KONTEYNIR YÜKLEME PROBLEMİNE SEZGİSEL BİR YAKLAŞIM

Mustafa ENSARİ

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCAK 2007
ANKARA

Mustafa ENSARİ tarafından hazırlanan KONTEYNIR YÜKLEME PROBLEMİNE SEZGİSEL BİR YAKLAŞIM adlı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Serpil EROL
Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği ile Endüstri Mühendisliği Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr. Taner ALTINOK

Üye : Prof Dr. Serpil EROL

Üye : Yrd. Doç. Dr. Ediz ATMACA

Tarih : ...15...../…01…./…2007…

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Mustafa ENSARİ

KONTEYNIR YÜKLEME PROBLEMİNE
SEZGİSEL BİR YAKLAŞIM
(Yüksek Lisans Tezi)

Mustafa ENSARİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Ocak 2007

ÖZET

Konteynır yükleme probleminin ele alındığı bu çalışmada geçmiş çalışmalardan farklı bir yaklaşım geliştirilerek üç boyutlu bu problemde daha etkin bir yapı ortaya konulmaya çalışılmıştır. Bu çalışmada problem daha önceki çoğu çalışma gibi iki boyutlu ve tek boyutlu olmak üzere ikiye ayrılmamıştır. Üç boyutlu olarak ele alınan problemde tüm boyutlar eşit öncelikli olarak düşünülmüştür. Geliştirilen çözüm yaklaşımında bir kurala göre seçilen birimlerden önce büyük birimler oluşturulur, daha sonra bu büyük birimler konteynıra yine belirlenen kurala göre yüklenir. Konteynır, hiçbir büyük birim konteynıra yüklenemeyecek doluluğa ulaştığında büyük birimler ayrılarak "birimlere" dönüştürülür ve konteynıra yüklenir. Son aşamada algoritma kaydırma prosesleri kullanarak boş alanları mümkün olduğunca birleştirmeye çalışır ve birleştirilen boş alanlara yine açıkta kalan birimler yüklenir. Geliştirilen bu algoritma ile literatürdeki test problemleri denenmiş ve önceki çalışmalarla karşılaştırılarak algoritmanın etkinliği belirlenmiştir.

Bilim Kodu : **906.1.141**
Anahtar Kelimeler : **konteynır yükleme, sezgisel, lojistik**
Sayfa Adedi : **119**
Tez Yöneticisi : **Prof. Dr. Serpil EROL**

A HEURISTIC APPROACH FOR CONTAINER LOADING PROBLEM
(M.Sc. Thesis)

Mustafa ENSARİ

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
January 2007

ABSTRACT

This study deals with three dimensional container loading problem, and a different heuristic approach is generated. It aims to create a more effective structure about three dimensional container loading problem. This problem has not been divided into two smaller problems like the most of previous ones (two dimensional and one dimensional container loading problem). All the dimensions has the same priority. In this approach first big items will be created from items that has been selected with a definite rule. Than these big items will be loaded to the container. After the "loading big items prosess" the items that has not been loaded will be loaded to the container. The last phase is to move the items that divides the spare space in the container and to load the possible items to these spare space. With this newly generated algorithm, the testing problems in the literature have been tested and by comparing with the previous studies, the efficiency of the algorithm is proved.

Science Code : 906.1.141
Key Words : container loading, heuristic, logistic
Page Number: 119
Adviser : Prof. Dr. Serpil Erol

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren Hocam Prof. Dr. Serpil Erol'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ ix

ŞEKİLLERİN LİSTESİ xi

1. GİRİŞ 1

2. KONTEYNIR YÜKLEME PROBLEMİ 4

2.1. Yükleme Problem Tipleri 4

2.1.1.Tek boyutlu yükleme problemleri 4

2.1.2.İki boyutlu yükleme problemleri 4

2.1.3.Üç boyutlu yükleme problemleri 4

2.2.Problemin Tanımı 6

3. LİTERATÜR ARAŞTIRMASI 11

4. PROBLEME YENİ BİR YAKLAŞIM (EE YAKLAŞIMI) 40

4.1. Algoritma 45

4.2. Akış Çizelgesi 47

4.3. İstatistiksel Analiz 49

4.4. EE Uygulaması 52

5.EE ALGORİTMASININ DİĞER ALGORİTMALARLA KARŞILAŞTIRILMASI 58

6.SONUÇ VE ÖNERİLER 60

Sayfa

KAYNAKLAR 62

EKLER 64
EK-1 Diziler, sabitler ve değişkenler 65
EK-2 Fonksiyonlar 70
EK-3 C program kodu 78
EK-4 Sonuçlar 117

ÖZGEÇMİŞ 119

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 3.1. Bischoff ve Marriot'un 1.test grubu 15

Çizelge 3.2. Bischoff ve Marriot'un 2.test grubu 16

Çizelge 3.3. Bischoff ve Marriot'un 1. tip denemeleri 18

Çizelge 3.4. Bischoff ve Marriot'un 2. tip denemeleri 19

Çizelge 3.5. Gehring ve arkadaşlarının kullandığı konteynır tipleri 22

Çizelge 3.6. Gehring ve arkadaşlarının kullandığı birim boyutları 22

Çizelge 3.7. Gehring ve arkadaşlarının elde ettiği 1.tip konteynır yükleme planları 23

Çizelge 3.8. Gehring ve arkadaşlarının elde ettiği 2.tip konteynır yükleme planları 23

Çizelge 3.9. Chen ve arkadaşlarının kullandığı birimler 28

Çizelge 3.10. Chen ve arkadaşlarının elde ettiği yerleşim planı 29

Çizelge 3.11. Pisinger'ın kullandığı birimler 33

Çizelge 3.12. Pisinger'ın kısmen heterojen yerleşimi 34

Çizelge 3.13. Pisinger'ın heterojen yerleşimi 35

Çizelge 3.14. Pisinger'ın homojen yerleşim 36

Çizelge 3.15. Baltacıoğlu'nun elde ettiği sonuçlar 38

Çizelge 3.16. BR problem seti sonuçları karşılaştırması 38

Çizelge 4.1. Anlamlılık testi 51

Çizelge 4.2. Annova testi 51

Çizelge 4.3. Regresyon 52

Çizelge **Sayfa**

Çizelge 5.1. EE algoritması ile diğer algoritmaların karşılaştırılması..... 58

Çizelge 5.2. EE algoritması ile diğer algoritmalar arasındaki sayısal farklar…...59

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. Konteynıra birimlerin yüklenmesi 7

Şekil 2.2. İkinci yükleme için uygun noktalar..... 8

Şekil 2.3. Uygun noktaların ağaç gösterimi 8

Şekil 2.4. Rotasyon 9

Şekil 3.1. George ve Robinson'ın sınıflandırma kuralına göre sıralı yerleşimi 11

Şekil 3.2. Gehring ve arkadaşlarının yaklaşımında boş alanların gösterimi 21

Şekil 3.3. Gehring ve arkadaşlarının yükleme planları..... 24

Şekil 3.4. Chen ve arkadaşlarının modelindeki değişkenlerin gösterimi..... 26

Şekil 3.5. Chen ve arkadaşlarının optimum yerleşim 29

Şekil 4.1. Büyük birim oluşturma..... 41

Şekil 4.2. Uygun nokta seçimi..... 42

Şekil 4.3. Birimleri kaydırma prosesi 43

Şekil 4.4. Uygun noktaları kaydırma prosesi 44

Şekil 4.5. Akış çizelgesi..... 47

Şekil 4.6. Verilerin türetilmesi 52

Şekil 4.7. Birim verileri..... 53

Şekil 4.8. Diagonal 54

Şekil 4.9. Birim boyutları ve konteynır içerisindeki koordinatları..... 54

Şekil 4.10. Büyük birimler..... 55

Şekil 4.11. Analiz 56

Şekil **Sayfa**

Şekil 4.12. Üç boyutlu görünüm...57

1.GİRİŞ

Günümüzde artan rekabet ve taleplere hızlı cevap verme gereksinimlerinin kuvvetlendirdiği zorunluluklardan birisi de maliyetlerin minimize edilmesidir. Bu anlamda her sektörde ve her sektörün çeşitli alanlarında sayısız çalışmalar yapılmıştır.

Lojistik sektöründe de maliyetlerin minimize edilmesi anlamında rotalama, yerleşim ve yükleme problemleri göze çarpan unsurlar olmaktadır.

Rotalama ve yerleşim problemleri çok bilinen ve üzerinde çok sayıda çalışmalar yapılmış olan problem türleridir. Rotalama, dağıtım yapacak yüklü bir aracın dağıtım merkezlerine minimum maliyeti sağlayacak sıralamayı arayan problem türüdür. Yerleşim problemi de dağıtım merkezlerinin, dağıtımı en iyileyecek noktalara yüklemeye çalışan problem türüdür.

Yükleme problemi ise bir araca belli sayıda birimi yine minimum maliyeti sağlayacak şekilde yükleme problemidir. Bu çalışmada bu konu ele alınacaktır.

Her geçen gün çok sayıda birim bir yerden bir yere taşınmaktadır. Global dünyada birimlerin temel maliyetleri kadar artık lojistik maliyetleri de çok büyük önem kazanmıştır. Bu durumda taşıma maliyetlerinin minimize edilmesinin bir ayağı da konteynırların içerisine yüklenen birimlerin mümkün olduğunca konteynırda boş alan bırakmaması olacaktır. Çünkü konteynırların içerisinde kalan her boş santimetreküp o konteynırda taşınan ürünlerin maliyetlerine yansıyacaktır.

Giderek globalleşen dünyamızda lojistik harcamalarının gittikçe gösterdiği artışa paralel olarak konteynır yükleme problemine gösterilen ilgi de artmıştır. Bu konuda çeşitli yaklaşımlar ve çeşitli teknikler geliştirilmiş, optimum çözüme ulaşılmaya çalışılmıştır. Ne var ki günümüz bilgisayarları kullanılarak bile geliştirilen optimum çözüme ulaşma yöntemleri pratik hayatta kullanılmaya uygun çözüm zamanları sunamamaktadır. Örneğin Ballew (2000) optimum çözüme ulaşmak amacıyla bir

genetik algoritma geliştirmiş fakat yalnızca 11 adet birimi yüklemek için bile 45 dakika gibi bir zamanı sunmuştur [1].

Yukarıda açıklanan sebep, yükleme problemini sezgisel yöntemlerle çözmeye yöneltmiştir. Bir çok yazar probleme sezgisel yaklaşan algoritmalar geliştirmişler ve kabul edilebilir sonuçlara ulaşmışlardır.

Geliştirilen çoğu sezgisel algoritmalar *wall building* olarak adlandırılan bir yöntem yardımıyla meydana getirilmiştir. Bu yöntem basit olarak problemi iki bölüme ayırmak suretiyle çözmeye yöneliktir. İlk bölüm konteynırı derinlik boyutunda bölmelere ayırmak suretiyle ikinci bölümde oluşturulacak duvarların kalınlıklarını belirlemek, ikinci bölüm de belirlenen kalınlıklarda duvarlar inşa etmek olacaktır. Yani problemin ilk bölümü tek boyutlu yükleme problemi, ikinci bölümü de iki boyutlu yükleme problemine dönüşecektir.

Bir anlamda üç boyutlu yükleme problemi iki boyutlu ve tek boyutlu olmak üzere iki ayrı probleme dönüştürülmektedir. Yani duvarların oluşturulması problemi iki boyutlu yükleme problemi olarak, duvarların kalınlıklarının belirlenmesi de tek boyutlu yükleme problemi olarak düşünülmektedir.

Burada tahmin edileceği gibi oluşturulacak duvarların kalınlıklarının belirlenmesi çok önemli bir seçimdir. Yapılan çalışmalarda bu seçimin en sağlıklı şekilde yapılabilmesi için yöntemler geliştirilmiştir. Tabi ki oluşturulan duvarların minimum kayıpla inşa edilmesi de problemin diğer boyutu olacaktır.

Bu tez çalışmasının 2. bölümünde konteynır yükleme probleminin yapısı, genel özellikleri, türleri açıklanmıştır.

3. bölümünde konteynır yükleme problemi ile ilgili literatür araştırması sunulmuştur.

4. bölümünde geliştirilen yaklaşım prosesleri sırasıyla anlatılmış, algoritma ve akış çizelgesi verilmiş, sonuçlar istatistiki olarak analiz edilmiş ve C dilinde kodlanan bir program uygulaması açıklanmıştır.

5. bölümünde geliştirilen algoritma ile önceki çalışmalar karşılaştırılmıştır.

6. bölümünde ise elde edilen sonuçlar ve konteynır yükleme problemi için öneriler sunulmuştur.

2.KONTEYNIR YÜKLEME PROBLEMİ

Yükleme problemleri tek boyutlu, iki boyutlu ve üç boyutlu olarak üç kısımda incelenebilir.

2.1.Yükleme Problem Tipleri

2.1.1.Tek boyutlu yükleme problemleri

Tek boyutlu yükleme problemleri; yüklemesi yapılacak olan nesnelerin sadece tek boyutları ele alınmakta ve buna bağlı olarak optimum yükleme yapılmaya çalışılmaktadır. Bir konteynıra yüklenecek birimlerin sadece ağırlıklarının önemli olduğunu düşünüldüğünde, toplam ağırlığın belirlenmiş yasal sınırın altında, aynı zamanda maksimum seviyeye getirilmesi amaç olmalıdır.

2.1.2.İki boyutlu yükleme problemleri

İki boyutlu yükleme problemlerinde cismin iki boyutu (eni ve boyu) ele alınmakta olup, kayıplar (boş alanlar) minimize edilmeye çalışılmaktadır. Çok bilinen kesme ve yükleme problemleri (Cutting and Packing Problems) buna örnektir. $M \ x \ N$ boyutunda bir kumaşı ele alalım. Bu kumaştan kesilecek değişik boyutlarda kumaşlar olsun. Buradaki amaç tahmin edilebileceği gibi gerekli kumaşlar kesildikten sonra geriye kalan atıl bölümün asgariye indirilmesi olacaktır.

2.1.3.Üç boyutlu yükleme problemleri

Bu tez çalışmasında üç boyutlu olarak ele alınacak olan problemde değişik şekillerde üç boyutlu cisimler büyük bir konteynırın içine yüklenebilirler. Ancak burada yüklenecek bütün birimler ve konteynır dikdörtgenler prizması olarak ele alınmaktadır. Bu kabullerle farklı türlerde problem tipleri ortaya çıkmıştır:

Birimleri n adet küçük konteynırlara yükleme (Bin Packing Problem)

Bu problem tipinde *n* adet birim *m* adet küçük konteynırlara yüklemeye çalışılmaktadır. Problemde her küçük konteynırın içinde kalan boş alanlar asgariye indirilmeli ve buna bağlı olarak *m* minimize edilmelidir.

Değişken maliyetli yükleme problemi (Knapsack Loading)

Burada bir konteynır ve n adet birim vardır. Bu birimlerin her birinin de firmaya getireceği kar oranları değişkendir. Bu problemin amacı da konteynıra yüklenecek birimlerle kar oranları toplamının maksimize edilmesidir.

Konteynır yükleme problemi(Container Loading Problem)

Bu problem tipinde de bir konteynır ve yine n adet birim vardır. Konteynırın eni ve boyu sabit olup yüksekliği sonsuz olarak ele alınmakta ve amaç yükleme sonunda minimum yüksekliğe ulaşmak dolayısıyla en uygun yükleme elde etmektir.

Değişken maliyetli konteynır yükleme problemi (Knapsack Container Loading Problem)

Değişken maliyetli konteynır yükleme problemi (KCLP); konteynır yükleme probleminin bir anlamda yüksekliğinin sabit olduğu durumudur, denilebilir. KCLP; n adet birimin her birinin kar oranını o birimin hacmi olarak ele almakta ve eni, boyu ve yüksekliği sabit olan bir konteynıra toplam kar oranını maksimize edecek şekilde yüklemeye çalışır.

Ayrıca bu problem tiplerinde bazı kısıtlar da kullanılmaktadır. Gerçek hayatta konteynırların belli bir ağırlığın üstüne çıkmaları yasalarla kısıtlanmıştır. Bu nedenle problemin çözümünde ağırlık kısıtı eklenebilir.

Hava, deniz veya kara yollarıyla yapılan taşımacılıkta denge de çok önemli bir faktör olarak ortaya çıkmaktadır. Yapılan yüklemenin ağırlık merkezinin konteynırın ağırlık

merkezine yakın olması nakliye güvenliğini arttıracaktır. Probleme ağırlık merkezini belli sınırlar içinde tutacak bir kısıt eklenmesi mümkündür.

Bütün birimlerin dikdörtgenler prizması şeklinde olduğu kabul edilmiştir. Bu birimlerden hepsinin doksan derece dönüşlerle rotasyonlarına izin verilebileceği gibi bazı durumlarda da hiçbir birimin rotasyonuna izin verilmemesi istenebilir. Bazı birimlerin rotasyonuna izin verilirken bazı birimlerin sabit tutulması istenebilir veya bazı birimlerin sadece belli yönlerde rotasyonlarına izin verilebilir. Rotasyonları bu şekilde sınırlandıracak bir kısıt, gerçek hayattaki problemlerin çözümüne biraz daha yaklaştırmaktadır.

Belli birimlerin üstünde başka bir birimin olmaması veya üzerine sadece sınırlı bir ağırlık yüklenmesi istenebilir. Bu durumda, o birimin üzerindeki toplam ağırlık da probleme kısıt olarak eklenmelidir.

Bu şekilde çeşitli kısıtlar eklenerek problem farklı şekillerde sınıflandırılabilir. Gerçek hayatta bu kısıtların hepsinin karşılanması istenebilir veya sadece bazılarının sağlanması gerekebilir. Ancak akademik olarak elde edilen sonuçları karşılaştırma amacı ile bu kısıtlar genellikle kullanılmaktadır.

Bu tez çalışmasında üç boyutlu yükleme problemi ele alınmış olup aşağıda açıklanmıştır.

2.2.Problemin Tanımı

Konteynır yükleme problemi, bir yükleme problemi çeşidi olup birim ağırlıkları ve/veya hacimleri belirli n adet birimin bir konteynıra en verimli şekilde yüklenmesi problemidir. Problemde amaç; genişliği W, derinliği D ve yüksekliği H olan bir konteynıra yine genişlikleri w_i, derinlikleri d_i ve yükseklikleri h_i olan n adet birimi, karı maksimize edecek şekilde, başka bir deyişle boş alanları minimize edecek biçimde konumlandırmaya çalışmaktır. Bu işlem sonucunda birimlerin hepsinin konumlandırılması zorunlu değildir. Dışarıda kalan birimler olabilir. Bu durumda

yüklenen birimler için amaç fonksiyonu Eş 2.1, kısıtlar Eş 2.2, Eş 2.3, Eş 2.4 deki gibi olacaktır.

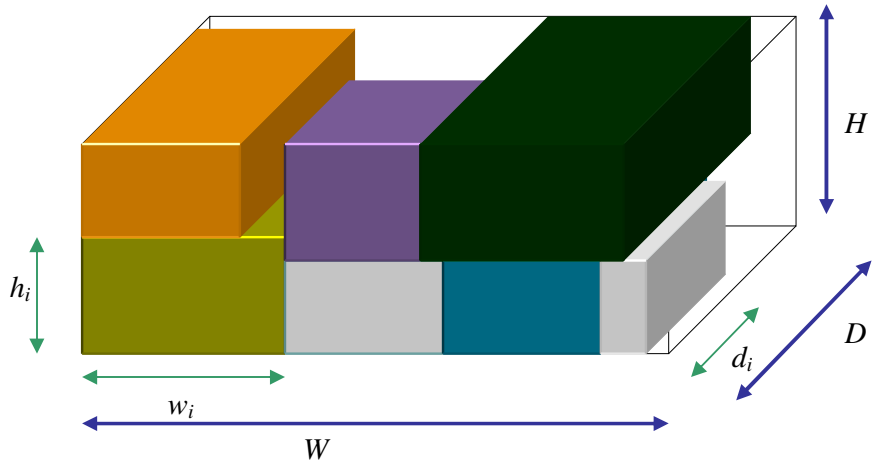
$$Z_{\max} = \sum (w_i \times d_i \times h_i) \tag{2.1}$$

$$\sum w_i \le W \tag{2.2}$$

$$\sum d_i \le D \tag{2.3}$$

$$\sum h_i \le H \tag{2.4}$$

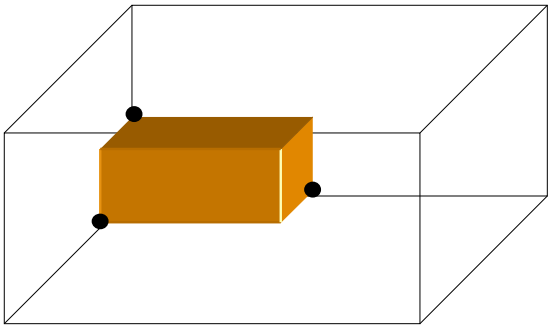
$$w_i, d_i, h_i \ge 0 \tag{2.5}$$



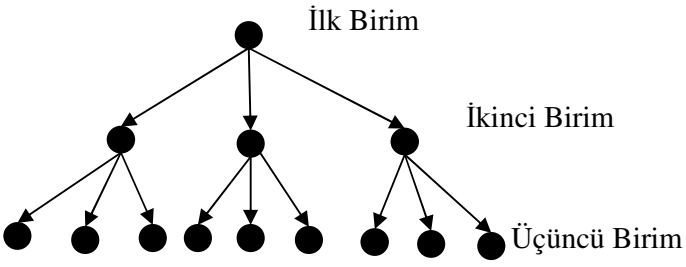
Şekil 2.1. Konteynıra birimlerin yüklenmesi [2]

Bu problem tipinin çözümünde genellikle sezgisel algoritmalara başvurulmuştur. Çünkü problem bir NP-Hard problemi olup optimum sonucun bilgisayar tarafından çözümü kabul edilebilir işlem zamanlarını aşmaktadır.

Problemde ilk birimin sol alt dip köşeye yüklendiği düşünüldüğünde, yüklenecek birim için sonsuz sayıda nokta olur. Bu düşünce ile problemin çözümü zaten olanaksız olacaktır. Bu yüzden ilk seçilen birimin sol alt dip köşeye (orijin) yüklenmesi ile başladıktan sonra diğer birimlerin sol alt dip köşelerinin ilk yüklenen birimin sağına, önüne ve üstüne yüklenmesi ile çözüme devam edilecektir. Yani her yüklemeden sonra bir sonraki yüklenecek olan birimin sol alt dip köşesi oluşan köşe noktalarından birine konumlandırılacak şekilde yüklenir. (Şekil 2.2) Her yüklemeden sonra üç adet uygun yükleme noktası oluşacaktır. Üçüncü birim düşünülecek olursa ikinci birimin her yükleme olasılığı için 3+2=5 yükleme olasılığı vardır. Bunu üçle çarpacak olursak üçüncü birim için toplam 15 yükleme noktası olacaktır. 15, *i.* birimin sol alt köşeye yüklenmesi durumunda oluşan yükleme olasılıkları sayısıdır. n birim için düşünülecek olursa bu sayı 15 x n olacaktır. Yüklenecek birim sayısı n'nin 500 olduğu düşünülürse bu olasılık 15 x 500=7500 olur. Bu durumda bile çok sayıda çözüm oluşmakta ve problemin NP-Hard olduğu Şekil 2.3'de görülmektedir. [3]



Şekil 2.2. İkinci yükleme için uygun noktalar [2]

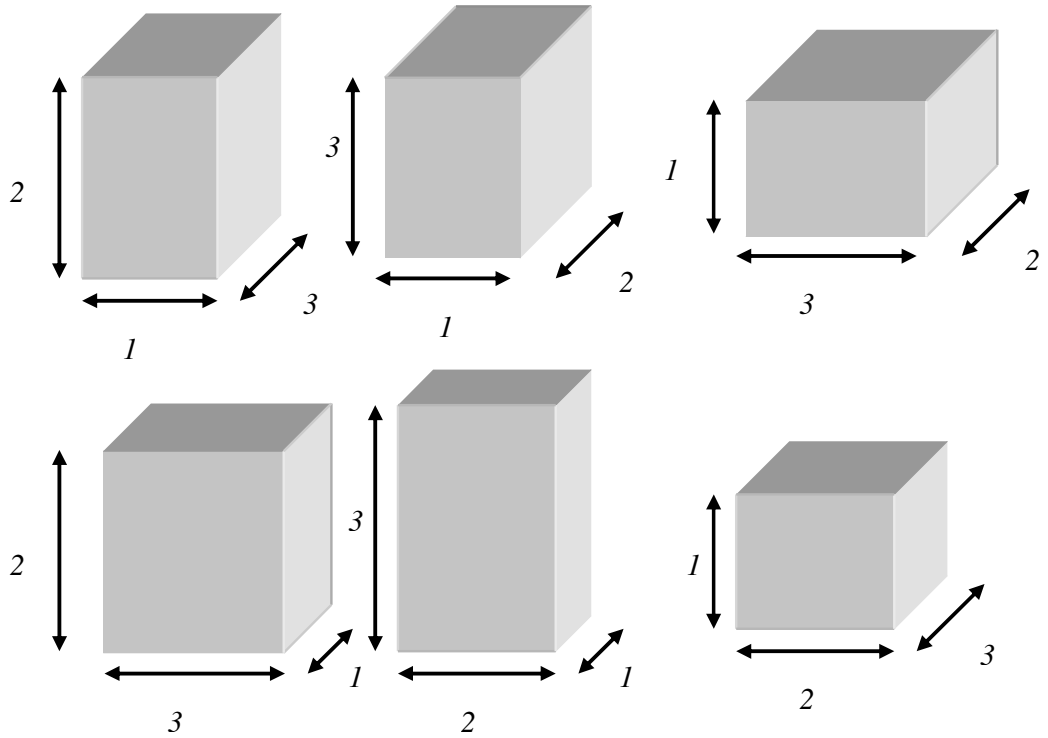


Şekil 2.3. Uygun noktaların ağaç gösterimi [2]

Şekil 2.3'de görüldüğü gibi birinci yüklemede oluşan üç noktadan birine yükleme yapılırsa geriye iki nokta kalır. Bunlara da ikinci yüklemede oluşan noktalar eklenirse her seçenek için toplam 5 adet olasılık oluşur. Problemin bir de her birim için 90 derecelik dönüşlerle 6 pozisyonunun olduğu göz ardı edilmemelidir. Bu durumda çok daha fazla olasılık meydana gelir.

Rotasyon

Rotasyon, yüklenmek için bekleyen birimlerin 90 derecelik dönüşlerle farklı pozisyonlarını ifade etmektedir. Problemde rotasyonun kullanılmasının hem avantajı, hem de dezavantajı vardır. Avantajı; yüklemede daha fazla olasılık meydana getirmesi ve dolayısıyla optimuma en yakın yüklemeyi elde etme olasılığını önemli ölçüde arttırmasıdır. Dezavantajı ise problemin büyüklüğünü ciddi manada arttırmasıdır.



Şekil 2.4. Rotasyon [2]

Her birimin 90 derecelik dönüşlerle altı ayrı pozisyonu olacaktır:

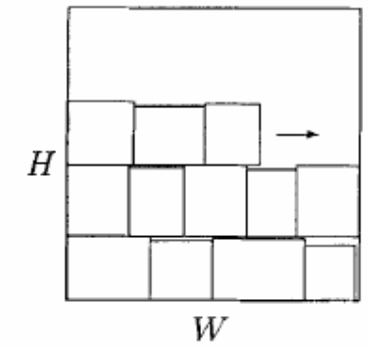
Bir birimin *x, y* ve *z* koordinatlarındaki uzunlukları (*1, 2, 3*) olsun. Bu durumda *3*'ün *1*'li permütasyonu kadar farklı olasılık ortaya çıkacaktır. Bunlar; *(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)* şeklinde olacaktır. Bu durum Şekil 2.4'de gösterilmiştir.

3.LİTERATÜR ARAŞTIRMASI

Üç boyutlu yükleme problemlerinden olan konteynır yükleme problemi ile ilgili literatür tarih sırasına göre verilmiştir.

George ve Robinson (1980) tarafından geliştirilen *Wall Building (Duvar inşa)* yaklaşımı literatürde en çok kullanılan yaklaşımdır. Bu yaklaşımda yükleme, konteynırı derinlik anlamında katlara (layer) ayırarak ya da bir başka deyişle birimlerden katlar oluşturarak gerçekleştirilmektedir [4].

Burada tahmin edileceği gibi katların derinliklerinin seçilmesi en önemli unsurdur. Çünkü kat derinliği katta boş alanları minimize edecek ve aynı zamanda katların konteynıra yüklenmesi sırasında iyi bir performans sağlayacak şekilde seçilmelidir.



Şekil 3.1. George ve Robinson'ın sınıflandırma kuralına göre sıralı yerleşim [4]

George ve Robinson, (1980) kat derinliğini saptamak için bir sınıflandırma kuralı geliştirmiştir. Bu kurala göre yüklenmemiş birimler arasından en küçük boyutu (üç boyut arasından en kısa olanı) en büyük değere sahip birim seçilir. Seçilen bu birim en büyük boyutu derinlik boyutu olacak şekilde döndürülür ve derinlik boyutu kat derinliği olarak belirlenir. Bunun amacı bu birimlerin yüklenmesinin daha sonra zor olacağı, dolayısıyla ilk başta bu birimlerin yüklenmesi ile sıkıntı oluşturacak bir durumdan kurtulmaktır. Kat derinliğinin belirlenmesinin ardından yukarıda bahsedilen sınıflandırma kuralına göre en üst sınıf değerine sahip olan birimden en alt sınıf değerine sahip birime doğru seçilerek, katın yüklenmesi yatay olarak gerçekleştirildiği Şekil 3.1'de görülmektedir [4].

Kat derinliği belirlendikten sonra kat içindeki yüklemelerde mümkün olduğunca aynı tip birim yüklenmeye çalışılır. Daha önce yüklenmiş tip birimin durumu *açık* olarak belirlenir. Yüklemede durumu *açık* olarak belirlenen birimler tercih edilir. Burada akla şu soru gelecektir. Eğer durumu *açık* olan hiçbir tür birim yoksa ne yapılmalıdır? Bu durumda algoritma, önceki sınıflandırma kuralını önermektedir. Yani bütün birimlerin en kısa olan boyutları sıralanarak bu sıralamada en uzun boyuta sahip olan birim türü seçilecektir. Algoritmada bir başka seçim yapılması gereken nokta şudur: Eğer birden fazla *açık* birim türü varsa algoritma hangisini seçmelidir? Bu durum için George ve Robinson (1980) iki ayrı seçim yolu önermektedir. Birinci çözüm, *açık* durumda olan birim tiplerinden yüklenmemiş olan birim sayısının fazla olduğu tipi seçmek, ikinci çözüm ise yine yukarıda kullanılan sınıflandırma kuralını kullanarak en yüksek sınıf değerine sahip olan birimi seçmektir. Katların yüklenmesinden sonra algoritma boş kalan alanları saptayarak birimleri yüklemeye çalışır [4].

Bischoff ve Dowsland, (1982) duvar inşa yaklaşımını kullanan bir başka algoritma geliştirmiştir [5] Bu algoritma, George ve Robinson'ın (1980) geliştirdiği yöntemle temelde aynı özellikleri taşımakla beraber, iki önemli noktada ayrılmaktadır [4,5] Birincisi, katların oluşturulmasında George ve Robinson (1980) farklı tip birimlerin aynı katta bulunmasına izin vermesine karşın; Bischoff ve Dowsland (1982) buna izin vermemiş, katları sadece tek tip birimlerden inşa etme yoluna gitmiştir. İkincisi, her katın inşası iki boyutlu yükleme problemi olarak ele alınmıştır [4,5]. Yani her katın inşasında kullanılan birimlerin derinlikleri aynı olacak ve alt problem olarak iki boyutlu yükleme problemine indirgenecektir.

Burada her katın derinliğinin seçimi çok önemlidir. Yüklenmeye aday olarak seçilen birimin her boyuttaki uzunluğu aday kat derinliği olarak ele alınır. Derinlik seçildikten sonra daha önce seçilen birim tipinden olan birimler kata iki boyutlu yükleme problemine indirgenmiş şekliyle yüklenmeye çalışılır. Bir başka deyişle, konteynırın boyu ve yüksekliği sırasıyla W ve H ile, seçilen birim tipinin en, boy ve yüksekliği de w, l ve h ile gösterildiğinde eğer örneğin w kat derinliği olarak belirlenirse bu birim tipinden n adet $l \times h$ dikdörtgeni $H \times W$ dikdörtgenine

yüklenmeye çalışılacaktır. Eğer birden fazla olası derinlik seçimi bulunuyorsa algoritmanın iki derinlikten birini seçmesi gerekecektir. Bu seçim için bir kriter katın doluluk oranını maksimize eden derinlik seçiminin yapılması olacaktır. Diğer bir kriter de kata yüklenen birim sayısının maksimize edilmesi olarak belirlenmiştir.

Bu durumda her katın boşluk kalmayacak şekilde yüklenmesi yani doluluk oranının %100'e ulaşması amaçlanmaktadır. Fakat doluluk oranının tam olması tahmin edilebileceği gibi her zaman mümkün olmamaktadır. Bu oranın tam olmadığı katlarda yüklenemeyen birimler George ve Robinson (1980) yaklaşımı kullanılarak yüklenmektedir [4].

Bischoff ve Marriot, (1990) George ve Robinson (1980) yaklaşımı ile Bischoff ve Dowsland (1982) yaklaşımını inceleyerek bu iki yaklaşımın avantajları ve dezavantajlarını ortaya koymuştur [4-6]. Ayrıca bu iki yaklaşımda önerilen çözümleri on dört ayrı şekilde çaprazlayarak sonuçları karşılaştırmıştır.

Kat derinliklerinin belirlenmesinde Bischoff ve Dowsland (1982) ve George Robinson (1980) yaklaşımlarından benzer varyasyonlar oluşturulabilir [4,5].

(i) Birimleri en kısa boyutlarının azalan şekilde sınıflandırma.
(ii) Yüklenmek üzere bekleyen birimlerin sayılarına göre azalan şekilde sınıflandırma.
(iii) Birimlerin hacimlerine göre azalan şekilde sıralanması.
(iv) (i)'nin tersi.
(v) (ii)'nin tersi.
(vi) (iii)'nin tersi.

George-Robinson (1980) yaklaşımı A olarak temsil edilsin. İkinci yaklaşım olan Bischoff ve Dowsland (1982) yaklaşımının iki metodu da B1 ve B2 ile temsil edilsin. B1 yaklaşımı katın inşasında alan doluluk oranının maksimize edildiği durumu, B2 yaklaşımı da yine katın inşasında yüklenmek üzere bekleyen birimlerin sayılarını

kullanan yaklaşımdır. Bu durumda Bischoff ve Marriot (1990) ondört ayrı sezgisel belirlemiş ve bunları karşılaştırmıştır [4-6]. Bu sezgiseller;

(1) Sınıflandırma kuralı (i) kullanılarak A metodu
(2) Sınıflandırma kuralı (ii) kullanılarak A metodu
(3) Sınıflandırma kuralı (iii) kullanılarak A metodu
(4) Sınıflandırma kuralı (iv) kullanılarak A metodu
(5) Sınıflandırma kuralı (v) kullanılarak A metodu
(6) Sınıflandırma kuralı (vi) kullanılarak A metodu
(7) Sınıflandırma kuralı (i) kullanılarak B1 metodu
(8) Sınıflandırma kuralını (ii) kullanılarak B1 metodu
(9) Sınıflandırma kuralını (iii) kullanılarak B1 metodu
(10) Sınıflandırma kuralı (i) kullanılarak B2 metodu
(11) Sınıflandırma kuralı (ii) kullanılarak B2 metodu
(12) Sınıflandırma kuralı (iii) kullanılarak B2 metodu
(13) A metodu kullanılarak Rassal olarak belirlenen sınıflandırma kuralı seçilerek algoritmanın 10 kere çalıştırılması sonucu elde edilen en iyi sonuç
(14) A metodu kullanılarak Rassal olarak belirlenen sınıflandırma kuralı seçilerek algoritmanın 100 kere çalıştırılması sonucu elde edilen en iyi sonuç

Üç metodun ve sınıflandırma kurallarının çaprazlanması ile elde edilen bu sonuçlar farklı test problemleri ile denenmiş ve sonuçlar karşılaştırılmıştır. Test problemleri 1,2,3,4,5,10,15 ve 20 farklı tip birim içermektedir. İki farklı tip prosedür kullanılmış ve 1. tip ve ikinci tip olarak isimlendirilmiştir. İki test grubu da toplam yüklenecek kargo hacmi bütün problemlerde aynı ya da birbirine yakın olacak şekilde ayarlanmıştır. 1. tip test grubu, yüklenecek birimlerin toplamının 500 adet olacağı şekilde oluşturulmakta ve buna göre toplam kargo hacmine ulaşmak için her tip birim için boyutlar belirlenmektedir. 2. tip test grubu oluşturulurken önce her tip birim için boyutlar belirlenmekte, sonra toplam kargo hacmine ulaşılacak şekilde her birim türünden kaç adet oluşturulacağı belirlenmektedir. Bu kategoride yüklenecek toplam kaç adet birim olacağı daha önceden bilinmemektedir.

Bütün test problemlerinde konteynırın en ve yüksekliği sabit tutulmuş ve fakat boyu sonsuz kabul edilmiştir. Performans kriteri olarak da minimize edilmeye çalışılan konteynırın yüksekliği belirlenmiştir.

Çizelge 3.1 ve Çizelge 3.2, iki tip test grubu ile 14 farklı sezgisel kullanılarak elde edilen sonuçları özetlemektedir. Her sütun 100 ayrı test problemi kullanılarak ilgili sezgisel ile elde edilen konteynır boyları toplamlarını göstermektedir.

Çizelge 3.1. Bischoff ve Marriot'un 1.test grubu [6]

Sezgisel	Birim tipi sayıları							
	1	2	3	4	5	10	15	20
1	853,5	679,5	541,5	559,5	502	362,5	304	278
2	853,5	834	748	711,5	678	585,5	576,5	439
3	853,5	684,5	557,5	513,5	462,5	332,5	291,5	275
4	853,5	929,5	976	943	1028	1063,5	1087	1117,5
5	853,5	820	821	829	852,5	853	849,5	879,5
6	853,5	969,5	1067,5	1052	1077	1157,5	1133	1170
7	286	273	298,5	291,5	309,5	378	384,5	423
8	286	335,5	472,5	455	460,5	585,5	664	750,5
9	286	274	308,5	312,5	313	409	455	454
10	938	900	941	983	1027	1098,5	1097	1069,5
11	938	949,5	1091	1123,5	1172	1270	1271	1298
12	938	909,5	947	1017	1039,5	1111,5	1144,5	1118,5
13	853,5	1001	930	918	869,5	751	754	720,5
14	853,5	940,5	800	791	709	542	488,5	506

Çizelge 3.2. Bischoff ve Marriot'un 2.test grubu [6]

Sezgisel	Birim tipi sayıları							
	1	2	3	4	5	10	15	20
1	849	566,5	507	515	466,5	338,5	255	220
2	849	692,5	710	714	702,5	573	467	396,5
3	849	582,5	506	457	443	281,5	242	194,5
4	849	887	939,5	1008	981	1046,5	1076,5	1030,5
5	849	807,5	790	804	833,5	816	789	803,5
6	849	919	973	1078	1071,5	1093,5	1115,5	1086
7	293	323,5	283	304	341,5	439	536,5	607
8	293	398,5	447,5	500	502	671,5	759	847,5
9	293	332,5	287	331	361	460,5	543	648,5
10	293	1044	1115,5	1087,5	1114,5	1157,5	1144	1161
11	293	1104	1218,5	1179,5	1228,5	1318,5	1328	1348
12	293	1064	1124	1110	1151	1192	1188	1216
13	849	918,5	864	817	766	674	654	590,5
14	849	860	735	597	537,5	438	402,5	350,5

Bischoff ve Marriot, (1990) bu test sonuçlarından elde edilen verilere göre aşağıdaki çıkarımları yapmıştır [6].

(a) Verilerin elde edilme yollarına göre ikiye ayırdığımız test grupları kullanılarak elde edilen sonuçlar arasında önemsenecek bir farklılık bulunmamaktadır.
(b) Sezgiseller birbirleri ile karşılaştırıldıklarında en iyi sonucun 7. ve 8. sezgiseller ile elde edildiği söylenebilir.
(c) 7,8, ve 9. sezgiseller, kullanılan birim tipi sayısı arttıkça azalan bir grafik oluşturmaktadır. Buna karşılık 1., 2. ve 3. sezgisellerde artan bir grafik söz konusudur.
(d) 4,6,10,11 ve 12. sezgiseller bütün durumlarda zayıf çözümler bulmaktadır.
(e) 13 ve 14. sezgiseller daha uzun çözüm zamanları kullandıkları halde 1. ve 3. sezgisellerden daha iyi sonuçlar elde edememişlerdir.

Ayrıca Bischoff ve Marriot (1990) her iki tip test problemi için Çizelge 3.1 ve Çizelge 3.2'ye benzer bir şekilde her sezgisel için problemi 100 ayrı veri tipi üreterek denemeler yapmış ve Çizelge 3.3 ve Çizelge 3.4 ü elde etmiştir. Her sezgisel için *b* ve *e* olarak temsil edilen iki ayrı sıra mevcuttur. *b* ile gösterilen birinci sıra o sıradaki sezgiselin diğer bütün sezgisellerden daha iyi sonuç verme frekansını göstermektedir. *e* ile gösterilen sıra ise elde edilen en iyi sonuca ulaşma frekansını vermektedir [6].

Çizelge 3.3. Bischoff ve Marriot'un 1. tip denemeleri [6]

Sezgisel	Birim tipi sayıları							
	1	2	3	4	5	10	15	20
1 b	0	0	0	2	2	16	18	24
e	13	13	20	12	22	11	7	4
2 b	0	0	1	5	3	6	8	16
e	13	7	10	5	6	4	0	0
3 b	0	0	1	4	6	23	27	26
e	13	13	21	12	21	10	8	4
4 b	0	0	0	1	0	0	0	0
e	13	2	0	1	0	0	0	0
5 b	0	0	0	3	0	1	1	0
e	13	8	8	0	1	0	0	0
6 b	0	0	0	0	0	0	0	0
e	13	2	0	1	0	0	0	0
7 b	0	2	8	13	9	12	14	9
e	99	65	53	34	41	6	4	3
8 b	0	11	3	13	5	7	3	3
e	99	44	18	3	7	0	0	0
9 b	0	2	1	6	3	10	9	7
e	99	68	56	34	40	6	4	3
10 b	0	0	0	0	0	0	0	0
e	20	9	5	2	2	0	0	0
11 b	0	0	1	0	0	0	0	0
e	20	6	1	1	0	0	0	0
12 b	0	0	0	0	0	0	0	0
e	20	10	6	2	2	0	0	0
13 b	0	0	0	0	1	2	0	2
e	13	3	7	3	2	0	0	0
14 b	0	0	4	1	6	6	8	6
e	13	4	8	4	2	0	1	0

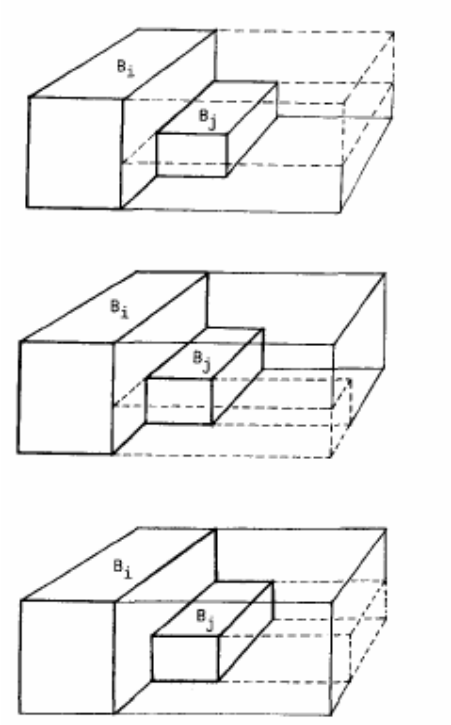
Çizelge 3.4. Bischoff ve Marriot'un 2. tip denemeleri [6]

Sezgisel	Birim tipi sayıları							
	1	2	3	4	5	10	15	20
1 b	0	0	0	0	6	16	26	32
e	16	25	23	20	15	15	15	8
2 b	0	0	1	0	5	7	4	8
e	16	19	10	8	10	1	0	0
3 b	0	0	0	3	7	21	29	44
e	16	25	23	22	16	15	15	7
4 b	0	1	0	0	1	0	0	0
e	16	3	0	2	0	0	0	0
5 b	0	0	2	2	4	1	1	1
e	16	8	4	4	0	0	0	0
6 b	0	0	1	0	0	0	0	0
e	16	2	0	0	0	0	0	0
7 b	0	0	6	7	8	10	3	0
e	99	60	52	42	28	9	3	0
8 b	0	6	6	10	9	5	1	0
e	99	37	5	7	5	1	0	0
9 b	0	1	4	3	6	2	9	1
e	99	55	51	41	27	10	3	0
10 b	0	0	0	0	0	0	0	0
e	17	6	0	0	0	0	0	0
11 b	0	0	0	0	0	0	0	0
e	17	4	0	0	0	0	0	0
12 b	0	0	0	0	0	0	0	0
e	17	5	0	0	0	0	0	0
13 b	0	0	1	0	1	1	2	1
e	16	5	2	2	2	0	0	0
14 b	0	2	2	8	8	12	7	6
e	16	5	2	2	2	0	0	1

Gehring ve ark. (1990) yine duvar inşa yaklaşımını kullanarak bir çözüm sunmuştur. Bu çalışmada konteynırın doluluk oranının dışında ağırlık ve denge unsurları da ele alınmıştır. Fakat bu unsurlar ikinci planda düşünülmüştür. Burada hacim optimizasyonu için getirilen yaklaşım incelenecektir. Algoritma aşağıdaki temeller üzerine kurulmuştur [7].

(1) Bütün birimler öncelikle hacimlerine göre sıralanırlar. En büyük hacimli olan birim listenin en üstünde yer alır. Diğer birimler de hacimlerinin büyüklüklerine göre listede aşağıya doğru sıralanırlar. Her birim yüklendikten sonra bu listeden çıkarılır. En büyük hacimli birimden başlayarak yükleme mantığının daha iyi bir kullanım yüzdesi ile sonuçlanacağı kabul edilmiştir.
(2) Birimler dikey olarak oluşturulan katlar biçiminde yüklenir ve hiçbir birimin aynı anda iki katın üyesi olmasına izin verilmez. Yani birimin bir kısmının bir katın sınırladığı alan içinde, diğer kısmının başka bir katın sınırladığı alan içinde bulunması mümkün değildir.
(3) Her katın derinliği seçilen bir birimin derinliği olarak kabul edilir. Bu birime kat belirleyici birim (LDB) adı verilmiştir. LDB, kata yüklenen ilk birim olup katın sol alt dip köşesine yüklenir. Kata yüklenecek diğer birimler bu birim yüklendikten sonra oluşturulan boş alanlara konumlandırılacaklardır. LDB nin pozisyonu değiştirildiğinde buna bağlı olarak kat derinliğinin de değişeceği göz önünde bulundurulmalıdır.
(4) LDB yüklendikten ve tabi ki kat derinliği belirlendikten sonraki proses katın diğer birimlerle doldurulması olacaktır. Bu proses için her birim yüklendikten sonra boş alanlar oluşturulur. Bu boş alanlar yüklenen birimin sırasıyla önündeki ve üstündeki boş alanlardır. Fakat LDB nin önü bu boş alanlar listesinde bulunmayacaktır. Boş alanlar listesinden seçilen bir boş alana birimler ikili olarak denenir ve en fazla hacim kullanımı sağlayan ikili bu boş alana yüklenir.
(5) Yukarıdaki durumlar için farklı yükleme durumları ortaya çıkacaktır. Her katın LDB si için 6 farklı yükleme pozisyonu olacağına göre her pozisyon için farklı yükleme planları meydana gelir. Algoritma bu yükleme planlarını ortaya çıkartırken bir yandan da elde ettiği sonuçları gösterir. Eğer kullanıcı elde edilen sonucu yeterli bulursa çözüm zamanını kısaltmak adına algoritmayı durdurabilir.

Her kat derinliğinin seçilen bir LDB ile belirlendiği açıklanmıştı. Bunun istisnası son kattır. Son kat derinliği belirlenirken konteynırın geri kalan derinliğine bakılır. Bu derinlik eğer en küçük boyuta sahip birimden kısa ise kat derinliği LDB derinliği olarak değil de geri kalan bütün derinlik olarak kabul edilir. Birimler yüklendikten sonra oluşturulan boş alanlar Şekil 3.2'de görülebilir.



Şekil 3.2. Gehring ve arkadaşlarının yaklaşımında boş alanların gösterimi [6]

Gehring ve ark. (1990) tarafından önerilen bu sezgisel yöntem iki farklı problem seti ile kullanılmıştır. Birinci problem seti 21 adet birimden oluşup birinci tip konteynıra yüklenmeye çalışılmıştır. İkinci problem seti ise 48 adet birimden oluşmuş ve ikinci tip konteynıra yüklenmeye çalışılmıştır. Konteynır tipleri, kullanılan birinci tip veriler ve iki problem seti için elde edilen sonuçlar Çizelge 3.5, 3.6, 3.7 ve 3.8 de görülebilir [7].

Çizelge 3.5. Gehring ve arkadaşlarının kullandığı konteynır tipleri [7]

Konteynır Tipi	Genişlik (m)	Yükseklik (m)	Derinlik (m)	Hacim (m^2)
1.Tip	2,352	2,388	5,899	33,13
2.Tip	2,373	2,405	12,069	68,88

Çizelge 3.6. Gehring ve arkadaşlarının kullandığı birim boyutları [7]

Birim No	Genişlik (m)	Yükseklik (m)	Derinlik (m)	Hacim (m3)
1	0,3	0,35	0,9	0,095
2	0,46	0,66	1,44	0,437
3	0,6	0,66	0,95	0,376
4	0,32	0,33	0,96	0,101
5	0,32	0,33	0,96	0,101
6	0,97	1,03	1,22	1,219
7	0,68	0,68	0,68	0,314
8	0,68	0,68	0,68	0,314
9	0,64	0,64	1,15	0,471
10	1,2	2,2	2,22	5,861
11	0,44	0,55	0,65	0,157
12	0,4	0,55	0,6	0,132
13	0,54	0,64	0,7	0,242
14	0,54	0,8	1,2	0,518
15	1,2	2,2	2,21	5,834
16	1,63	2,05	2,22	7,418
17	0,4	0,69	1,9	0,524
18	0,36	0,69	1,73	0,430
19	1	1,4	1,5	2,100
20	0,62	0,78	0,84	0,406
21	0,5	0,62	0,78	0,242

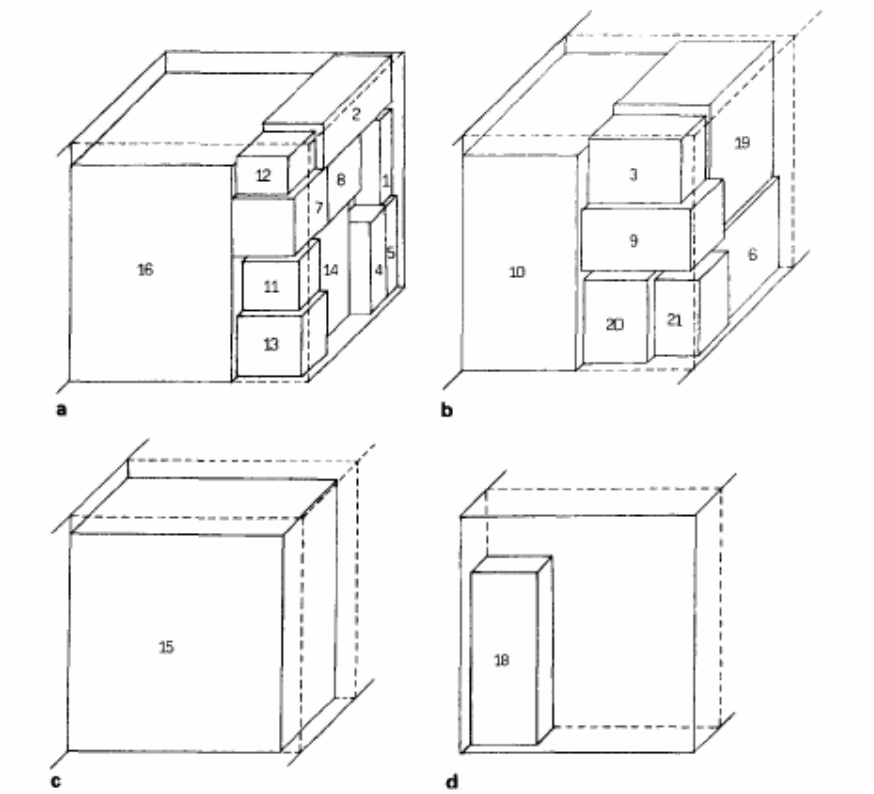
Çizelge 3.7. Gehring ve arkadaşlarının elde ettiği 1.tip konteynır yükleme planları [7]

Yükleme Planı No	Toplam Hacim (m3)	Hacim Kullanımı %	Yüklenen Birimler	Yüklenemeyen Birimler
1	27,29	82,39	21	0
2	26,88	81,13	19	2
3	26,74	80,71	19	2
4	26,98	81,44	20	1
19	21,46	64,78	20	1

Çizelge 3.8. Gehring ve arkadaşlarının elde ettiği 2.tip konteynır yükleme planları [7]

Yükleme Planı No	Toplam Hacim (m3)	Hacim Kullanımı %	Yüklenen Birimler	Yüklenemeyen Birimler
1	51,9	75,43	41	7
2	55,39	80,51	44	4
5	51,52	74,89	40	8
7	53,28	77,44	42	6
8	54,34	78,98	43	5
9	56,45	82,05	45	3

Birinci problem seti ve birinci tip konteynır kullanılarak oluşturulan 1.yükleme planı Şekil 3.3'de üç boyutlu olarak görülebilir.



Şekil 3.3. Gehring ve arkadaşlarının yükleme planları [7]

Han, Knott ve Egbalu, (1989) tek bir çeşit birimin konteynıra yüklenmesi için bir model geliştirilmiştir (Üretici yükleme modeli). Burada yine duvar inşa yaklaşımı kullanılmış olup ve fakat bu duvarların *W* ve/veya *H*'ye bağımlı kalması gerekmediği savunulmaktadır. Yaklaşım, duvarın *L* şeklinde modüllerden oluşturulmasını ve oluşturulan ilk modülün de konteynırın tabanının tümünü kapsayan L modülü olması gerektiğini önermiştir [8].

Mohanty ve ark., (1994) farklı tip birden fazla konteynırlara birimleri boş alanları minimize edecek şekilde yüklemeyi amaçlamışlardır. Bir sütun oluşturma prosedürü önerilmekte ve bu sütunlar daha sonra konteynırlara yüklenmektedir [9].

Chen ve ark. (1995) konteynır yükleme problemine daha önce getirilen sezgisel yaklaşımlardan farklı olarak analitik bir yaklaşım öne sürmüştür. Matematiksel bir

model olarak sunulan bu yaklaşımda optimum çözüme ulaşılmaktadır. Fakat çözüm zamanları çok fazladır [10].

Konteynır yükleme problemi öncelikle belli sayıda ve boyutları birbirinden farklı olabilecek konteynırlara yine boyutları birbirinden farklı olabilecek belli sayıda birimlerin yüklenmesi olarak ele alınmıştır. Birimlerin ve konteynırların en uzun boyutları derinlik olarak, ikinci uzun boyut genişlik ve en kısa boyut da yükseklik olarak ele alınmıştır. Modelde kullanılan parametreler ve değişkenler aşağıdaki gibidir:

N :Yüklenecek olan birimlerin toplam sayısı.

m :Toplam konteynır sayısı.

M :Çok büyük bir sayı.

s_{ij} :İki değer alabilen (0 ve 1) bir değişken. Eğer i. birim j. konteynıra yüklenmiş ise 1, aksi durumda 0 değerini alacaktır.

n_j :İki değer alabilen (0 ve 1) bir değişken. Eğer j. konteynır kullanılmış ise 1, kullanılmamış ise 0 değerini alır.

(p_i, q_i, r_i) :i. birimin derinlik, genişlik ve yükseklik boyutlarını temsil eden değişken.

(L_j, W_j, H_j) :j. konteynırın derinlik, genişlik ve yüksekliklerini temsil eden değişken.

(x_i, y_i, z_i) :i. birimin sol alt dip köşelerinin koordinatlarını gösteren parametreler.

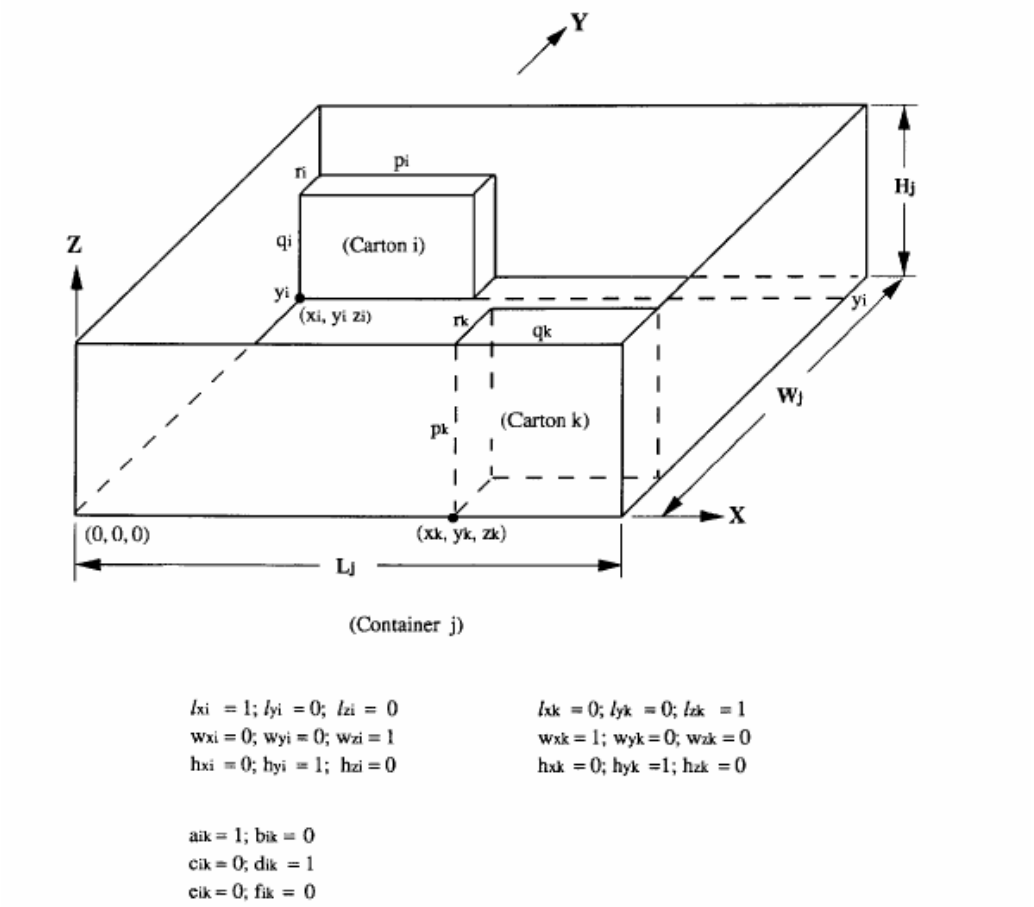
(l_{xi}, l_{yi}, l_{zi}) :i. birimin derinliğinin x, y ve z eksenlerine paralel olup olmadığını gösteren ikili değişkenler. Paralel ise 1, değilse 0 değerini alırlar.

(w_{xi}, w_{yi}, w_{zi}) :i. birimin genişliğinin x, y ve z eksenlerine paralel olup olmadığını gösteren ikili değişkenler. Paralel ise 1, değilse 0 değerini alırlar.

(h_{xi}, h_{yi}, h_{zi}) :i. birimin yüksekliğinin x, y ve z eksenlerine paralel olup olmadığını gösteren ikili değişkenler. Paralel ise 1, değilse 0 değerini alırlar.

Bunların dışında a_{ik}, b_{ik}, c_{ik}, d_{ik}, e_{ik}, f_{ik} değişkenleri birimlerin birbirlerine komşuluklarını belirtirler. a_{ik} değişkeni i. birim k. birimin solunda olduğu durumda 1, tersi durumda ise 0 değerini alır. Benzer şekilde b_{ik}, c_{ik}, d_{ik}, e_{ik}, f_{ik} değişkenleri i.

birim k. birimin sırasıyla sağında, arkasında, önünde, altında ve üstünde yer aldığı durumlarda 1 değerini alacaktır. Bu değişkenler sadece i<k durumları için tanımlanmıştır. Aksi durumda değişkenler gereksiz yere tekrarlanmış olurdu. Chen ve arkadaşlarının modelindeki değişkenlerin gösterimi Şekil 3.4'de görülebilir



Şekil 3.4. Chen ve arkadaşlarının modelindeki değişkenlerin gösterimi [10]

Yukarıda açıklanan değişkenler ile ifade edilen lineer tamsayılı modelin amaç fonksiyonu ve kısıtları şöyledir;

Amaç Fonksiyonu

$$\min\left(\sum_{j=1}^{m} L_j \times W_j \times H_j \times n_j - \sum_{i=1}^{N} p_i \times q_i \times r_i\right)$$

Kısıtlar

1. $x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} + r_i \cdot h_{xi} \le x_k + (1 - a_{ik}) \cdot M$ i<k olan bütün i ve k değerleri için.

2. $x_k + p_k \cdot l_{xk} + q_k \cdot w_{xk} + r_k \cdot h_{xk} \le x_i + (1 - b_{ik}) \cdot M$ i<k olan bütün i ve k değerleri için.

3. $y_i + q_i \cdot w_{yi} + p_i \cdot l_{yi} + r_i \cdot h_{yi} \le y_k + (1 - c_{ik}) \cdot M$ i<k olan bütün i ve k değerleri için.

4. $y_k + q_k \cdot w_{yk} + p_k \cdot l_{yk} + r_k \cdot h_{yk} \le y_i + (1 - d_{ik}) \cdot M$ i<k olan bütün i ve k değerleri için.

5. $z_i + r_i \cdot h_{zi} + q_i \cdot w_{zi} + p_i \cdot l_{zi} \le z_k + (1 - e_{ik}) \cdot M$ i<k olan bütün i ve k değerleri için.

6. $z_k + r_k \cdot h_{zk} + q_k \cdot w_{zk} + p_k \cdot l_{zk} \le z_i + (1 - f_{ik}) \cdot M$ i<k olan bütün i ve k değerleri için.

7. $a_{ik} + b_{ik} + c_{ik} + d_{ik} + e_{ik} + f_{ik} \ge s_{ij} + s_{kj} - 1$ i<k olan bütün i ve k değerleri için.

8. $\sum_{j=1}^{m} s_{ij} = 1$ Bütün i değerleri için.

9. $\sum_{i=1}^{N} s_{ij} \le M \cdot n_j$ Bütün j değerleri için.

10. $x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} + r_i \cdot h_{xi} \le L_j + (1 - s_{ij}) \cdot M$ Bütün i ve j değerleri için.

11. $y_i + q_i \cdot w_{yi} + p_i \cdot l_{xi} + r_i \cdot h_{yi} \le W_j + (1 - s_{ij}) \cdot M$ Bütün i ve j değerleri için.

12. $z_i + r_i \cdot h_{zi} + q_i \cdot w_{zi} + p_i \cdot l_{zi} \le H_j + (1 - s_{ij}) \cdot M$ Bütün i ve j değerleri için.

13. $l_{xi}, l_{yi}, l_{zi}, w_{xi}, w_{yi}, w_{zi}, h_{xi}, h_{yi}, h_{zi}, a_{ik}, b_{ik}, c_{ik}, d_{ik}, e_{ik}, f_{ik}, s_{ij}, n_j = 0$ veya 1 Bütün i,j,k değerleri için.

14. $x_i, y_i, z_i \ge 0$ Bütün i değerleri için.

Modelin amaç fonksiyonu konteynırlardaki toplam kullanılmayan alanları minimize etmektir. 1-6 kısıtları birimlerin birbirlerinin içine girmeleri durumunu

engellemektedir. Bu durum 7. kısıt ile belirtildiği gibi sadece aynı konteynıra yüklenen birimler için geçerli olacaktır. 8. kısıt her birimin sadece tek konteynıra yüklenmesini sağlamaktadır. Eğer bir birim bir konteynıra yüklenirse bu konteynır kullanılmış olarak ele alınır. Bu durum da 9. kısıt ile ifade edilmektedir. 10-12 kısıtları birimlerin yüklemelerinin konteynır boyutlarını aşmayacak şekilde gerçekleştirilmesini sağlamaktadır.

Chen ve ark. (1995) üç adet farklı boyutlara sahip konteynır ve 6 adet yine farklı boyutlarda birim kullanarak bir örnek problemi LINGO paket programı ile çözmüştür. Bu kadar küçük bir problem boyutu ile çözüm zamanı 15 dakika olmuştur [10]

Problemi bir boyutun minimizasyonu olarak da düşünen Chen ve ark. (1995) küçük boyutlu bir problemin çözümünü sunmuştur. Bu problemde genişlik ve yüksekliği 20 ve 10 olarak kabul edilen tek bir konteynıra ve boyutları aşağıda verilen 6 adet birim yüklenmiş ve en küçük derinlik elde edilmeye çalışılmıştır. Çözümde bu derinlik 35 olarak bulunmuştur. Bu çözümde birimlerin rotasyonuna izin verilmemiştir.. Chen ve arkadaşlarının kullandığı birimler Çizelge 3.9'da görülebilir. Chen ve arkadaşlarının optimum yerleşimi Çizelge 3.10'da görülebilir [10].

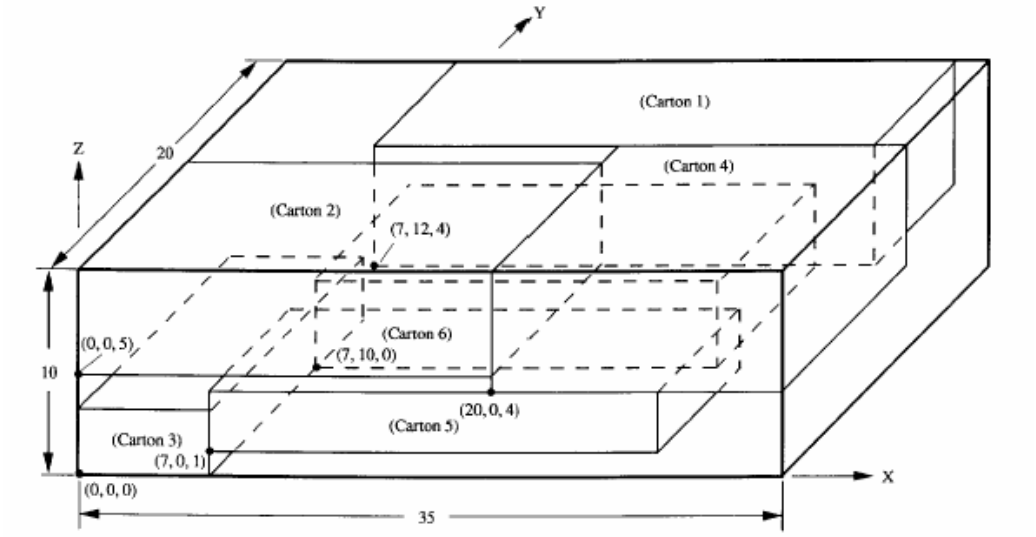
Çizelge 3.9. Chen ve arkadaşlarının kullandığı birimler [10]

Birim No	Yükseklik	Derinlik	Genişlik
1	6	25	8
2	5	20	10
3	3	16	7
4	6	15	12
5	3	22	8
6	4	20	10

Bu çözümden elde edilen sonuçlar da Çizelge 3.10'da gösterilmiştir.

Çizelge 3.10. Chen ve arkadaşlarının elde ettiği yerleşim planı [10]

Birim No	Birimlerin Koordinatları		
i	X_i	Y_i	Z_i
1	7	12	4
2	0	0	5
3	0	0	0
4	20	0	4
5	7	0	1
6	7	10	0



Şekil 3.5. Chen ve arkadaşlarının optimum yerleşimi [10]

Bischoff ve ark., (1995) bir konteynıra farklı boyutlarda birimleri yüklemek için üç farklı seviye algoritması geliştirmiştir. Bu algoritmalardan ilki her seviyede en az iki farklı boyut tipinde birim kullanılarak, ikincisi her seviyede aynı yüksekliğe sahip en fazla iki tip birim kullanılarak, üçüncüsü ise her seviyede tek tip birim kullanılarak oluşturulmuştur. Birim seçimlerinde ise kullanılan alanın en iyi olduğu birimler seçilmektedir. En iyi sonucun birinci algoritma ile elde edildiği görülmüştür. Fakat birim tiplerinin arttığı durumda algoritmanın performansı buna bağlı olarak düşmektedir [11].

Terno ve ark., (1995) farklı bir sezgisel algoritma geliştirmiştir. Üç boyutun yanı sıra ağırlık kısıtı ve ağırlık dengelenmesini de ele almışlardır. Bu çalışmada da bir seviye algoritması kullanılmış ve bu algoritmada dal-sınır algoritması uygulanmıştır[12]. Bischoff ark. (1995) de kullanılan veri setleri kullanılmış ve daha iyi bir sonuç elde edilmiştir. Fakat bu model de *üreticinin yükleme problemi* olarak ele alınmış olduğundan birim çeşidi arttıkça buna bağlı olarak sonuç kalitesi de düşmektedir [11].

Osogami ve Okano, (1999) bir yerel arama algoritması geliştirmiştir. Burada komşuluklar kullanılarak optimum değere yaklaşılmaya çalışılmıştır. Yükleme problemi için daha önce yapılmış yöntemler karşılaştırılmıştır [13].

Martello ve ark., (2000) üç boyutlu yükleme problemi için bir dal sınır algoritması geliştirmiştir. Bu çalışmada n adet birim minimum sayıda alt konteynırlara yüklenmeye çalışılmıştır (Bin Packing Problem). n değeri 20 ve 30 dan küçük alınarak algoritma test amaçlı çözülmüştür. Fakat çalışmada birimlerin rotasyonuna izin verilmemiştir [3].

Faina, (2000) üç boyutlu yükleme probleminin genel çözümü için bir geometrik model geliştirmiştir. Burada birimlerin rotasyonlarına izin verilmiş, probleme kısıtların eklenmesi ile problemin daha kötü sonuçlar vereceği savunulmuştur. Fakat bu görüş kanıtlanmamıştır. Problem için bir tavlama benzetimi algoritması geliştirilmiştir. Problem çözümünün minimum atıl alan çözümünü vereceği garanti edilmemektedir. Algoritma, ilk birimin konteynırın merkezine yüklenmesiyle başlanmaktadır. Bu uygulamada birim sayısı 70 gibi bir sayının üzerine çıktığında algoritmanın getirdiği çözümler kabul edilebilir çözümlerin altında kalmaktadır [14].

Ballew, (2000) Chen ve ark. (1995) çalışmasına yakın bir çalışma olarak matematiksel bir formulasyon geliştirmiştir. Bu çalışma gerçekçi rakamlarda bir yüklemeyi desteklememektedir. Birim sayısı arttıkça değişkenlerin sayısı çok fazla artış göstermekte ve bu da problemin çözümünü çok zorlamaktadır [1,10].

Pisinger, (2002) duvar inşa metodunu genişleterek ağaç metodu ile kat derinlikleri ve katların içine yüklenmek üzere oluşturulan sütunları (strip) optimum duruma getirmeye çalışan bir sezgisel geliştirmiştir. Problemi daha kısa sürede ve daha ekonomik çözebilmek adına, bu ağaç yapısının çözümünde dallanma noktalarında bazı kısıtlamalar kullanılmıştır [15].

Bu nedenle, her dallanma noktasında özelleştirilmiş bir sınıflandırma kuralı uygulanarak, M_1 adet kat derinliği seçilmiş ve bu derinlikler kullanılarak elde edilen sonuçlar denenmiştir. M_1 adet kat derinliği birimlerden oluşturulan sütunlar kullanılarak doldurulmuştur. Burada oluşturulan sütunlar dikey veya yatay olarak yüklenmesine imkan tanınmıştır. Ayrıca oluşturulan M_2 adet farklı sütun genişlikleri (yüklemeye bağlı olarak yükseklikleri) M_1 adet dallanma noktasından çıkarılarak alt dallanma noktaları oluşturulmuştur.

Kat derinlikleri ve sütun genişlikleri (veya yükseklikleri) için bazı sınıflandırma kuralları geliştirilmiştir. Bu sınıflandırma kuralları, yüklenmek üzere bekleyen birimlerin istatistik verilerini kullanarak farklı derinlik ve sütun genişlikleri oluşturmaktadır. Bu durumda α ve β yüklenecek birimlerin en kısa ve en uzun boyutlarını, w, h ve d de sırasıyla genişlik, yükseklik ve derinlik boyutlarını göstermek üzere üç adet sınıflandırma kuralı geliştirilmiştir.

1- k adet birim yüklenmek üzere beklediğinde f^1 frekans fonksiyonu boyutun k adet birimde kaç kere tekrarlandığını göstermektedir.

$$f_k^1 = \sum_{i=1}^{n} 1_{(w_i = k \vee h_i = k \vee d_i = k)} \quad k = \alpha, \ldots\ldots, \beta \tag{3.1}$$

2- İkinci frekans fonksiyonu olan f^2 sadece birimlerin en büyük boyutlarını ele alarak boyutun kaç kere tekrarlandığını göstermektedir.

$$f_k^2 = \sum_{i=1}^{n} 1_{(\max\{w_i, h_i, d_i\}=k)} \qquad k=\alpha,.....,\beta \tag{3.2}$$

3-Son olarak f^3 frekans fonksiyonu sadece birimlerin en küçük boyutlarını ele alarak boyutun kaç kere tekrarlandığını göstermektedir.

$$f_k^3 = \sum_{i=1}^{n} 1_{(\min\{w_i, h_i, d_i\}=k)} \qquad k=\alpha,.....,\beta \tag{3.3}$$

Yukarıdaki üç frekans fonksiyonu i. birimin genişlik ve yüksekliklerini göz önüne alarak M_2 adet sütun çeşitlemelerini sınıflandırmak amacıyla kullanılmaktadır. Bu tarama kat derinlik uzunluğunun sabitlendiği, aynı zamanda birimin derinliğinin mevcut kat derinliğine uyduğu ve en iyi genişlik veya yükseklik uzunluğunun araştırılması için geçerlidir.

Bu üç frekans fonksiyonuna bağımlı olarak bazı öncelik kuralları geliştirilmiştir. Bu kurallar birimlerin en uzun boyutlu olanlarına (büyük birimlerin yüklemelerinin daha önce gerçekleştirilmesi adına) veya en çok tekrarlanan boyutlara (sütunların homojen bir yapıda oluşturulması adına) öncelik verilerek kısıtlar oluşturmaktadır.

(a) En uzun boyutu seç

(b) Artan frekansa göre en uzun boyutu seç. ($f_k \geq 1$ olan en büyük boyutu seç. Daha sonra $f_{k'} \geq f_k$ olan en büyük boyutu seç. Bu prosesi M adet boyut için uygula

(c) $f_k \geq 1$ olan en büyük boyutu seç. Daha sonra $f_{k'} \geq 2$ olan en büyük boyutu seç. Bu durumda seçilen i. Boyut $f_k \geq i$ frekansına sahip olacaktır

(d) $f_k \geq 1$ olan en büyük boyutu seç. Daha sonra $f_{k'} \geq 4i/M$ olan en büyük boyutu seç. Burada M seçilecek boyut sayısıdır. Bunun (c) den farkı daha az aralıklarla frekansları taramaktır

(e) En çok tekrar eden boyutu seç

(f) Bu öncelik kuralı (b) ile aynıdır. Fakat M adet boyuttan daha az seçim durumunda (a) öncelik kuralına başvurulmaktadır

(g) Bu öncelik kuralı (c) ile aynıdır. Fakat M adet boyuttan daha az seçim durumunda (a) öncelik kuralına başvurulmaktadır

(h) Bu öncelik kuralı (b) ile aynıdır. Fakat geri kalan seçimler için (e) öncelik kuralına başvurulmaktadır

(i) Bu öncelik kuralı (d) ile aynıdır. Fakat geri kalan seçimler için (e) öncelik kuralına başvurulmaktadır

Bu öncelik kuralları kullanılarak algoritmalar test edilmiştir. Veri örnekleri birimlerin toplam hacminin konteynırın hacmine yakın olacağı şekilde oluşturulmuştur. Aşağıdaki tablo kısmen heterojen bir veri örneğini göstermektedir. Bu veri örneğinde rassal olarak oluşturulan birimlerin toplam hacmi konteynırın hacminin %91,8 idir. Kullanılan konteynırın genişlik ve yüksekliği 230 cm, derinliği de 590 cm olarak belirlenmiştir. Pisinger'ın (2002) kullandığı birimler Çizelge 3.11'de görülebilir [15].

Çizelge 3.11. Pisinger'ın kullandığı birimler [15]

Birim Tipi	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Genişlik	85	36	68	71	28	109	94	32	52	29	42	97	106	105	107	39	29	31	39	101
Yükseklik	60	69	26	100	31	29	52	106	115	68	41	59	53	68	43	62	56	47	41	38
Derinlik	26	28	46	100	42	51	43	113	45	44	115	48	40	32	110	80	73	42	114	40
Birim Sayısı	5	7	6	5	6	8	9	6	13	9	12	8	5	3	3	6	7	10	8	10

Çizelge 3.12, 3.13 ve 3.14 frekans fonksiyonları ve öncelik kurallarına göre elde edilen sonuçları göstermektedir. Bu test üretilen 100 ayrı veri örneği kullanılarak elde edilen sonuçların ortalamalarını göstermektedir. Birinci sütun frekans fonksiyonunu, ikinci sütun da öncelik kuralını göstermektedir.

Çizelge 3.12. Pisinger'ın kısmen heterojen yerleşimi [15]

Kısmen Heterojen

Frekans Fonk.	Öncelik Kuralı	Doldurulan Alan	Yüklenemeyen Birimler	Harcanan Zaman (saniye)
	a	90,44	0,6	13,25
	b	90,1	1,1	0,76
	c	90,7	0,3	1,27
	d	90,73	0,2	4,81
f1	e	90,73	0,2	13,39
	f	90,68	0,3	6,1
	g	90,77	0,1	4,6
	h	90,79	0	9,33
	i	90,76	0,1	9,46
	a	90,11	1	13,34
	b	88,75	2,8	0,46
	c	90,11	1,3	1,51
	d	90,36	0,8	2,95
f2	e	90,54	0,6	7,36
	f	90,36	0,9	9,61
		90,42	0,7	8,31
	h	90,52	0,5	7,57
	i	90,51	0,6	9,08
	a	89,86	1,4	81,6
	b	80	2,5	18,6
	c	89,83	1,4	40,04
	d	90,04	1,1	53,96
f3	e	88,86	1,8	102,33
	f	89,91	1,3	80,16
	g	89,95	1,1	73,95
	h	89,99	1,2	76,9
	i	89,99	1,1	79,35

Çizelge 3.13. Pisinger'ın heterojen yerleşimi [15]

		Heterojen		
Frekans Fonk.	Öncelik Kuralı	Doldurulan Alan	Yüklenemeyen Birimler	Harcanan Zaman (saniye)
	a	89,44	2,3	14,18
	b	90,34	1,5	0,68
	c	90,58	1,2	0,91
	d	90,87	0,5	2,48
f1	e	90,51	1	88,07
	f	90,87	0,6	8,25
	g	90,88	0,5	6,67
	h	90,98	0,3	24,86
	i	90,97	0,3	23,86
	a	89,42	2,5	13,81
	b	88,5	4	0,1
	c	89,17	3,5	0,28
	d	89,79	2,5	0,82
f2	e	90,77	0,8	39,79
	f	90,18	1,8	11,86
	g	90,34	1,8	11,58
	h	90,86	0,6	29,03
	i	90,82	0,7	26,41
	a	90,55	1	77,44
	b	89,17	3	2,34
	c	89,34	2,7	3,36
	d	89,93	1,8	11,23
f3	e	83,72	3,3	119,43
	f	90,51	1,1	86,27
	g	90,46	1,2	88,28
	h	90,11	1,4	105,06
	i	90,19	1,5	102,93

Çizelge 3.14. Pisinger'ın homojen yerleşimi [15]

		Homojen		
Frekans Fonk.	Öncelik Kuralı	Doldurulan Alan	Yüklenemeyen Birimler	Harcanan Zaman (saniye)
	a	83,79	5,9	35,38
	b	79,17	9,5	1,57
	c	83,79	5,9	35,37
	d	83,79	5,9	35,56
f1	e	83,74	6,1	35,95
	f	83,79	5,9	35,41
	g	83,79	5,9	35,39
	h	83,79	5,9	35,41
	i	83,79	5,9	35,61
	a	78,89	10,7	0,26
	b	78,89	10,7	0,26
	c	78,89	10,7	0,26
	d	78,89	10,7	0,26
f2	e	78,89	10,7	0,26
	f	78,89	10,7	0,26
	g	78,89	10,7	0,26
	h	78,89	10,7	0,26
	i	78,89	10,7	0,26
	a	75,66	17,6	0,16
	b	75,66	17,6	0,16
	c	75,66	17,6	0,16
	d	75,66	17,6	0,16
f3	e	75,66	17,6	0,16
	f	75,66	17,6	0,16
	g	75,66	17,6	0,16
	h	75,66	17,6	0,16
	i	75,66	17,6	0,16

Baltacıoğlu, (2001) birimleri n adet konteynırlara yükleme problemine (bin packing problem) yine duvar inşa yaklaşımı ile farklı bir çözüm getirmeye çalışmıştır. Bu yaklaşımda katlar daha önceki yaklaşımlar gibi sadece derinlik boyutuna dikey olarak değil de üç boyutu da ele alarak oluşturulmaktadır. Çalışmada *layer in layer* olarak adlandırılan bir metot ile yüklemesi tamamlanan kat içinde boşluklar bulunması durumunda bu katın içinde bir alt kat oluşturularak yüklemenin verimliliğinin arttırılması yoluna gidilmiştir [16].

Kullanılan bir diğer metotta yüklenmek üzere seçilecek olan birimlerin insan zekasının yüklemedeki davranışı taklit edilerek seçilmesi, algoritmanın bu yönde yüklemenin gerçekleştirmesi sağlanmıştır. Burada yüklemedeki insan davranışının yükleme için seçilen birimlerin yüzeyi mümkün olduğunca düzlemeye çalıştığı düşünülerek algoritmada da önce boşlukların boyutları incelenmekte ve daha sonra birimler taranarak en uygun birim seçilmektedir.

Kat kalınlıkları da yüklenmek üzere beklemekte olan birimlerin boyutlarından elde edilen istatistikler kullanılarak belirlenmektedir. Ancak belirlenen bu kalınlık değerleri esnek olup daha verimli bir yükleme için daha sonraki proseslerde arttırılabilmektedir.

Algoritma farklı veri örnekleri kullanılarak test edilmiştir. Ayrıca daha önce yapılan çalışmalar ile de elde edilen sonuçlar karşılaştırılmaktadır. Çizelge 3.15'de elde edilen sonuçlar görülmektedir.

Çizelge 3.15. Baltacıoğlu'nun elde ettiği sonuçlar [16]

Birim Setleri	Birim Sayısı	Birim Tipi Sayısı	Alan Kullanımı(%)	Harcanan Zaman(sn)
Set 1	307	5	89,5	2
Set 2	1728	5	97,5	189
Set 3	637	11	92,4	44
Set 4	1493	21	96,4	255
En Kötü Durum	31	31	68,7	<1

Bischoff ve Ratcliff'in (1995) kullandığı veri örnekleri kullanılarak elde edilen sonuçların Baltacıoğlu'nun (2001) sonuçları ile karşılaştırılması Çizelge 3.16'da verilmiştir [15,16]. Burada Bischoff ve Ratcliff (1995) B/R ile, Gehring ve Bortfeld (1997) G/B ile, Terno ve ark. (1995) T/S/S/R ile Baltacıoğlu (2001) B ile temsil edilmiştir [12,16-18].

Çizelge 3.16. BR problem seti sonuçları karşılaştırması [17]

Birim	B/R			G/B			B/G			T/S/S/R			B		
Setleri	min	ort	maks	min	ort	maks	min	ort	maks	min	ort	maks	min	ort	maks
BR-3	73,7	85,4	94,4	76,7	85,8	94,3	78,7	89	95,7	75,7	89,9	95,9	78,9	89	95,3
BR-5	73,8	86,3	93,8	78,4	87,3	95,2	79,7	88,7	95	81,9	89,6	94,7	84,8	89	94
BR-8	75,3	85,9	92,6	81,1	88,1	92,9	82,4	88,2	94	83,2	89,2	93	84,5	88,4	92,1
BR-10	78,4	85,1	90,1	82,7	88	91,6	80,9	87,4	92	83,1	88,9	92,7	84,4	88,2	91,9
BR-20	75,7	83	88,3	84,4	87,7	90,7	79,9	83,9	88,4	80,6	86,3	89	84,3	87,1	90,2

Lodi ve ark. (2002), yükleme problemine iki değişik yaklaşım sunmuştur. Bu yaklaşımlardan biri tabu arama algoritması, diğeri de bir sezgisel algoritmadır. Problemde birimlerin rotasyonlarına izin verilmemiş, birimler yüksekliklerine göre gruplanıp konteynır içinde seviyeler oluşturularak yüksekten düşüğe doğru konteynır tabanından başlanarak çözüme gidilmiştir. Burada gruplama bir β değerine göre yapılmakta, bu değerin 1 olması durumunda gruplama zayıf olacak, 0 olması durumunda ise gruplar çok büyük olacak ve dolayısıyla çözüm çok zorlaşacaktır. β değerini algoritma en uygun şekilde belirlemeye çalışılmaktadır [19].

Yeung ve Tang, (2005) üç boyutlu yükleme problemi için bir hibrit algoritma geliştirmiştir. Algoritma iki ana kısımdan oluşmaktadır. Birinci kısım bir genetik algoritma, ikinci kısım ise sezgisel bir algoritmadır. Üçüncü bir genetik algoritma bu iki algoritmayı birbiriyle ilişkilendirmek için kullanılmıştır. Çözümde birimlerin rotasyonlarına müsaade edilmemiştir. Konteynırın en ve boy oranları sabit, yükseklik sonsuz olarak kabul edilmekte ve en iyi çözüme, yüksekliği en aza indirgeyerek gidilmeye çalışılmaktadır. Çalışmada genetik algoritma, en iyi yüklenmesini sağlayacak birimlerin yükleme sırasını belirlemekte ve bu sırayı sezgisel algoritmaya göndermektedir. Sezgisel algoritma, sıralama verilerini alarak en alt sol köşeden başlayarak yüklemesi gerçekleştirmektedir. Bu yaklaşımda daha önce de bahsettiğimiz konteynır içinde seviyeler oluşturma metodu ile yükleme gerçekleştirilmektedir. Fakat birimlerin belirlenen seviyeleri aşmaları durumuna izin verilmekte, seviyeyi aşan birimleri kesme metodu ile bir sonraki seviyede kalan kısımları hesaplamakta ve bir sonraki seviyenin yüklemesine geçilmektedir. Seviye yükseklikleri ise yükseklikleri birbirlerine yakın olan birimler gruplanarak belirlenmektedir. En büyük yüksekliğe sahip seviye en alta yüklenerek başlanmakta ve konteynırın yukarısına doğru azalarak devam etmektedir [20].

Boschetti, (2003) daha önceden çözülmüş olan yükleme problemine birimleri 90 derece dönebilme özelliğini de ekleyerek yeni yerel çözümler ve buna bağlı olarak optimuma daha yakın çözüm sunmakta. Aynı zamanda algoritmanın hızlanmasını sağlayan yeni bir kısıt eklemektedir [21].

Pımpawat ve Chaıyaratana, (2004) diğer 3d problemlerinden farklı olarak birimleri *küçük boyutlu*, *orta boyutlu* ve *büyük boyutlu* olarak üç gruba ayırmaktadır. Ve problemi de böylelikle daha küçük alt problemlere ayırmakta ve bu alt problemleri çözmek için de bir genetik algoritma kullanmaktadır. Algoritma bu ayırımı yaptıktan sonra seviye mantığıyla işlemektedir [22].

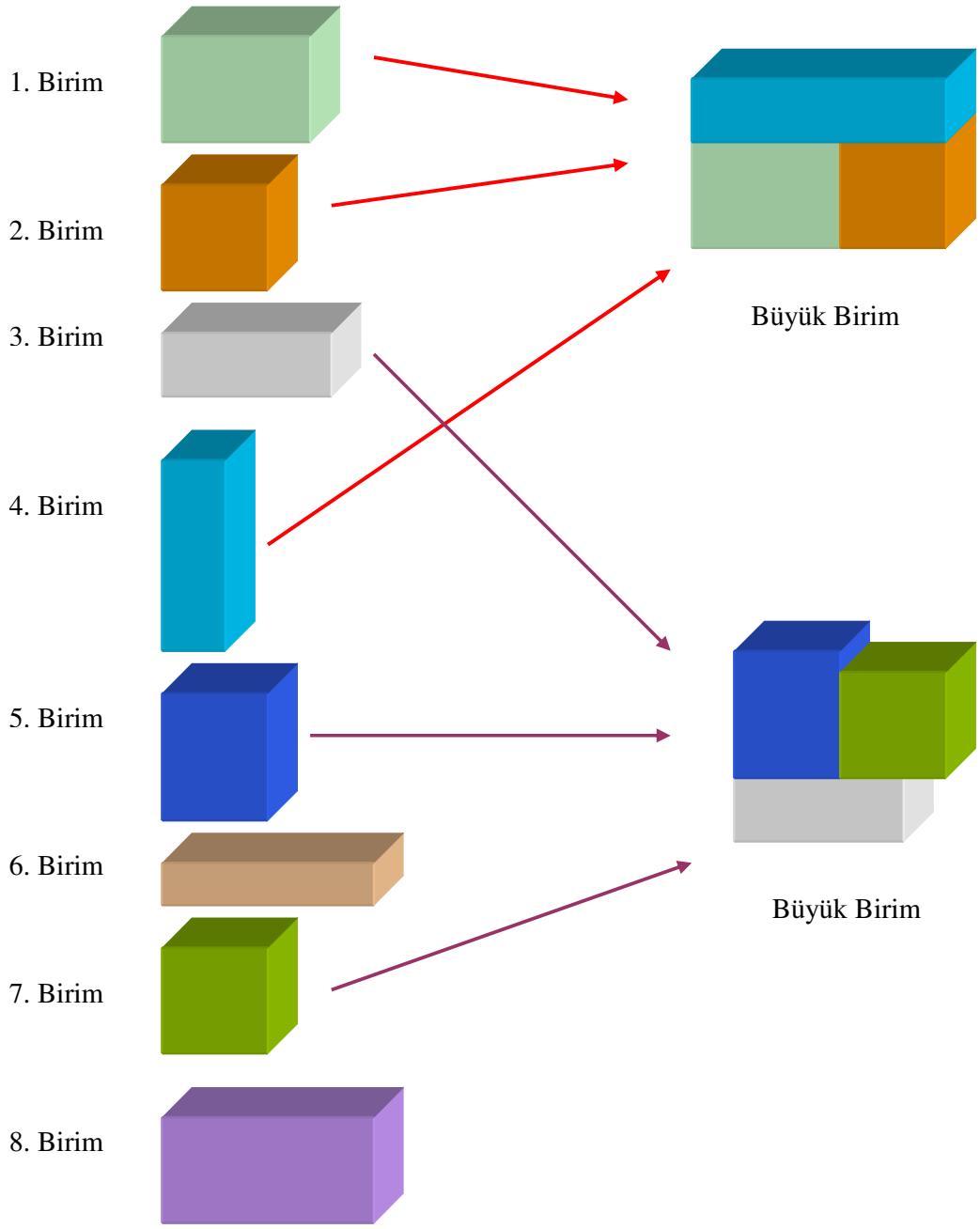
4. PROBLEME YENİ BİR YAKLAŞIM (EE YAKLAŞIMI)

Konteynır yükleme problemine önceki bölümde de açıklandığı gibi çeşitli çözüm yolları sunulmuştur. Bu çalışmada, probleme farklı bir sezgisel yaklaşımla çözüm getirmek amaçlanmıştır. Geliştirilen sezgisel yaklaşım ardarda tamamlanacak 5 prosesten oluşmaktadır.

Büyük birim oluşturma

Yüklenmeyi bekleyen *n* adet birim öncelikle azalan hacimde sıralanmasının ardından bölümlenecektir. Her bölüm *m* adet birim içerecektir.Burada *m* daha önceden belirlenmiş bir sabit sayıdır. Her bölüm için bu *m* adet birimden *büyük birim* olarak adlandırılan en büyük dikdörtgenler prizması oluşturulmaya çalışılacaktır. Burada amaç en büyük dikdörtgenler prizmasına en yakın yüklemeyi sağlamaktır.

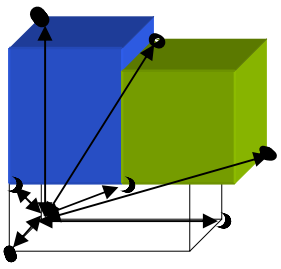
Yukarıda kısaca açıklanan yaklaşım bir örnekle açıklanacak olursa; problemde *n=8* ve *m=3* kabul edilsin. Bu kabul, yüklenmek üzere bekleyen 8 adet birimin olduğunu ve bu sekiz adet birimden üçer üçer seçim yaparak büyük birimler elde edilmeye çalışılacağını anlatmaktadır. Sekiz birim için oluşturulan büyük birimler Şekil 4.1'dedir.



Şekil 4.1. Büyük birim oluşturma [2]

Şekil 4.1'de de açıklandığı gibi ilk deneme birinci, ikinci ve dördüncü birimlerden oluşturulmuş ve bu deneme ile büyük birim meydana getirilmiştir. Burada birinci ve ikinci birimler kendi pozisyonlarında kalmış, üçüncü birimde ise rotasyon gerçekleştirilmiştir. Bu birim tam olarak dikdörtgenler prizmasıdır. Amaç dikdörtgenler prizmasına en yakın birimi elde etmek idi. Bu durumda en iyi durum elde edilmiştir. Bir diğer seçim olan üçüncü, beşinci ve yedinci birim seçiminde oluşturulan büyük birim ise tam olarak dikdörtgenler prizması oluşturulamamış fakat dikdörtgenler prizmasına yakın bir birim elde edilmiştir.

Büyük birimlerin oluşturulması prosesinde ilk seçilen birimden sonra yüklenecek birimler için uygun noktalar belirlenmektedir. Bu uygun noktalar her yüklenen birimin x, y ve z eksenlerindeki komşularıdır. Seçilen birimin yükleneceği nokta bu uygun noktalardan birisi olacaktır. Bu noktalardan hangisinin seçileceğine ise sistem şöyle karar vermektedir: Bütün uygun noktaların ilk seçilen birimin sol alt arka köşesinin bulunduğu noktaya (0,0,0) uzaklıkları hesaplanır ve seçilen birim en kısa uzaklığa sahip noktaya yüklenir (Şekil 4.2). Eğer bu noktaya seçilen birim yüklenemiyorsa başka bir birim seçilir ve bu noktaya yüklenir. Bu noktaya hiçbir birimin yüklenememesi durumunda ise en yakın ikinci noktaya geçilir ve proses böyle devam eder.



Şekil 4.2. Uygun nokta seçimi [2]

Büyük birim oluşturma prosesi bütün birimler bir büyük birimin üyesi oluncaya dek devam eder.

Büyük birimlerin yüklenmesi

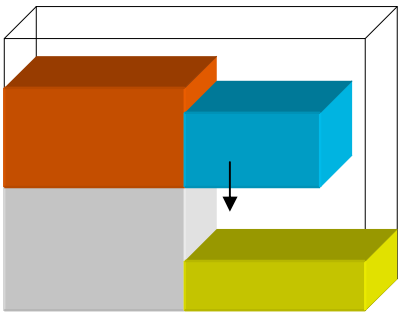
Büyük birimler oluşturulduktan sonraki proses büyük birimlerin konteynıra yüklenmesi prosesidir. Öncelikle oluşturulan büyük birimler dikdörtgenler prizmasına yakınlıklarına göre sıralanırlar. Yüklemeye dikdörtgenler prizmasına en yakın olan büyük birimden başlanır. Büyük birimlerin yüklemesinde bir önceki adımda da uygulanan seçim yaklaşımı kullanılmaktadır. Yani uygun noktalardan orijin noktasına en yakın olan noktalara öncelik verilerek yükleme yapılır. Büyük birimlerin yüklemesi hiçbir büyük birim boş alanlara yüklenemeyecek duruma gelinceye kadar devam edecektir.

Son yükleme prosesi

Büyük birimlerin yüklemesini gerçekleştirdikten sonra açıkta kalan büyük birimler tekrar eski durumlarındaki gibi birimler olarak ele alınırlar. Uygun olan bütün noktalar (boş alanlar) taranarak açıkta kalan birimler son durumda yüklenmeye çalışılır.

Birimleri kaydırma prosesi

Son yükleme prosesinin uygulanmasının ardından birimler sol alt dip köşeye (orijin) tekrar mümkün olduğunca yaklaştırılmaya çalışılır. Bu prosesteki amaç kullanılmayan alanları birleştirebilmek ve açıkta kalan birimler için müsait yükleme alanları oluşturmaktır. Şekil 4.3'de bu proses görsel olarak verilmiştir.

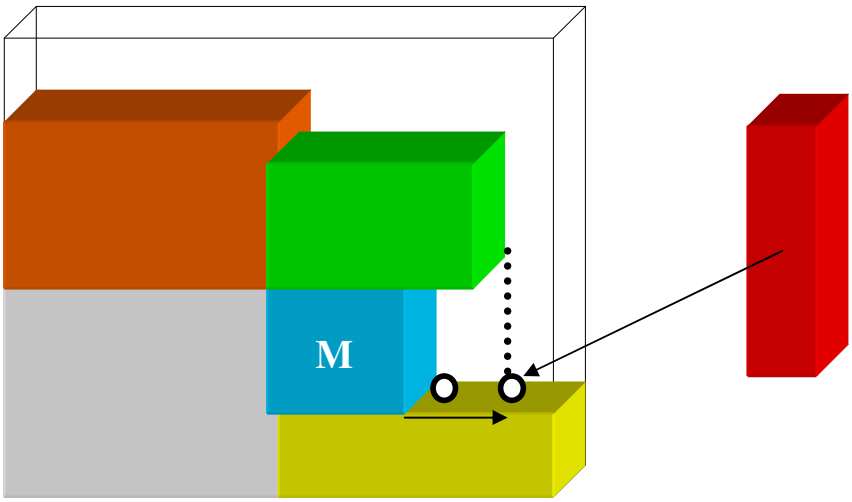


Şekil 4.3. Birimleri kaydırma prosesi [2]

Şekil 4.3'de sağ üst kısımda görünen birim kullanılmayan alanları bölmektedir. Bu durumu engellemek amacı ile birimi y ekseninde aşağıya doğru kaydırmak üst taraftaki kullanılmayan alanı genişletecek ve üst tarafa başka bir birim daha yükleme imkanı sağlamış olacaktır.

Uygun noktaları kaydırma prosesi

Bazı durumlarda birimler yüklendikten sonra açıkta kalan birimlerin yüklenebilmesi için belirlenen uygun noktalar açıkta kalan birimlerin yüklenmesine hacim olarak olanak vermemekte fakat bu uygun noktaların x, y ve z eksenlerindeki komşuları açıkta kalan birimleri kapsayabilmektedir. Şekil 4.4'de bu durum daha iyi görülmektedir. Yuvarlak ile belirtilmiş nokta M harfi ile temsil edilen birim yüklendikten sonra x ekseninde oluşturulan uygun noktadır. Fakat dışarıdaki birimin bu noktaya rotasyon gerçekleştirilerek bile uymayacağı açıktır. Bu durumda bu uygun noktayı sağa kaydırarak belirlenen noktalara bu birimin yüklenmesi denenecektir.



Şekil 4.4. Uygun noktaları kaydırma prosesi [2]

4.1.Algoritma

*Adım 1:*Birim boyutlarını dosyadan oku.

*Adım 2:*Birimleri hacimlerine göre azalan şekilde sırala doğru sırala.

*Adım 3:*Bütün birimleri her bölüm MAX adet birim içerecek şekilde sıralanmış şekilde bölümlere ayır.

*Adım 4:*Sırasıyla bölümlerden büyük birim oluştur. Eğer büyük birimde istenen değerlerden fazla bir kayba (büyük birimi içine alacak dikdörtgenler prizması hacmi ile büyük birimin hacminin farkı) sahipse tekrar dene. Büyük birim oluşturamadığında bir sonraki bölüme geç.

*Adım 4.1:*Bulunduğun bölümden rassal olarak bir birim seç ve bu birimi ilk uygun nokta olan (0,0,0) noktasına yükle ve daha sonra sonraki yükle için uygun olan noktalara yüklenen birimin üç boyutta (yanı,üstü ve önü) komşularını ekle. Belirlediğin tüm uygun noktaları orijin (0,0,0) noktasından uzaklıklarına göre sırala. yüklenmediği durumda diğer bölüme geç.

*Adım 4.2:*Bulunduğun bölümden rassal olarak bir birim seç ve bu birimi bir önceki adımda belirlenen uygun noktalardan baştan başlayarak yüklemeyi dene. Yükleme gerçekleşirse sonraki yükleme için uygun olan noktalara yüklenen birimin üç boyutta (yanı,üstü ve önü) komşularını ekle, belirlediğin tüm uygun noktaları orijin noktasından uzaklıklarına göre sırala, bölümün tüm birimleri yüklenene kadar bu adımı tekrar et. Yükleme gerçekleştirilemezse bir sonraki bölüme geç.

Adım 4.3: Seçilen bölümün bütün birimleri yüklendiyse büyük birim oluşturulmuş anlamına gelir. Oluşturulan büyük birimin kaybını, üyelerini ve büyük birim içersindeki koordinatlarını, uygun noktalarını büyük birime kaydet.

*Adım 5:*Oluşturulan büyük birimleri kaybı en az olandan en fazla olana doğru sırala.

*Adım 6:*Büyük birimlerin yüklemesi için (0,0,0) noktasından başlayarak yüklemeye başla.

*Adım 7:*Her yüklemeden sonra uygun bir sonraki yükleme için uygun noktaları belirle ve bu noktaları (0,0,0) noktasından uzaklıklarına göre sırala.

*Adım 8:*Bir sonraki yüklemesi sıradan uygun nokta seçerek gerçekleştir. Eğer gerçekleşmezse bir sonraki noktayı dene.

*Adım 9:*Yüklenen büyük birimlerin üyeleri olan birimlerin gerçek koordinatlarını bağlı oldukları büyük birimlerin koordinatlarına göre hesapla ve son yükleme için uygun noktaları belirle.

*Adım 10:*Bütün birimleri hacmi en büyük olandan en küçük olana doğru sırala.

*Adım 11:*Sıralanmış olan birimlerden yüklenmemiş olanlarını baştan itibaren sırasıyla seçerek uygun noktaların birine yüklemeyi dene. Eğer yüklenemezse diğer birime geç.

*Adım 12:*Yüklenmemiş olan birimleri mümkün olduğu kadar sol alt dip köşeye yaklaştırmak için boş alan varsa kaydır.

Adım 13: 11. adımı tekrar et.

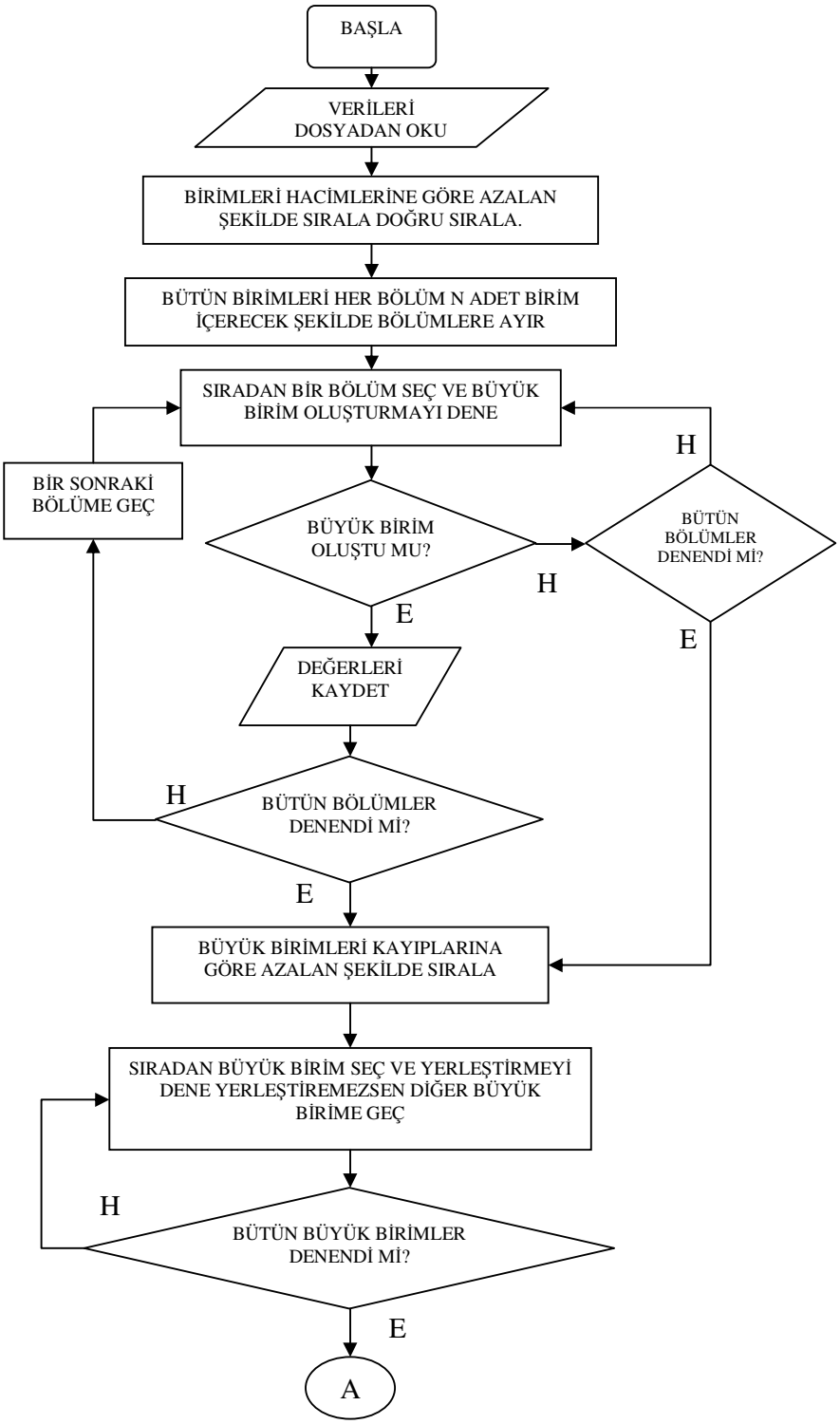
*Adım 14:*12. adımı tekrar et.

*Adım 15:*13.adımı tekrar et.

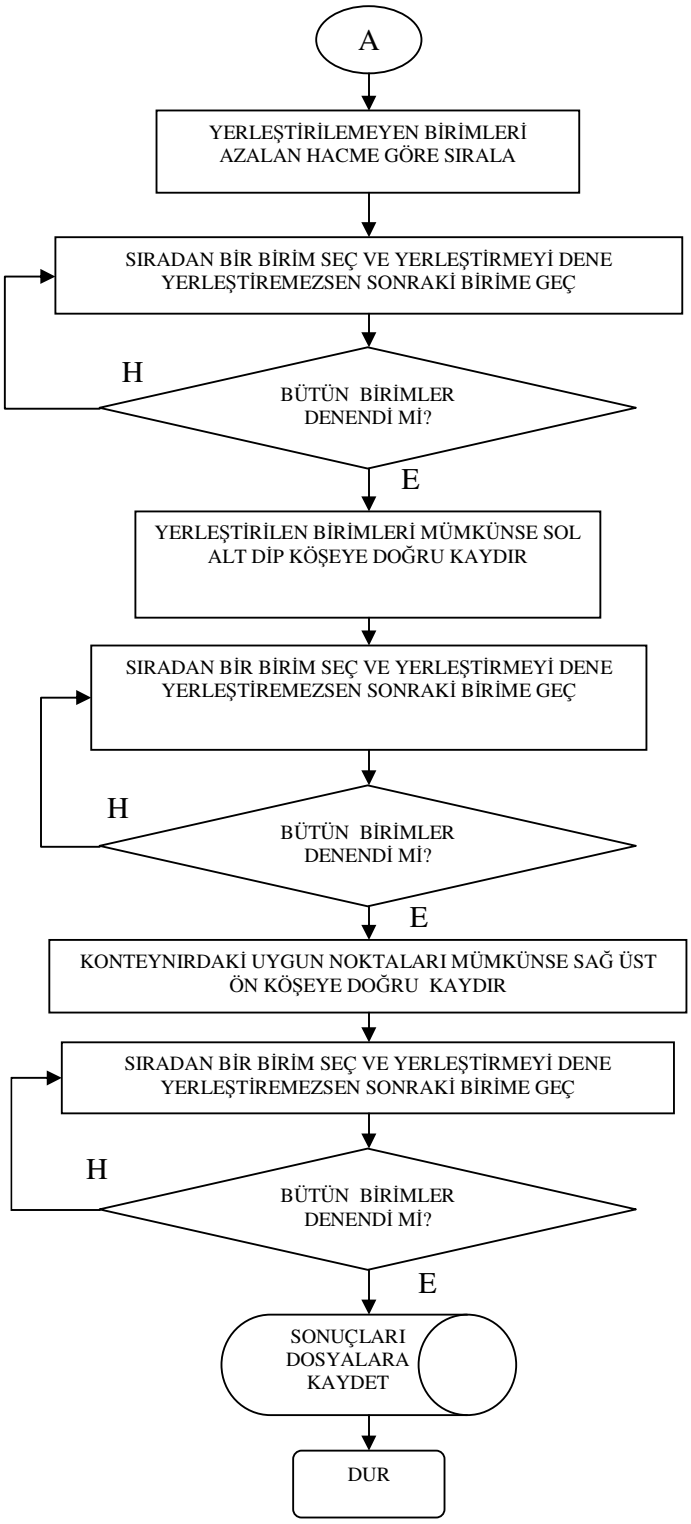
*Adım 16:*Uygun noktaları x, y ve z eksenlerinde kaydırarak boş alanları tekrar tara ve yüklenebilecek birimleri yükle.

*Adım 17:*Elde edilen sonuçları ekrana ve dosyalara yazdır.

4.2.Akış Çizelgesi



Şekil 4.5. Akış çizelgesi [2]



Şekil 4.5. (Devam) Akış çizelgesi [2]

4.3.İstatistiksel Analiz

Problemde farklı birim ve konteynır boyutlarının uzunluk değerleri girilerek çözüm yapıldığında elde edilen verim değerleri karşılaştırıldı. Bu değerler ile kullanılan veriler arasında bir bağıntı olup olmadığı aşağıda açıklanacak olan parametreler yardımıyla araştırıldı. Buna bağlı olarak iki adet bağımsız değişken ve bir adet bağımlı değişken belirlendi. Belirlenen bağımsız değişkenler de *hacimoran* ve *ortkare* olarak isimlendirildi. Yükleme sonucunda elde edilen verim değeri de bağımlı değişken olan *verim* olarak adlandırıldı.

Hacimoran

Bu parametre son durumda konteynıra yüklenen birimlerin ortalama hacmi ile konteynırın hacminin birbirine bölünmesi ile elde edilen orandır. x_n, y_n, z_n n inci birimin x,y ve z boyutlarındaki ebatlarını, N ise konteynıra yüklenen toplam birim sayısını gösterdiğinde hacimoran değişkeni Eş.4.1'deki gibi olacaktır.

$$Hacimoran = \frac{\sum_{n=1}^{N}(x_n \times y_n \times z_n)}{\frac{N}{X_n \times Y_n \times Z_n}} \tag{4.1}$$

Ortkare

Bu parametre her birimin küp şekline olan yakınlığını, bir anlamda ebatlarının birbirinden uzaklığını belirleyen parametredir. Her birim için ayrı ayrı hesaplanır ve bunların ortalaması alınır.

$$a_n = \frac{\min(x_n, y_n)}{\max(x_n, y_n)} \tag{4.2}$$

$$b_n = \frac{\min(x_n, z_n)}{\max(x_n, z_n)} \tag{4.3}$$

$$c_n = \frac{\min(z_n, y_n)}{\max(z_n, y_n)} \tag{4.4}$$

$$Ortkare = \frac{\sum_{n=1}^{N}(a_n + b_n + c_n)}{N} \tag{4.5}$$

şeklinde olacaktır. Eş. 4.2, 4.3, 4.4 ve 4.5'den de anlaşılacağı üzere ortkare değişkeni sıfır ile üç arasında olacaktır.

Verim

Bu bağımlı değişken konteynırın doluluk oranıdır. Konteynıra yüklenen birimlerin hacimleri toplamının konteynırın hacmine oranıdır.

$$Verim = \frac{\sum_{n=1}^{N}(x_n \times y_n \times z_n)}{X_n \times Y_n \times Z_n} \tag{4.6}$$

Program farklı bağımsız değişkenler ile çalıştırıldığında elde edilen verimler arasında SPSS paket programı kullanılarak Eş.5.1'deki istatistiksel bağıntı bulundu.

$$Verim = 101.896 - 5.994 \times \log(Ortkare) - 6.759 \times \log(Hacimoran) \tag{4.7}$$

Elde edilen bu regresyon denklemine göre bir üreticinin ya da dağıtıcının konteynır yüklemesinde verimliliği arttırmak için mümkün olduğunca konteynır boyutuna göre küçük ve aynı zamanda boyutları birbirine yakın olmayan birimler kullanması önerilmektedir.

Çizelge 4.1. Anlamlılık testi [2]

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	,910[a]	,829	,822	6,90823

a. Predictors: (Constant), HCMORAN, ORTKARE

Çizelge 4.1'e göre bağımsız değişkenlerin bağımlı değişkeni açıklama gücü %82,2 dir. Model anlamlı bulunmuştur. (F=128,370) Elde edilen regresyon denklemine göre diğer değişken sabitken Ortkare %1 arttığında verim %5,994 azalmaktadır. Diğer değişken sabitken Hacimoran %1 arttığında verim %6,759 azalmaktadır. Modelin SPSS çıktıları da Çizelge 4.1, 4.2 ve 4.3'de görülmektedir.

Çizelge 4.2. Annova testi [2]

ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	12252,626	2	6126,313	128,370	,000[a]
	Residual	2529,356	53	47,724		
	Total	14781,982	55			

a. Predictors: (Constant), HCMORAN, ORTKARE

b. Dependent Variable: VERIM

Çizelge 4.3. Regresyon [2]

Coefficients[a]

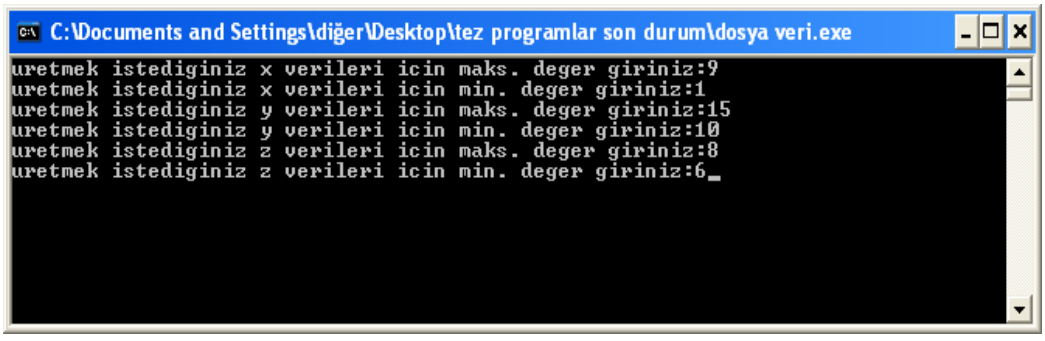
Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	101,896	2,081		48,972	,000
	ORTKARE	-5,994	1,045	-,371	-5,738	,000
	HCMORAN	-6,759	,651	-,672	-10,382	,000

a. Dependent Variable: VERIM

$t_{ortkare}$=-5,738 ve $t_{hacimoran}$=-10,382 olduğundan ortkare ve hacimoran regresyonda bulunması gereken anlamlı değişkenlerdir.

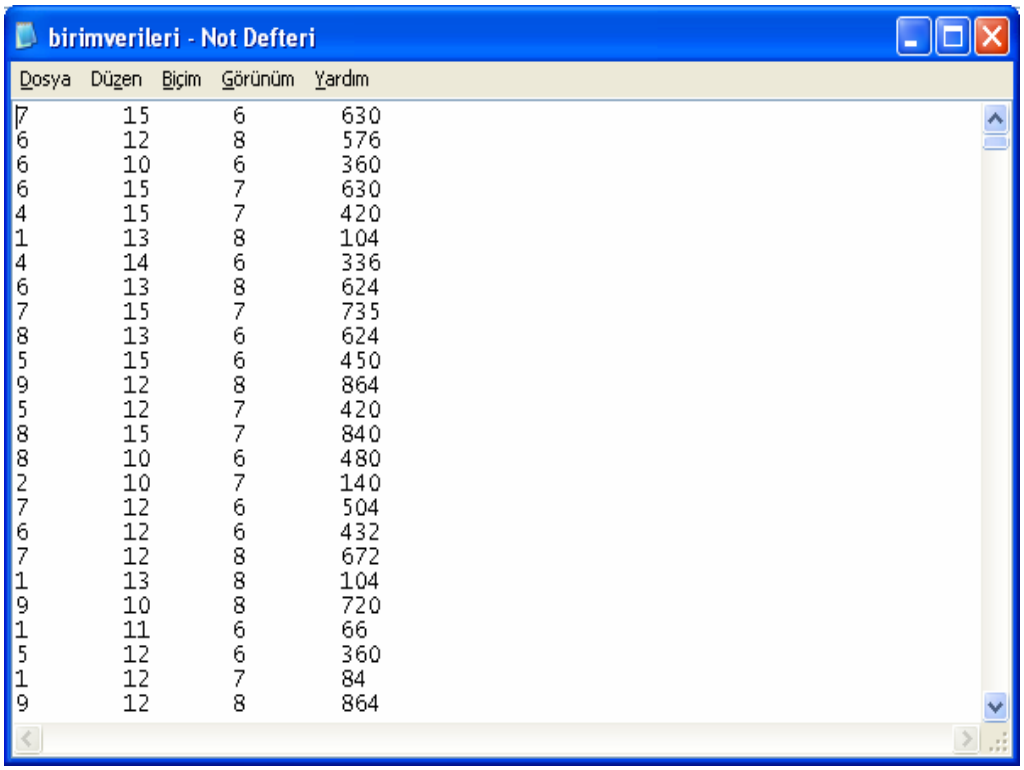
4.4.EE Uygulaması

C programlama diliyle kodlanarak (EK-3) oluşturulan programın çalışabilmesi için yükleme için bekleyen birimlerin boyutlarının programa gösterilmesi gerekmektedir. Program bu verileri *birimverileri.txt* dosyasından okumaktadır. Bu verileri kullanıcı kendisi girebileceği gibi Şekil 4.6'da ekran çıktısı görülen *dosya veri* adında hazırlanan diğer bir program yardımıyla rassal veri de ürettirebilir. *dosya veri.exe* çalıştırıldığında program, üretilmek istenilen birimlerin x,y ve z boyutları için maksimum ve minimum değerleri isteyecektir. Bu değerler girildiğinde *birimverileri.txt* adında bir dosya oluşacaktır.



Şekil 4.6. Verilerin türetilmesi [2]

Eğer programa veriler el yordamıyla girilmek istenirse *birimverileri.txt* adında *txt* uzantılı bir dosya oluşturulup veriler, Şekil 4.7'deki formatta yazılmalıdır. İlk üç sütun birimlerin sırasıyla x,y ve z boyutlarını göstermektedir. Son sütun ise birimlerin hacimlerini göstermektedir. Dosya şekil 4.7'de görülebilir.

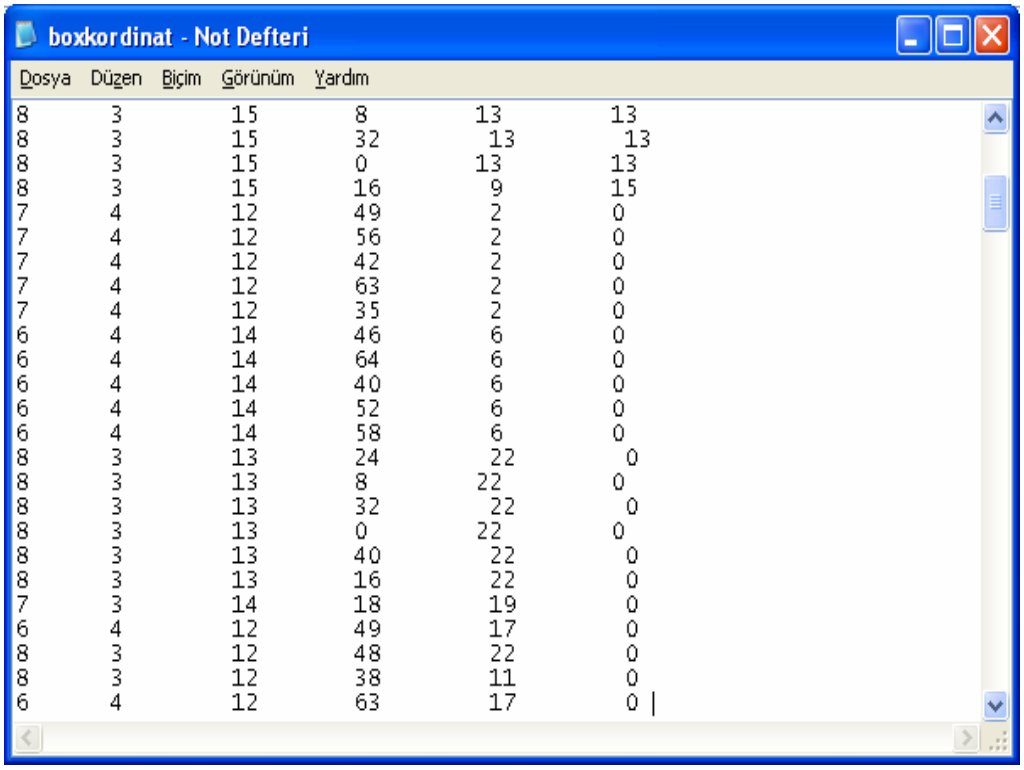


Şekil 4.7. Birim verileri [2]

Bu birimler girildikten veya bilgisayara ürettirildikten sonraki aşama asıl programın çalıştırılması olacaktır. *diagonal.exe* adlı program çalıştırıldığında ekran çıktısı olarak, Şekil 4.8'deki pencere açılacaktır.

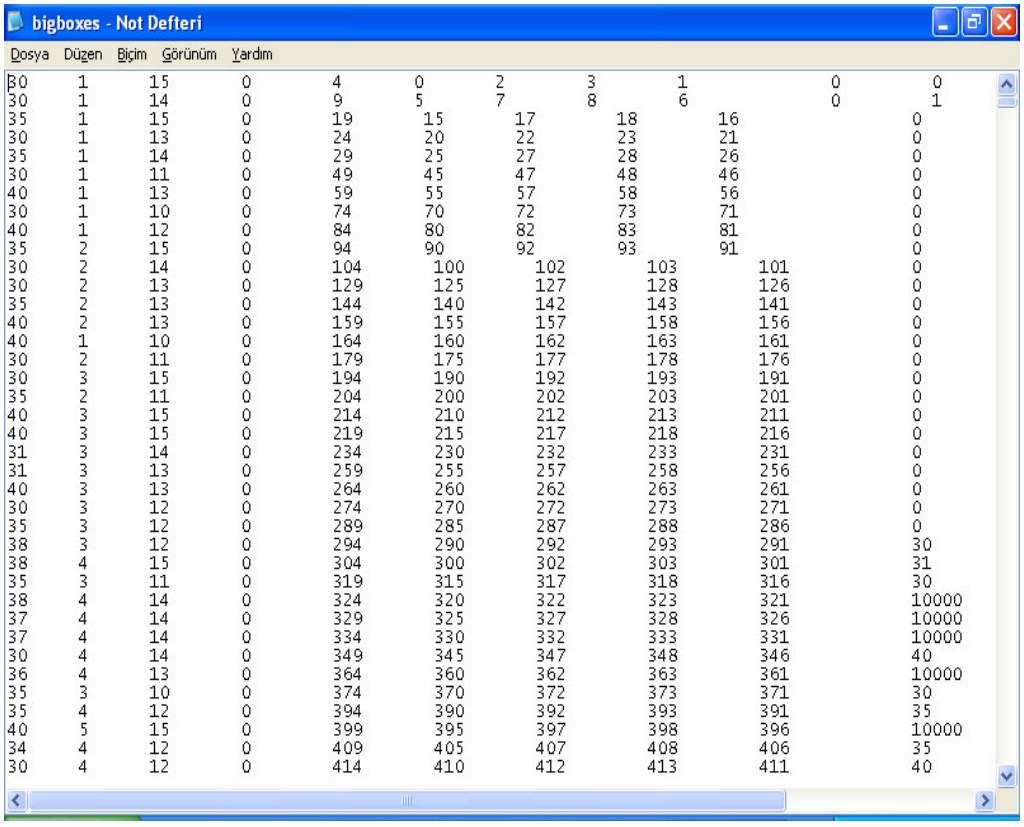
Şekil 4.8. Diagonal [2]

Program Şekil 4.8 deki ekran çıktısının dışında *txt* uzantılı bazı dosyalar oluşturacaktır. *boxkordinat.txt* adındaki dosya yüklenen birimlerin boyutlarını ve koordinatlarını göstermektedir. (Şekil 4.9) İlk üç sütun birimlerin boyutlarını, ikinci üç sütun ise birimlerin yüklenen koordinatlarını göstermektedir.



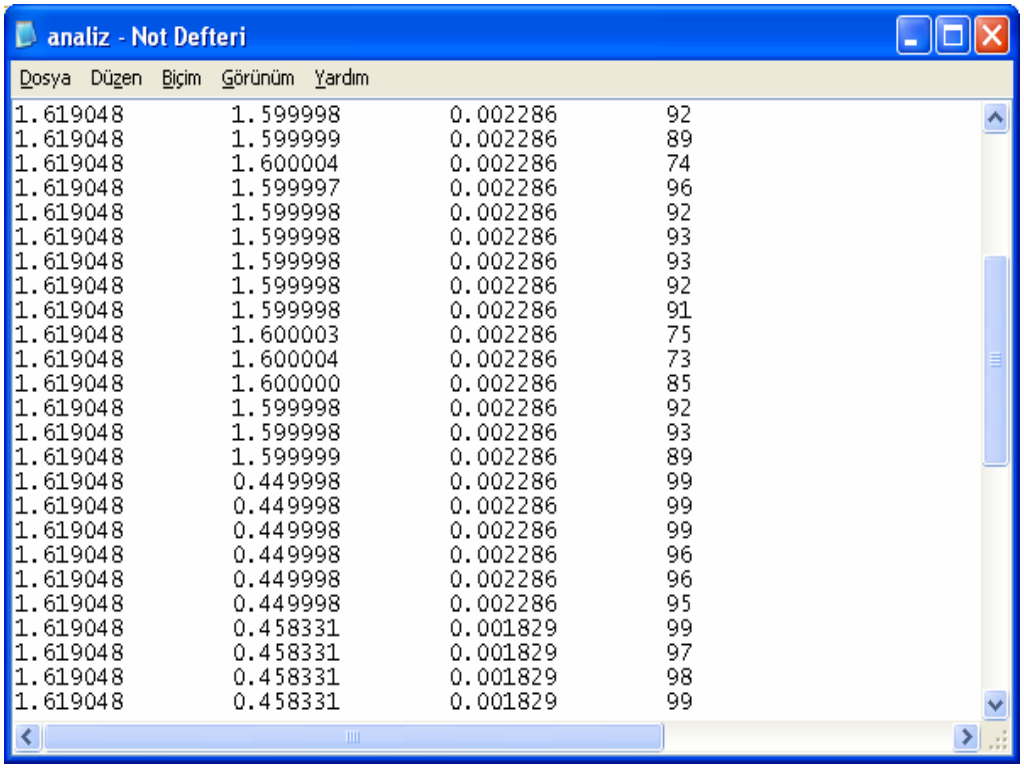
Şekil 4.9. Birim boyutları ve konteynır içerisindeki koordinatları [2]

Oluşturulan bir başka dosya *bigboxes.txt* adındaki dosyadır. Bu dosya oluşturulan büyük birimleri göstermektedir (Şekil 4.10). İlk üç sütun büyük birimin boyutlarını, dördüncü sütun büyük birimin kaybını, sonraki sütunlar ise büyük birimin üyesi olan birimlerin numaralarını ve son üç sütun da büyük birimin koordinatlarını göstermektedir. Bazı koordinat değerlerinde çok büyük değere rastlanmaktadır. Bunun anlamı o büyük birimin yüklenmediğidir. .



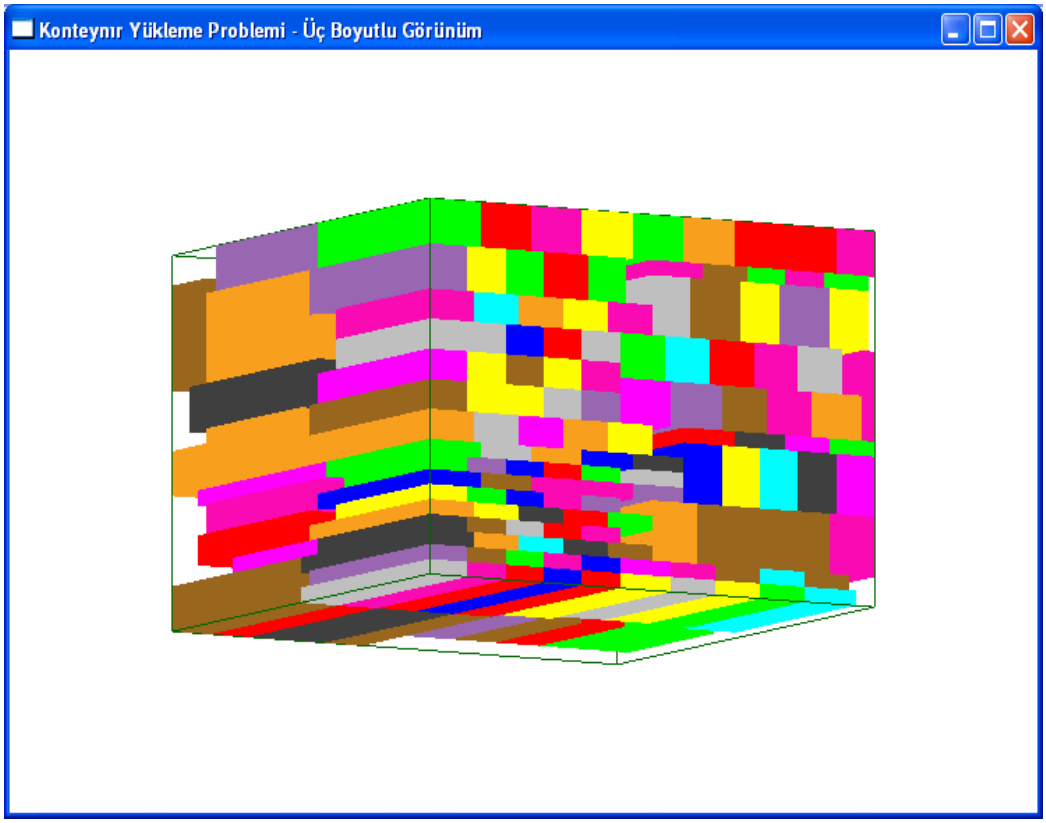
Şekil 4.10. Büyük birimler [2]

Son olarak *analiz* adında bir dosya oluşturulur (Şekil 4.11). Bu dosya belirlediğimiz parametrelere karşılık elde edilen verimi göstermektedir. Program her çalıştırıldığında bu dosyaya bir satır eklenir. Burada ilk sütun, *maxkare* değişkenini, ikinci sütun, *ortkare* değişkenini, üçüncü sütun, *hacimoran* değişkenini, beşinci sütun verimi göstermektedir.



Şekil 4.11. Analiz [2]

Elde edilen yüklemeyi üç boyutlu olarak görme imkanı da bulunmaktadır. *konteynır görünüm.exe* adlı dosya çalıştırıldığında Şekil 4.12'deki ekran çıktısı görünecektir. Görüntü üç boyutlu olduğu için ekranda farklı açılardan görmek algılamayı güçlendirecektir. Bunun için klavyedeki x, y, z, X, Y, Z tuşları kullanılır. x,y,z tuşları aynı isimli eksenlerde şekli döndürecektir. X,Y,Z tuşları ise yine aynı isimli eksenlerde ters yönde döndürecektir.



Şekil 4.12. Üç boyutlu görünüm [2]

5.EE ALGORİTMASININ DİĞER ALGORİTMALARLA KARŞILAŞTIRILMASI

Algoritma Bischoff ve Ratcliff'in (1995) çalışmalarında kullandıkları veriler kullanılarak çalıştırıldı ve elde edilen sonuçlar daha önce yapılan çalışmalarla karşılaştırılmıştır [17]. Bischoff ve Ratcliff (1995) 3,5,8,10,12 ve 15 farklı birim çeşidi kullanarak ve her birim çeşidinden 100'er problem seti oluşturarak toplam 700 farklı çeşit problem seti kullanmıştır [17]. Çizelge 4.4 her problem setinden elde edilen verimlerin ortalamalarını göstermektedir. Bütün problemlerde 233×587×220 boyutunda konteynır kullanılmıştır. Burada Bischoff ve Ratcliff (1995) B/R ile, Gehring ve Bortfeld (1997) G/B ile, Bortfeld ve Gehring (1998) B/G ile, Terno ve ark. (1995) T/S/S/R ile Baltacıoğlu (2001) B ile, Erol ve Ensari, EE ile temsil edilmiştir [12,16-18,23,].

Çizelge 5.1. EE algoritması ile diğer algoritmaların karşılaştırılması [2]

Birim	B/R			G/B			B/G			T/S/S/R			B			EE		
Setleri	min.	ort.	maks.	min.	ort.	maks.	min.	ort.	maks.	min.	ort.	maks.	min.	ort.	maks.	Min	ort.	maks
BR-3	73,7	85,4	94,4	76,7	85,8	94,3	78,7	89,0	95,7	75,7	89,9	95,9	78,9	89,0	95,3	89,57	91,6	94,8
BR-5	73,8	86,3	93,8	78,4	87,3	95,2	79,7	88,7	95,0	81,9	89,6	94,7	84,8	89,0	94,0	88,32	89,2	90,6
BR-8	75,3	85,9	92,6	81,1	88,1	92,9	82,4	88,2	94,0	83,2	89,2	93,0	84,5	88,4	92,1	87,4	88,2	89,7
BR-10	78,4	85,1	90,1	82,7	88,0	91,6	80,9	87,4	92,0	83,1	88,9	92,7	84,4	88,2	91,9	86,9	87,8	91,3
BR-20	75,7	83,0	88,3	84,4	87,7	90,7	79,9	83,9	88,4	80,6	86,3	89,0	84,3	87,1	90,2	85,2	86,2	87,4

Bu sonuçlardan görüleceği gibi EE, daha az çeşit birim içeren problem setlerinde daha iyi sonuç vermektedir. Çeşit sayısının artması ile birlikte verim bu problemde her zaman düşüş eğilimindedir. Bu çalışmada elde edilen sonuçlarda da durum böyle olmuştur. BR-3 probleminde ortalama verim 91,55 iken BR-20 probleminde bu değer 86,18 e gerilemiştir. Yani toplamda %5,37 gibi bir fark oluşmuştur. Bu farkın çok büyük bir fark olmadığını kabul edilse de çeşit sayısı arttıkça EE'nin elde ettiği sonuçlar daha önceki algoritmalardan çok küçük de olsa farklar içermektedir.

Çizelge 5.2. EE algoritması ile diğer algoritmalar arasındaki farklar [2]

Birim Setleri	B/R ort.	G/B ort.	B/G ort.	T/S/S/R ort.	B ort.
BR-3	6,15	5,75	2,55	1,65	2,55
BR-5	2,94	1,94	0,54	-0,36	0,24
BR-8	2,33	0,13	0,03	-0,97	-0,17
BR-10	2,71	-0,19	0,41	-1,09	-0,39
BR-20	3,18	-1,52	2,28	-0,12	-0,92

Çizelge 5.2'de EE ile diğer algoritmalar arasındaki farklar görülebilir. Çizelge5.2'de pozitif değerler EE'nin daha iyi sonuçlarını, negatif değerler ise daha kötü sonuçları göstermektedir. Aşağıdaki tablo da göz önünde bulundurularak EE'nin üreticinin yükleme probleminde dağıtıcının yükleme problemine göre daha iyi sonuçlar vereceği söylenebilir. Bunun yanında dağıtıcının yükleme probleminde de kullanılması mümkündür. Yukarıdaki tablodan da görülebileceği gibi EE diğer algoritmalardan daha istikrarlıdır. Maksimum ve minimum değerler arasındaki farklar incelendiğinde bu aralığın en küçük olduğu algoritma EE'dir.

6.SONUÇ VE ÖNERİLER

Bu çalışma lojistik sektöründe konteynır yükleme problemine farklı bir çözüm yaklaşımı getirmiştir. Elde edilen sonuçlar geçmiş çalışmalarla karşılaştırılmış ve birim çeşidi daha az olan problemlerde daha iyi sonuç verdiği görülmüştür. Buna rağmen birim çeşidi fazla olan problemlerde de geçmiş çalışmalarla çok yakın sonuçlar elde edilmiştir.

Verimi etkileyen iki adet parametre ve regresyon denklemi literatüre kazandırılmıştır. Bu parametreler sonucu birebir etkilemekte olduğundan yüklenecek birimler parametrelere göre seçildiğinde daha iyi sonuç elde edilecektir.

Çalışmanın geçmiş çalışmalardan en önemli farkı birimlerden büyük birimler oluşturulması ve bu büyük birimlerin konteynıra yerleştirilmesinden sonraki proseslerle etkinliğin arttırılmasıdır. Önceki çalışmaların temel mantığını oluşturan duvar inşa yaklaşımı (wall building aproach) ile en önemli fark konteynırın bütün boyutlarının aynı öncelikle düşünülmesi yani bir anlamda problemin alt problemlere (tek boyutlu ve iki boyutlu konteynır yükleme problemi) ayrılmadan çözüme gidilmesidir.

Verimi arttırmaya yönelik ek prosesler (uygun noktaları kaydırma prosesi, birimleri kaydırma prosesi, son yerleştirme prosesi) verimi önemli ölçüde arttırmıştır.

Büyük birimlerin oluşturulma prosesinde 4. bölümde açıklandığı gibi birimler m'şer adet birim içeren bölümlere ayrılarak büyük birimler oluşturulmaktadır. Bu m değerinin en iyi verimin elde edileceği şekilde değişken veya sabit olarak belirlenmesi sonraki çalışmalarda incelenebilir.

Pratik hayatta birimlerin konteynırdan indirilme sıralarına göre yerleştirilmesi (en son indirilecek birim en arkaya, en önce indirilecek birim en öne) kuralı proses zamanlarını kısaltacaktır. Fakat aynı zamanda bu uygulamanın konteynırın yüklenme

oranını (verim) azaltacağı da göz önünde bulundurulmalıdır. Bu çalışma ileriki çalışmalar için bu yönde bir zemin oluşturmaktadır.

Ayrıca algoritmada rassal olarak ve denenerek en iyisi belirlenen uygun noktalar, rotasyon gibi işlemler için de bir kural oluşturulması çözüm zamanlarını kısaltacaktır.

KAYNAKLAR

1.Ballew B.P.,"The distributor's three dimonsional pallet packing problem:a mathematical formulation and heuristic solution aproach", ***Air force Inst. of Tech. Wright-Patterson of School of Engineering,*** Msc Thesis,1-48 (2000).

2.Ensari Arşivi, (2007).

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

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

5.Bischoff E., Dowsland E.B., "The application of the micro to product design and distribution", ***European Journal of Operational Research,*** 33:271-280,(1982).

6.Bischoff E.E., Marriott E.D., "A comperative evaluation of heuristic for container loading",***European Journal of Operation Research,*** 44:267-276,(1990).

7.Gehring M., Menscher K., Meyer M., "A computer-based heuristic for packing pooled shipment containers", ***European Journal of Operational Research,*** 44:277–288, (1990).

8.Han C.P., Knott K., Egbalu P.J., "A heuristic approach to the three dimensional cargo loading problem",***International Journal of Production Research,*** 27(5):757-774,(1989).

9.Mohanty B.B., Mathur K., Ivancic N.J., "Value considerations in three dimensional packing a heuristic procedure using the fractional knapsack problem", ***International Journal of Production Research,*** 74:143-151,(1994).

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

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

12.Terno J., Scheithauer G., Sommerweiβ U., Riehme J.,"An efficient approach for the multi-pallet loading problem", ***European Journal of Operational Research,*** 84:681-692,(1995).

13.Osogami T., Okano H.,"Local search algorithms for the bin packing problem and their relationship to various construction heuristics", ***Journal of Heuristics,*** 9:29-49,(2003).

14.Faina L., "Three dimensional bin packing problem", ***Operational Research,*** 48:256-267,(2000).

15.Pisinger D., "Heuristic for the container loading problem", ***European Journal of Operational Research,*** 141:382-392,(2002).

16.Baltacıoğlu E.,"The distributer's three dimensional pallet packing problem: a human intelligence based heuristic approach", ***Air force Institute of technology,*** M.Sc. Thesis,1-53(2001).

17.Bischoff E.E., Ratcliff M.S.W., "Loading multiple pallets",***Journal of the Operational Research Society,*** 46:1322–1336,(1995).

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

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

20.Yeung L.H.W., Tang W.K.S., "A hybrid genetic approach for container loading in logistics industry", ***IEEE,*** Nisan sayısı,617-627(2005).

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

22.Pimpawat C., Chaiyaratana N., "Three dimensional container loading using a cooperative co-evolitionary genetic algorithm", ***Artifficial Intelligence,*** 18:581-601, (2004).

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

EKLER

EK-1. Diziler, sabitler ve değişkenler

Sabitler

Maxbox

Yerleştirilmek istenen birimlerin toplam sayısını döndürür.

Max

Birimlerin kaçarlı birimler halinde gruplanacağı başka bir deyişle bir büyük birimin kaç birimden oluşacağını döndürür.

Buyuk

Çok büyük bir sayıyı döndürür.

Maxx

Birimlerin yerleştirileceği konteynırın genişliğini döndürür.

Maxy

Birimlerin yerleştirileceği konteynırın yüksekliğini döndürür.

Maxz

Birimlerin yerleştirileceği konteynırın derinliğini döndürür.

Uygun

Bir büyük birim oluşturulurken meydana gelebilecek uygun noktaların kaydedileceği dizinin büyüklüğünü döndürür.

EK-1 (Devam) Diziler, sabitler ve değişkenler

Uygunbig

Büyük birimler konteynıra yerleştirilirken meydana gelebilecek uygun noktaların kaydedileceği dizinin büyüklüğünü döndürür.

Uygunson

Son yerleştirme prosesi gerçekleştirilirken meydana gelebilecek uygun noktaların kaydedileceği dizinin büyüklüğünü döndürür.

Diziler

Kordinat

Birimlerin yerleştirileceği konteynırı bir koordinat düzlemi olarak belirler ve arabelleğe konumlandırır.

Secbox

Büyük birimlerin oluşturulması için seçilen birimlerin kaydedileceği dizidir.

EK-1 (Devam) Diziler, sabitler ve değişkenler

İcsecim

Büyük birimler oluşturulurken yerleştirilmiş birimlerin ve yerleştirilmemiş birimlerin ayırt edilmesini sağlayan dizidir. Burada yerleştirilmiş birimler için 1, yerleştirilmemiş birimler için 0 değerleri tutulur.

Uygunx

Büyük birimler oluşturulurken uygun noktaların x koordinatlarının kaydedildiği dizidir.

Uyguny

Büyük birimler oluşturulurken uygun noktaların y koordinatlarının kaydedildiği dizidir.

Uygunz

Büyük birimler oluşturulurken uygun noktaların z koordinatlarının kaydedildiği dizidir.

Uygund

Büyük birimler oluşturulurken meydana getirilen uygun noktaların orijin noktasından uzaklıklarının kaydedildiği dizidir.

Uygunbigx

Büyük birimler konteynıra yerleştirilirken meydana getirilen uygun noktaların x koordinatlarının kaydedildiği dizidir.

EK-1 (Devam) Diziler, sabitler ve değişkenler

Uygunbigy

Büyük birimler konteynıra yerleştirilirken meydana getirilen uygun noktaların y koordinatlarının kaydedildiği dizidir.

Uygunbigz

Büyük birimler konteynıra yerleştirilirken meydana getirilen uygun noktaların z koordinatlarının kaydedildiği dizidir.

Uygunbigd

Büyük birimler konteynıra yerleştirilirken meydana getirilen uygun noktaların orijin noktasından uzaklıklarının tutulduğu dizidir.

Uygunsonx

Son yerleştirme prosesi gerçekleştirilirken meydana gelen uygun noktaların x koordinatlarının tutulduğu dizidir.

EK-1 (Devam) Diziler, sabitler ve değişkenler

Uygunsony

Son yerleştirme prosesi gerçekleştirilirken meydana gelen uygun noktaların y koordinatlarının tutulduğu dizidir.

Uygunsonz

Son yerleştirme prosesi gerçekleştirilirken meydana gelen uygun noktaların z koordinatlarının tutulduğu dizidir.

Yapılar

Boxes

Bütün birimlerin genişlik, yükseklik, derinliklerini, eğer yerleştirilmişlerse x,y,z koordinatlarını ve hacimlerini tutan yapıdır.

Bboxes

Oluşturulan bütün büyük birimlerin genişlik, yükseklik, derinliklerini, eğer yerleştirilmişlerse x,y,z koordinatlarını, hacimlerini, hangi birimler kullanılarak meydana getirildiklerini, içerisinde oluşturulan uygun noktaları ve büyük birimi kapsayan en küçük dikdörtgenler prizması arasındaki hacim farkını tutan yapıdır.

EK-2 Fonksiyonlar

Verioku

Bu fonksiyon yerleştirilmek istenen veri grubunun program tarafından okunmasını ve arabelleğe alınmasını sağlar.

Rassalboxsec

Bütün birimleri her grup n şer adet birim içerecek şekilde bölmüştük. Bu fonksiyon da parametre olarak istenilen grup numarasını kullanarak o grubu çağırır. Örneğin n=5 olsun. Bu durumda rassalboxsec(8), 8. grubu yani;

$8 \times 5 = 40$

$8 \times 5 + 1 = 41$

$8 \times 5 + 2 = 42$

$8 \times 5 + 3 = 43$

$8 \times 5 + 4 = 44$

40,41,42,43 ve 44. birimleri çağıracaktır.

Rotasyonsec

Bu fonksiyon istediğimiz birim için rassal olarak bir rotasyon seçer. Parametre olarak birim numarasını kullanır. Örneğin 14. birim için rotasyonsec(14) terimi 14. birimi rassal olarak belirlediği bir yönde döndürecektir. Bu birimin z ekseninde 90 derece eksi yönlü döndürüldüğünü düşünürsek birimin x boyutundaki uzunluğu y, y boyutundaki uzunluğu da x olacaktır.

İcrasal

Rassalboxsec fonksiyonu ile istenilen grup çağırıldıktan sonra bu gruptan büyük birim oluşturmak amacı ile teker teker birimler çağırılacaktır. İcrassal fonksiyonu rassalboxsec fonksiyonundan sonra çağırılmaktadır. Fonksiyonun parametresi

EK-2 (Devam) Fonksiyonlar

yoktur. Grubun içinden yerleştirilmeyen birimler arasından rassal olarak bir birim çağırır.

Uygunilk

Bu fonksiyon konteynırda hiçbir yerleşim yapılmadan önceki tek uygun nokta yani (0,0,0) noktasının olduğu duruma döner.

Yerlestir

Yerlestir fonksiyonu ilk olarak uygun noktaları orijin noktasından uzaklıklarına göre sıralar. Daha sonra önceden çağırılan gruptan icrassal fonksiyonunu kullanarak bir

EK-2 (Devam) Fonksiyonlar

birim çağırır. Bu birimi sıraladığı uygun noktalara baştan itibaren deneyerek yerleştirmeye çalışır. Hiçbir uygun noktaya yerleşme imkanı bulamazsa yine icrassal fonksiyonunu kullanarak başka bir birim seçer ve yukarıdaki prosesi tekrarlar. Parametre içermez.

Grupla

Bu fonksiyon her birim için "ortkare" değişkeninin değerini hesaplar ve bu değere göre birimleri sıralar ve gruplandırır. Parametre içermez.

Uygunbigilk

Bu fonksiyon uygunilk fonksiyonuna benzer olarak büyük birimlerin yerleşimleri için ilk duruma yani tek uygun nokta olarak orijin noktasının bulunduğu duruma döner. Parametre içermez.

Uygunbigsirala

Bu fonksiyon büyük birimlerin yerleşimleri için uygun noktaları belli bir kurala göre sıralar. Daha sonra bu sıraya göre büyük birimler yerleştirilmeye çalışılacaktır. Sıralama kuralı uygun noktaların orijin noktasından uzaklıklarına göre belirlenir. Orijin noktasına en yakın olan uygun nokta en başa, en uzak nokta da en sona atanacaktır. Uygun noktanın orijinden uzaklığı şöyle hesaplanır: noktamız (x,y,z) olsun. Bu noktanın orijinden uzaklığına da d diyelim. Bu durumda;

$d = \sqrt{x^2 + y^2 + z^2}$ olacaktır.

Fonksiyon parametre kullanmamaktadır.

Bigboxsirala

Büyük birimlerin yerleşimlerinde önceliği belirleyen bu sıralama kuralı her büyük birimin hacminin büyük birimi kapsayan en küçük dikdörtgenler prizmasının

EK-2 (Devam) Fonksiyonlar

hacminden farkına göre sıralanmasından ibarettir. Örneğin büyük birimin, hacimleri v_1, v_2, v_3, v_4, v_5 olan birimlerden oluşturulduğu ve büyük birimi kapsayacak en küçük dikdörtgenler prizmasının hacminin de V olduğu varsayılırsa fark;

$Fark = V - \sum_{n=1}^{5} v_n$ şeklinde olacaktır. En küçük fark değerine sahip büyük birimlere yerleştirmede öncelik tanınacaktır. Fonksiyon parametre içermemektedir.

EK-2 (Devam) Fonksiyonlar

Yerlestirbig

Bu fonksiyon büyük birimleri konteynıra sıralama kurallarına göre yerleştirir. Yukarıda açıkladığımız bigboxsirala ve uygunbigsirala fonksiyonlarını çağırarak öncelikle öncelik kurallarının gerçekleştirilmesini sağlar. Daha sonra bu sıralamaya göre büyük birimleri konteynıra yerleştirir. Fonksiyon, yerleştirecek büyük birim kalmayıncaya veya kalan büyük birimlerin hiçbir uygun noktaya yerleştirilme imkanı bulunmayıncaya kadar kendini tekrar eder. Parametre içermez.

Uygunsonilk

Bu fonksiyon uygunilk ve uygunbigilk fonksiyonlarına benzer şekilde ilk yerleşim için tek uygun nokta olarak orijin noktasını belirler. Parametre içermez.

Uygunsonbossec

Üçüncü adım yerleştirme prosesi için uygun noktalar arasından birini çağırır. Parametre içermez.

Kordinatkayit

Yerleştirilmiş olan büyük birimlerin kapsadığı birimlerin koordinatları koordinat düzlemine bu fonksiyon çağırılarak kaydedilir. Ayrıca büyük birim içerisinde oluşturulan uygun noktalar da bu fonksiyon sayesinde son yerleştirme için belirlenen uygun noktalar arasındaki yerini alır. Fonksiyon parametre içermez.

Vsirala

Son yerleştirmeden önce yerleştirilemeyen bütün birimler hacimlerinin büyüklüklerine göre bir sıraya konur. En büyük hacimli birim en üstte yer alır ve diğer birimlerde azalan hacimlere göre sıralanır. Son yerleştirmeye ilk olarak en

EK-2 (Devam) Fonksiyonlar

büyük hacimli birimden başlanacaktır. Çünkü büyük hacimli birimlerin yerleşimlerinin sonradan yapılması nispeten zor olacaktır. Fonksiyon parametre içermemektedir.

Yerlestirson

Fonksiyon, büyük birimlerin yerleştirilmesinden sonraki proses olan son yerleştirme prosesini gerçekleştirir. Vsirala fonksiyonu ile hacimlerine göre sıralanmış olan birimleri boş kalan alanlara yerleştirmeye çalışır. Yerleştirilecek birim kalmadığında veya kalan birimlerden hiçbirinin uygun noktaların hiçbirine yerleşememesi durumuna gelinceye kadar fonksiyon kendini tekrar eder. Parametre içermez.

EK-2 (Devam) Fonksiyonlar

Vgecicihesapkontrol

Fonksiyon, parametre olarak girilen büyük birimi kapsayan en küçük dikdörtgenler prizmasını bulur ve değer olarak prizmanın hacmini döndürür. İki parametre içerir. Birinci parametre işlemin yapılması istenen büyük birimin numarasıdır. İkinci parametre ise yalnızca 0 ve 1 değerlerini alan ikili bir değişkendir. Eğer büyük birim yerleşmemişse bu ikili değişken 0 değerini, yerleşmişse 1 değerini alır. Fonksiyonu bu ikinci parametre 0 olarak girildiğinde 0 fonksiyon da 0 değerini döndürür.

Bigboxkayit

Oluşturulmuş olan büyük birimlerin hangi birimlerden oluştuğunu ve bu büyük birimin uygun noktalarını arabelleğe kaydeder. Parametre içermez.

Uygunensonbelirlex

X ekseninde kaydırılabilecek uygun noktaları belirler. Parametre içermez.

Uygunensonbelirley

Y ekseninde kaydırılabilecek uygun noktaları belirler. Parametre içermez.

Uygunensonbelirlez

Z ekseninde kaydırılabilecek uygun noktaları belirler. Parametre içermez.

Uygunkaydirx

Uygunensonbelirlex fonksiyonuyla belirlenen noktaları x ekseninde kaydırır.
Parametre içermez.

EK-2 (Devam) Fonksiyonlar

Uygunkaydiry

Uygunensonbelirley fonksiyonuyla belirlenen noktaları y ekseninde kaydırır. Parametre içermez. Parametre içermez.

Uygunkaydirz

Uygunensonbelirlez fonksiyonuyla belirlenen noktaları z ekseninde kaydırır. Parametre içermez.

Kaydır

Parametre olarak belirlenen birimi x, y ve z eksenlerinde sol alt dip köşeye yakınlaştırarak boş alanlar oluşturur.

EK-3. C program kodu

```
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>

#define MAXBOX 476
#define MAX 1
#define BUYUK 100000
#define YOK  10000
#define MAXX 234
#define MAXY 588
#define MAXZ 221

#define UYGUN 30
#define UYGUNBIG 800
#define UYGUNSON 1000

#define tipsayisi 3

int kordinat[MAXX][MAXY][MAXZ];


int secbox[MAX];
int icsecim[MAX];

struct boxes{
	int x;
	int y;
	int z;
	int dx;
	int dy;
	int dz;
	int v;
	float d;

};

struct bboxes{
	int x;
	int y;
	int z;
	int dx;
	int dy;
	int dz;
```

EK-3 (Devam) C program kodu

```
        int v;
        int kayip;
        int origin;
        int uyeler[MAX-1];

};



struct boxes allboxes[MAXBOX];
struct boxes *gallboxes[MAXBOX];
struct bboxes bigboxes[MAXBOX/MAX];
struct bboxes *gbigboxes[MAXBOX/MAX];
int uygunx[UYGUN],uyguny[UYGUN],uygunz[UYGUN];
float uygund[UYGUN];

int uygunbigx[UYGUNBIG],uygunbigy[UYGUNBIG],uygunbigz[UYGUNBIG];
float uygunbigd[UYGUNBIG];

int uygunsonx[UYGUNSON],uygunsony[UYGUNSON],uygunsonz[UYGUNSON];
float uygunsond[UYGUNSON];

void ilkkordinat(void){

int index1=0,index2=0,index3=0;

for(index1=0;index1<MAXX;index1++){
   for(index2=0;index2<MAXY;index2++){
      for(index3=0;index3<MAXZ;index3++){
         kordinat[index1][index2][index3]=0;
      }
   }
}
}


void verioku(void){

FILE *gdosya;
int index1=0;


if((gdosya=fopen("birimverileri.txt","r"))==NULL) {
     printf("dosya acilamadi");
     exit(1);}
```

EK-3 (Devam) C program kodu

```
rewind(gdosya);

for(index1=0;index1<MAXBOX;index1++){
   fscanf(gdosya,"%d",&gallboxes[index1]->dx);
   fscanf(gdosya,"%d",&gallboxes[index1]->dy);
   fscanf(gdosya,"%d",&gallboxes[index1]->dz);
   fscanf(gdosya,"%d",&gallboxes[index1]->v);
   if(gallboxes[index1]->v==0){
                        gallboxes[index1]->dx=BUYUK;
                        gallboxes[index1]->dy=BUYUK;
                        gallboxes[index1]->dz=BUYUK;
                        gallboxes[index1]->v=BUYUK;
                        }
   }
   fclose(gdosya);
}

void rassalboxsec(int b){
int index;

for(index=0;index<MAX;index++)
secbox[index]=b*MAX+index;



}

void rotasyonsec(int boxno, int a){
int yedek;



switch(a) {
case 1: break;
case 2: yedek=gallboxes[boxno]->dy;
       gallboxes[boxno]->dy=gallboxes[boxno]->dz;
       gallboxes[boxno]->dz=yedek;
       break;

case 3: yedek=gallboxes[boxno]->dx;
       gallboxes[boxno]->dx=gallboxes[boxno]->dz;
       gallboxes[boxno]->dz=yedek;
       break;
```

EK-3 (Devam) C program kodu

```
case 4: yedek=gallboxes[boxno]->dy;
        gallboxes[boxno]->dy=gallboxes[boxno]->dx;
        gallboxes[boxno]->dx=yedek;

        yedek=gallboxes[boxno]->dx;
        gallboxes[boxno]->dx=gallboxes[boxno]->dz;
        gallboxes[boxno]->dz=yedek;
        break;

case 5: yedek=gallboxes[boxno]->dx;
        gallboxes[boxno]->dx=gallboxes[boxno]->dz;
        gallboxes[boxno]->dz=yedek;

        yedek=gallboxes[boxno]->dx;
        gallboxes[boxno]->dx=gallboxes[boxno]->dy;
        gallboxes[boxno]->dy=yedek;
        break;

case 6: yedek=gallboxes[boxno]->dx;
        gallboxes[boxno]->dx=gallboxes[boxno]->dy;
        gallboxes[boxno]->dy=yedek;
        break;

default: printf("hatali islem");
exit(1);
}
}


int icrassal(void){

int index1=0;
int *a;
int b;

a=&b;

srand(time(NULL));
for(index1=0;index1<BUYUK;index1++){
b=rand()%MAX;

if ((*(icsecim+(*a)))==0)
break;
}
```

EK-3 (Devam) C program kodu

```
rotasyonsec(secbox[*a],1);

return *a;
}

void uygunilk(void){

int index1=1;


*uygunx=0;
*uyguny=0;
*uygunz=0;


for(index1=1;index1<UYGUN;index1++){
*(uygunx+index1)=BUYUK;
}

}




int yerlestir(void){

int
ask=BUYUK,k,t,p,index,index1=0,index2=0,index3=0,index4=0,i,in,yedek,yerlesti
mi;
int *gk,*gt,*gp,*gi;
int dx=0,dy=0,dz=0;
float yedek1;
gp=&p;
gi=&i;
gk=&k;
gt=&t;


for(index1=0;index1<UYGUN;index1++){
if(*(uygunx+index1)!=BUYUK)
*(uygund+index1)=sqrt((*(uygunx+index1))*
(*(uygunx+index1))+(*(uyguny+index1))*
(*(uyguny+index1))+(*(uygunz+index1))* (*(uygunz+index1)));
else
```

EK-3 (Devam) C program kodu

```
*(uygund+index1)=BUYUK;
}
for(index1=0;index1<UYGUN;index1++){
 for(index2=index1;index2<UYGUN;index2++){
 if(uygund[index1]>uygund[index2]){
    yedek=*(uygunx+index1);
    *(uygunx+index1)=*(uygunx+index2);
    *(uygunx+index2)=yedek;

    yedek=*(uyguny+index1);
    *(uyguny+index1)=*(uyguny+index2);
    *(uyguny+index2)=yedek;

    yedek=*(uygunz+index1);
    *(uygunz+index1)=*(uygunz+index2);
    *(uygunz+index2)=yedek;

    yedek1=*(uygund+index1);
    *(uygund+index1)=*(uygund+index2);
    *(uygund+index2)=yedek1;
    }
  }
}

yerlestimi=0;
k=icrassal();
t=*(secbox+*gk);
for(index=0;index<UYGUN;index++){
for(in=1;in<7;in++){

rotasyonsec(*gt,in);


      if(uygunx[index]!=BUYUK && uygunx[index]+gallboxes[*gt]->dx<MAXX
&& uyguny[index]+gallboxes[*gt]->dy<MAXY && uygunz[index]+gallboxes[*gt]-
>dz<MAXZ){
                  ask=0;
                  for(index1=uygunx[index];index1<uygunx[index]+gallboxes[*gt]-
>dx;index1++){
                  for(index2=uyguny[index];index2<uyguny[index]+gallboxes[*gt]-
>dy;index2++){
                  for(index3=uygunz[index];index3<uygunz[index]+gallboxes[*gt]-
>dz;index3++){
                  ask+=kordinat[index1][index2][index3]; }}}
   }
```

EK-3 (Devam) C program kodu

```
else ask=BUYUK;

        if(uygunx[index]+gallboxes[*gt]->dx<MAXX &&
uyguny[index]+gallboxes[*gt]->dy<MAXY && uygunz[index]+gallboxes[*gt]-
>dz<MAXZ && ask==0){
                *(icsecim+(k))=1;
                gallboxes[*gt]->x=*(uygunx+index);
                gallboxes[*gt]->y=*(uyguny+index);
                gallboxes[*gt]->z=*(uygunz+index);
     *(uygunx+index)=BUYUK;
                for(index1=gallboxes[*gt]->x;index1<gallboxes[*gt]-
>x+gallboxes[*gt]->dx;index1++){
                for(index2=gallboxes[*gt]->y;index2<gallboxes[*gt]-
>y+gallboxes[*gt]->dy;index2++){
                for(index3=gallboxes[*gt]->z;index3<gallboxes[*gt]-
>z+gallboxes[*gt]->dz;index3++){
     kordinat[index1][index2][index3]=1; }}}

                for(index1=0;index1<UYGUN;index1++){
                if(*(uygunx+index1)==BUYUK){
                break;}}
                *(uygunx+index1)=(gallboxes[*gt]->x)+(gallboxes[*gt]->dx);
                *(uyguny+index1)=(gallboxes[*gt]->y);
                *(uygunz+index1)=(gallboxes[*gt]->z);

                for(index1=0;index1<UYGUN;index1++){
                if(*(uygunx+index1)==BUYUK){
                break;}
                }
                *(uygunx+index1)=(gallboxes[*gt]->x);
                *(uyguny+index1)=(gallboxes[*gt]->y)+(gallboxes[*gt]->dy);
                *(uygunz+index1)=(gallboxes[*gt]->z);

                for(index1=0;index1<UYGUN;index1++){
                if(*(uygunx+index1)==BUYUK){
                break;}}
                *(uygunx+index1)=(gallboxes[*gt]->x);
                *(uyguny+index1)=(gallboxes[*gt]->y);
                *(uygunz+index1)=(gallboxes[*gt]->z)+(gallboxes[*gt]->dz);
                yerlestimi=1;
                break;
                }
                else {continue;}
        }
```

EK-3 (Devam) C program kodu

```
        if(yerlestimi==1) break; else continue;
}

return yerlestimi;
}

void grupla(void){

int index1,index2,yedek,a1,a2,a3,b1,b2,b3;
float c1,c2,c3,d1,d2,d3;


for(index1=0;index1<MAXBOX;index1++){
for(index2=index1;index2<MAXBOX;index2++){
a1=gallboxes[index1]->dx;
a2=gallboxes[index1]->dy;
a3=gallboxes[index1]->dz;

if(a1>a2)   c1=(float)a2/a1;
else        c1=(float)a1/a2;

if(a1>a3)   c2=(float)a3/a1;
else        c2=(float)a1/a3;

if(a2>a3)   c3=(float)a3/a2;
else        c3=(float)a2/a3;

b1=gallboxes[index2]->dx;
b2=gallboxes[index2]->dy;
b3=gallboxes[index2]->dz;

if(b1>b2)   d1=(float)b2/b1;
else        d1=(float)b1/b2;

if(b1>b3)   d2=(float)b3/b1;
else        d2=(float)b1/b3;

if(b2>b3)   d3=(float)b3/b2;
else        d3=(float)b2/b3;


if(c1+c2+c3>d1+d2+d3){
yedek=gallboxes[index2]->dx;
gallboxes[index2]->dx=gallboxes[index1]->dx;
```

EK-3 (Devam) C program kodu

```
gallboxes[index1]->dx=yedek;

yedek=gallboxes[index2]->dy;
gallboxes[index2]->dy=gallboxes[index1]->dy;
gallboxes[index1]->dy=yedek;

yedek=gallboxes[index2]->dz;
gallboxes[index2]->dz=gallboxes[index1]->dz;
gallboxes[index1]->dz=yedek;

yedek=gallboxes[index2]->v;
gallboxes[index2]->v=gallboxes[index1]->v;
gallboxes[index1]->v=yedek;
}
}}
}

void uygunbigilk(void){

int index1=1;


*uygunbigx=0;
*uygunbigy=0;
*uygunbigz=0;


for(index1=1;index1<UYGUNBIG;index1++){
*(uygunbigx+index1)=BUYUK;
}

}



void uygunbigsirala(void){
int yedek;
int j;
int index1,index2;
float yedek1;



for(index1=0;index1<UYGUNBIG;index1++){
if(uygunbigx[index1]!=BUYUK)
```

EK-3 (Devam) C program kodu

```
*(uygunbigd+index1)=sqrt((*(uygunbigx+index1))*
(*(uygunbigx+index1))+(*(uygunbigy+index1))*
(*(uygunbigy+index1))+(*(uygunbigz+index1))* (*(uygunbigz+index1)));
else {
*(uygunbigd+index1)=BUYUK;
}
}
for(index1=0;index1<UYGUNBIG;index1++){

    for(index2=index1+1;index2<UYGUNBIG;index2++){
    if(*(uygunbigd+index1)>*(uygunbigd+index2)){

    yedek=*(uygunbigx+index2);
    *(uygunbigx+index2)=*(uygunbigx+index1);
    *(uygunbigx+index1)=yedek;

    yedek=*(uygunbigy+index2);
    *(uygunbigy+index2)=*(uygunbigy+index1);
    *(uygunbigy+index1)=yedek;

    yedek=*(uygunbigz+index2);
    *(uygunbigz+index2)=*(uygunbigz+index1);
    *(uygunbigz+index1)=yedek;

    yedek1=*(uygunbigd+index2);
    *(uygunbigd+index2)=*(uygunbigd+index1);
    *(uygunbigd+index1)=yedek1; }
    }}


}

void bigboxsirala(void){
struct bboxes yedek;
int index1,index2;

for(index1=0;index1<MAXBOX/MAX;index1++){
for(index2=index1+1;index2<MAXBOX/MAX;index2++){

if(gbigboxes[index1]->v<gbigboxes[index2]->v){

yedek=bigboxes[index1];
bigboxes[index1]=bigboxes[index2];
bigboxes[index2]=yedek;    }
```

EK-3 (Devam) C program kodu

```
}}
}

void yerlestirbig(void){
int index1,index2,index3,index4,index5;
int ask;

uygunbigilk();

uygunbigsirala();
bigboxsirala();

for(index4=0;index4<MAXBOX/MAX;index4++){

if(gbigboxes[index4]->x!=BUYUK)
continue;

for(index5=0;index5<UYGUNBIG;index5++){


ask=BUYUK;
if(*(uygunbigx+index5)+gbigboxes[index4]->dx>=MAXX ||
*(uygunbigy+index5)+gbigboxes[index4]->dy>=MAXY ||
*(uygunbigz+index5)+gbigboxes[index4]->dz>=MAXZ  ||
uygunbigx[index5]==BUYUK)
continue;
else{
ask=0;
for(index1=*(uygunbigx+index5);index1<*(uygunbigx+index5)+gbigboxes[index4]-
>dx;index1++){
for(index2=*(uygunbigy+index5);index2<*(uygunbigy+index5)+gbigboxes[index4]-
>dy;index2++){
for(index3=*(uygunbigz+index5);index3<*(uygunbigz+index5)+gbigboxes[index4]-
>dz;index3++){

ask+=kordinat[index1][index2][index3];
 }}}
}
if(ask==0){

gbigboxes[index4]->x=*(uygunbigx+index5);
gbigboxes[index4]->y=*(uygunbigy+index5);
gbigboxes[index4]->z=*(uygunbigz+index5);
```

EK-3 (Devam) C program kodu

```
for(index1=gbigboxes[index4]->x;index1<gbigboxes[index4]-
>x+gbigboxes[index4]->dx;index1++){
for(index2=gbigboxes[index4]->y;index2<gbigboxes[index4]-
>y+gbigboxes[index4]->dy;index2++){
for(index3=gbigboxes[index4]->z;index3<gbigboxes[index4]-
>z+gbigboxes[index4]->dz;index3++){

kordinat[index1][index2][index3]=1; }}}

*(uygunbigx+index5)=BUYUK;


for(index1=0;index1<UYGUNBIG;index1++){
  if(*(uygunbigx+index1)==BUYUK){
  break;}}
  *(uygunbigx+index1)=(gbigboxes[index4]->x)+(gbigboxes[index4]->dx);
  *(uygunbigy+index1)=(gbigboxes[index4]->y);
  *(uygunbigz+index1)=(gbigboxes[index4]->z);

for(index1=0;index1<UYGUNBIG;index1++){
  if(*(uygunbigx+index1)==BUYUK){
  break;}}
  *(uygunbigx+index1)=(gbigboxes[index4]->x);
  *(uygunbigy+index1)=(gbigboxes[index4]->y)+(gbigboxes[index4]->dy);
  *(uygunbigz+index1)=(gbigboxes[index4]->z);

for(index1=0;index1<UYGUNBIG;index1++){
  if(*(uygunbigx+index1)==BUYUK){
  break;}}
  *(uygunbigx+index1)=(gbigboxes[index4]->x);
  *(uygunbigy+index1)=(gbigboxes[index4]->y);
  *(uygunbigz+index1)=(gbigboxes[index4]->z)+(gbigboxes[index4]->dz);

  uygunbigsirala();
   break;
}


}
}

}

void uygunsonilk(void){
int index1;
```

EK-3 (Devam) C program kodu

```
for(index1=0;index1<UYGUNSON;index1++)
*(uygunsonx+index1)=BUYUK;
}

int uygunsonbossec(void){
int index1;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)==BUYUK)
break; }
return index1;
}
void boxdilk(void){

int i=0,i1=0;

for(i=0;i<MAXBOX;i++)
gallboxes[i]->d=BUYUK;
}
void boxdkayittek(int a){

gallboxes[a]->d=sqrt((gallboxes[a]->x)*(gallboxes[a]->x)+(gallboxes[a]-
>y)*(gallboxes[a]->y)+(gallboxes[a]->z)*(gallboxes[a]->z));
}

void boxdsirala(void){

int i=0,i1=0;
struct boxes yedek;

for(i=0;i<MAXBOX;i++){
for(i1=i;i1<MAXBOX;i1++){

if(gallboxes[i]->d>gallboxes[i1]->d){

yedek=allboxes[i];
allboxes[i]=allboxes[i1];
allboxes[i1]=yedek;
}
}}
}
void kaydir(int a){

int index,index1,index2,index3,ask=0,x1=0,y1=0,z1=0;
```

EK-3 (Devam) C program kodu

```
x1=0;
y1=0;
z1=0;

//x ekseninde kaydir//
ask=0;
if(gallboxes[a]->x!=0){

for(index1=gallboxes[a]->x-1;index1>=0;index1--){
for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]->dy;index2++){
for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]->dz;index3++){

ask+=kordinat[index1][index2][index3];

}}if(ask==0) x1++;if(ask!=0) break;}
}

if(x1!=0){

for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
    for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]-
>dy;index2++){
    for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]-
>dz;index3++){
       kordinat[index1][index2][index3]=0; }}}


gallboxes[a]->x=(gallboxes[a]->x)-x1;


for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
    for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]-
>dy;index2++){
    for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]-
>dz;index3++){
       kordinat[index1][index2][index3]=1; }}}
}
ask=0;
if(gallboxes[a]->y!=0){

for(index2=gallboxes[a]->y-1;index2>=0;index2--){
for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]->dz;index3++){

ask+=kordinat[index1][index2][index3];
```

EK-3 (Devam) C program kodu

```
}}if(ask==0) y1++; if(ask!=0) break;}
}
if(y1!=0){

for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
    for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]-
>dy;index2++){
    for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]-
>dz;index3++){
       kordinat[index1][index2][index3]=0; }}}

gallboxes[a]->y=(gallboxes[a]->y)-y1;


for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
    for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]-
>dy;index2++){
    for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]-
>dz;index3++){
       kordinat[index1][index2][index3]=1; }}}
}


ask=0;
if(gallboxes[a]->z!=0){

for(index3=gallboxes[a]->z-1;index3>=0;index3--){
for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]->dy;index2++){

ask+=kordinat[index1][index2][index3];

}}if(ask==0) z1++;if(ask!=0) break;}
}
if(z1!=0){

for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
    for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]-
>dy;index2++){
    for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]-
>dz;index3++){
       kordinat[index1][index2][index3]=0; }}}

gallboxes[a]->z=(gallboxes[a]->z)-z1;
```

EK-3 (Devam) C program kodu

```
for(index1=gallboxes[a]->x;index1<gallboxes[a]->x+gallboxes[a]->dx;index1++){
    for(index2=gallboxes[a]->y;index2<gallboxes[a]->y+gallboxes[a]-
>dy;index2++){
    for(index3=gallboxes[a]->z;index3<gallboxes[a]->z+gallboxes[a]-
>dz;index3++){
      kordinat[index1][index2][index3]=1; }}}
}
}

void uygunsonbelirle(void){

int index1,index2,a,b,c,a1,b1,c1,k;
int *ga,*gb,*gc,*ga1,*gb1,*gc1;

ga=&a;
gb=&b;
gc=&c;
ga1=&a1;
gb1=&b1;
gc1=&c1;

for(index1=0;index1<MAXBOX;index1++){
if(gallboxes[index1]->x!=BUYUK){

*ga=gallboxes[index1]->x;
*ga1=gallboxes[index1]->x+gallboxes[index1]->dx;

*gb=gallboxes[index1]->y;
*gb1=gallboxes[index1]->y+gallboxes[index1]->dy;

*gc=gallboxes[index1]->z;
*gc1=gallboxes[index1]->z+gallboxes[index1]->dz;

if(kordinat[*ga1][*gb][*gc]==0){
k=uygunsonbossec();

*(uygunsonx+k)=*ga1;
*(uygunsony+k)=*gb;
*(uygunsonz+k)=*gc;}

if(kordinat[*ga][*gb1][*gc]==0){
k=uygunsonbossec();

*(uygunsonx+k)=*ga;
*(uygunsony+k)=*gb1;
```

EK-3 (Devam) C program kodu

```
*(uygunsonz+k)=*gc;}

if(kordinat[*ga][*gb][*gc1]==0){
k=uygunsonbossec();

*(uygunsonx+k)=*ga;
*(uygunsony+k)=*gb;
*(uygunsonz+k)=*gc1;}
}}
}

void kordinatkayit(void){
int index1,index2,index3,index4,index5,a,b,c;
int k;

ilkkordinat();
uygunsonilk();

for(index1=0;index1<MAXBOX/MAX;index1++){
if(gbigboxes[index1]->x!=BUYUK && gbigboxes[index1]->x!=YOK){
gallboxes[gbigboxes[index1]->origin]->x=gbigboxes[index1]->x;
gallboxes[gbigboxes[index1]->origin]->y=gbigboxes[index1]->y;
gallboxes[gbigboxes[index1]->origin]->z=gbigboxes[index1]->z;
boxdkayittek(gbigboxes[index1]->origin);

for(index3=gallboxes[gbigboxes[index1]->origin]-
>x;index3<gallboxes[gbigboxes[index1]->origin]->x+gallboxes[gbigboxes[index1]-
>origin]->dx;index3++){
for(index4=gallboxes[gbigboxes[index1]->origin]-
>y;index4<gallboxes[gbigboxes[index1]->origin]->y+gallboxes[gbigboxes[index1]-
>origin]->dy;index4++){
for(index5=gallboxes[gbigboxes[index1]->origin]-
>z;index5<gallboxes[gbigboxes[index1]->origin]->z+gallboxes[gbigboxes[index1]-
>origin]->dz;index5++){

kordinat[index3][index4][index5]=1; }}}



for(index2=0;index2<MAX-1;index2++){

gallboxes[gbigboxes[index1]->uyeler[index2]]->x=gallboxes[gbigboxes[index1]-
>uyeler[index2]]->x+gbigboxes[index1]->x;
gallboxes[gbigboxes[index1]->uyeler[index2]]->y=gallboxes[gbigboxes[index1]-
>uyeler[index2]]->y+gbigboxes[index1]->y;
```

EK-3 (Devam) C program kodu

```
gallboxes[gbigboxes[index1]->uyeler[index2]]->z=gallboxes[gbigboxes[index1]-
>uyeler[index2]]->z+gbigboxes[index1]->z;
boxdkayittek(gbigboxes[index1]->uyeler[index2]);

for(index3=gallboxes[gbigboxes[index1]->uyeler[index2]]-
>x;index3<gallboxes[gbigboxes[index1]->uyeler[index2]]-
>x+gallboxes[gbigboxes[index1]->uyeler[index2]]->dx;index3++){
for(index4=gallboxes[gbigboxes[index1]->uyeler[index2]]-
>y;index4<gallboxes[gbigboxes[index1]->uyeler[index2]]-
>y+gallboxes[gbigboxes[index1]->uyeler[index2]]->dy;index4++){
for(index5=gallboxes[gbigboxes[index1]->uyeler[index2]]-
>z;index5<gallboxes[gbigboxes[index1]->uyeler[index2]]-
>z+gallboxes[gbigboxes[index1]->uyeler[index2]]->dz;index5++){

kordinat[index3][index4][index5]=1; }}}

 }
}

    if(gbigboxes[index1]->x==BUYUK){
gallboxes[gbigboxes[index1]->origin]->x=BUYUK;
for(index2=0;index2<MAX-1;index2++)

gallboxes[gbigboxes[index1]->uyeler[index2]]->x=BUYUK;
}}
 boxdsirala();


   uygunsonbelirle();

 }

void vsirala(void){
int index1,index2;
struct boxes yedek;
struct boxes *gyedek;

for(index1=0;index1<MAXBOX;index1++){
for(index2=index1;index2<MAXBOX;index2++){
if(gallboxes[index1]->v<gallboxes[index2]->v){
gyedek=gallboxes[index1];
gallboxes[index1]=gallboxes[index2];
gallboxes[index2]=gyedek;   }
}}
}
```

EK-3 (Devam) C program kodu

```
void uygunsondkayittek(int a){

*(uygunsond+a)=sqrt((*(uygunsonx+a))*(*(uygunsonx+a))+(*(uygunsony+a))*(*(u
ygunsony+a))+(*(uygunsonz+a))*(*(uygunsonz+a)));
}

void uygunsondkayit(void){
    int index;

    for(index=0;index<UYGUNSON;index++){
                                if(*(uygunsonx+index)!=BUYUK)

*(uygunsond+index)=sqrt((*(uygunsonx+index))*(*(uygunsonx+index))+(*(uyguns
ony+index))*(*(uygunsony+index))+(*(uygunsonz+index))*(*(uygunsonz+index)));
        else *(uygunsond+index)=BUYUK;
}
}

void uygunsondsirala(void){
    int index,index1,index2,yedek;
    float yedek1;

    for(index=0;index<UYGUNSON;index++){
    for(index1=index;index1<UYGUNSON;index1++){
      if(uygunsond[index]>uygunsond[index1]){
         yedek=*(uygunsonx+index);
         *(uygunsonx+index)=*(uygunsonx+index1);
         *(uygunsonx+index1)=yedek;

         yedek=*(uygunsony+index);
         *(uygunsony+index)=*(uygunsony+index1);
         *(uygunsony+index1)=yedek;

         yedek=*(uygunsonz+index);
         *(uygunsonz+index)=*(uygunsonz+index1);
         *(uygunsonz+index1)=yedek;

         yedek1=*(uygunsond+index);
         *(uygunsond+index)=*(uygunsond+index1);
         *(uygunsond+index1)=yedek1;
}}}
}

void yerlestirson(void){
int index1,index2,index3,index4,index5,index6,ask,k;
```

EK-3 (Devam) C program kodu

```
vsirala();

uygunsondkayit();

ask=BUYUK;
for(index1=0;index1<MAXBOX;index1++){

if(gallboxes[index1]->x==BUYUK && gallboxes[index1]->dx!=BUYUK){

for(index2=0;index2<UYGUNSON;index2++){

for(index6=1;index6<7;index6++){
                        ask=BUYUK;
rotasyonsec(index1,index6);


   if(*(uygunsonx+index2)+gallboxes[index1]->dx<MAXX &&
*(uygunsony+index2)+gallboxes[index1]->dy<MAXY &&
*(uygunsonz+index2)+gallboxes[index1]->dz<MAXZ  &&
*(uygunsonx+index2)!=BUYUK){

    ask=0;

for(index3=*(uygunsonx+index2);index3<*(uygunsonx+index2)+gallboxes[index1]-
>dx;index3++){

for(index4=*(uygunsony+index2);index4<*(uygunsony+index2)+gallboxes[index1]-
>dy;index4++){

for(index5=*(uygunsonz+index2);index5<*(uygunsonz+index2)+gallboxes[index1]-
>dz;index5++){
      ask+=kordinat[index3][index4][index5];
 }}}
    if(ask==0){


for(index3=*(uygunsonx+index2);index3<*(uygunsonx+index2)+gallboxes[index1]-
>dx;index3++){

for(index4=*(uygunsony+index2);index4<*(uygunsony+index2)+gallboxes[index1]-
>dy;index4++){

for(index5=*(uygunsonz+index2);index5<*(uygunsonz+index2)+gallboxes[index1]-
>dz;index5++){
      kordinat[index3][index4][index5]=1; }}}
```

EK-3 (Devam) C program kodu

```
    gallboxes[index1]->x=*(uygunsonx+index2);
    gallboxes[index1]->y=*(uygunsony+index2);
    gallboxes[index1]->z=*(uygunsonz+index2);



    uygunsonx[index2]=BUYUK;
if(kordinat[gallboxes[index1]->x+gallboxes[index1]->dx][gallboxes[index1]-
>y][gallboxes[index1]->z]==0){
    k=uygunsonbossec();
    *(uygunsonx+k)=gallboxes[index1]->x+gallboxes[index1]->dx;
    *(uygunsony+k)=gallboxes[index1]->y;
    *(uygunsonz+k)=gallboxes[index1]->z;
    printf("%d\n",gallboxes[index1]->z);
    uygunsondkayittek(k);}
  if(kordinat[gallboxes[index1]->x][gallboxes[index1]->y+gallboxes[index1]-
>dy][gallboxes[index1]->z]==0){
    k=uygunsonbossec();
    *(uygunsonx+k)=gallboxes[index1]->x;
    *(uygunsony+k)=gallboxes[index1]->y+gallboxes[index1]->dy;
    *(uygunsonz+k)=gallboxes[index1]->z;
uygunsondkayittek(k);}
if(kordinat[gallboxes[index1]->x][gallboxes[index1]->y][gallboxes[index1]-
>z+gallboxes[index1]->dz]==0){
    k=uygunsonbossec();
    *(uygunsonx+k)=gallboxes[index1]->x;
    *(uygunsony+k)=gallboxes[index1]->y;
    *(uygunsonz+k)=gallboxes[index1]->z+gallboxes[index1]->dz;
uygunsondkayittek(k);
  }

uygunsondsirala();



    break; } //if ask=0
    }       //if uygunson tamamsa
    if(ask==0)
    break;
    }       //for rotasyonsec
    if(ask==0)
    break;
    }
```

EK-3 (Devam) C program kodu

```
            //for uygunson
    }        //gallbox buyuk degilse
    }        //for galbox
}

int vgecicihesapkontrol(int in,int s){

int dx,dy,dz;
int index1,toplam=0,vgecici;

if(s==0) return 0;

 dx=gallboxes[secbox[0]]->x+gallboxes[secbox[0]]->dx;
       for(index1=0;index1<MAX;index1++){
       if(dx<gallboxes[secbox[index1]]->x+gallboxes[secbox[index1]]->dx)
       dx=gallboxes[secbox[index1]]->x+gallboxes[secbox[index1]]->dx;
       }

       dy=gallboxes[secbox[0]]->y+gallboxes[secbox[0]]->dy;
       for(index1=0;index1<MAX;index1++){
       if(dy<gallboxes[secbox[index1]]->y+gallboxes[secbox[index1]]->dy)
       dy=gallboxes[secbox[index1]]->y+gallboxes[secbox[index1]]->dy;
       }

       dz=gallboxes[secbox[0]]->z+gallboxes[secbox[0]]->dz;
       for(index1=0;index1<MAX;index1++){
       if(dz<gallboxes[secbox[index1]]->z+gallboxes[secbox[index1]]->dz)
       dz=gallboxes[secbox[index1]]->z+gallboxes[secbox[index1]]->dz;
       }
       vgecici=dx*dy*dz;
       toplam=0;

for(index1=0;index1<MAX;index1++){
       toplam+=gallboxes[secbox[index1]]->v; }
       if(vgecici-toplam<100000000){
       gbigboxes[in]->dx=dx;
       gbigboxes[in]->dy=dy;
       gbigboxes[in]->dz=dz;
       gbigboxes[in]->v=vgecici;
       gbigboxes[in]->kayip=vgecici-toplam;

       return vgecici;}
else    return 0;

}
```

EK-3 (Devam) C program kodu

```
void bigboxkayit(int in){

int index3,index5,b;
int *a;



        for(index5=0;index5<MAX;index5++){
        if(gallboxes[secbox[index5]]->x==0 && gallboxes[secbox[index5]]->y==0
&& gallboxes[secbox[index5]]->z==0){
        gbigboxes[in]->origin=secbox[index5];}
        else{
        a=&b;


srand(time(NULL));
for(index3=0;index3<BUYUK;index3++){
b=rand()%(MAX-1);

if (gbigboxes[in]->uyeler[*a]==BUYUK){
gbigboxes[in]->uyeler[*a]=secbox[index5];

break;}
}}}}

void br(int problemno){

    FILE *gdosya;
int problemsayisi=100,i,i1,k=0;
int a;

problemno=problemno-1;
struct tumveriler{
        int x[tipsayisi];
        int y[tipsayisi];
        int z[tipsayisi];
        int kactane[tipsayisi];


};
struct tumveriler veriler[problemsayisi];
struct tumveriler *gveriler[problemsayisi];
for(i=0;i<problemsayisi;i++)
gveriler[i]=&(veriler[i]);
```

EK-3 (Devam) C program kodu

```
if((gdosya=fopen("br3.txt","r"))==NULL) {
     printf("dosya acilamadi");
     exit(1);}


     rewind(gdosya);

     fscanf(gdosya,"%d",&a);

     fscanf(gdosya,"%d",&a);
     for(i=0;i<problemsayisi;i++){
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&a);

      for(i1=0;i1<tipsayisi;i1++){
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&gveriler[i]->x[i1]);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&gveriler[i]->y[i1]);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&gveriler[i]->z[i1]);
      fscanf(gdosya,"%d",&a);
      fscanf(gdosya,"%d",&gveriler[i]->kactane[i1]);
      }
      }
      fclose(gdosya);
      if((gdosya=fopen("birimverileri.txt","w"))==NULL) {
     printf("dosya acilamadi");
     exit(1);}
     rewind(gdosya);

     for(i=0;i<tipsayisi;i++){
     for(i1=0;i1<(gveriler[problemno]->kactane[i]);i1++){
     fprintf(gdosya,"%d\t",gveriler[problemno]->x[i]);
     fprintf(gdosya,"%d\t",gveriler[problemno]->y[i]);
     fprintf(gdosya,"%d\t",gveriler[problemno]->z[i]);
k++;
     fprintf(gdosya,"%d\n",gveriler[problemno]->x[i]*gveriler[problemno]-
>y[i]*gveriler[problemno]->z[i]);
      }
      }
```

EK-3 (Devam) C program kodu

```
      fclose(gdosya);
printf("%d adet birim uretildi",k);
}
void uygunensonbelirlex(void){
int index1,index3,ask1;
for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK){

 ask1=0;
    for(index3=*(uygunsonx+index1);index3<MAXX;index3++){
    ask1+=kordinat[index3][*(uygunsony+index1)][*(uygunsonz+index1)];
}
if(ask1!=0) *(uygunsonx+index1)=BUYUK;
}
}
}

void uygunensonbelirley(void){
int index1,index3,ask2;
for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK){

 ask2=0;
    for(index3=*(uygunsony+index1);index3<MAXY;index3++){
    ask2+=kordinat[*(uygunsonx+index1)][index3][*(uygunsonz+index1)];
}
if(ask2!=0) *(uygunsonx+index1)=BUYUK;
}
}
}

void uygunensonbelirlez(void){
int index1,index3,ask3;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK){

ask3=0;
    for(index3=*(uygunsonz+index1);index3<MAXZ;index3++){
    ask3+=kordinat[*(uygunsonx+index1)][*(uygunsony+index1)][index3];
}
if(ask3!=0) *(uygunsonx+index1)=BUYUK;
}
}
}
```

EK-3 (Devam) C program kodu

```
void uygunkaydirx(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsonx+index1)+1<=MAXX-20){

ask=kordinat[*(uygunsonx+index1)+1][*(uygunsony+index1)][*(uygunsonz+index1
)];
if(ask==0)
*(uygunsonx+index1)=*(uygunsonx+index1)+1;

else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
}
}

void uygunkaydiry(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsony+index1)+1<=MAXY-20){
ask=kordinat[*(uygunsonx+index1)][*(uygunsony+index1)+1][*(uygunsonz+index1
)];
if(ask==0)
*(uygunsony+index1)=*(uygunsony+index1)+1;

else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
}
}


void uygunkaydircapraz(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsonx+index1)+1<=MAXX-20 &&
*(uygunsony+index1)+1<=MAXY-20 && *(uygunsonz+index1)+1<=MAXZ-20){
ask=kordinat[*(uygunsonx+index1)+1][*(uygunsony+index1)+1][*(uygunsonz+inde
x1)+1];
```

EK-3 (Devam) C program kodu

```
if(ask==0){
*(uygunsonx+index1)=*(uygunsonx+index1)+1;
*(uygunsony+index1)=*(uygunsony+index1)+1;
*(uygunsonz+index1)=*(uygunsonz+index1)+1;
}
else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
}
}


void uygunkaydirz(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsonz+index1)+1<=MAXZ-20){
ask=kordinat[*(uygunsonx+index1)][*(uygunsony+index1)][*(uygunsonz+index1)+
1];
if(ask==0)
*(uygunsonz+index1)=*(uygunsonz+index1)+1;

else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
}
}

void uygunkaydirxy(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsonx+index1)+1<=MAXX-20 &&
*(uygunsony+index1)+1<=MAXY-20){
ask=kordinat[*(uygunsonx+index1)+1][*(uygunsony+index1)+1][*(uygunsonz+inde
x1)];
if(ask==0){
*(uygunsonx+index1)=*(uygunsonx+index1)+1;
*(uygunsony+index1)=*(uygunsony+index1)+1;
}
else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
```

EK-3 (Devam) C program kodu

```
}
}

void uygunkaydirxz(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsonx+index1)+1<=MAXX-20 &&
*(uygunsonz+index1)+1<=MAXZ-20){
ask=kordinat[*(uygunsonx+index1)+1][*(uygunsony+index1)][*(uygunsonz+index1
)+1];
if(ask==0){
*(uygunsonx+index1)=*(uygunsonx+index1)+1;
*(uygunsonz+index1)=*(uygunsonz+index1)+1;}

else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
}
}

void uygunkaydiryz(void){

int index1,ask;

for(index1=0;index1<UYGUNSON;index1++){
if(*(uygunsonx+index1)!=BUYUK && *(uygunsony+index1)+1<=MAXY-20 &&
*(uygunsonz+index1)+1<=MAXZ-20){
ask=kordinat[*(uygunsonx+index1)][*(uygunsony+index1)+1][*(uygunsonz+index1
)+1];
if(ask==0){
*(uygunsony+index1)=*(uygunsony+index1)+1;
*(uygunsonz+index1)=*(uygunsonz+index1)+1;
}
else *(uygunsonx+index1)=BUYUK;
}
else *(uygunsonx+index1)=BUYUK;
}
}
void yerlestirenson(void){

int index1;
uygunsonbelirle();
uygunensonbelirlex();
```

EK-3 (Devam) C program kodu

```
for(index1=0;index1<100;index1++){

uygunkaydirx();

yerlestirson();

}

uygunsonbelirle();
uygunensonbelirley();
for(index1=0;index1<20;index1++){

uygunkaydiry();

yerlestirson();
}

uygunsonbelirle();
uygunensonbelirlez();
for(index1=0;index1<20;index1++){

uygunkaydirz();

yerlestirson();
}

uygunsonbelirle();
uygunensonbelirlex();
for(index1=0;index1<20;index1++){

uygunkaydircapraz();

yerlestirson();

}
uygunsonbelirle();
uygunensonbelirley();
for(index1=0;index1<20;index1++){

uygunkaydircapraz();

yerlestirson();

}
uygunsonbelirle();
```

EK-3 (Devam) C program kodu

```
uygunensonbelirlez();
for(index1=0;index1<20;index1++){

uygunkaydircapraz();

yerlestirson();

}
uygunsonbelirle();
uygunensonbelirlex();
for(index1=0;index1<20;index1++){

uygunkaydirxy();

yerlestirson();

}
uygunsonbelirle();
uygunensonbelirley();
for(index1=0;index1<20;index1++){

uygunkaydirxy();

yerlestirson();

}
uygunsonbelirle();
uygunensonbelirlex();
for(index1=0;index1<20;index1++){

uygunkaydirxz();

yerlestirson();

}
uygunsonbelirle();
uygunensonbelirlez();
for(index1=0;index1<20;index1++){

uygunkaydirxz();

yerlestirson();

}
uygunsonbelirle();
```

EK-3 (Devam) C program kodu

```
uygunensonbelirley();
for(index1=0;index1<20;index1++){

uygunkaydiryz();

yerlestirson();

}
uygunsonbelirle();
uygunensonbelirlez();
for(index1=0;index1<20;index1++){

uygunkaydiryz();

yerlestirson();

}
}
int main()
{
FILE *gdosya;

int
ind=0,index=0,index1=0,index3=0,index4=0,index5=0,dx,dy,dz,vgecici,toplam=0,to
plam1=0,toplam2=0,yedek,lev=1,k=0,index10,maxb;;
int index2=0,j=0,b,yerhacim=0,dogrula,toplamx=0,toplamy=0,toplamz=0,r=0;
int *a;
float yedek1,a1,a2,a3,maxkare,ortx,orty,ortz,verim;
float kare[MAXBOX];
float toplamkare,ortalamakare,hacimoran;


printf("lutfen bekleyiniz....\n");


for(index=0;index<MAXBOX;index++){

gallboxes[index]=&(allboxes[index]);
gbigboxes[index]=&(bigboxes[index]);
 }

 for(index10=1;index10<101;index10++){
 br(index10);
 maxb=0;
for(index=0;index<MAXBOX/MAX;index++){
```

EK-3 (Devam) C program kodu

```
gbigboxes[index]->origin=BUYUK;
for(index1=0;index1<MAX-1;index1++)
gbigboxes[index]->uyeler[index1]=BUYUK;

gbigboxes[index]->x=BUYUK;
}
for(index=0;index<MAXBOX;index++){
gallboxes[index]->x=BUYUK;
gallboxes[index]->y=BUYUK;
gallboxes[index]->z=BUYUK;
gallboxes[index]->dx=BUYUK;
gallboxes[index]->dy=BUYUK;
gallboxes[index]->dz=BUYUK;
gallboxes[index]->v=BUYUK;
gallboxes[index]->d=BUYUK;



}

printf("veriler dosyadan okunuyor...\n");
     verioku();
      toplamx=0;
        toplamy=0;
        toplamz=0;

      for(index1=0;index1<MAXBOX;index1++){
if(gallboxes[index1]->dx!=BUYUK){
        toplamx+=gallboxes[index1]->dx;
        toplamy+=gallboxes[index1]->dy;
        toplamz+=gallboxes[index1]->dz;}}

        for(index1=0;index1<MAXBOX;index1++){
        if(gallboxes[index1]->dx!=BUYUK)
        maxb++;}

        ortx=(float)toplamx/maxb;
        orty=(float)toplamy/maxb;
        ortz=(float)toplamz/maxb;



     for (index=0;index<MAX;index++){
     *(icsecim+index)=0;
     }
```

EK-3 (Devam) C program kodu

```
    grupla();
    printf("buyuk birimler olusturuluyor...\n");

    for(ind=0;ind<MAXBOX/MAX;ind++){
    lev=100; j=0;  uygunilk();  ilkkordinat();  for
(index=0;index<MAX;index++){*(icsecim+index)=0;}
    rassalboxsec(ind);


    for(index=0;index<100;index++){
      r=1;

      for(index1=0;index1<MAX;index1++){

      if(yerlestir()==0) {r=0;break;}}
       if(vgecicihesapkontrol(ind,r)!=0 && r==1){ bigboxkayit(ind);  break;  }
      if(r==0) {for(index2=0;index2<MAX;index2++){gallboxes[secbox[index2]]-
>x=BUYUK;}gbigboxes[ind]->x=YOK;}
      j++;  if((j+1)%5==0)  {  lev++;}
       }

       }

       for(index=0;index<100;index++){
       ilkkordinat();
       boxdilk();

       printf("buyuk birimler yerlestiriliyor...\n");
       yerlestirbig();


       ilkkordinat();

       kordinatkayit();
       printf("buyuk birimler yerlestirildi, ucuncu asama yerlestirme islemi
yapiliyor...\n");
       yerlestirson();
       printf("kaydirma islemi gerceklestiriliyor...\n");
       for(index1=0;index1<MAXBOX;index1++){
      if(gallboxes[index1]->x!=BUYUK)
      kaydir(index1);}
printf("4. yerlesitirme islemi gerceklestiriliyor...\n");
      uygunsonilk();

      uygunsonbelirle();
```

EK-3 (Devam) C program kodu

```
yerlestirson();


for(index1=0;index1<MAXBOX;index1++){
        if(gallboxes[index1]->x!=BUYUK)
        kaydir(index1);}
printf("4. yerlesitirme islemi gerceklestiriliyor...\n");
        uygunsonilk();

        uygunsonbelirle();
yerlestirson();
printf("son yerlesitirme islemi gerceklestiriliyor...\n");
      yerlestirenson();
          toplam=0;
         for(index1=0;index1<MAXBOX;index1++){
         if(gallboxes[index1]->x!=BUYUK)
         toplam+=gallboxes[index1]->v;   }
         if(toplam/(MAXX*MAXY*MAXZ)>=0)
         break;
         }




          printf("dogrulaniyor....\n");
          printf("sonuclar yazdiriliyor...\n");




  k=0;
     for(index1=0;index1<MAXX;index1++){
     for(index2=0;index2<MAXY;index2++){
     for(index3=0;index3<MAXZ;index3++){
        k+=kordinat[index1][index2][index3];}}}
        if(toplam-k==0) printf("1. dogrulama tamam\n");



      if((gdosya=fopen("bigboxes.txt","w"))==NULL) {
      printf("dosya acilamadi");
      exit(1);}
```

EK-3 (Devam) C program kodu

```
    for(index=0;index<MAXBOX/MAX;index++){
    fprintf(gdosya,"\n%d    %d    %d    %d    %d\t\t %d    %d
%d\t",gbigboxes[index]->dx,gbigboxes[index]->dy,gbigboxes[index]-
>dz,gbigboxes[index]->kayip,gbigboxes[index]->origin,gbigboxes[index]-
>x,gbigboxes[index]->y,gbigboxes[index]->z);
for(index1=0;index1<MAX-1;index1++)
fprintf(gdosya,"%d\t",gbigboxes[index]->uyeler[index1]);
}
    for(index=0;index<UYGUNBIG;index++)
    fprintf(gdosya,"uygunbig:
%d\t%d\t%d\n",uygunbigx[index],uygunbigy[index],uygunbigz[index]);

    for(index=0;index<UYGUNSON;index++)
    fprintf(gdosya,"uygunson:
%d\t%d\t%d\n",uygunsonx[index],uygunsony[index],uygunsonz[index]);


    toplam=0;
    for(index=0;index<MAXBOX/MAX;index++)
    toplam+=gbigboxes[index]->v;
    fprintf(gdosya,"bigbox toplam hacmi:%d\n",toplam);

    dx=gbigboxes[0]->x+gbigboxes[0]->dx;
      for(index1=0;index1<MAXBOX/MAX;index1++){
      if(dx<gbigboxes[index1]->x+gbigboxes[index1]->dx && gbigboxes[index1]-
>x!=BUYUK)
      dx=gbigboxes[index1]->x+gbigboxes[index1]->dx;
      }
      dy=gbigboxes[0]->y+gbigboxes[0]->dy;
      for(index1=0;index1<MAXBOX/MAX;index1++){
      if(dy<gbigboxes[index1]->y+gallboxes[index1]->dy && gbigboxes[index1]-
>x!=BUYUK)
      dy=gbigboxes[index1]->y+gbigboxes[index1]->dy;
      }
      dz=gbigboxes[0]->z+gbigboxes[0]->dz;
      for(index1=0;index1<MAXBOX/MAX;index1++){
      if(dz<gbigboxes[index1]->z+gbigboxes[index1]->dz && gbigboxes[index1]-
>x!=BUYUK)
      dz=gbigboxes[index1]->z+gbigboxes[index1]->dz;
      }
      vgecici=dx*dy*dz;
      fprintf(gdosya,"%d    %d    %d    %d\n",dx,dy,dz,vgecici);

      toplam=0;
      for(index=0;index<MAXBOX;index++){
```

EK-3 (Devam) C program kodu

```
        if(gallboxes[index]->x!=BUYUK)
        toplam+=gallboxes[index]->v;   }
        fprintf(gdosya,"yerlestirilen boxlar:%d\n",toplam);

    fclose(gdosya);

    if((gdosya=fopen("boxkordinat.txt","w"))==NULL) {
    printf("dosya acilamadi");
    exit(1);}

    j=0;
    fprintf(gdosya,"yerlestirilen birimler\n");
    yerhacim=0;
    for(index=0;index<MAXBOX;index++){
    if(gallboxes[index]->x!=BUYUK){

    fprintf(gdosya,"%d     %d       %d       %d       %d       %d
%d\n",gallboxes[index]->dx,gallboxes[index]->dy,gallboxes[index]-
>dz,gallboxes[index]->x,gallboxes[index]->y,gallboxes[index]->z,index);
    j++;
    yerhacim+=gallboxes[index]->v;
      }
    }
    fprintf(gdosya,"yerlestirilemeyen birimler\n");
    for(index=0;index<MAXBOX;index++){
    if(gallboxes[index]->x==BUYUK){

    fprintf(gdosya,"%d     %d       %d\n",gallboxes[index]->dx,gallboxes[index]-
>dy,gallboxes[index]->dz);
    }}

    fprintf(gdosya,"%d adet birim yerlesti\n",j);
    fprintf(gdosya,"yerlesen birimlerin toplam hacmi=%d\n",yerhacim);
    verim=(double)(yerhacim)/(double)((MAXX-1)*(MAXY-1)*(MAXZ-1));
    verim=verim*100;
    fprintf(gdosya,"VERİM=%f\n",verim);


    fclose(gdosya);
    dogrula=0;
    for(index1=0;index1<MAXBOX;index1++){
    for(index2=index1+1;index2<MAXBOX;index2++){
    if(gallboxes[index1]->x!=BUYUK && gallboxes[index2]->x!=BUYUK &&
gallboxes[index1]->x==gallboxes[index2]->x && gallboxes[index1]-
>y==gallboxes[index2]->y && gallboxes[index1]->z==gallboxes[index2]->z)
```

EK-3 (Devam) C program kodu

```
    dogrula++;  }}
    if(dogrula==0)
    printf("2. dogrulama tamam\n");
    else
    printf("yanlis hesaplama\n");

    if(MAXX>MAXY)
    a1=(float)MAXY/MAXX;
    else
    a1=(float)MAXX/MAXY;


    if(MAXX>MAXZ)
    a2=(float)MAXZ/MAXX;
    else
    a2=(float)MAXX/MAXZ;


    if(MAXZ>MAXY)
    a3=(float)MAXY/MAXZ;
    else
    a3=(float)MAXZ/MAXY;

    maxkare=a1+a2+a3;

    printf("max kare=%f\n",a1+a2+a3);
toplamkare=0;
    for(index1=0;index1<MAXBOX;index1++)
    kare[index1]=0;

    for(index1=0;index1<MAXBOX;index1++){
    if(gallboxes[index1]->x!=BUYUK){
      if(gallboxes[index1]->dx>gallboxes[index1]->dy)
      a1=(float)gallboxes[index1]->dy/gallboxes[index1]->dx;
      else
      a1=(float)gallboxes[index1]->dx/gallboxes[index1]->dy;

      if(gallboxes[index1]->dx>gallboxes[index1]->dz)
      a2=(float)gallboxes[index1]->dz/gallboxes[index1]->dx;
      else
      a2=(float)gallboxes[index1]->dx/gallboxes[index1]->dz;

      if(gallboxes[index1]->dy>gallboxes[index1]->dz)
      a3=(float)gallboxes[index1]->dz/gallboxes[index1]->dy;
      else
```

EK-3 (Devam) C program kodu

```
        a3=(float)gallboxes[index1]->dy/gallboxes[index1]->dz;

        kare[index1]=a1+a2+a3;
        toplamkare+=kare[index1];
        }}
        if(j!=0) {ortalamakare=(float)toplamkare/j;
hacimoran=(float)(toplam/j)/(MAXX*MAXY*MAXZ);}
        else {ortalamakare=0; hacimoran=0;}
        if(ortalamakare!=0) printf("ortalama kare=%f\n",ortalamakare);
        if(ortalamakare==0) {printf("ortalama kare hesaplanamadi\n"); printf("yerlesim
gerceklestirilemedi, verilerinizi tekrar kontrol edin...\n");}




      if((gdosya=fopen("analiz.txt","a"))==NULL) {
    printf("dosya acilamadi");
    exit(1);}

       fprintf(gdosya,"\n%d-%f\t",index10,maxkare);
       fprintf(gdosya,"%f\t",ortalamakare);
       fprintf(gdosya,"%f\t",hacimoran);
       fprintf(gdosya,"%f\t",verim);
       fprintf(gdosya,"%f\t",ortx);
       fprintf(gdosya,"%f\t",orty);
       fprintf(gdosya,"%f\t",ortz);

       fclose(gdosya);

       if((gdosya=fopen("boxkordinatlar.txt","w"))==NULL) {
    printf("dosya acilamadi");
    exit(1);}
    fprintf(gdosya,"%d\n",j);

    for(index=0;index<MAXBOX;index++){
    if(gallboxes[index]->x!=BUYUK){
    fprintf(gdosya,"%d\t",gallboxes[index]->x);
    fprintf(gdosya,"%d\t",gallboxes[index]->y);
    fprintf(gdosya,"%d\t",gallboxes[index]->z);
    fprintf(gdosya,"%d\t",gallboxes[index]->dx);
    fprintf(gdosya,"%d\t",gallboxes[index]->dy);
    fprintf(gdosya,"%d\n",gallboxes[index]->dz);

    }}
    fprintf(gdosya,"%d\t%d\t%d\n",MAXX-1,MAXY-1,MAXZ-1);
```

EK-3 (Devam) C program kodu

```
    fclose(gdosya);

}
    printf("\a");

    system("PAUSE");
    return 0;
}
```

EK-4. Sonuçlar

ORTKARE	HACİMORAN	VERİM	ORTALAMA X	ORTALAMA Y	ORTALAMA Z
1.730512	0.01003	78.233162	103.01786	68.01786	37.232143
2.054023	0.008014	80.94043	70.442032	51.246376	42.391304
1.545004	0.008508	76.569672	104.196854	67.881889	34.086613
1.693596	0.007036	86.540215	74.522842	54.284264	30.055838
1.826276	0.007581	84.908714	89.330879	61.948528	39.544117
1.960712	0.007596	85.830399	80.285713	62	38.904762
2.039224	0.007355	82.375595	87.53968	55.087303	44.373016
2.156897	0.005809	85.977623	69.644447	55.277779	40.255554
1.962359	0.010426	82.366707	89.683167	78.47525	42.405941
1.918292	0.008417	86.695496	93.746155	59.369232	39.46154
1.916732	0.009759	76.11927	83.294121	67.441177	47.64706
1.739881	0.010137	90.217583	107.21154	68.07692	41.471153
1.589739	0.004207	88.337029	71.169014	44.718311	27.647888
1.702195	0.009076	86.22171	84.060608	57.242424	37.454544
2.001705	0.008914	78.446785	84.705879	62.15126	46.008404
2.213555	0.006729	75.361702	69.226418	60.08176	42.289307
1.856086	0.004776	88.828644	74.901405	49.267605	35.314552
2.136493	0.013779	88.187592	94.048782	78.231705	47.817074
2.035983	0.009215	81.094704	78.384613	54.384617	44.615383
1.988106	0.011083	83.12513	95	66.454544	53.727272
1.888025	0.015181	85.013863	107.683548	68.645569	46.987343
1.953806	0.00907	76.183899	83.517242	68.560349	44.827587
1.690254	0.00873	89.04641	89.828125	67.609375	34.578125
1.859453	0.010318	79.450256	87.605263	67.236839	45.552631
1.982662	0.008151	79.068893	83.669067	60.338131	41.899281
2.111154	0.008227	78.981468	86.540321	69.685486	42.44355
1.656057	0.009782	85.101059	103.864075	67.650482	40.029125
2.415249	0.01205	72.298714	73.603607	60.144146	55.981983
1.795176	0.008957	79.717697	98.238533	60.752293	46.21101
1.995095	0.002809	82.313599	55.281483	42.874073	30.182716
2.072365	0.007869	82.628822	80.744186	62.348839	46.720932
1.777692	0.007077	87.754791	83.567741	58.954838	36.645161
2.148511	0.003529	87.884956	60.897163	45.351063	34.758865
1.78038	0.00662	91.354378	86.71875	56.068748	35.71875
1.725114	0.007211	84.365891	92.279999	62.173332	35.253334
1.963217	0.005881	87.034126	82.947678	49.348839	36.05814
2.091797	0.010778	84.064667	86.414139	68.141411	47.737373
1.772744	0.013363	80.176804	107.483871	73.107529	40.021507
1.997386	0.003866	87.755966	67.687241	43.950619	35.539093
2.084593	0.010196	82.584724	87.532707	71.149536	44.710281
2.011947	0.011521	85.255745	87.954132	59.972477	43.899082
2.287772	0.007045	83.133522	76.238098	55.208332	41.69643
2.035073	0.007249	84.809853	74.418442	63.439716	39.539005
1.416337	0.007294	80.233734	100.669014	75.929581	27.866198
1.715627	0.007879	87.45443	91.021431	57.728573	38.057144

EK-4 (Devam) Sonuçlar

ORTKARE	HACİMORAN	VERİM	ORTALAMA X	ORTALAMA Y	ORTALAMA Z
1.800252	0.010166	88.442032	89.181038	68.75	38.051723
1.46992	0.007551	87.589729	116.204376	61.014599	33.773724
2.140845	0.006611	76.024437	79.235291	57.235294	43.633987
2.255649	0.013706	78.125114	90.720932	68.790695	53.581394
2.111254	0.006543	85.058105	69.047623	55.57143	40.142857
2.030584	0.008549	82.066719	78.542633	66.395348	42.728683
1.759338	0.006385	91.946342	83.721924	54.406418	31.673798
1.64475	0.007206	87.19854	85.247055	62.299999	30.917646
1.693195	0.007898	82.13681	87.922539	61.53521	38.711269
1.739544	0.008706	87.931618	97.666664	73.947365	37.666668
1.953329	0.00266	86.45842	64.710785	34.580883	31.014706
2.105587	0.008093	81.736389	82.669357	65.80645	45.088711
1.725739	0.010871	84.790581	95.91304	70.565216	41.586956
2.197231	0.005272	86.467613	68.379486	54.928204	41.292309
2.114384	0.009219	79.283287	86.390472	64.276192	51.24762

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ENSARİ,Mustafa

Uyruğu : T.C.

Doğum tarihi ve yeri : 13.02.1981,MARDİN

Medeni hali : Bekar

e-mail : ensmustafa@mardincimento.com.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Sakarya Üniversitesi	2003
Lise	Ankara Ömer Seyfettin Lisesi	1998

İş Deneyimi

Yıl	Yer	Görev
2005	YAZAKI Otomotiv-BURSA	Endüstri Müh.
2006	OYAK Mardin Çimento	Satın Alma

Yabancı Dil

İngilizce

Hobiler

Yüzme, Bilgisayar oyunları, Satranç.