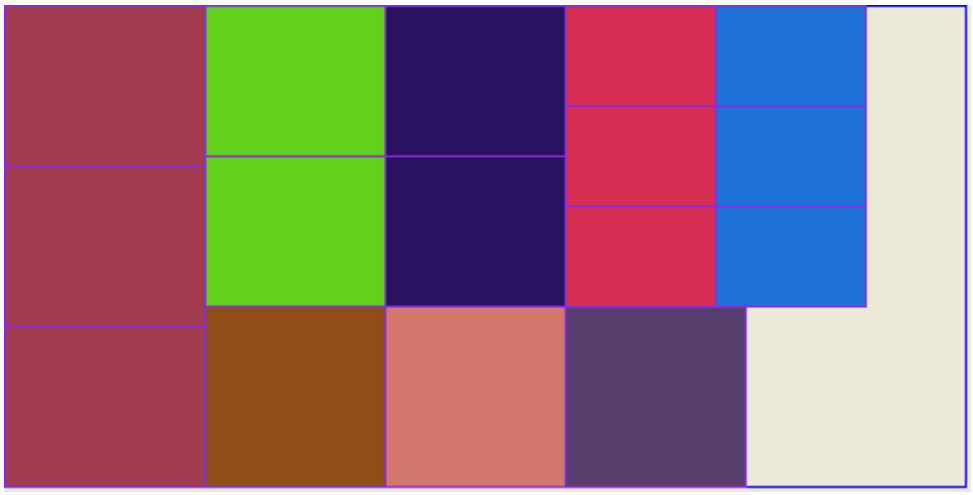


**Şekil 3. 11 : Tablo 3.8 verilerine göre dördüncü adımda oluşan yerleşim planı**

Şekil 3.11'de ki sonucu elde etmek için kullanılan parçalar daha önceki adımlardan farklı olarak bir tanesinin genişliği diğerlerinin genişliğinden farklıdır. Bu farklılık bundan sonraki adımda bizifarklı bir sonuçla karşı karşıya bırakacaktır.

**ADIM 5 :**

Burada ilk L yönünde arama yaparken karşımıza çıkıcak ilk boş alan (1, 281) noktası olacaktır. Bu noktadan itibaren boş yükseklik ise 150 değerine sahiptir. Diğer bir değişle yeni sınır değerimiz 150 dir ve $\Psi_k(y)$ fonksiyonu 150 değerini dolduracak en iyi parçaları seçecektir.

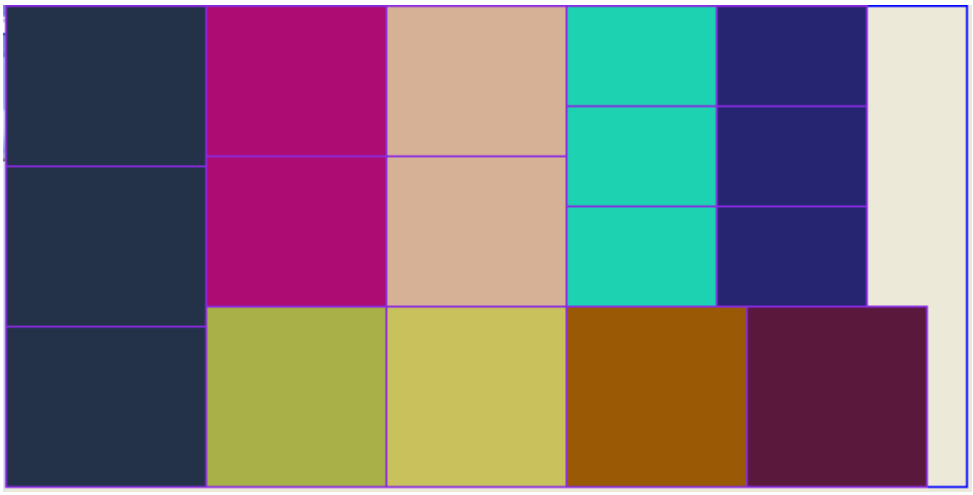


**Şekil 3. 12 : Tablo 3.8 verilerine göre beşinci adımda oluşan yerleşim planı**

Şekil 3.12'de görüldüğü üzere $\Psi_k(y)$ fonksiyonu 150 değerini tam olarak dolduracak parçaları seçmiştir.

**ADIM 6 :**

Bir önceki adımdaki durum bu adımda da karşımıza çıkmaktadır. L yönünde arama yaparken karşımıza çıkıcak ilk boş alan (151, 371) noktasıdır. Sınır değerimiz ise 90 birimdir.

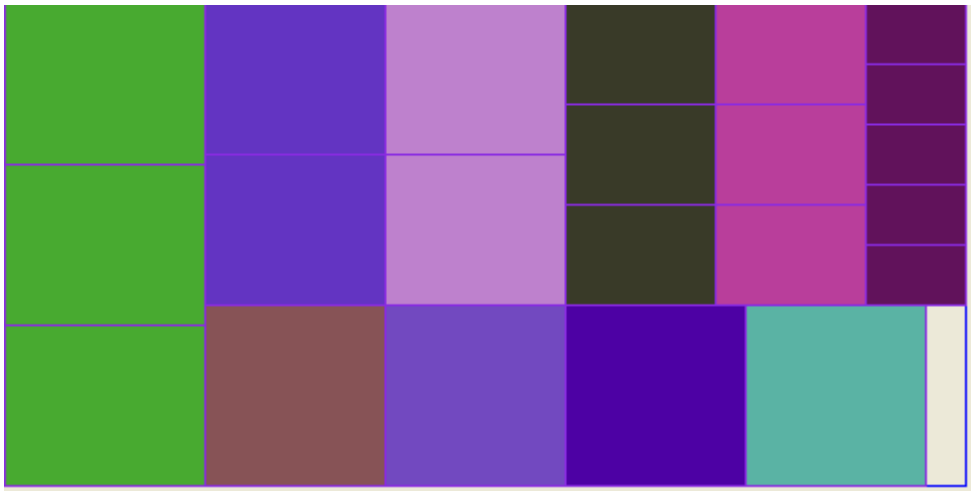


**Şekil 3. 13 : Tablo 3.8 verilerine göre altıncı adımda oluşan yerleşim planı**

$\Psi_k$ ( y ) fonksiyonu uygulandıktan sonra alanı tam dolduran bir tek parça bulundu ve alana yerleştirildi.

**ADIM 7 :**

Adım 4'de kine benzer bir durumla karşılaştık. Gerekli işlemlerden sonra aşağıdaki şekilde sonuca ulaşırız.



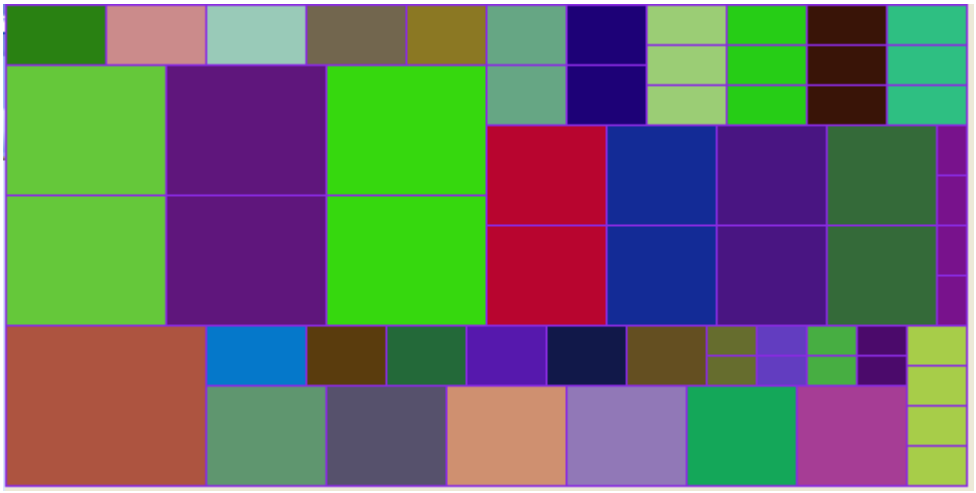
**Şekil 3. 14 : Tablo 3.8 verilerine göre yedinci adımda oluşan yerleşim planı**

**ADIM 8 :**

Bu adımda ise diğer adımlarda yapılan işlemler yapılır, başlangıç noktası ve sınır bulunur. Uygun parçalar $\Psi_k$ ( y ) fonksiyonuna gönderilerek uygun parçalar seçilir.

**Şekil 3. 15 : Tablo 3.8 verilerine göre sekizinci adımda oluşan yerleşim planı**

Sonuçta bütün alan %100 doldurulmuştur. Kalan parçalarada aynı işlemi uygularsak şekil 3.15 elde edilir.



**Şekil 3. 16 : Tablo 3.8 verilerine göre kalan parçaların ikinci büyük parça üzerindeki yerleşim planı**

Şekil 3.16'da görüldüğü üzere kalan parçalarada aynı işlemler uygulandığı zaman ikinci alanda %100 dolmuştur. Yani kalan parçalar için üçüncü bir alan gerekmektedir.

**Şekil 3. 17 : Tablo 3.8 verilerine göre kalan parçaların üçüncü büyük parça üzerindeki yerleşim planı**

Şekil 3.17 de ise elimizde kalan parçaların üçüncü bir alana yerleşimi ile oluşmuş yerleşim planı verilmiştir. Kalan parçalar için birden fazla büyük alanın kullanılması kullanıcıya bırakılmıştır.

**SONUÇ :**

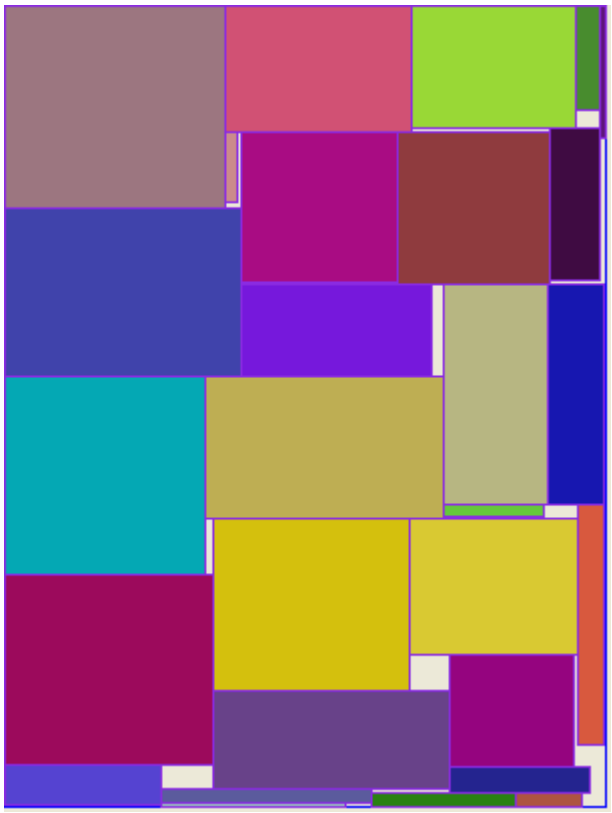
Dinamik programlamanın 2BKP e uygulanması sonucunda ilk 2 alan firesiz olarak yerleşim planı hazırladığı görülmüştür. Bu bize olası en iyi yerleşim planının bu küme için çözüldüğünü göstermektedir.

### 3.1.3 - İlk Sığan Algoritması Ve Dinamik Programlama Yöntemlerinin Karşılaştırması

İlk sığan algoritmasının(aşağı sol dolgu) diğer aşağı sol algoritmalarına göre daha yoğun bir yerleşim planı sağladığı bilinmektedir ancak dinamik programlama da bir uzunlukta en iyi şekilde ufak dikdörtgenleri yerleşimini sağlayarak yoğun bir yerleşim planı

oluşturmaktadır. Bu bölümde oluşturulan örnek veriler ile iki yerleşim planı karşılaştırılması yapılmıştır.

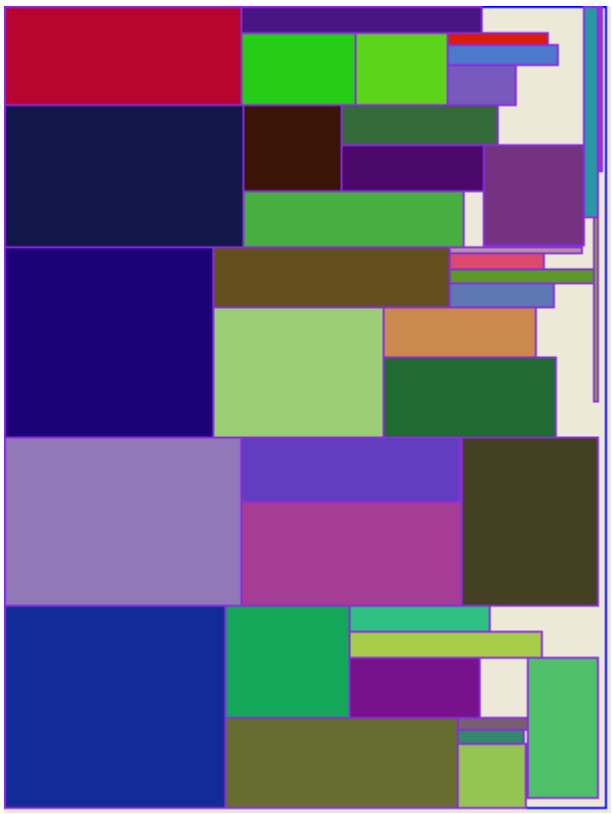
Ek-1'de verilen rastgele şekilde oluşturulmuş 50 dikdörtgenden oluşan veri kümesi uygulaması her iki algoritma için denenmiştir. Yerleşim(kesim) yapılacak büyük dikdörtgenin alanın boyutları L = 400 ve W = 300 olarak alınmıştır.



**Şekil 3. 18 : Ek-1 verisinin ilk sığan algoritması ile oluşturulan yerleşim planı**

İlk sığan yöntemi ile yapılan uygulamada Şekil 3.18'deki yerleşim planı ile karşılaşırız ve alanımızın yaklaşık %2 lik bir fire ile problem çözülmüştür.

Aynı verileri dinamik programlama ile yerleştirme planı oluşturduğumuzda ise aşağıdaki Şekil 3.19 ile karşılaşırız.



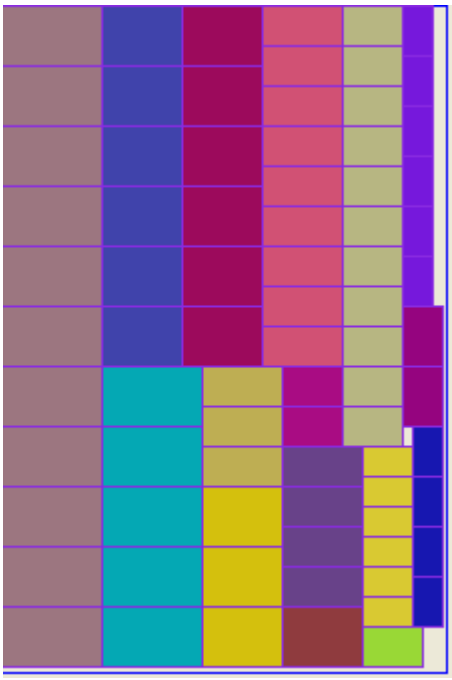
**Şekil 3. 19 : Ek-1 verisinin dinamik programlama ile oluşturulan yerleşim planı**

Dinamik programlama (Şekil 3.19) ile alanın ancak %93'ü dolmaktadır. Diğer bir söylem ile %7 lik bir fire ile yerleşim planı oluşturulmuştur. Bu problem için ilk sığan algoritması daha iyi bir sonuç vermiştir diyebiliriz. Bunun sebebi sınır değerini doldurmak için seçilebilecek kutuların genişliklerinin yüksekliklerine oranın çok büyük olmasıdır. Birbirinden farklı boyutlara sahip oldukları için bulunması gereken sınır değeri azalır ve bu değeri dolduracak uygun dikdörtgenler bulunamaz ve boş alanlar artar. Bu durum dinamik programlamanın sadece tek bir değeri sağlayacak olmasından kaynaklanmaktadır. Eğer aynı boyutlara sahip dikdörtgen sayısı artar ise dinamik programlama çok daha iyi sonuçlar verecektir.

Tablo 3.2'de verilen küçük parça değerleri üzerine yapılan diğer bir uygulamada kesilecek alanın boyutları ise L=333 ve W=222 olarak alınmıştır.

İlk sığan yöntemi uygulandığı zaman Şekil 3.3'de ki yerleşim planını elde ederiz. (Veriler yatay öncelikli ve yerleştirme yönü yukarıdan aşağıya olarak seçilmiştir.) Alanın sadece %93 kısmı doldurabilmiş fakat dışarıda da 7 adet dikdörtgen bırakmıştır.

Dinamik programlama ile bu problemin çözümü aşağıdaki Şekil 3.20'de ki gibi olacaktır.



**Şekil 3. 20 : Tablo 3.2 verilerin dinamik programlamaya göre oluşan yerleşim planı**

Görüldüğü üzere bütün parçalar alana yerleştirilmiştir, dışarıda hiçbir kutu kalmamış ve alanın %97 si doldurulmuştur. (Şekil 3.20) Dinamik programlama bu örnek için daha iyi bir yerleşim planı oluşturmuştur.

Sonuç olarak iki algoritmada polinomial algoritmalardır fakat dinamik programlama daha fazla bellek alanına ihtiyaç duyar. Buna rağmen ilk sığan algoritmasının süresel

karmaşıklığı $O(n^3)$ dür. İki algoritmanın birbirinin önüne geçtiği durumlar vardır. Ayrıca iki algoritmanın aynı sonuçları verdiği durumlarda mevcuttur. Bu nedenle hazırlanan algoritmada yerleşim algoritması olarak iki yönteme de yer verilmiştir.

## 3.2 – SEÇİM ALGORİTMASI

Bu noktaya 2 BKP için tek bir kümenin bir sıralaması için elde edilen sonuçlar ele alınmıştır. Bu noktada çeşitliliği arttırmak ve yerel bir optimuma takılmamak için genetik algoritma kullanılarak bir seçim algoritması bölümü oluşturulmuştur.

### 3.2.1 – Genetik Algoritma

Genetik algoritma, doğadaki evrim yöntemlerini kullanan bir arama yöntemidir. Genetik algoritma tekniği, Michigan Üniversitesinde yer alan John Holland tarafından 1970 li yıllarda ortaya çıkmış ve 1975 de Holland Doğal ve Yapay Sistemlerin Uygulanması adlı kitabında incelemere yer vererek yayınlamıştır. Mekanik öğrenme konusunda çalışan Holland, Darwin in evrim teoreminden etkilenerek canlılarda yaşanan genetik süreci bilgisayar ortamında gerçekleştirmeyi düşündü.

GA hem problem çözmek hem de modelleme için kullanılmaktadır. Günümüzde genetik algoritmaların uygulama alanları genişlemektedir. Bunlardan bazıları : Atölye Çizelgeleme, Yapay Sinir Ağları Tasarımı, Görüntü Kontrolü, Elektronik Devre Tasarımı, Optimizasyon, Uzman Sistemler, Paketleme Problemleri, Makine ve Robot Öğrenmesi, Gezgin Satıcı Problemi, Ekonomik Model Çıkarma v.b sayılabilir (Mitchell ve Forest,1994).

Genetik algoritmalar, klasik optimizasyon algoritmalarından dört temel noktada ayrılır (Oğuz ve Akbaş, 1997):

- GA parametrelerin kendileri ile değil, parametre takımının kodlanmış bir haliyle uğraşırlar.
- GA aramaya tek bir noktada değil, bir nokta ailesinden başlarlar. Dolayısıyla yerel bir optimuma takılmadan çalışabilirler.
- GA amaç fonksiyonunun (objective function) türevlerini ve bir takım ek bilgileri değil, doğrudan amaç fonksiyonunun kendisini kullanırlar.
- GA da deterministik değil rastlantısal geçiş kurallar kullanılır.

Genel bir genetik algoritma adımları aşağıdaki şekilde verilebir:

1. Başlangıç populasyonunu rastlantısal olarak üret.
2. Populasyon içindeki tüm kromozomların amaç fonksiyonu değerlerini hesapla.
3. Tekrar üreme, çaprazlama ve mutasyon operatörlerini uygula.
4. Oluşturulan her yeni kromozomun amaç fonksiyonu değerlerini bul.
5. Amaç fonksiyonu değerleri kötü olan kromozomlar populasyondan çıkar.

Genetik algoritmaların esnekliği sayesinde birçok problemin çözümü veya modellenmesi için, algoritma üzerinde çeşitli değişikliklere izin vermektedir. Bu sebeple 2BKP için bir çok genetik algoritma modeli geliştirlmiştir.

#### 3.2.1.1 – Genetik algoritma operatörleri

Kullanılan genetik operatörler, varolan populasyon üzerine uygulanan işlemlerdir. Bu işlemlerin amacı daha iyi özelliğe sahip yeni nesiller üretmek ve arama algoritmasının alanını genişletmektir (İşçi ve Korukoğlu, 2003). Farklı uygulamalarda farklı operatörler kullanılmakla birlikte genelde genetik algoritmada 3 standart operatör kullanılır. Bu operatörler:

- (Yeniden) Üretim ((Re)-production)
- Çaprazlama (Crossover)
- Mutasyon (Mutation)

Yapılan bu çalışmada bu üç operatör kullanılmıştır. Operatörlerin genel kullanım ve açıklamaları aşağıdaki şekilde yapılabilir.

**Yeniden Üretim:**

Bu aşamada küme elemanlarının ya da verilerin kromozoma dönüştürelerek seçildiği bölümdür. Bir kromozom temsil ettiği çözüm hakkında bir şekilde bilgi içermelidir. En çok kullanılan kodlama ikili karakter dizisidir. Bu yöntemle kromozom şu şekilde görülmektedir:
Kromozom 1 : 1101100100110110
Kromozom 2 : 1101111000011110

Her kromozom ikili karakter dizisi şeklinde temsil edilmektedir. Karakter dizisindeki her bir çözümün bir özelliğini temsil eder. Bir başka olasılık tüm karakter dizisinin bir sayıyı temsil etmesidir. Elbette, birçok başka kodlama yöntemi vardır. Kodlama daha çok çözülen probleme bağlıdır. Örneğin bazı problemler için tamsayı veya gerçek sayı şeklinde kodlamak gerekirken, bazı problemlerde permütasyon şeklinde kodlamaya ihtiyaç vardır.

**Çaprazlama:**

Üretim (ya da yeniden üretim) sonra, çaprazlama işlemiyle devam edelir. Çaprazlama, atalardaki seçili genler üzerinde işlem yapar ve yeni çocuklar oluşturur. Bunun en basit şekli, rasgele bir kesme noktası (çaprazlama noktası) seçip, bu noktadan önceki her şeyi ilk bireyden, sonraki her şeyi ikinci bireyden alıp birleştirerek çocuğu oluşturmaktır.

Çaprazlama aşağıdaki şekilde gösterilebilir: ( | kesme noktasıdır):

Kromozom 1: 11011 | 00100110110
Kromozom 2: 11011 | 11000011110
Çocuk 1 : 11011 | 11000011110
Çocuk 2 : 11011 | 00100110110

Çaprazlamanın birçok yolu mevcuttur, örneğin birden fazla kesme noktası seçilebilir. Çaprazlama daha da karmaşık olabilir ve tamamen kromozomların kodlanmasına bağlıdır. Özel problemler için yapılmış özel çaprazlamalar genetik algoritmanın başarımını arttırabilir.

**Mutasyon:**

Çaprazlama işlemi gerçekleştirildikten sonra, mutasyon işlemi yapılır. Mutasyonun amacı, toplumdaki tüm çözümlerin çözülen problemlerin bir yerel uygun değerine düşmesinin önüne geçmektir. Mutasyon işlemi çaprazlama sonucu oluşan çocuğu
rasgele değiştirmektedir. İkili kodlamada rasgele seçilmiş bir kaç biti 1'i 0'a, 0'ı 1'e
şeklinde değiştirmek bir mutasyondur.

Asıl Çocuk 1: 1101111000011110
Mutasyon Geçirmiş Çocuk 1: 1100111000011110
Asıl Çocuk 2: 1101100100110110
Mutasyon Geçirmiş Çocuk 2:1101101100110110

Mutasyon tekniği (çaprazlama tekniği de) kromozomların kodlamasına çoğunlukla
bağlıdır. Örneğin permütasyon şeklinde kodlamada mutasyon rasgele seçilen iki genin yer değiştirmesi olarak gerçekleştirilir.

### 3.2.2 – Seçim Algoritması Olarak Genetik Algoritma

Genetik algoritmalar Holland tarafından 70 li yılların başında geliştirilmiş olmasına rağmen 80 yılların ortalarına kadar paketleme problemlerine uygulanmamıştır (Söke, 2003). Smith, 1985 yılında ilk olarak genetik algoritmaları kullanarak paketleme problemlerine çözüm aramıştır. Bu çalışmada seçim algoritması olarak genetik algoritma seçilmesinin sebebi küçük dikdörtgenlerin bulunduğu uzayda yerel optimuma takılmadan, birden fazla alt uzayı rastlantısal olarak oluşturulup, optimum noktayı buralarda aramaktır.

Problem geniş bir çözüm uzayından alt kümelere ayrılarak, her biri için bulunan optimal sonuçların karşılaştırlması koşulu ele alınarak incelenmiştir. Genetik algoritma üzerinde yapılan değişiklikler ile iki boyutlu kesme problemine adapte edilmiştir. Aşağıda verildiği şekilde operatörler kullanılarak genetik algoritma 2BKP için seçim algoritması olarak kullanılmıştır.

**(Yeniden) Üretim:**

Programın ilk adımında uygulanan bu operatör, rastsal bir şekilde istenilen sayıda birey üretmektedir. Seçilen her bir küçük dikdörtgen sıra numarası ile indekslenir ve bize bireydeki kromozonları oluşturur. Tüm karakter dizisi bir sayıyı temsil etmektedir ki bu da ufak dikdörtgenlerin hangi sıra ile yerleşim bölümüne gönderileceğini belirler. Bu şekilde istenilen birey sayısı kadar üretim devam eder. İlk bireyin sıralaması büyükten küçüğe yapılarak deterministik bir sıralama yapılmıştır. Kalan bireyler ise tamamen rasgele oluşturulur.

**Çaprazlama:**

Bu bölümden önce yerleşim planına bakılarak en az fireyi (boş alan) veren bireylere göre sıralanır. Fire hesabı aşağıdaki şekilde hesaplanmaktadır:

Fire = (Büyük dikdörtgenin alanı – Kullanılan alan) / Büyük dikdörtgenin alanı

Bu şekilde üretim bölümünde verilen popülasyon sayısı sabit tutulur ve kötü kromozonlar (bireyler) işlemden atılır. İkililer tek noktalı çaprazlama tekniği ile işleme alınır ve 2 bireyden 2 çocuk üretilir. Örneğin 10 başlangıç popülasyonu için 10 çocuk üretilir ve daha sonra yerleşim sonuçlarına göre toplam 20 bireyden 10 tanesi bir sonraki çaprazlama işlemcisine alınırlar.

Bir sonraki adımda, literatürde sıraya dayalı çaprazlama olarak bilinen teknik uygulanır. Sıraya dayalı çaprazlamada kesim noktası belirlenir. Yapılan çalışmada kesim noktası toplam dikdörtgen sayısının yarısı olarak belirlenmiştir. Çaprazlamanın yapılacağı baba olarak isimlendirilen ilk genden kesim noktasına kadar olan genler yeni bireyin diğer bir deyişle çocuğun ilk genlerini oluşturur. Çocuğun diğer genleri annenin (ikinci bireyin) kesim noktasından başlayarak son gene kadar aynı sırayla genlerin alınmasıyla oluşur. Tekrar eden genler için ilk önce babanın kullanılmayan genlerine daha sonra annenin genlerine sırası ile bakılır. İkinci çocuk ise annenin kesim noktasına kadar olan gen e kadar sırası ile alınır. Kalan genler ise babanın kesim noktasından itibaren son gene kadar olan kısmı alınarak oluşturulur. Tekrar eden genler için burada ilk önce annenin kullanılmayan genlerine daha sonra babanın kullanılmayan genlerine sırası ile bakılır.

Çaprazlama operasyonunda bilindiği üzere, bu operatöre alınan iki bireyin sıra numarası değiştirilir. Ancak 2BKP probleminde ufak parçaların yükseklik ve genişlik değerleri bir önceki dizilimlerindeki hallerini korumak zorundadırlar. Bu sebeple bireylerin kendi dizilim planlarına göre olan sırası değiştirilirken, küçük parçalar üzerinde döndürme yapılmaz.

**Mutasyon:**

Mutasyon işlemcisine girebilme koşulu, aynı yerleşim sonucunu üreten (birbirini tekrar eden) bireylerin varlığıdır. Ufak dikdörtgenlerin sayısının dörtte bir sayısı kadar kromozomu rastsal şekilde yer değiştirirler. Bu sayede yerleşim planı için çeşitlilik oranı arttırılmış olur. Yine aynı şekilde çeşitlilik ve parçaların en verimli şekilde kullanılması için ufak dikdörtgenlerin sayısının dörtte bir sayısı kadar parça üzerinde döndürme işlemi uygulanır.

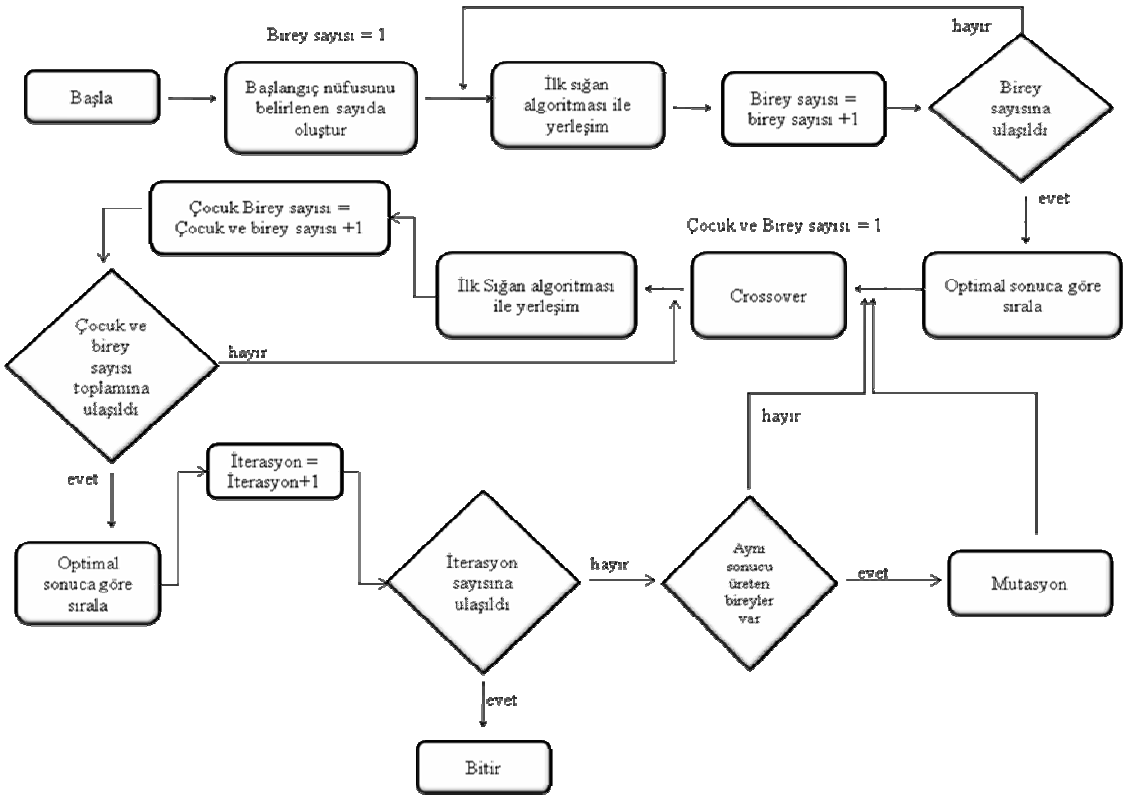
# BÖLÜM 4 : UYGULAMA

Hazırlanan program daha öncede belirtildiği gibi seçim ve yerleşim algoritmaları olarak iki ayrı bölümden oluşmaktadır.

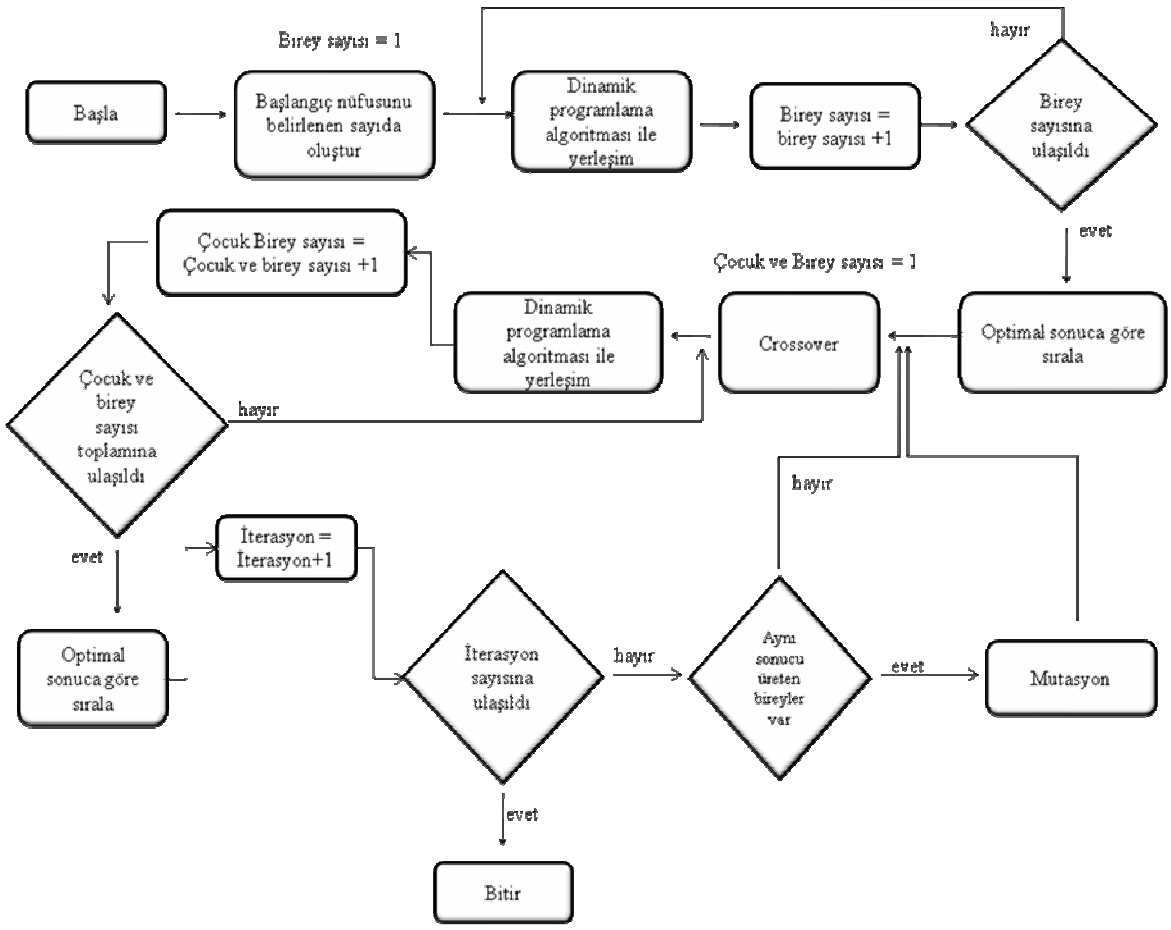
Seçim algoritması yerel bir optimuma takılmamak için genetik algoritma seçilmiştir.
Yerleşim algoritması ilk sığan (aşağı sol dolgu) ve dinamik algoritma yardımı ile hazırlanan diğer bir sezgisel yöntem ile oluşturulmuştur.

## 4.1 – ALGORİTMA

İki farklı yerleşim planı için hazırlanan algoritmaların akış diyagramları Şekil 4.1 ve Şekil 4.2'de verilmiştir.




**Şekil 4. 1 : İlk Sığan Algoritması ile Hazırlanan Uygulamanın Akış Diyagramı**




**Şekil 4. 2 : Dinamik Programlama ile Hazırlanan Uygulamanın Akış Diyagramı**

### 4.1.1 – Seçim Bölümü

Bu bölümde seçim kısmının çalışması aşağıdaki adımlar ile sıralanmıştır.

1. İlk adımda problem için tanımlanan küçük dikdörtgen parçalar alınır ve buradan başlangıç nüfusunu oluşturmak üzere rasgele aynı elemen sayısına sahip kümeler oluşturlur. Aynı eleman sayısına sahip ancak sıralamaları farklı olan küme elemanları küme numarasına bağlı olarak indekslenerek ilk yerleşim algoritmasına gönderilir.

2. Yerleşim algoritması sonuçlarına göre en az boş alan bırakan (en az fire veren) dizilim kısıtına göre sıralanır.

3. En iyi dizilimler bir ve iki, üç ve dört vb. şeklinde gruplandırılarak çaprazlama(crossover) işlemcisine gönderilir.

4. Her bir ikiliden ikişer adet çocuk üretilir ve tekrar yerleşim işlemcisine gönderilir.

5. Çocuk ve birey sayısının toplamına ulaşıldığı zaman elde edilen sonuçlar tekrar sıraya dizilir ve eğer aynı değerleri üreten yerleşimler var ise mutasyon işlemcisine gönderilir. Mutasyon işlemcisi, parça sayısının dörtte bir sayısı kadar parçanın rasgele olarak yerinin değişiminden oluşmaktadır. Ayrıca yine aynı sayıda ufak dikdörtgenlerin yükseklik ve genişlik değerleri yer değiştirilir. Yani döndürme işlemi uygulanır. Bu sayede bireyler arasındaki çeşitlilik de arttırılmış olmaktadır. Eğer tüm yerleşim sonuçları ayrı sonuçlar üretmiş ise tekrar crossover işlemcesine dönülür.

6. İterasyon sayısına ulaşılır ise algoritma sonlandırılır, aksi halde yerleşim devam eder.

### 4.1.2 – Yerleşim Bölümü

Yerleşim bölümünde ilk sığan ve dinamik algoritma karşılaştırmalı olarak uygulanmıştır. İlk sığan algoritmasının uygulanmasında yerleşim yönü ufak dikdörtgenlerin önceliği kısıtları uygulanmamıştır. Bunun sebebi, seçim algoritmasının gerekli çeşitliliği sağlamasıdır. Tüm parçaların tek bir kısıta bağlı olmadan, bağımsız olarak mutasyon işlemcisinde döndürülebilir olması bize bir parçanın birden fazla yerde ve birden fazla şekilde yerleşimini olası kılar.

Dinamik programlama yerleşiminde ise yerleşim kuralları aynı şekilde uygulanmıştır.

Bilindiği üzere $\Psi_k$ ( y ) tablosunun oluşumunda ve buna bağlı olarak parçaların seçiminde parçaların fonksiyona giriş sırası, yükseklik ve genişlik değerleri önemlidir.

$\Psi_k$ ( y ) tablosundan yerleşim için olan parçaların seçimi belli kısıtlar altında last-in-first-out mantığındadır. Bu sebeple GA bize $\Psi_k$ ( y ) tablosundaki çeşitliliği ve buna bağlı olarak yerleşim planındaki varyasyonları arttırmaktadır. Ayrıca mutasyon işlemindeki döndürme işlemcisi sayesinde genişlik veya yükseklik kısıtına da bağlı kalınmadan yerleşim yapılmaktadır. Bu sayede ilk mantıksal sıralama ile elde edilen yerel optimuma bağlı kalmadan, daha geniş bir uzayda optimum noktası aranmış olur.

# BÖLÜM 5: UYGULAMANIN SONUÇLARI VE PERFORMANS KARŞILAŞTIRILMASI

## 5.1 DİNAMİK PROGRAMLAMA İLE YERLEŞİM VE GENETİK ALGORİTMA İLE SEÇİM ALGORİTMASI SONUÇLARI

Yapılan bu çalışmada, Ek-1 den Ek-10 a kadar verilmiş olan, rasgele olarak oluşturulmuş herbiri 50 parçadan oluşan 10 test verisi hazırlanmıştır. Dinamik algoritmanın yerleşim algoritması olarak kullanıldığı bu programda başlangıç popülasyon sayısı 10 ve 20 birey olarak belirlenmiştir ve sonuçların karşılaştırılmaları yapılmıştır. İterasyon sayısı 100 olarak belirlenmiştir. Yerleştirilecek büyük alan L = 400 ve W = 300 birim boyutlarındadır. Yerleştirilecek olan paketlerin toplam alanı, yerleşimin yapılacağı büyük dikdörtgenin alanından daha büyük olduğu bilinmektedir.

### 5.1.1 Başlangıç Popülasyonunun Sonuca Etkileri

Tablo 5.1'de başlangıç popülasyonuna göre elde edilen en iyi sonuçlar birim kare cinsinden verilmiştir. Aynı şekilde tablo 5.2'de başlangıç popülasyonuna bağlı olarak elde edilen en iyi sonuçların fire değerleri verilmiştir. Test sonuçlarından anlaşılacağı üzere bazı problemler için başlangıç popülasyonu artışı daha az fire üretirken, bazı test verileri için daha yüksek fire değerleri elde edilmiştir. Bu sebeple başlangıç popülasyonun artışı fire değerlerini her zaman küçültür diyemeyiz.

**Tablo 5. 1 : Test verilerinin başlangıç popülasyonuna göre birim kare üzerinden yerleşim sonuçları**

| | | Başlangıç popülasyonu | |
|---|---|---|---|
| | | 10 | 20 |
| Test numarası | 1 | 115093 | 115215 |
| | 2 | 115175 | 113181 |
| | 3 | 112220 | 110796 |
| | 4 | 116023 | 115082 |
| | 5 | 116569 | 116336 |
| | 6 | 116992 | 117688 |
| | 7 | 112042 | 114023 |
| | 8 | 115544 | 114416 |
| | 9 | 111704 | 116140 |
| | 10 | 115988 | 115154 |

**Tablo 5. 2 : Test verilerinin başlangıç popülasyonuna göre fire değerleri**

| | | Başlangıç popülasyonu | |
|---|---|---|---|
| | | 10 | 20 |
| Test numarası | 1 | 0,040891667 | 0,039875 |
| | 2 | 0,040208333 | 0,056825 |
| | 3 | 0,064833333 | 0,0767 |
| | 4 | 0,033141667 | 0,04098333 |
| | 5 | 0,028591667 | 0,03053333 |
| | 6 | 0,025066667 | 0,01926667 |
| | 7 | 0,066316667 | 0,04980833 |
| | 8 | 0,037133333 | 0,04653333 |
| | 9 | 0,069133333 | 0,03216667 |
| | 10 | 0,033433333 | 0,04038333 |

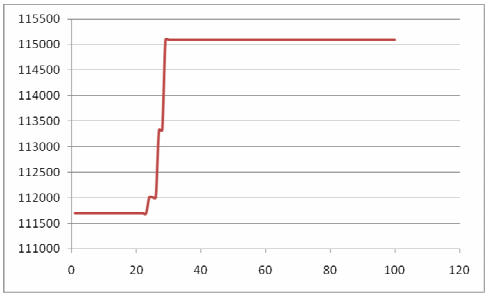
Test verisi olarak kullandığımız 10 problemden 1, 6, 7 ve 9 numaralı problemler için başlangıç popülasyonu artışı olumlu etki yaparken, diğer veriler için olumsuz etki yapmıştır. Ancak Tablo 5.3'de verilen en büyük fire farkı olan 0,036966667 değeri 9 numaralı problem için başlangıç popülasyonu artışı olumlu şekilde etkileyerek elde edilmiştir.

**Tablo 5. 3 : Başlangıç popülasyonuna bağlı sonuçların fire değerleri farkı**

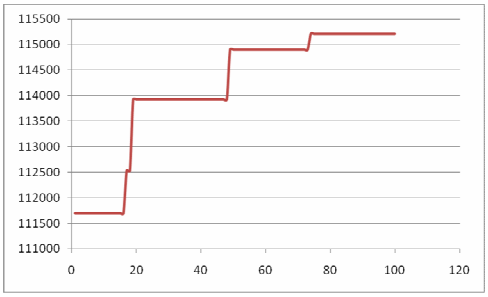
| | | Fire değerleri farkları |
|---|---|---|
| Test numarası | 1 | 0,001016667 |
| | 2 | -0,016616667 |
| | 3 | -0,011866667 |
| | 4 | -0,007841667 |
| | 5 | -0,001941667 |
| | 6 | 0,0058 |
| | 7 | 0,016508333 |
| | 8 | -0,0094 |
| | 9 | 0,036966667 |
| | 10 | -0,00695 |

### 5.1.2 İterasyon Sayısının Sonuca Etkileri

Şekil 5.1'den Şekil 5.10'a kadar verilen grafikler incelendiğinde problemin doğası gereği hangi iterasyonda en iyi sonuca ulaşacağı önceden belirlenemez. Örneğin 2 nolu problem için 20 başlangıç popülasyonu için en iyi sonuç 100. iterasyonda elde edilirken, 10 numaralı problem için 20 başlangıç popülasyonu için en iyi sonuç 6. iterasyonda elde edilmiştir.

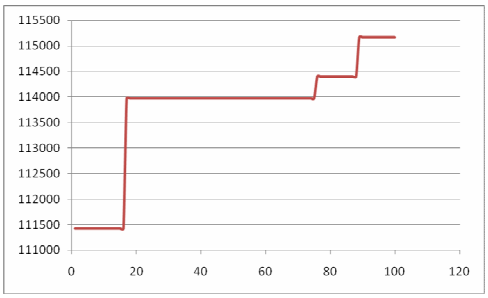



a

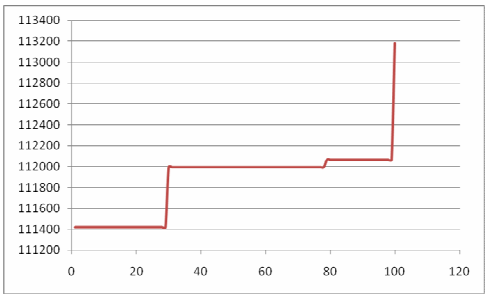



b

**Şekil 5. 1 : 1 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

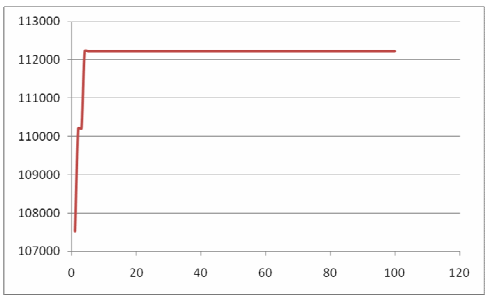



a

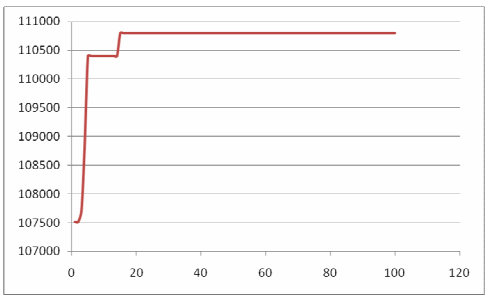



b

**Şekil 5. 2 : 2 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

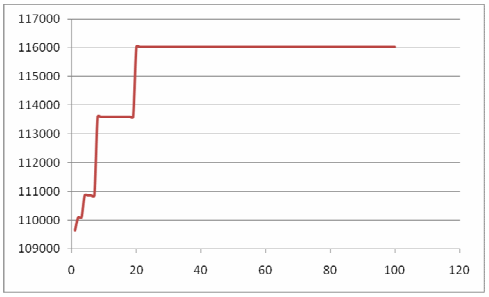



a

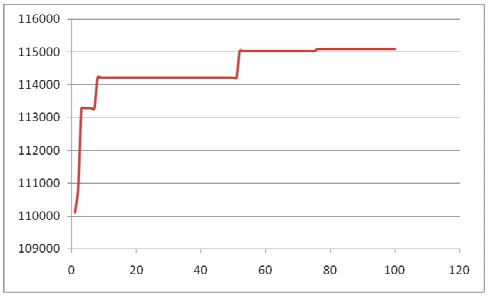



b

**Şekil 5. 3 : 3 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafi**

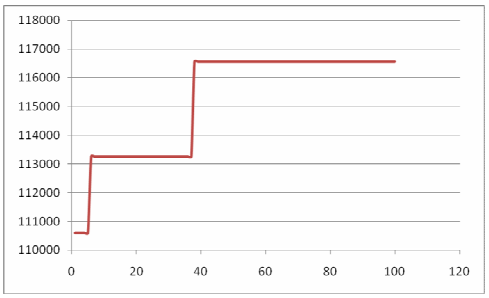



a

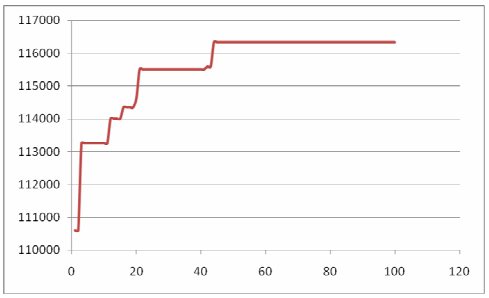



b

**Şekil 5. 4 : 4 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

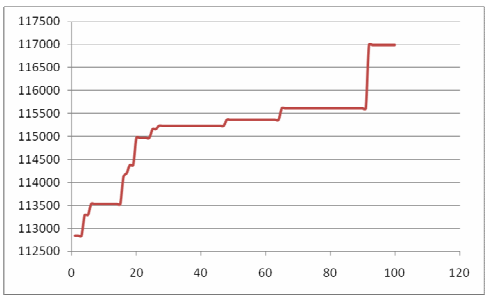



a

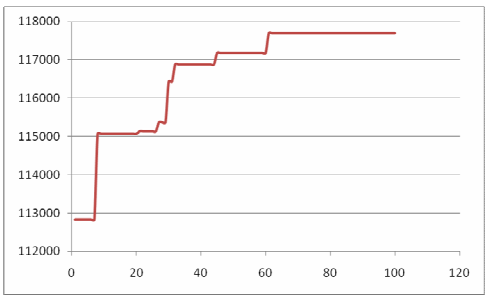



b

**Şekil 5. 5 : 5 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

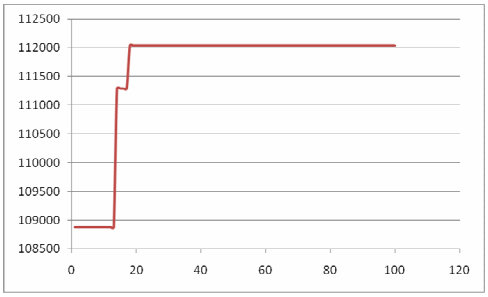



a

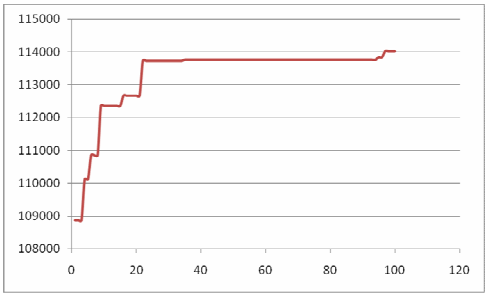



b

**Şekil 5. 6: 6 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

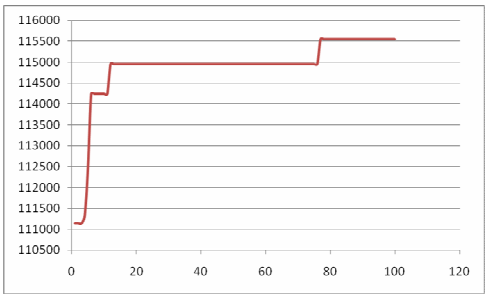



a

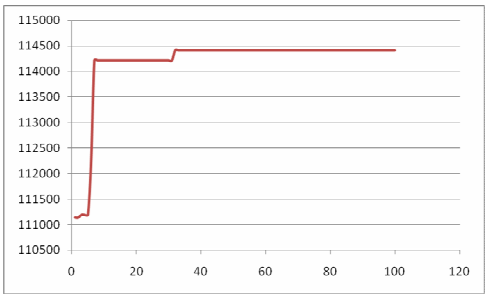



b

**Şekil 5. 7: 7 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

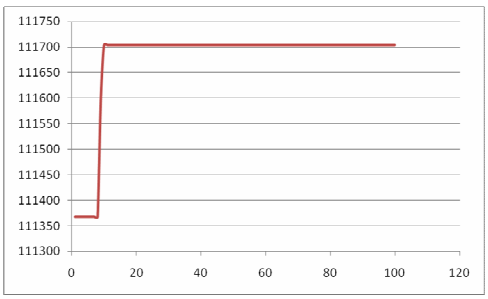



a

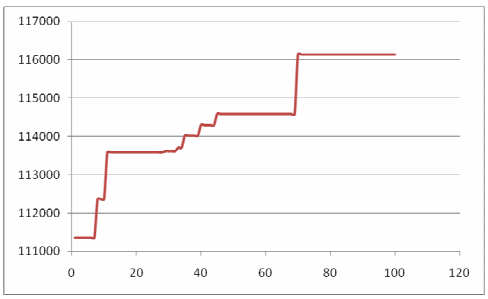



b

**Şekil 5. 8: 8 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

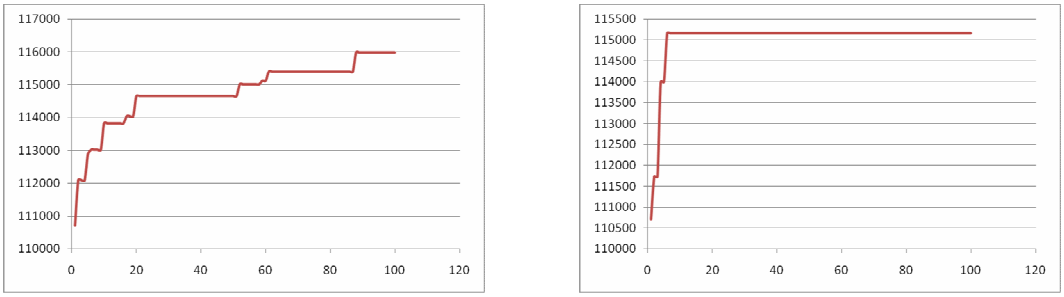



a




b

**Şekil 5. 9: 9 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**




a b

**Şekil 5. 10 : 10 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

## 5.2 İLK SIĞAN ALGORİTMASI İLE YERLEŞİM VE GENETİK ALGORİTMA İLE SEÇİM ALGORİTMASI SONUÇLARI

Yapılan bu çalışmada rasgele olarak oluşturulmuş Bölüm 5.1'de dinamik programlama ile yerleşim ve genetik algoritma ile seçim programında kullanılan, Ek-1 den Ek-10 a kadar verilmiş, herbiri 50 parçadan oluşan 10 test verisi kullanılmıştır. Aynı şekilde bu programda da başlangıç popülasyon sayısı 10 ve 20 birey olarak belirlenmiştir ve sonuçların karşılaştırılmaları yapılmıştır. İterasyon sayısı 100 olarak belirlenmiştir. Yerleştirilecek büyük alan L = 400 ve W = 300 birim boyutlarındadır. Yerleştirilecek olan paketlerin toplam alanı, yerleşimin yapılacağı dikdörtgenin alanından daha büyük olduğu bilinmektedir.

### 5.2.1 Başlangıç Popülasyonunun Sonuca Etkileri

Tablo 5.4'de başlangıç popülasyonuna göre elde edilen en iyi sonuçlar birim kare cinsinden verilmiştir. Aynı şekilde Tablo 5.5'de başlangıç popülasyonuna bağlı olarak elde edilen en iyi sonuçların fire değerleri verilmiştir. Test sonuçlarından anlaşılacağı üzere dinamik programlama yerleşiminde olduğu gibi bazı problemler için başlangıç popülasyonu artışı daha az fire üretirken, bazı test verileri için daha yüksek fire değerleri elde edilmiştir. Aynı

sebeplerden ötürü başlangıç popülasyonun artışı fire değerlerini her zaman küçültür diyemeyiz.

**Tablo 5. 4 : Test verilerinin başlangıç popülasyonuna göre birim kare üzerinden yerleşim sonuçları**

| | | Başlangıç popülasyonu | |
|---|---|---|---|
| | | 10 | 20 |
| Test numarası | 1 | 117790 | 117733 |
| | 2 | 116990 | 117340 |
| | 3 | 116346 | 116984 |
| | 4 | 116883 | 118077 |
| | 5 | 117841 | 117948 |
| | 6 | 118510 | 118232 |
| | 7 | 116608 | 117337 |
| | 8 | 116767 | 116996 |
| | 9 | 117668 | 117542 |
| | 10 | 117689 | 117389 |

**Tablo 5. 5 : Test verilerinin başlangıç popülasyonuna göre fire değerleri**

| | | Başlangıç popülasyonu | |
|---|---|---|---|
| | | 10 | 20 |
| Test numarası | 1 | 0,018416667 | 0,018891667 |
| | 2 | 0,025083333 | 0,022166667 |
| | 3 | 0,03045 | 0,025133333 |
| | 4 | 0,025975 | 0,016025 |
| | 5 | 0,017991667 | 0,0171 |
| | 6 | 0,012416667 | 0,014733333 |
| | 7 | 0,028266667 | 0,022191667 |
| | 8 | 0,026941667 | 0,025033333 |
| | 9 | 0,019433333 | 0,020483333 |
| | 10 | 0,019258333 | 0,021758333 |

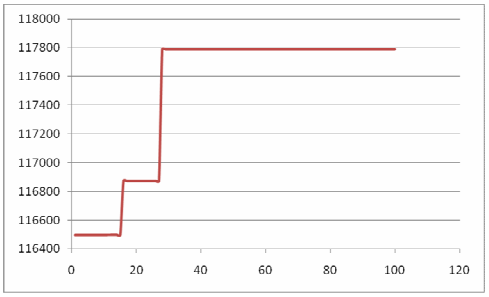
Test verisi olarak kullandığımız 10 problemden 1, 6, 9 ve 10 numaralı problemler hariç başlangıç popülasyonu, sonucu olumlu katkı yapmıştır. Tablo 5.6'da verilen en büyük fire farkı olan 0,006075 değeri 7 numaralı problem için başlangıç popülasyonu artışı sonucu olumlu şekilde etkileyerek elde edilmiştir. Bu sebep ile tüm problem kümesi için başlangıç popülasyonu sayısının belirlenmesi zordur.

**Tablo 5. 6 : Başlangıç popülasyonuna bağlı sonuçların fire değerleri farkı**

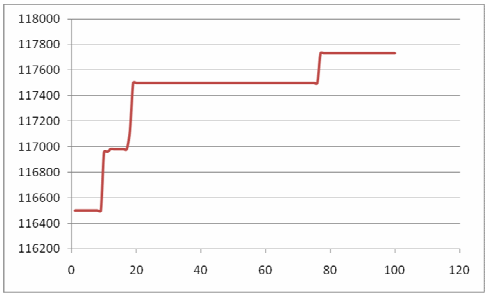
| | | Fire değerleri farkları |
|---|---|---|
| Test numarası | 1 | -0,000475 |
| | 2 | 0,002916667 |
| | 3 | 0,005316667 |
| | 4 | 0,00995 |
| | 5 | 0,000891667 |
| | 6 | -0,002316667 |
| | 7 | 0,006075 |
| | 8 | 0,001908333 |
| | 9 | -0,00105 |
| | 10 | -0,0025 |

### 5.2.2 İterasyon Sayısının Sonuca Etkileri

Şekil 5.11'den Şekil 5.20'a kadar verilen grafikler incelendiğinde problemin doğası gereği hangi iterasyonda en iyi sonucu üreteceği dinamik programlama uygulamasındaki gibi belirlenemez. Örneğin Şekil 5.16(a)'de iterasyona bağlı alan grafiği verilen verilen 6 numaralı problem ve 10 başlangıç popülasyonu için yapılan çalışmada en iyi sonuç 20. iterasyonda bulunurken, Şekil 5.15(a)'da iterasyona bağlı alan grafiği verilen 5 numaralı problem ve 10 başlangıç popülasyonu için yapılan çalışmada en iyi sonuç 97. iterasyonda bulunmuştur.

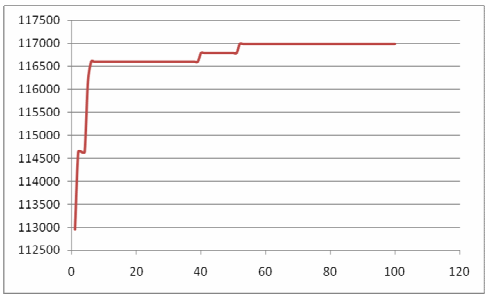



a

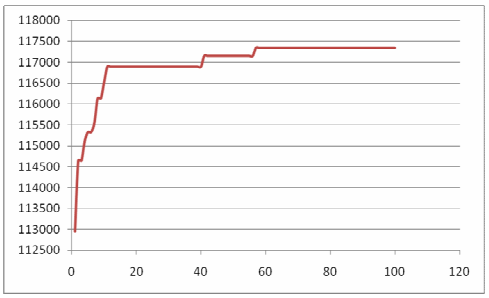



b

**Şekil 5. 2 : 1 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

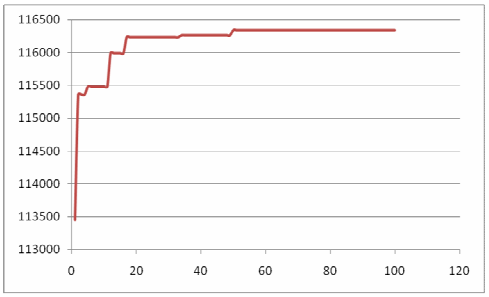



a

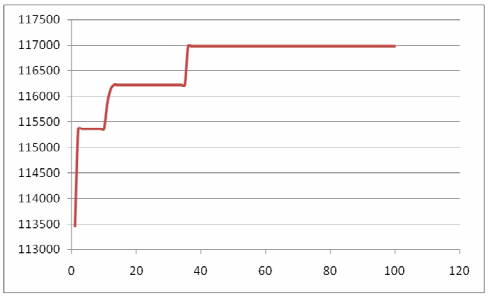



b

**Şekil 5. 12: 2 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

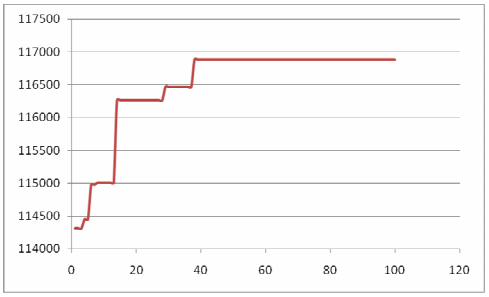



a

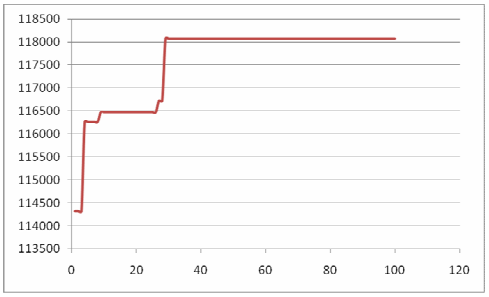



b

**Şekil 5. 13 : 3 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

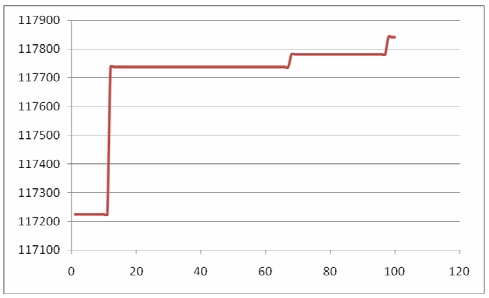



a

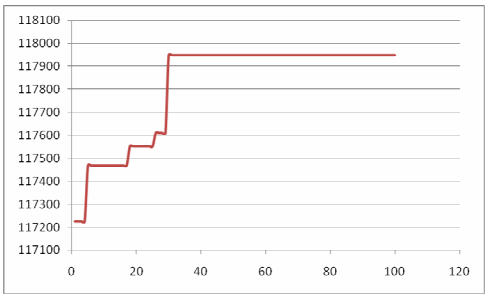



b

**Şekil 5. 14 : 4 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

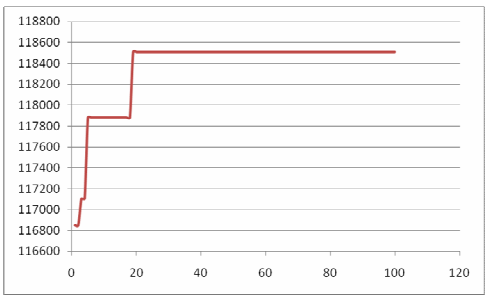



a

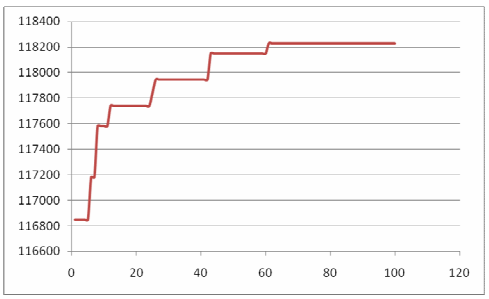



b

**Şekil 5. 15: 5 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

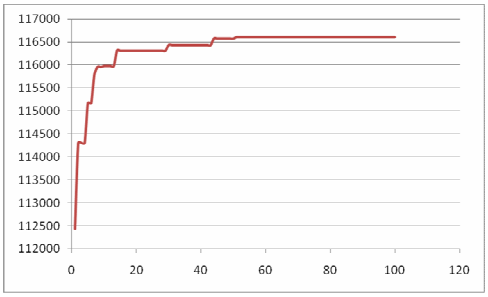



a

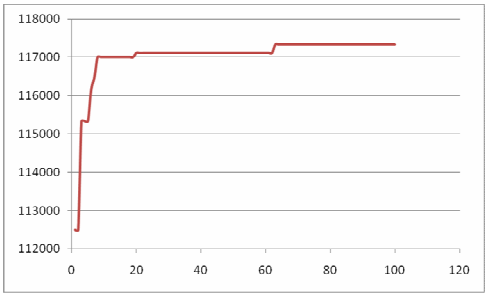



b

**Şekil 5. 16: 6 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

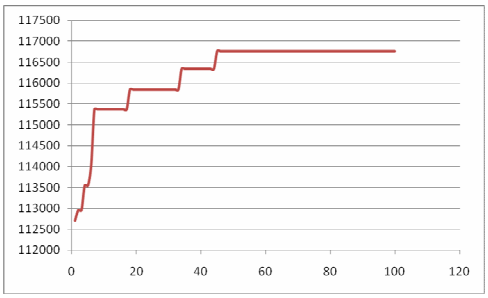



a

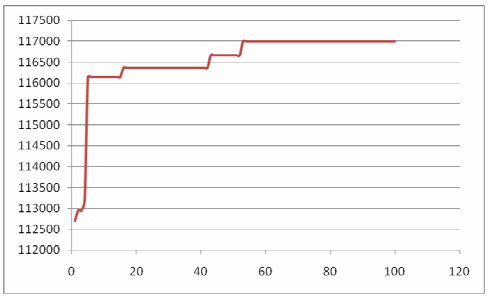



b

**Şekil 5. 17: 7 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

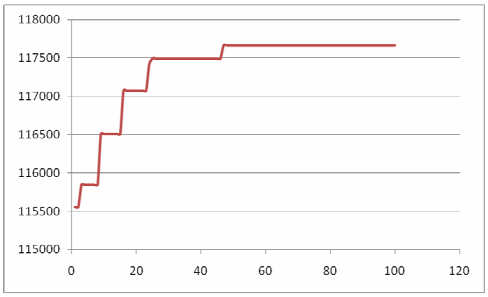



a

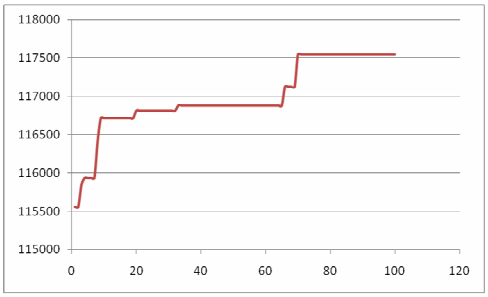



b

**Şekil 5. 18: 8 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

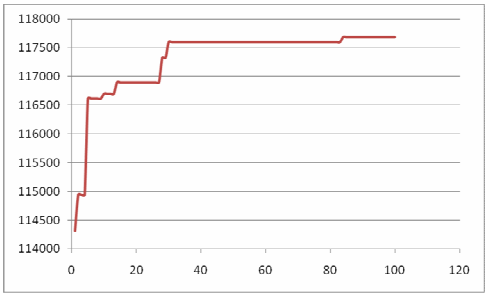



a

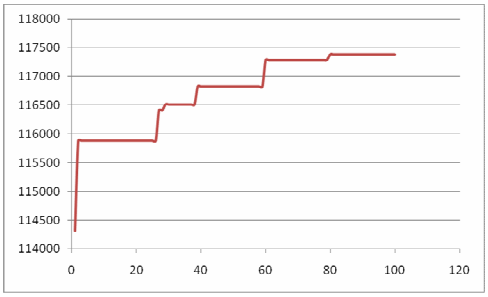



b

**Şekil 5. 19: 9 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**




a




b

**Şekil 5. 20 : 10 nolu problemin başlangıç popülasyonu 10 (a) ve 20 (b) iterasyona bağlı değişim grafiği**

## 5.3 - SEÇİM ALGORİTMALARI SONUÇLARININ KARŞILAŞTIRILMASI

Hazırlanan iki yerleşim programı için rastsal şekilde oluşturulmuş 10 test verisi için algoritmalar 10 ve 20 başlangıç popülasyonu ve 100 iterasyon için çalıştırılmıştır. Tablo 5.7'de elde edilen en iyi yerleşim alanları birimkare cinsinden verilmiştir.

**Tablo 5. 7 : Yerleşim Algoritmalarına göre elde edilmiş en iyi yerleşim sonuçları**

| | | Yerleşim Algoritması | |
|---|---|---|---|
| | | Dinamik Programlama | İlk Sığan |
| Test numarası | 1 | 115215 | 117790 |
| | 2 | 115175 | 117340 |
| | 3 | 112220 | 116984 |
| | 4 | 116023 | 118077 |
| | 5 | 116569 | 117948 |
| | 6 | 117688 | 118510 |
| | 7 | 114023 | 117337 |
| | 8 | 115544 | 116996 |
| | 9 | 116140 | 117668 |
| | 10 | 115988 | 117689 |

Elde edilen sonuçlara dayanarak 10 test problem içinde ilk sığan algoritması dinamik programlama ile yapılan yerleşimden daha iyi bir yerleşim oluşturmuştur. Bunun sebebi bölüm 3.1.3'de bahsedildiği üzere, sınır değerini doldurmak için seçilebilecek kutuların genişliklerinin yüksekliklerine oranın çok büyük olmasıdır. Birbirinden farklı boyutlara sahip oldukları için bulunması gereken sınır değeri azalır ve bu değeri dolduracak uygun dikdörtgenler bulunamaz ve boş alanlar artar. Bu durum dinamik programlamanın sadece tek bir değeri sağlayacak olmasından kaynaklanmaktadır. Eğer aynı boyutlara sahip dikdörtgen sayısı artar ise dinamik programlama daha iyi sonuçlar verecektir.

Dinamik programlama $O(n^2)$ karmaşıklığında çalışırken, ilk sığan algoritması $O(n^3)$ zaman karmaşıklığında çalışmaktadır.

## 5.4 – GENETİK ALGORİTMANIN SONUÇLARA KATKISI

Bu bölümde başlangıç popülasyonunda elde edilen en iyi değer ile belirlenen en son iterasyonda elde edilen değer arasındaki farklar ortaya koyularak genetik algoritmanın yerleşime olan katkısı gösterilmiştir.

Tablo 5.8 ve Tablo 5.9'de ilk sığan algoritması ile yapılan uygulamada en iyi ilk birey ve en iyi son birey arasındaki fire farkları verilmiştir.

Tablo 5.10 ve Tablo 5.11'de dinamik programlama ile yapılan uygulamada en iyi ilk birey ve en iyi son birey arasındaki fire farkları verilmiştir.

**Tablo 5. 8 : İlk sığan yerleşimi ile 1O Başlangıç popülasyonuna göre elde edilen endüşük ve en yüksek fire değerleri ve farkları**

| | | 10 Başlangıç popülasyonuna göre | | |
|---|---|---|---|---|
| | | İlk en iyi bireyin fire değeri | Son en iyi bireyin fire değeri | fire farkı |
| Test numarası | 1 | 0,029183333 | 0,018416667 | 0,010766667 |
| | 2 | 0,058775 | 0,025083333 | 0,033691667 |
| | 3 | 0,05455 | 0,03045 | 0,0241 |
| | 4 | 0,047358333 | 0,025975 | 0,021383333 |
| | 5 | 0,023133333 | 0,017991667 | 0,005141667 |
| | 6 | 0,02625 | 0,012416667 | 0,013833333 |
| | 7 | 0,063083333 | 0,028266667 | 0,034816667 |
| | 8 | 0,060816667 | 0,026941667 | 0,033875 |
| | 9 | 0,037025 | 0,019433333 | 0,017591667 |
| | 10 | 0,047383333 | 0,019258333 | 0,028125 |

**Tablo 5. 9 : İlk sığan yerleşimi ile 2O Başlangıç popülasyonuna göre elde edilen endüşük ve en yüksek fire değerleri ve farkları**

| | | 20 Başlangıç popülasyonuna göre | | |
|---|---|---|---|---|
| | | ilk en iyi bireyin fire değeri | son en iyi bireyin fire değeri | fire farkı |
| Test numarası | 1 | 0,029183333 | 0,018891667 | 0,010291667 |
| | 2 | 0,058775 | 0,022166667 | 0,036608333 |
| | 3 | 0,05455 | 0,025133333 | 0,029416667 |
| | 4 | 0,047358333 | 0,016025 | 0,031333333 |
| | 5 | 0,023133333 | 0,0171 | 0,006033333 |
| | 6 | 0,02625 | 0,014733333 | 0,011516667 |
| | 7 | 0,062508333 | 0,022191667 | 0,040316667 |
| | 8 | 0,060816667 | 0,025033333 | 0,035783333 |
| | 9 | 0,037025 | 0,020483333 | 0,016541667 |
| | 10 | 0,047383333 | 0,021758333 | 0,025625 |

**Tablo 5. 10 : Dinamik programlama ile 1O Başlangıç popülasyonuna göre elde edilen endüşük ve en yüksek fire değerleri ve farkları**

| | | 10 Başlangıç popülasyonuna göre | | |
|---|---|---|---|---|
| | | ilk en iyi bireyin fire değeri | son en iyi bireyin fire değeri | fire farkı |
| Test numarası | 1 | 0,069166667 | 0,040891667 | 0,028275 |
| | 2 | 0,071483333 | 0,040208333 | 0,031275 |
| | 3 | 0,104 | 0,064833333 | 0,039166667 |
| | 4 | 0,082425 | 0,033141667 | 0,049283333 |
| | 5 | 0,078308333 | 0,028591667 | 0,049716667 |
| | 6 | 0,059683333 | 0,025066667 | 0,034616667 |
| | 7 | 0,092716667 | 0,066316667 | 0,0264 |
| | 8 | 0,07385 | 0,037133333 | 0,036716667 |
| | 9 | 0,071941667 | 0,069133333 | 0,002808333 |
| | 10 | 0,077466667 | 0,033433333 | 0,044033333 |

**Tablo 5. 11 : Dinamik programlama ile 2O Başlangıç popülasyonuna göre elde edilen endüşük ve en yüksek fire değerleri ve farkları**

| | | 20 Başlangıç popülasyonuna göre | | |
|---|---|---|---|---|
| | | ilk en iyi bireyin fire değeri | son en iyi bireyin fire değeri | fire farkı |
| Test numarası | 1 | 0,069166667 | 0,039875 | 0,029291667 |
| | 2 | 0,071483333 | 0,056825 | 0,014658333 |
| | 3 | 0,104 | 0,0767 | 0,0273 |
| | 4 | 0,082425 | 0,040983333 | 0,041441667 |
| | 5 | 0,078308333 | 0,030533333 | 0,047775 |
| | 6 | 0,059683333 | 0,019266667 | 0,040416667 |
| | 7 | 0,092716667 | 0,049808333 | 0,042908333 |
| | 8 | 0,07385 | 0,046533333 | 0,027316667 |
| | 9 | 0,071941667 | 0,032166667 | 0,039775 |
| | 10 | 0,077466667 | 0,040383333 | 0,037083333 |

Elde edilen verilere dayanarak genetik algoritmanın %5 lere kadar bir katkı sağladığı görülmüştür. Seçim algoritmasının katksı en fazla 5 nolu test verisinin 10 iterasyon için dinamik algoritma ile yerleşimde görülmüştür. Burada en iyi ilk birey ve son en iyi birey

arasındaki fire farkı 0,049716667 olarak belirlenmiştir. Herbir test verisi için farklı bir katkı sağlayan genetik algoritma tüm testler için yerel maksimumları olan ilk bireyin yerleşim planından daha iyi bir yerleşim planı oluşturmasını sağlamıştır.

## 5.5 - GCUT PROBLEM SERİSİ İÇİN DİNAMİK PROGRAMLAMA İLE YERLEŞİM VE GENETİK ALGORİTMA İLE SEÇİM ALGORİTMASI SONUÇLARI

GCUT, 13 adet test verisinden oluşmakta olup, Beasly tarafından 1985 yılında yaptığı iki boyutlu giyotinsiz kesme problemleri için hazırladığı çalışmada tanımlamıştır. Ancak her bir parça sayısı bu çalışmada sonsuz olarak alınmıştır.(Beasly, 1985)

Caprara ve Monaci iki boyutlu sırt çantası problem için hazırladıkları çalışmada aynı test verisini herbir parçadan yalnızca bir kez kullanma koşulu ekleyerek, hazırladıkları dört farklı dal ve sınır algoritması için çözüm aramışlardır. (Caprara ve Monaci, 2004)
Hazırlanan ilk sığan ve dinamik algoritma yerleşim ve genetiik algoritma ile seçim algoritması gcut1-gcut13 dataları ile test edilmiş ve elde edilen sonuçlar birim kare cinsinden tablo 5.5.1 de verilmiştir. Ayrıca program çalışması süresi için 6 saatlik bir sınır koyulmuştur. 6 saatin sonunda bir veri elde edilemez ise tabloda bu alan boş bırakılmıştır. Aynı şekilde eğer sure sonunda iterasyon sayısına ulaşılmamış ise yine program kesilip en son elde edilen en iyi yerleşim değeri tablo da verilmiştir. Bunun sebebi ilk sığan algoritmasının büyük parçalar ve büyük yerleşim alanları verilerinde hızın olumsuz şekilde düşmesinden kaynaklanmaktadır.

**Tablo 5. 12 : GCUT problemlerinin 3 farklı algoritma sonucu bulunan çözümleri**

| Problem | Büyük parçanın boyutları | Kutu tipleri | Toplam kutu sayısı | Dal-sınır algoritması sonuçları | İlk Sığan algoritması sonuçları | Dinamik Algoritma sonuçları |
|---|---|---|---|---|---|---|
| GCUT1 | ( 250, 250) | 10 | 10 | 48368 | 58136 | 58136 |
| GCUT2 | ( 250, 250) | 20 | 20 | 59798 | 59692 | 58928 |
| GCUT3 | ( 250, 250) | 30 | 30 | 61275 | 60663 | 59587 |
| GCUT4 | ( 250, 250) | 50 | 50 | 61380 | 61710 | 61710 |
| GCUT5 | ( 500, 500) | 10 | 10 | 195582 | 233969 | 18339 |
| GCUT6 | ( 500, 500) | 20 | 20 | 236305 | 238030 | 223859 |
| GCUT7 | ( 500, 500) | 30 | 30 | 240143 | 238252 | 238543 |
| GCUT8 | ( 500, 500) | 50 | 50 | 245758 | 239957 | 243714 |
| GCUT9 | ( 1000, 1000) | 10 | 10 | 939600 | 907611 | 917905 |
| GCUT10 | ( 1000, 1000) | 20 | 20 | 937349 | 886406 | 906422 |
| GCUT11 | ( 1000, 1000) | 30 | 30 | 969709 | 869104 | 969273 |
| GCUT12 | ( 1000, 1000) | 50 | 50 | 979521 | 924472 | 962657 |
| GCUT13 | ( 3000, 3000) | 32 | 32 | <8408316 | ##### | 8505301 |

Gcut verisi ile yapılan çalışma göstermiştirki, yerleştirilecek olan paçanın boyutları ve yerleşimin yapılacağı parça büyürse, ilk sığan algoritması beklendiği üzere yavaşlamaktadır. Gcut verilerinin büyük bir kısmında istenilen iterasyona ulaşamadan süre sınırına takılmış ve program sonlandırılmıştır. Ancak dinamik programlama ile istenilen 10 başlangıç popülasyonu ve 100 iterasyon sayısına ulaşmıştır. Dal-sınır algoritmasının ve ilk sığan algoritmasının verilen sure sınırı içinde bir sonuç bulamamasına rağmen dinamik programlama 8505301 değerine ulaşmıştır. Gcut1, 4, 5 ve 6 için ilk sığan algoritması daha verimli yerleşim planı oluştururken Gcut1 ve Gcut13 için dinamik programlama en iyi yerleşim planı sonuçlarını bulmuştur.

# BÖLÜM 6: SONUÇ

Kaynak sıkıntısı ve bunun sonucu olarak ortaya çıkan ekonomik zorlukların arttığı günümüzde optimizasyon konusu üzerine yapılan çalışma sayısında artış sağlamıştır. Özellikle üretim yapan firmaların stokların verimli kullanımına olanak veren problemlerin çözümü kesme ve yerleştirme problemleri altında incelenmektedir. Endüstride bir çok farklı kısıt altında problem incelenebilmektedir. Bu çalışmada stok kesimi için kesilecek ya da yerleşimin yapılacağı büyük dikdörtgenin uzunluğu belli olarak kabul edilmiştir. Ancak kumaş kesim uygulaması için kumaşın ölçüleri sadece genişlik olarak verilmektedir ve uzunluk sonsuz olarak alınmalıdır. Bu nedenle her bir farklı uygulama için amaç fonksiyonunun yeniden tanımlanma zorunluluğu ortaya çıkar. Aynı şekilde deri kesim için stoğun her bir noktası aynı kalitede olmadığı ve kusurlu kısımlar bulunabileceği için bu problem için farklı kısıt tanımlamaları gerekmektedir.

Yapılan bütün bu çalışmalar bize gösteriyorki çok az sayıda örnek uygulanmasına rağmen 2BKP NP-zor sınıfında bulunmaya devam ediyor.Ancak yazılan polinomail algoritmalar bize bazı problemler için optimal sonuçları vermektedir, bu noktadan yola çıkarak probleme çözüm aramak yarar sağlayabilir.

Yapılan bu çalışmada problem iki aşamada incelenmiştir:

1. **Seçim Bölümü**
2. **Yerleşim Bölümü**

Problemi iki bölüme ayırmak her bir nokta için oluşan kısıt ve durumun analizini kolaylaştırmıştır. Bu sayede probleme farklı yaklaşım modelleri eklenebilmesi esnekliği kazandırılmıştır.

Seçim algoritması olarak genetik algoritmadan yararlanarak, iki boyutlu kesme problemi için paketlerin dizilimlerinin oluşturulduğu bir algoritma hazırlanmıştır. Elde edilen test

sonuçlarına göre genetik algoritma yerel bir maksimum noktasına takılmadan programın daha iyi yerleşim planları üretmesine olanak vermiştir.

Yerleşim algoritması olarak dinamik programlamadan yararlanarak sezgisel bir yerleşim algoritması oluşturulmuştur. Bu yaklaşım deterministik ve sezgisel algoritmaları kapsamaktadır. Bir boyutlu knapsack problemi için hazırlanmış dinamik programlama modeli belli sezgisel kriterler eklenerek iki boyutlu kesme prolemine uygun hale getirilmiştir. Ancak bu yaklaşım bize sadece bir boyutta kesin olarak en iyi dizilimi vermektedir, iki boyut için gereken zamansal karmaşıklık bu sayede indirgenmiştir.

Hazırlanan ikinci yerleşim algoritması ilk sığan ya da aşağı sol dolgu algoritmasıdır. Tamamen sezgisel bir yaklaşım olan bu algoritma keyfi sırada yerleşim ya da mantıksal sırada yerleşim sonucunda oluşmaktadır. Söke 2003 yılında yaptığı çalışmada üç farklı aşağı sol algoritmasına yer vermiştir. Bunların içinde en fazla süresel karmaşıklığa sahip algoritma aşağı sol dolgu olmasına rağmen en yoğun yerleşim planını yine aynı algoritma sağlamaktadır.

Süresel olarak ilk sığan algoritması daha uzun zamanda sonuçlar üretmesine karşın dinamik algoritma ile oluşturulmuş programın verdiği sonuçlara göre bir çok test datası için daha iyi sonuçlar üretmiştir. Ancak dinamik programlamanın daha kısa zamanda daha verimli yerleşim planları elde ettiği veriler mevcuttur. Buna dayanarak iki yerleşim programının birbirine üztün geldiği veya daha zayıf kaldığı noktalar vardır diyebiliriz.

Her iki program içinde yapılan bu çalışmalarda fire oranlarının yoksayılabilir oran olan %8 in altında olduğu görülmüştür. Ancak her veri gurubu için optimal sonucu vereceği garanti edilemez.

İleride yapılabilecek her iki algoritmayıda birleştiren sezgisel bir çalışma daha verimli sonuçlar üretebilir. Genetik algoritma üzerinde yapılabilecek değişiklikler ve problem verilerinin durumuna bağlı karar verebilen bir yapının oluşturulması ile gelecekte

yapılabilecek bir çalışmanın hem süresel hem de daha iyi bir yerleşim planı ortaya çıkarabileceği göz önüne alınmalıdır.

Günümüzde çeşitli alanlarda uygulama alanına sahip 2BKP nin şu anda optimal sonucu verebilecek bir polinamial algoritma yoktur. Ancak yapılan çalışmalar umut vericidir. Ancak doğanın kuralları bu problem içinde geçerlidir ; “bir noktadan kazanç başka bir noktadan kayıp demektir”.

## KAYNAKÇA


Beasley, D., Bull, D. R. & Martin, R. R., 1993a, An Overview of Genetic Algorithms: Part 1, *Fundamentals. University Computing*, vol.**15**(2), pp. 58-69,UK.

Beasley, D., Bull, D. R. & Martin, R. R., 1993b, An Overview of Genetic Algorithms: Part 2, *Research Topics. University Computing*, vol.**15**(4), pp. 170-181,UK.

Beasley, J. E., 1985, Algorithms for Unconstrained Two-Dimensional Guillotine Cutting, *Journal of the Operational Research Society*, vol. **36**, pp. 297-306.

Beasly, J.E. , 1985 , An exact two dimensionalnon-guillotine cutting tree search procedure, *Operations Research*, pp. 49-64.

Beasly, J.E., 1985, An algorithm for the two-dimensional assortment problem, *European Journal of Operational Research*, vol.**19**, pp. 253-261.

Beasley, J. E. & Hoare, N. P., 2001, Placing boxes on shelves: a case study, *Journal of the Operational Research Society*, Vol. **52**, pp. 605–614.

Bellman, R. , 1957, *Dynamic Programming*, Princeton University Press. Dover paperback edition (2003), UK.

Bischoff, E. & Dowsland, W.B., 1980, An application of the microcomputer to product design and distribution, *Journal of the Operational Research Society*, vol. **33**, pp. 271-280.

Callahgan, A.R., Nair, A.R. & Lewis, K. E. , 1999, An Extension of The Orthogonal Packing Problem Through Dimensional Flexibility, *1999 ASME Design Engineering Technical Conferences*, pp. 12-15.

Caprara, A. & Monaci, M., 2004, On the two-dimensional Knapsack Problem. *Operations Research Letters*, vol. **32**, pp. 5-14.

Chazelle, B., 1983, The bottom-left bin packing heuristic: An Efficent implementation, *IEEE Trans. on Comput.* vol. **32**, pp. 697–707.

Chen, C., Sarin, S. & Ram, B. , 1991, The pallet packing problem for non-uniform box sizes, *International Journal of Production Research*, vol. **29**(10), pp. 1963-1968.

Cormen, T.H., Leiserson, C.E., Rivest, L.R. & Stein, C., 2001, *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, ISBN 0-262-03293-7. Section 29.3: The simplex algorithm, pp.790–804.

Downsland, K.A. , 1987, A combined data-base and algorithmic approach to the pallet loading problem, *Journal of the Operational Research Society*, vol. **38**(4), pp. 341-345.

Fekete, S. P. & Schepers J., 1997, *On higher-dimensional packing II:Bounds*.Technical report, Univ. of Cologne , Center for Parallel Computing ,http://www.math.tu-bs.de/~fekete/publications.html.

Garey, M. R. & Johnson, D. S., 1979, *Computers And Intractability – A Guide To The Theory Of NP – Completeness*, New York, W.H. Freeman.

Gasımov, R.N., Siphaioğlu ve A., Saraç, T. 2004, 1.5 Boyutlu Stok Malzemesi Seçimi Problemi İçin Çok Ölçütlü Bir Karar Modeli Ve Çözüm Yaklaşımı, *YA/EM'2004 - Yöneylem Araştırması/Endüstri Mühendisliği - XXIV Ulusal Kongresi.*

Gilmore P.C. & Gomory R.E., 1961, A linear programming approachto the cutting stock problem, *Operation Research*, vol. **9**, pp. 849-859.

Gilmore P.C. & Gomory R.E., 1963, A linear programming approachto the cutting stock problem part II , *Operation Research*, vol. **11**, pp. 863-888.

Haessler, R.W. & Sweeney, P.E., 1991, Cutting Stock Problems and Solution Procedures. *European Journal of Operational Research,* vol. **54**, pp. 141-150.

Hodgson, T.J., 1982, A combined approach to the pallet loading problem, IIE Transaction vol. **14**(3), pp. 175-182.

Holland, J.H., 1975, *Adaption in Natural and Artificial Systems*, University of Michigan Pres, Ann Arbor, MI, 1975.

Hopper E. & Turton B., 1997, Application of Genetic Algorithms to Packing Problems - A Review, *Proceedings of the 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing,* Springer Verlag, London, pp. 279-288.

Hopper, E. & Turton, B., 1999., A Genetic Algorithm for A 2D Industrial Packing Problem, *Computers & Industrial Engineering*, Vol. **37**, pp. 375-378.

HU T. C., 1970, *Integer Programming And Network Flows*,Addison-Wesley Publishing Company Reading, Massachusetts, 1970.

İşçi, Ö. ve Korukoğlu, S., 2003, Genetik Algoritma Yaklaşımı ve Yöneylem Araştırmasında Bir Uygulama, *CBÜ İİBF Yönetim ve Ekonomi Dergisi*(ISSN-1302-0064), Cilt **10** sayı 2, ss 191-208.

Jain, S. & Gea, H. C., 1998, Two-Dimensional Packing Problems Using Genetic Algorithm, *Engineering with Computers*, vol. **14**, pp. 206-213.

Leung, T.W., Yung, C.H. & Chan, C.K., 1999. Applications of Genetic Algorithm and Simulated Annealing to The Two-Dimensional Non-Guillotine Cutting Stock Problem, Presented at *IFORS'99*, Beijing China.

Leung, T.W., Yung, C.H. & Troutt, M.D., 2001. Applications of Genetic Search ad Simulated Annealing to The Two-Dimensional Non-Guillotine Cutting Stock Problem. *Computers and Industrial Engineering*, Vol.**40**, pp.201-214.

Lines, M. E., 1990, *Bir Sayı Tut*, 10, Arık, N.( Çev.), Tübitak, Ankara, 2004.

Lodi A., 2000, Algorithms for Two-Dimensional Bin Packing and Assignment Problems , *Ph.D. Thesis*, University of Bologna, Italy.

Lodi, A. & Monaci, M., 2003, Integer linear programming models for the 2-staged two-dimensional knapsack problem, *Math. Prog*., vol. **94**, pp. 257–278.

Lodi, A., Martello, S. & Monaci, M., 2002, Two-dimensional packing problems: a survey, *European Journal of Operational Research*, Vol. **141**(2), pp. 241-252.

Lodi, A., Martello, S. & Vigo, D., 1999, Heuristic and meta-heuristic approaches for a class of two dimensional bin packing problems, *INFORMS Journal on Computing*, vol. **11**, pp. 345–357.

Mitchell, M. & Forest S., 1994, *Genetic Algorithms and Artificial Life. Vol. 1*, No. 3, pp. 267-289. Reprinted in C. G. Langton (Ed.) Artificial Life: an Overview, MIT Press, Cambridge, MA (1995).

Nepomuceno, N. V., Pinheiro, P. R. & Coelho, A. L. V., 2008, A Hybrid Optimization Framework for Cutting and Packing Problems: Case Study on Constrained 2D Non-guillotine Cutting, *Recent Advances in Evolutionary Computation for Combinatorial Optimization.* Springer-Verlag, v. **153**, p. 87-99.

Oğuz, M. ve Abaş, S., 1997, Genetik Algoritmalar, *Bitirme Tezi*, YTÜ Endüstri Mühendisliği, İstanbul.

Pisinger D. & Sigurd M., (2005) , The two-dimensional bin packing problem with variable bin sizes and costs, Department of Computer Science, University of Copenhagen, *Operation Research,* p155-p163.

Pisinger D., 1995, An expanding-core algorithm for the exact 0-1 knapsack problem, *European Journal of Operational Research*, Vol. **87**, p175–187.

Pisinger, D., Keller, H. and Fersehy, V. P., 2004, *Knapsack Problems*, Springer, Berlin Heidelberg New York, 2004.

Smith, D., 1985. Bin Packing with Adaptive Search, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pp. 202-206, Lawrence Erlbaum.

Söke, A., 2003, Genetik Algoritma Ve Benzetilmiş Tavlama ile İki Boyutlu Giyotinsiz Kesme Problemlerine Olasılıksal Yaklaşım,*Yüksek Lisans Tezi* ,Kocaeli Üniversitesi, Kocaeli

Steudel, H., 1979, Generating pallet loading pattems:A special case for the two dimensional cutting stock problem, *Management Science*, Vol. **25**(10), pp 997-1004.

## EK – 1 1 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 85 | 65 | 1 |
| 2 | 70 | 35 | 1 |
| 3 | 37 | 93 | 1 |
| 4 | 2 | 92 | 1 |
| 5 | 98 | 86 | 1 |
| 6 | 6 | 50 | 1 |
| 7 | 104 | 95 | 1 |
| 8 | 45 | 116 | 1 |
| 9 | 105 | 7 | 1 |
| 10 | 114 | 44 | 1 |
| 11 | 63 | 93 | 1 |
| 12 | 7 | 72 | 1 |
| 13 | 57 | 36 | 1 |
| 14 | 75 | 78 | 1 |
| 15 | 32 | 34 | 1 |
| 16 | 100 | 99 | 1 |
| 17 | 71 | 119 | 1 |
| 18 | 110 | 28 | 1 |
| 19 | 84 | 118 | 1 |
| 20 | 30 | 65 | 1 |
| 21 | 13 | 120 | 1 |
| 22 | 82 | 2 | 1 |
| 23 | 70 | 13 | 1 |
| 24 | 13 | 96 | 1 |
| 25 | 35 | 6 | 1 |
| 26 | 36 | 46 | 1 |
| 27 | 37 | 114 | 1 |
| 28 | 118 | 49 | 1 |
| 29 | 34 | 20 | 1 |
| 30 | 20 | 78 | 1 |
| 31 | 50 | 50 | 1 |
| 32 | 86 | 40 | 1 |
| 33 | 76 | 25 | 1 |
| 34 | 23 | 71 | 1 |
| 35 | 10 | 55 | 1 |
| 36 | 109 | 32 | 1 |
| 37 | 95 | 46 | 1 |
| 38 | 35 | 111 | 1 |
| 39 | 76 | 76 | 1 |
| 40 | 52 | 12 | 1 |
| 41 | 68 | 84 | 1 |
| 42 | 110 | 101 | 1 |
| 43 | 3 | 66 | 1 |
| 44 | 110 | 52 | 1 |
| 45 | 82 | 61 | 1 |

| | | | |
|---|---|---|---|
| 46 | 62 | 56 | 1 |
| 47 | 43 | 49 | 1 |
| 48 | 33 | 7 | 1 |
| 49 | 30 | 118 | 1 |
| 50 | 8 | 47 | 1 |

## EK – 2 2 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 44 | 59 | 1 |
| 2 | 19 | 57 | 1 |
| 3 | 31 | 76 | 1 |
| 4 | 66 | 19 | 1 |
| 5 | 113 | 79 | 1 |
| 6 | 61 | 47 | 1 |
| 7 | 13 | 95 | 1 |
| 8 | 56 | 91 | 1 |
| 9 | 72 | 100 | 1 |
| 10 | 3 | 26 | 1 |
| 11 | 9 | 13 | 1 |
| 12 | 40 | 16 | 1 |
| 13 | 1 | 65 | 1 |
| 14 | 79 | 66 | 1 |
| 15 | 100 | 10 | 1 |
| 16 | 24 | 82 | 1 |
| 17 | 55 | 43 | 1 |
| 18 | 18 | 85 | 1 |
| 19 | 112 | 64 | 1 |
| 20 | 11 | 91 | 1 |
| 21 | 49 | 56 | 1 |
| 22 | 60 | 25 | 1 |
| 23 | 40 | 12 | 1 |
| 24 | 71 | 21 | 1 |
| 25 | 112 | 12 | 1 |
| 26 | 54 | 33 | 1 |
| 27 | 105 | 91 | 1 |
| 28 | 33 | 81 | 1 |
| 29 | 31 | 11 | 1 |
| 30 | 4 | 39 | 1 |
| 31 | 95 | 36 | 1 |
| 32 | 29 | 58 | 1 |
| 33 | 31 | 41 | 1 |
| 34 | 6 | 58 | 1 |
| 35 | 25 | 104 | 1 |
| 36 | 71 | 91 | 1 |
| 37 | 112 | 40 | 1 |
| 38 | 66 | 10 | 1 |
| 39 | 77 | 50 | 1 |
| 40 | 116 | 14 | 1 |
| 41 | 111 | 75 | 1 |
| 42 | 42 | 18 | 1 |
| 43 | 58 | 27 | 1 |
| 44 | 120 | 16 | 1 |

| | | | |
|---|---|---|---|
| 45 | 4 | 42 | 1 |
| 46 | 66 | 111 | 1 |
| 47 | 65 | 49 | 1 |
| 48 | 102 | 100 | 1 |
| 49 | 81 | 87 | 1 |
| 50 | 120 | 41 | 1 |

## EK – 3 3 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 60 | 50 | 1 |
| 2 | 84 | 22 | 1 |
| 3 | 51 | 66 | 1 |
| 4 | 98 | 65 | 1 |
| 5 | 52 | 62 | 1 |
| 6 | 28 | 75 | 1 |
| 7 | 59 | 82 | 1 |
| 8 | 107 | 45 | 1 |
| 9 | 37 | 36 | 1 |
| 10 | 19 | 64 | 1 |
| 11 | 27 | 71 | 1 |
| 12 | 44 | 106 | 1 |
| 13 | 58 | 23 | 1 |
| 14 | 83 | 90 | 1 |
| 15 | 74 | 94 | 1 |
| 16 | 20 | 97 | 1 |
| 17 | 25 | 115 | 1 |
| 18 | 8 | 8 | 1 |
| 19 | 96 | 46 | 1 |
| 20 | 56 | 15 | 1 |
| 21 | 14 | 21 | 1 |
| 22 | 6 | 86 | 1 |
| 23 | 64 | 68 | 1 |
| 24 | 27 | 57 | 1 |
| 25 | 90 | 91 | 1 |
| 26 | 48 | 109 | 1 |
| 27 | 90 | 11 | 1 |
| 28 | 77 | 86 | 1 |
| 29 | 2 | 52 | 1 |
| 30 | 49 | 34 | 1 |
| 31 | 119 | 97 | 1 |
| 32 | 84 | 51 | 1 |
| 33 | 89 | 34 | 1 |
| 34 | 43 | 53 | 1 |
| 35 | 114 | 15 | 1 |
| 36 | 78 | 42 | 1 |
| 37 | 13 | 23 | 1 |
| 38 | 10 | 52 | 1 |
| 39 | 116 | 66 | 1 |
| 40 | 60 | 117 | 1 |
| 41 | 27 | 46 | 1 |
| 42 | 48 | 34 | 1 |
| 43 | 61 | 17 | 1 |
| 44 | 63 | 116 | 1 |
| 45 | 67 | 110 | 1 |

| | | | |
|---|---|---|---|
| 46 | 79 | 53 | 1 |
| 47 | 84 | 8 | 1 |
| 48 | 91 | 85 | 1 |
| 49 | 60 | 19 | 1 |
| 50 | 27 | 40 | 1 |

## EK – 4 4 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 95 | 7 | 1 |
| 2 | 63 | 91 | 1 |
| 3 | 97 | 40 | 1 |
| 4 | 117 | 97 | 1 |
| 5 | 81 | 109 | 1 |
| 6 | 106 | 50 | 1 |
| 7 | 15 | 115 | 1 |
| 8 | 96 | 84 | 1 |
| 9 | 49 | 2 | 1 |
| 10 | 21 | 20 | 1 |
| 11 | 62 | 49 | 1 |
| 12 | 13 | 34 | 1 |
| 13 | 78 | 102 | 1 |
| 14 | 60 | 23 | 1 |
| 15 | 108 | 45 | 1 |
| 16 | 39 | 93 | 1 |
| 17 | 27 | 54 | 1 |
| 18 | 29 | 106 | 1 |
| 19 | 74 | 45 | 1 |
| 20 | 47 | 104 | 1 |
| 21 | 71 | 112 | 1 |
| 22 | 63 | 40 | 1 |
| 23 | 105 | 32 | 1 |
| 24 | 32 | 22 | 1 |
| 25 | 42 | 1 | 1 |
| 26 | 90 | 101 | 1 |
| 27 | 34 | 85 | 1 |
| 28 | 49 | 98 | 1 |
| 29 | 90 | 53 | 1 |
| 30 | 10 | 50 | 1 |
| 31 | 41 | 86 | 1 |
| 32 | 38 | 96 | 1 |
| 33 | 19 | 72 | 1 |
| 34 | 115 | 30 | 1 |
| 35 | 113 | 14 | 1 |
| 36 | 119 | 76 | 1 |
| 37 | 72 | 109 | 1 |
| 38 | 69 | 30 | 1 |
| 39 | 104 | 9 | 1 |
| 40 | 53 | 92 | 1 |
| 41 | 30 | 46 | 1 |
| 42 | 48 | 64 | 1 |
| 43 | 33 | 70 | 1 |
| 44 | 26 | 10 | 1 |
| 45 | 108 | 14 | 1 |

| | | | |
|---|---|---|---|
| 46 | 79 | 109 | 1 |
| 47 | 28 | 114 | 1 |
| 48 | 102 | 53 | 1 |
| 49 | 60 | 93 | 1 |

## EK – 5 5 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 24 | 40 | 1 |
| 2 | 50 | 19 | 1 |
| 3 | 75 | 12 | 1 |
| 4 | 25 | 84 | 1 |
| 5 | 61 | 22 | 1 |
| 6 | 119 | 57 | 1 |
| 7 | 1 | 52 | 1 |
| 8 | 35 | 91 | 1 |
| 9 | 107 | 99 | 1 |
| 10 | 22 | 14 | 1 |
| 11 | 2 | 19 | 1 |
| 12 | 46 | 23 | 1 |
| 13 | 118 | 105 | 1 |
| 14 | 70 | 88 | 1 |
| 15 | 18 | 31 | 1 |
| 16 | 4 | 100 | 1 |
| 17 | 95 | 95 | 1 |
| 18 | 40 | 55 | 1 |
| 19 | 67 | 97 | 1 |
| 20 | 68 | 25 | 1 |
| 21 | 25 | 44 | 1 |
| 22 | 69 | 42 | 1 |
| 23 | 64 | 52 | 1 |
| 24 | 69 | 7 | 1 |
| 25 | 66 | 25 | 1 |
| 26 | 116 | 112 | 1 |
| 27 | 120 | 101 | 1 |
| 28 | 50 | 97 | 1 |
| 29 | 119 | 32 | 1 |
| 30 | 21 | 84 | 1 |
| 31 | 3 | 41 | 1 |
| 32 | 84 | 14 | 1 |
| 33 | 14 | 35 | 1 |
| 34 | 48 | 68 | 1 |
| 35 | 38 | 36 | 1 |
| 36 | 97 | 38 | 1 |
| 37 | 46 | 49 | 1 |
| 38 | 101 | 50 | 1 |
| 39 | 87 | 80 | 1 |
| 40 | 49 | 115 | 1 |
| 41 | 78 | 62 | 1 |
| 42 | 27 | 70 | 1 |
| 43 | 90 | 55 | 1 |
| 44 | 109 | 34 | 1 |
| 45 | 81 | 107 | 1 |

| | | | |
|---|---|---|---|
| 46 | 34 | 45 | 1 |
| 47 | 2 | 36 | 1 |
| 48 | 119 | 92 | 1 |
| 49 | 70 | 73 | 1 |
| 50 | 33 | 61 | 1 |

## EK – 6 6 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 96 | 103 | 1 |
| 2 | 12 | 66 | 1 |
| 3 | 50 | 9 | 1 |
| 4 | 55 | 84 | 1 |
| 5 | 60 | 49 | 1 |
| 6 | 114 | 119 | 1 |
| 7 | 71 | 64 | 1 |
| 8 | 74 | 11 | 1 |
| 9 | 23 | 35 | 1 |
| 10 | 3 | 19 | 1 |
| 11 | 51 | 66 | 1 |
| 12 | 68 | 94 | 1 |
| 13 | 6 | 49 | 1 |
| 14 | 8 | 38 | 1 |
| 15 | 73 | 63 | 1 |
| 16 | 72 | 40 | 1 |
| 17 | 110 | 93 | 1 |
| 18 | 70 | 52 | 1 |
| 19 | 88 | 20 | 1 |
| 20 | 8 | 30 | 1 |
| 21 | 92 | 115 | 1 |
| 22 | 42 | 83 | 1 |
| 23 | 95 | 5 | 1 |
| 24 | 96 | 59 | 1 |
| 25 | 25 | 73 | 1 |
| 26 | 110 | 114 | 1 |
| 27 | 79 | 2 | 1 |
| 28 | 80 | 116 | 1 |
| 29 | 2 | 115 | 1 |
| 30 | 9 | 36 | 1 |
| 31 | 70 | 97 | 1 |
| 32 | 5 | 7 | 1 |
| 33 | 68 | 38 | 1 |
| 34 | 50 | 62 | 1 |
| 35 | 89 | 55 | 1 |
| 36 | 39 | 9 | 1 |
| 37 | 50 | 69 | 1 |
| 38 | 92 | 70 | 1 |
| 39 | 79 | 97 | 1 |
| 40 | 51 | 96 | 1 |
| 41 | 87 | 84 | 1 |
| 42 | 88 | 24 | 1 |
| 43 | 93 | 9 | 1 |
| 44 | 30 | 115 | 1 |
| 45 | 99 | 27 | 1 |

| | | | |
|---|---|---|---|
| 46 | 23 | 29 | 1 |
| 47 | 112 | 31 | 1 |
| 48 | 31 | 86 | 1 |
| 49 | 80 | 93 | 1 |
| 50 | 92 | 38 | 1 |

## EK – 7 7 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 99 | 87 | 1 |
| 2 | 108 | 27 | 1 |
| 3 | 43 | 67 | 1 |
| 4 | 43 | 25 | 1 |
| 5 | 70 | 76 | 1 |
| 6 | 17 | 8 | 1 |
| 7 | 77 | 101 | 1 |
| 8 | 77 | 51 | 1 |
| 9 | 23 | 60 | 1 |
| 10 | 117 | 93 | 1 |
| 11 | 46 | 59 | 1 |
| 12 | 66 | 81 | 1 |
| 13 | 5 | 78 | 1 |
| 14 | 108 | 41 | 1 |
| 15 | 15 | 52 | 1 |
| 16 | 93 | 43 | 1 |
| 17 | 50 | 90 | 1 |
| 18 | 53 | 87 | 1 |
| 19 | 61 | 21 | 1 |
| 20 | 34 | 11 | 1 |
| 21 | 118 | 35 | 1 |
| 22 | 75 | 37 | 1 |
| 23 | 103 | 39 | 1 |
| 24 | 96 | 21 | 1 |
| 25 | 37 | 9 | 1 |
| 26 | 60 | 55 | 1 |
| 27 | 72 | 4 | 1 |
| 28 | 39 | 17 | 1 |
| 29 | 40 | 92 | 1 |
| 30 | 76 | 116 | 1 |
| 31 | 100 | 67 | 1 |
| 32 | 73 | 23 | 1 |
| 33 | 109 | 39 | 1 |
| 34 | 73 | 72 | 1 |
| 35 | 93 | 86 | 1 |
| 36 | 83 | 92 | 1 |
| 37 | 31 | 66 | 1 |
| 38 | 62 | 68 | 1 |
| 39 | 34 | 50 | 1 |
| 40 | 43 | 117 | 1 |
| 41 | 63 | 70 | 1 |
| 42 | 56 | 4 | 1 |
| 43 | 46 | 17 | 1 |
| 44 | 79 | 19 | 1 |
| 45 | 24 | 46 | 1 |

| | | | |
|---|---|---|---|
| 46 | 12 | 74 | 1 |
| 47 | 85 | 53 | 1 |
| 48 | 51 | 41 | 1 |
| 49 | 85 | 26 | 1 |
| 50 | 6 | 107 | 1 |

## EK – 8 8 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 6 | 56 | 1 |
| 2 | 89 | 91 | 1 |
| 3 | 99 | 26 | 1 |
| 4 | 57 | 105 | 1 |
| 5 | 100 | 50 | 1 |
| 6 | 56 | 42 | 1 |
| 7 | 30 | 97 | 1 |
| 8 | 66 | 65 | 1 |
| 9 | 76 | 11 | 1 |
| 10 | 86 | 31 | 1 |
| 11 | 7 | 34 | 1 |
| 12 | 116 | 40 | 1 |
| 13 | 78 | 44 | 1 |
| 14 | 45 | 40 | 1 |
| 15 | 26 | 26 | 1 |
| 16 | 51 | 75 | 1 |
| 17 | 29 | 83 | 1 |
| 18 | 71 | 73 | 1 |
| 19 | 10 | 6 | 1 |
| 20 | 19 | 30 | 1 |
| 21 | 48 | 71 | 1 |
| 22 | 35 | 101 | 1 |
| 23 | 108 | 64 | 1 |
| 24 | 47 | 1 | 1 |
| 25 | 31 | 20 | 1 |
| 26 | 78 | 118 | 1 |
| 27 | 38 | 29 | 1 |
| 28 | 106 | 16 | 1 |
| 29 | 69 | 73 | 1 |
| 30 | 74 | 84 | 1 |
| 31 | 103 | 12 | 1 |
| 32 | 48 | 99 | 1 |
| 33 | 33 | 82 | 1 |
| 34 | 48 | 101 | 1 |
| 35 | 52 | 101 | 1 |
| 36 | 27 | 88 | 1 |
| 37 | 108 | 27 | 1 |
| 38 | 118 | 116 | 1 |
| 39 | 105 | 63 | 1 |
| 40 | 79 | 55 | 1 |
| 41 | 19 | 78 | 1 |
| 42 | 74 | 103 | 1 |
| 43 | 45 | 7 | 1 |
| 44 | 61 | 7 | 1 |
| 45 | 45 | 55 | 1 |

| | | | |
|---|---|---|---|
| 46 | 36 | 117 | 1 |
| 47 | 2 | 79 | 1 |
| 48 | 74 | 64 | 1 |
| 49 | 4 | 87 | 1 |
| 50 | 38 | 118 | 1 |

## EK – 9 9 NUMARALI TEST VERİLERİ

| No | Yükseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 35 | 44 | 1 |
| 2 | 39 | 43 | 1 |
| 3 | 29 | 72 | 1 |
| 4 | 85 | 33 | 1 |
| 5 | 70 | 70 | 1 |
| 6 | 119 | 112 | 1 |
| 7 | 30 | 98 | 1 |
| 8 | 93 | 49 | 1 |
| 9 | 76 | 60 | 1 |
| 10 | 60 | 16 | 1 |
| 11 | 117 | 22 | 1 |
| 12 | 71 | 82 | 1 |
| 13 | 14 | 79 | 1 |
| 14 | 89 | 94 | 1 |
| 15 | 52 | 10 | 1 |
| 16 | 60 | 69 | 1 |
| 17 | 8 | 50 | 1 |
| 18 | 15 | 18 | 1 |
| 19 | 38 | 119 | 1 |
| 20 | 73 | 33 | 1 |
| 21 | 50 | 57 | 1 |
| 22 | 8 | 55 | 1 |
| 23 | 53 | 114 | 1 |
| 24 | 33 | 21 | 1 |
| 25 | 61 | 42 | 1 |
| 26 | 62 | 48 | 1 |
| 27 | 32 | 9 | 1 |
| 28 | 77 | 44 | 1 |
| 29 | 66 | 106 | 1 |
| 30 | 51 | 72 | 1 |
| 31 | 46 | 7 | 1 |
| 32 | 14 | 18 | 1 |
| 33 | 18 | 92 | 1 |
| 34 | 32 | 17 | 1 |
| 35 | 6 | 84 | 1 |
| 36 | 102 | 63 | 1 |
| 37 | 97 | 119 | 1 |
| 38 | 63 | 9 | 1 |
| 39 | 76 | 40 | 1 |
| 40 | 34 | 82 | 1 |
| 41 | 49 | 47 | 1 |
| 42 | 25 | 106 | 1 |
| 43 | 19 | 49 | 1 |
| 44 | 74 | 66 | 1 |
| 45 | 88 | 15 | 1 |

| | | | |
|---|---|---|---|
| 46 | 39 | 1 | 1 |
| 47 | 95 | 111 | 1 |
| 48 | 36 | 28 | 1 |
| 49 | 115 | 23 | 1 |
| 50 | 51 | 44 | 1 |

## EK – 10 10 NUMARALI TEST VERİLERİ

| No | Yukseklik | Genislik | Adet |
|---|---|---|---|
| 1 | 39 | 118 | 1 |
| 2 | 70 | 64 | 1 |
| 3 | 17 | 108 | 1 |
| 4 | 4 | 96 | 1 |
| 5 | 87 | 30 | 1 |
| 6 | 114 | 57 | 1 |
| 7 | 87 | 48 | 1 |
| 8 | 28 | 73 | 1 |
| 9 | 67 | 56 | 1 |
| 10 | 78 | 92 | 1 |
| 11 | 8 | 19 | 1 |
| 12 | 13 | 7 | 1 |
| 13 | 66 | 115 | 1 |
| 14 | 28 | 42 | 1 |
| 15 | 107 | 58 | 1 |
| 16 | 28 | 102 | 1 |
| 17 | 9 | 16 | 1 |
| 18 | 116 | 101 | 1 |
| 19 | 99 | 120 | 1 |
| 20 | 29 | 80 | 1 |
| 21 | 76 | 117 | 1 |
| 22 | 51 | 7 | 1 |
| 23 | 106 | 20 | 1 |
| 24 | 30 | 73 | 1 |
| 25 | 44 | 64 | 1 |
| 26 | 111 | 66 | 1 |
| 27 | 116 | 92 | 1 |
| 28 | 49 | 91 | 1 |
| 29 | 69 | 54 | 1 |
| 30 | 6 | 5 | 1 |
| 31 | 44 | 50 | 1 |
| 32 | 44 | 76 | 1 |
| 33 | 22 | 53 | 1 |
| 34 | 19 | 117 | 1 |
| 35 | 33 | 8 | 1 |
| 36 | 97 | 86 | 1 |
| 37 | 31 | 95 | 1 |
| 38 | 105 | 119 | 1 |
| 39 | 33 | 56 | 1 |
| 40 | 110 | 111 | 1 |
| 41 | 61 | 107 | 1 |
| 42 | 69 | 51 | 1 |
| 43 | 99 | 88 | 1 |
| 44 | 108 | 19 | 1 |
| 45 | 93 | 80 | 1 |

| | | | |
|---|---|---|---|
| 46 | 110 | 101 | 1 |
| 47 | 94 | 94 | 1 |
| 48 | 57 | 89 | 1 |
| 49 | 68 | 20 | 1 |
| 50 | 38 | 57 | 1 |

## ÖZGEÇMİŞ

**Adı Soyadı** : Yakup Alper ERDOĞAN

**Sürekli Adresi** : Selami Ali Mah. Gazi Cad. No:56/2 Üsküdar / ISTANBUL

**Doğum Yeri ve Yılı** : İSTANBUL / 1981

**Yabancı Dili** : İngilizce

**İlk Öğretim** : Ahmet Cevdet Paşa İlköğretim Okulu - 1992

**Orta Öğretim** : Rezzan Has Lisesi - 1995

**Lise** : Kabataş Erkek Lisesi - 1999

**Lisans** : Ege Üniversitesi / Bilgisayar Bilimleri Ağırlıklı Matematik - 2006

**Yüksek Lisans** : Bahçeşehir Üniversitesi

**Enstitü Adı** : Fen Bilimleri Enstitüsü

**Program Adı** : Bilgi Teknolojileri

**Çalışma Hayatı** :

Nortel Netaş 2006 - ….