YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

UÇAK YÜKLEME OPTİMİZASYONU

Makine Mühendisi Alper DAÇE

FBE Makine Mühendisliği Anabilim Dalı Makine Teorisi ve Kontrol Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Vasfi Emre ÖMÜRLÜ

İstanbul, 2007

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

I	Moment ve moment kolu çarpımı sonucu olan indeks değeri
p	Ağırlığı 4626 kg ile 5145 kg arasında olan PMC tipi konteyner sayısı
r	Ağırlığı 4281 kg ile 4626 kg arasında olan PMC tipi konteyner sayısı
x	Ağırlığı 5670 kg ile 6804 kg arasında olan PMC tipi konteyner sayısı
x_1	Ağırlığı 4626 kg ile 6033 kg arasında olan PAG tipi konteyner sayısı
y	Ağırlığı 5446 kg ile 5670 kg arasında olan PMC tipi konteyner sayısı
y_1	Ağırlığı 4321 kg ile 4626 kg arasında olan PAG tipi konteyner sayısı
z	Ağırlığı 5145 kg ile 5446 kg arasında olan PMC tipi konteyner sayısı
L	Yerleştirilmesi gereken hafif konteyner sayısı
N_{kont}	Toplam konteyner sayısı
N_{PAG}	Yerleştirilmemiş PAG tipi konteyner sayısı
N_{PMC}	Yerleştirilmemiş PMC tipi konteyner sayısı
$N_{max.kont}$	İlgili yerleşim durumu için izin verilen azami konteyner sayısı

KISALTMA LİSTESİ

CG	Ağırlık Merkezi
MLW	Azami İniş Ağırlığı
MTOW	Azami Kalkış Ağırlığı
MZFW	Azami Yakıtsız Ağırlık
OW	Operasyon Ağırlığı
DOW	Kuru Operasyon Ağırlığı
BOW	Temel Operasyon Ağırlığı
TOW	Kalkış Ağırlığı
ZFW	Yakıtsız Ağırlık
LDW	İniş Ağırlığı
MAC	Ortalama Aerodinamik Kordo Hattı

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu çalışmada kargo uçaklarının yüklenmesindeki insan faktörünün azaltılması, yüklemenin ideal değerlere daha yakın yapılabilmesi ve olası en fazla yüklemenin yapılabilmesi amaçlanmıştır. Bu sayede daha emniyetli, daha ekonomik ve daha az süre de operasyon gerçekleştirilebilecektir. Bu amaç doğrultusunda yüklemenin nasıl daha iyi yapılabileceği araştırılmış, değişik çözüm yaklaşımları değerlendirilmiştir.

Buluşsal çözüme dayalı yaklaşımla yükleme algoritması tasarlanmış ve bilgisayar programı haline getirilmiştir. Gerçek örnekler üzerinden program test edilmiş ve oldukça iyi sonuçlar ortaya çıkmıştır. Sonuç olarak bilgisayar yardımı ile yapılan yüklemenin birçok avantajı olduğu saptanmış fakat bilgisayar teknolojisinin gelişmesi ile daha tatmin edici sonuçlar elde edilebileceği düşünülmektedir.

Tez çalışması sırasında programlama ve çözüm yollarıyla ilgili fedakarane yardımlarından dolayı Arş. Gör. Onur İsmail İLKORUR'a, teknik ve yükleme bilgileri için THY Plan ve Performans Müd.'ne, A310 Yükleme Uzmanı Erol SAVAŞ'a ve tüm yardımları için tez danışmanım Yrd.Doç.Dr. Vasfi Emre ÖMÜRLÜ'ye teşekkür ediyorum.

ÖZET

Kargo uçaklarının yüklenmesi hem toplam ağırlık, hem de ağırlık merkezi dikkate alındığında yolcu uçaklarına göre daha kritiktir. Mevcut yükleme ise insan eliyle yapılmakta, bu yüzden bir takım ekonomik ve zamansal kayıplar oluşmaktadır.

Bu çalışmada A310-304F modeli kargo uçağının yükleme optimizasyonu yapılmıştır. Amaç yüklemenin insana dayalı olmaktan çıkartılıp, bilgisayar çözümü ile en emniyetli en ekonomik yüklemenin yapılmasıdır.

Asıl çözümde ortaya çıkan amaç ise uçağa mümkün olabildiğince kargo yüklerken, CG'nin de ideale yaklaştırılması ile yakıt tasarrufu sağlamaktır. Çalışmada kullanılan değerler hali hazırda T.H.Y. A.O. 'da uçmakta olan kargo uçağından alınmıştır. Yükleme optimizasyonu için buluşsal çözüm algoritması tasarlanmış ve C++'da bilgisayar programı haline getirilmiştir. Program hangi kargoların yükleneceğine ve nerelere yükleneceğine karar vermektedir.

Programın güvenilirliğini test etmek üzere, daha önce yüklemeciler tarafından gerçekleştirilen yerleştirmeler bir de programa yaptırılmış ve sonuçlar karşılaştırılmıştır. Bilgisayar programı denemelerde daha iyi sonuçlar vermiş ve yüklemenin bu şekilde yapılması halinde hem daha emniyetli, hem de daha ekonomik operasyon yapılabileceği anlaşılmıştır..

Anahtar Kelimeler: Uçak, kargo, yükleme, optimizasyon, ağırlık merkezi, ağırlık-denge, buluşsal

ABSTRACT

The loading of cargo aircraft is more critical in terms of total weight and center of gravity. At the moment, loadmasters are executing the loading manually in a way that can result some economical and time loss.

This research is based on loading optimization of an A310-304F cargo aircraft. The main goal is to eliminate the human factor on loading, making it safer and more economic with the help of computers.

The goal of this research is to load as much freight as possible in aircraft while balancing the load in order to minimize fuel consumption during flight. The pre-used loading data used in this work has be obtained from an actual freighter aircraft operating for Turkish Airlines. A heuristic algorithm is designed for loading problem and C++ computer code is used to solve the problem. The program decides which containers in which position will be loaded and which to leave on the ground.

Former manual loading examples by the loadmasters are solved and loaded by the computer to test the reliability of the program and a comparison is done between both results. It is seen that the computer program results are much more ideal and computer loading will be safer, stable and economical.

Keywords: Aircraft, freight, cargo, loading, optimization, center of gravity, mass and balance, heuristic

1. GİRİŞ

1.1 Önceki Çalışmalar

Konuyla ilgili daha önce yapılan çalışmalar bulunmakla beraber, bu denli çok lineer olmayan değişken bulunan uygulama yapılmamıştır. Brosh (1981) ,konteyner yerine kargoların yığma şekilde yüklendiği sürekli bir optimizasyon problemini çözmüştür. Bir diğer çalışmada ise basit buluşsal çözüm* uygulanmıştır (Amiouny vd.,1992). Tüm kargo girdilerinin yüklenmesi gerektiği ve tek bir kompartımanın bulunduğu, bir hedef CG'den sapma hata döngüsü ile çözülen basit bir problemdir. Kevin (1992) ise, askeri kargo yüklemesiyle ilgili çalışmasında, dal sınır yöntemini kullanırken yüklemecilerin tecrübelerine dayanan bir metod benimsemiştir. Bu problemde yine eldeki tüm kargo girdisi yüklenmeli ve belirli bir sıralamada yapılmalıdır. Mathur (1998) ise, Amiouny vd.'nin çalışmasındaki aynı özel durum için, bu defa daha iyi bir hata sapma algoritması ile çözüm getirmiştir.

Fakat bu çalışmaların hiçbirinde kargo kompartımanı yüzeylerine gelen kesme kuvvetleri hesaba katılmamıştır. Çünkü bu durum, kompartıman pozisyonlarındaki ağırlık limitlerinin sabit olmaktan çıkıp, CG'nin lineer olmayan bir fonksiyonu olan bölgesel limitlerin göze alınmasını gerektirmektedir. Thomas vd., (1998) konuyla ilgili bir çalışma yapmışlar, ancak lineer olmayan sayısal bir programlama problemi ile karşı karşıya kalmamak için iki aşamalı çözüm geliştirmişlerdir. Birinci aşamada yüklenmesi gereken kargoları bir öncelik olarak kabul etmişlerdir. Bu aşamada problem uygun bir yerleştirme olması için konteyner atılarak devamlı çözülmüştür. Sundukları metod buluşsaldır ve ağırlığı maksimize etmek yerine kısıtları sağlayan bir tablolama arayüzü kullanmışlardır. Birinci aşamada çözümsüzlük olursa, kullanıcı bir veya birkaç konteyneri gruptan çıkarmalı ve birinci aşamayı tekrarlamalıdır. Tercihli pozisyon kısıtları daha sonra ikinci aşamadaki çözüme aktarılabilir.

Şimdiye kadarki çalışmalarda hep buluşsal çözüm metodları kullanılmıştır. Ancak Mongeau ve Bes (2003), A340-300 tipi uçakla ilgili yaptıkları çalışmada, buluşsal değil en uygun çözüme yönelmişlerdir. Kullanılan sayısal programlama ile iki avantaj sağlanmıştır. İlki girdi olarak verilen konteynerlerin hepsinin yükleneceği önermesi yapılmaması, programın eldeki çok miktar ve çeşitteki konteynerden en uygunlarını seçmesidir. Böylece ağırlık maksimize

---

* En iyi sonuca sadece yaklaştırabilen ama optimum çözümü vermesi kesin olmayan yöntemler bütünü. Hesaplaması en uygun sonuçları veren algoritmalara göre daha kısa sürdüğü için tercih edilir.

edilmektedir. İkincisi ise özel bir programlamaya ihtiyaç duymadan, hazır sayısal program yazılımıyla yaklaşık 10 dakika gibi makul bir zamanda hesaplamaya olanak vermesidir. Fakat programlamayı ve süreyi bu denli kısaltan, tek katlı kargo kompartımanı olması ve lineer olmayan ağırlık kısıtlarının olmamasıdır.

1.2 Amaç

Kargo uçaklarının yüklenmesi hem toplam ağırlık, hem de CG dikkate alındığında yolcu uçaklarına göre daha kritiktir. Kazanç sağlamaya yönelik yük fazla olduğu için limit ağırlıklara yaklaşılabilmektedir. Aynı yüklerin birçok değişik sıralama ve şekilde yüklenme olanağı olduğu için ise CG'nin ideal değerden uzaklaşma şansı oldukça yüksektir. CG bu değerden uzaklaştıkça uçuştaki yakıt sarfiyatı da sürükleme etkisi neticesinde artmaktadır. Tüm bunlar göz önüne alındığında birincil olarak uçuş emniyeti, ikincil olarak ise ekonomik bakımdan yüklemenin optimize edilmesi ihtiyacı ortaya çıkmaktadır. Bu ihtiyaç fark edilerek, yüklemeyi insan eli ile yapılmaktan çıkartıp bilgisayar ortamında yapılmasına olanak sağlayan bu çalışma yapılmıştır. Amaç uçağa mümkün olduğunca fazla kargo yüklerken, bir yandan da CG'nin önceden belirlenmiş bir ideale yaklaştırılmasıdır. Her ne kadar bahsi geçen Airbus uçaklarında havada yakıtın değişik noktalara transfer edilerek CG'nin değiştirilebilmesi mümkün ise de, bu çalışma sayesinde esneklik arttırılabilir ve daha iyi sonuçlar elde edilebilir.

1.3 Kapsam

Çalışmada hazırlanan program için T.H.Y. A.O. 'da uçmakta olan TC-JCT kuyruk isimli A310-304F tipi uçak temel alınmıştır. Programa girilen sabit değerler ve çıktılar bu uçağa özeldir. Yükleme kapsamında farklı boyutlarda 4 tip kargo konteyneri dikkate alınmıştır. Programın mümkün olan bütün değişik sayı ve çeşitte giriş için çözüm üretebilmesi için araştırma yapılmıştır.

1.4 Yöntem

En uygun sonuç için denemeler yapılmış fakat kombinasyon sayısının çok büyük olması, çözüme kabul edilebilir bir sürede ulaşmayı engellemektedir. Çalışmada incelenen uçağın iki katlı olması, kargo kapasitesinin çok olması, değişik konteyner tiplerinin bulunması bunda etkili olmuştur. Sonuç olarak çözüm için buluşsal yöntem benimsenmiştir. Buluşsal çözümü destekleyecek bir algoritma oluşturulmuş, C dilinde programlanmıştır. Algoritma, hacimsel

bir yaklaşım yerine ağırlıklara dayalı, ağırlık kısıtlarının dikkate alındığı bir yönteme göre çalışır. Bu doğrultuda program iki amaç gütmektedir. İlki olabilecek en ağır yüklemeyi yapmak, ikincisi ise ağırlık merkezini ideale mümkün olduğunca yaklaştırmaktır. Programa konteyner ağırlıkları ve tipleri girilmekte, sonuç olarak eğer aşırı yükleme veya yüklenememe durumu varsa hangi konteynerlerin yerde bırakılması gerektiği ve CG'nin değeri alınmaktadır. Önceden yeri belirlenmiş konteynerler varsa, programa girilirken yerleri belirtilir. Yüklemeyi yaparken yapısal, bölgesel ve kümülatif ağırlık kısıtlarını inceler. Yapısal ağırlıklar hesaplanırken uçuş için gerekli yakıt ve ekip gibi ağırlıklar da göz önüne alınır.

2. TEMEL UÇAK YÜKLEME ESASLARI

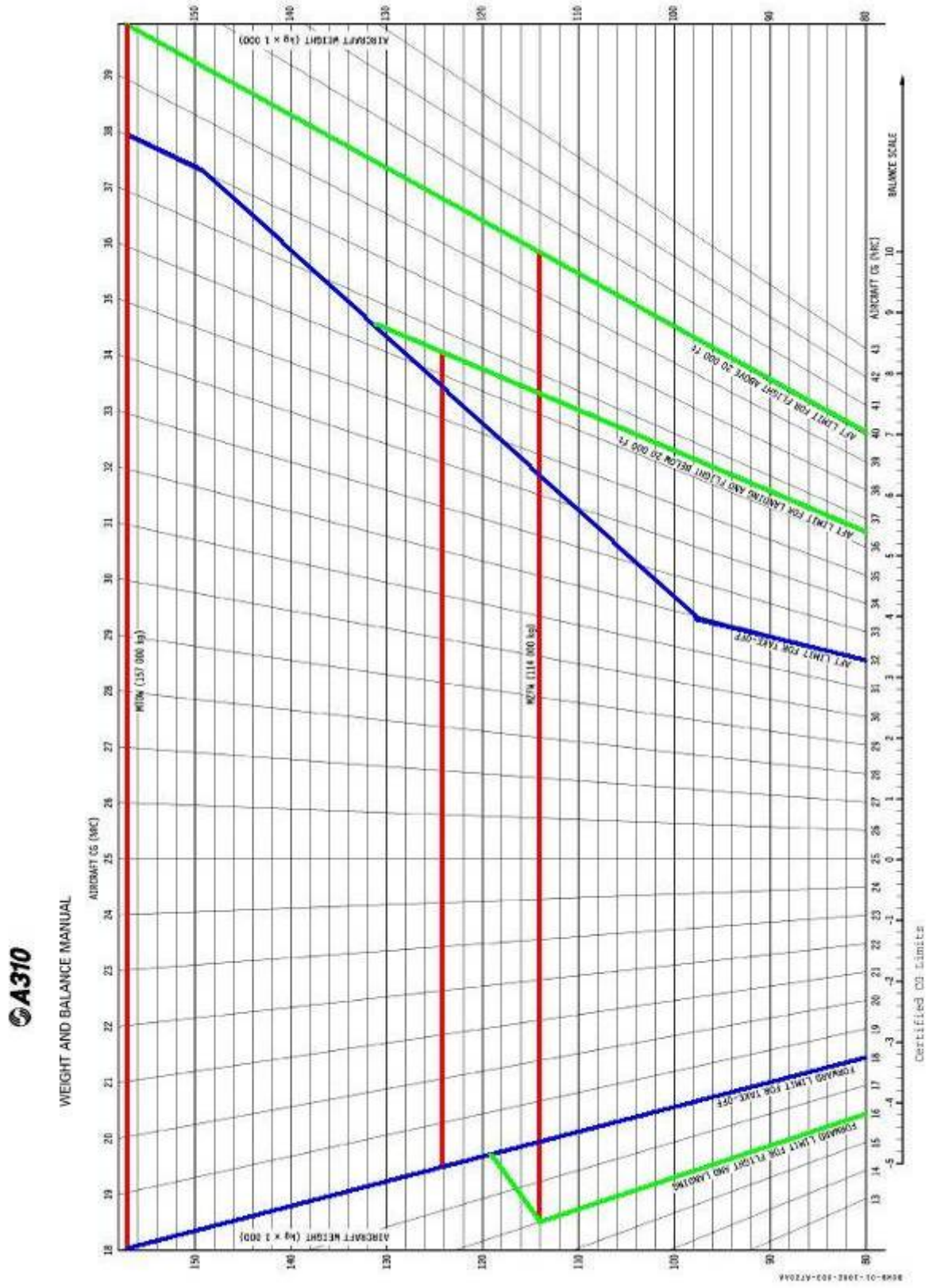
2.1 Tanımlar

- Temel Operasyon Ağırlığı (BOW): Uçağın yapımcısı tarafından belirlenip uçuş el kitaplarına geçirilmiş bir ağırlıktır. Ağırlık-Denge hesaplamalarında başlangıç noktasıdır. Bu ağırlığın içinde uçağın döşemeleri, kullanılmayan yağ, yakıt, hidrolik ve tuvalet suları, ikram araç gereçleri ve acil durum gereçleri bulunur.
- Kuru Operasyon Ağırlığı (DOW): Temel İşletme değerine işletme ağırlıkları (mürettebat, ikramlar, vb.) ilave edilerek bulunur. Uçuşun niteliğine göre değişir.
- Operasyon Ağırlığı (OW): Kuru İşletme Ağırlığı'na kalkış yakıtının dahil edilmesi ile bulunur.
- Azami Yakıtsız Ağırlık (MZFW): Uçağın yapımcısı tarafından belirlenmiş, uçak yakıtsızken yüklenebilecek azami yük ve yolcu ağırlığını ifade eder. Uçak kanatlarının taşıyabileceği yapısal yükle doğru orantılıdır.
- Azami Kalkış Ağırlığı (MTOW): Uçağın kalkış yapabileceği yapımcı tarafından belirlenmiş azami ağırlık limitidir.
- Azami İniş Ağırlığı (MLW): Uçağın iniş yapabileceği yapımcı tarafından belirlenmiş azami iniş ağırlığıdır.
- Trafik Yükü: Ticari sebeple uçakta taşınan tüm yüklerdir (yolcu, bagaj, kargo).
- Datum Çizgisi: Ağırlık-Denge hesaplamaları için uçak yapımcısı tarafından belirlenen, genel olarak uçağın burnuna teğet veya bir miktar önünde bulunan matematiksel, hayali bir başlangıç hattıdır.
- MAC: Uçak CG'sinin ifade edilmesini sağlamak üzere, uçak imalatçı firması tarafından tanımlanan, uzunluğu ve datum çizgisinden mesafesi sabit olan referans mesafedir. Kanat üzerine yayılmış yüzdelik skala ile gösterilir.
- MACTOW: Kalkış CG'sinin MAC cinsinden ifade edilmesidir.
- MACZFW: Yakıtsız CG'nin MAC cinsinden ifade edilmesidir.
- MACLW: İniş CG'sinin MAC cinsinden ifade edilmesidir.

- İndeks: Ağırlık-Denge hesaplarını kolaylaştırmak üzere, Datum Çizgisi temel alınarak hesaplanan momentleri ifade etmek için kullanılan birimdir.
- Stab Trim: CG hesaplandıktan sonra, bunun uçağın yunuslamasına etkisini ortadan kaldırmak üzere kuyruktaki yatay stabilazörden yapılan ayar.
- Trim Tank: Alınan toplam yakıt yaklaşık olarak 43,000 kg ‘ı geçtiğinde veya havada uçağın dengesini sağlamak üzere, yakıtın otomatik olarak transfer olduğu yedek yakıt tankı.
- Bulk Kargo: Konteyner ile paketlenmemiş, serbest halde yığma şeklinde yüklenen kargo.

2.2 A310 Kargo Uçağının Özellikleri

2.2.1 Uçak Limitleri



Şekil 2.1 A310-304F'in ağırlık-denge zarfı.

TC-JCT kuyruk isimli A310-304F kargo tipi uçağın limitleri aşağıdadır:

- MTOW……..157,000 kg
- MLW…….124,000 kg
- MZFW……...114,000 kg
- Azami yakıt kapasitesi……49,483 kg

Azami ağırlık değerleri Şekil 2.1'de kırmızı hatlarla gösterilmiştir.

2.2.2 Ağırlık-Denge Zarfının Amaç Bakımından Değerlendirmesi

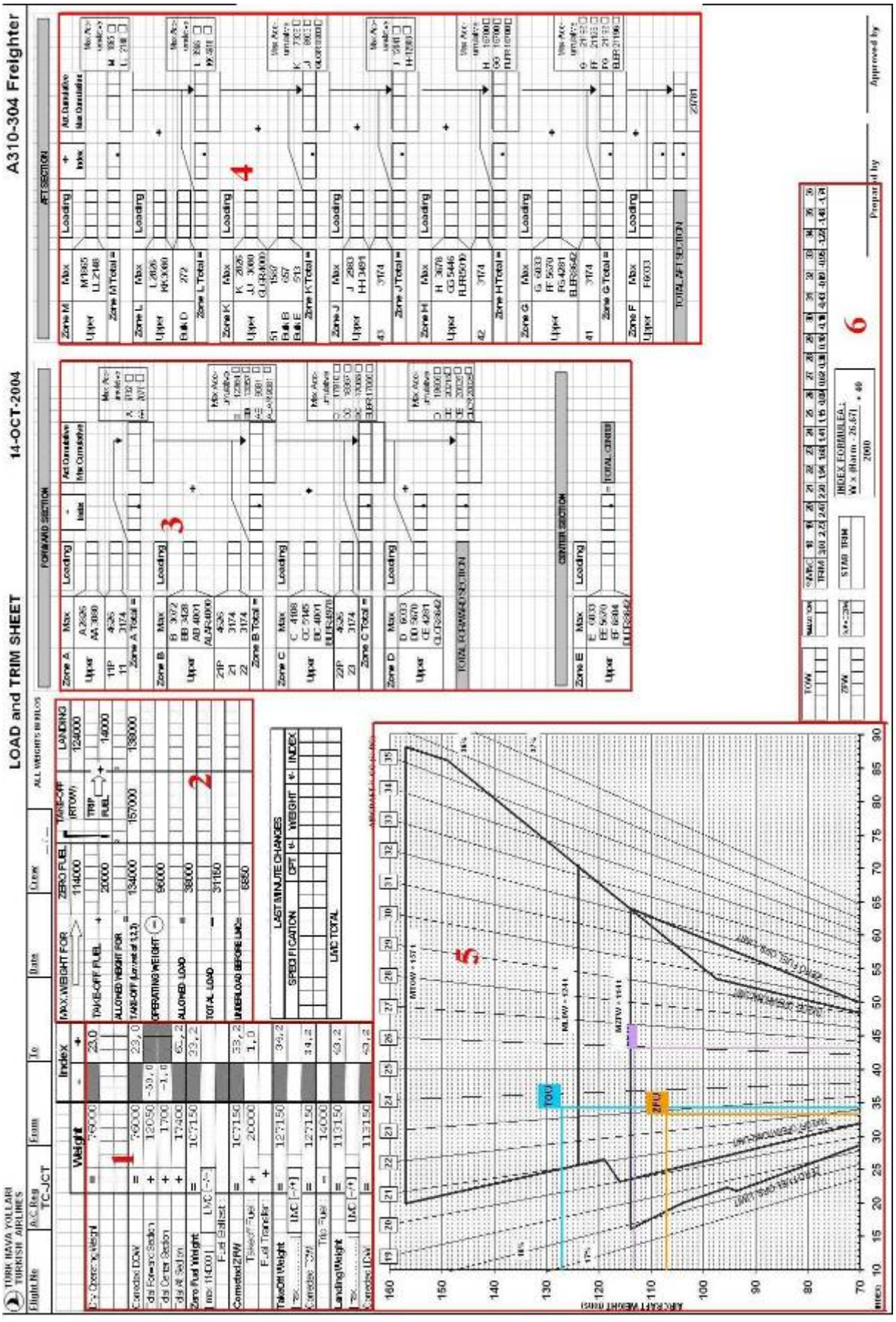
Her uçak için yapımcısı tarafından uçağın yapısal özelliklerini, yükleme ve uçuş sınırlarını tanımlamak üzere bir grafik oluşturulur. Bu zarf kritik ağırlık ve CG değerlerini barındırır ve ağırlık-denge zarfı olarak adlandırılır. Hem yükleme, hem de uçuş bakımından ağırlık-denge zarfının iyi analiz edilmesi çok önemlidir. Zarf üzerinde amaçlanan noktanın iyi tespiti, çözümün de doğru olmasını sağlar.

Şekil 2.1'de görüldüğü üzere ağırlık limitleri dikey eksendeki kırmızı hatlarla gösterilmiştir. Uçağın CG değeri ise MAC cinsinden yatay eksende görülmektedir. Mavi çizgiler kalkış zarfı sınırlarını, yeşil çizgiler ise uçuş ve iniş zarfını tanımlamaktadır. Zarf incelendiğinde kalkış zarfının CG bakımından daha kısıtlayıcı olduğu görülecektir. Dolayısı ile uçuş başlangıcında yapılan yüklemenin düzgün olması önem kazanmaktadır. Sayısal olarak tespit yapıldığında CG'nin tam merkezde olması için MAC değeri 29,6 ve buna karşılık gelen indeks değeri 61'dir. Yatay eksendeki değişimlerde, buna yakın değerler hedeflenmelidir.

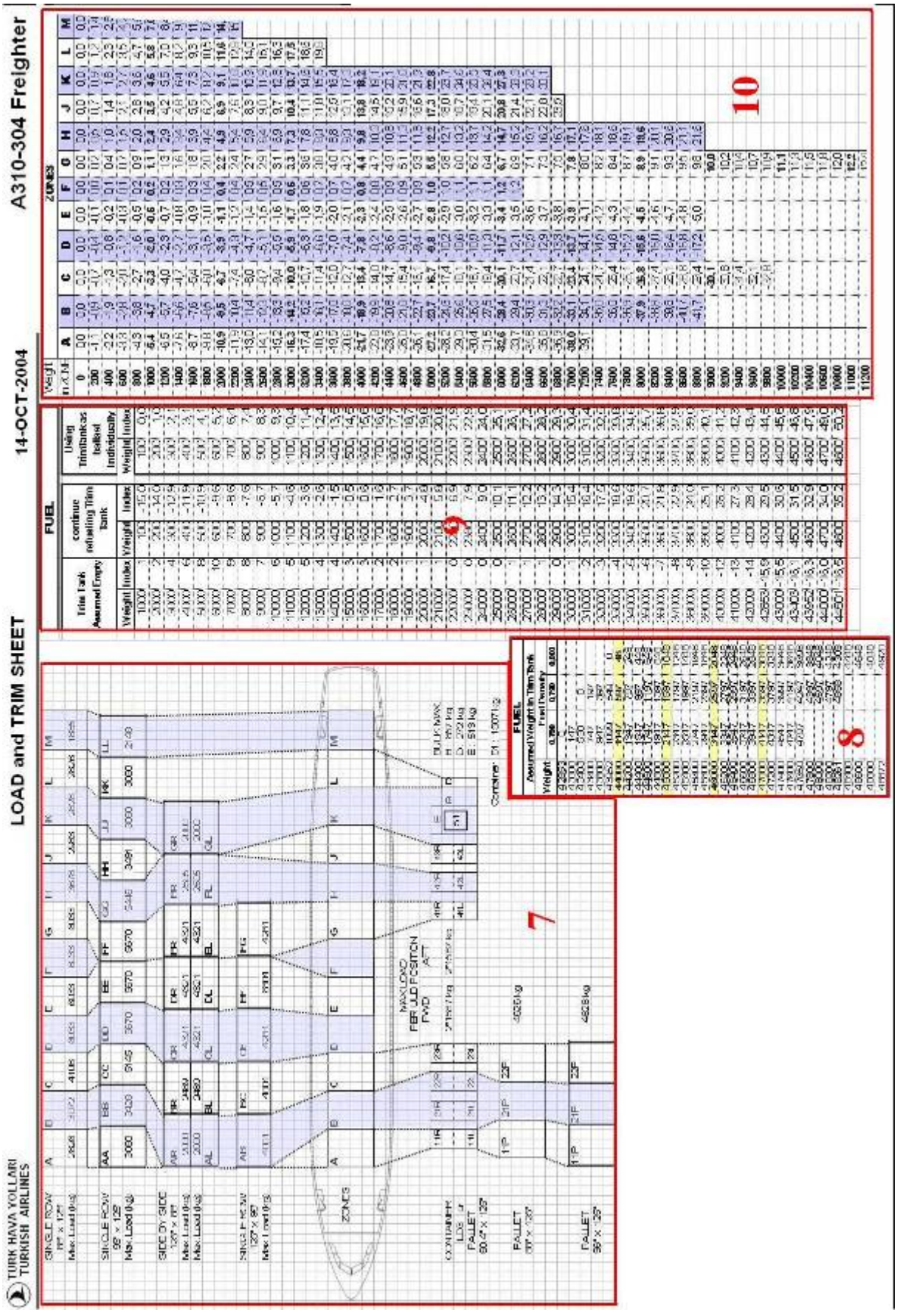
2.2.3 Yük-Trim Formu'nun Açıklaması

Şekil 2.2 ve Şekil 2.3'de numaralandırılmış yerlerin açıklamaları aşağıdadır:

1) Kargo, yakıt ve diğer ağırlıklarla indeks değerlerinin girildiği kısımlar
2) MTOW, ZFW, LDW değerlerinin girildiği ve izin verilen trafik yükünün bulunduğu kısım
3) Ön ve ortaya yüklenen kargo ağırlıklarının girildiği kısım
4) Arkaya yüklenen kargo ağırlıklarının girildiği kısım
5) Ağırlık-Denge zarfı
6) Hesaplanan MAC değeri için stab trim bulunan kısım



Şekil 2.2 A310-304F'in yük-trim formu ön yüzü.



Şekil 2.3 A310-304F'in yük-trim formu arka yüzü.

7) Kargo kompartımanlarının ve değişik konteynerlerin gösterildiği şema

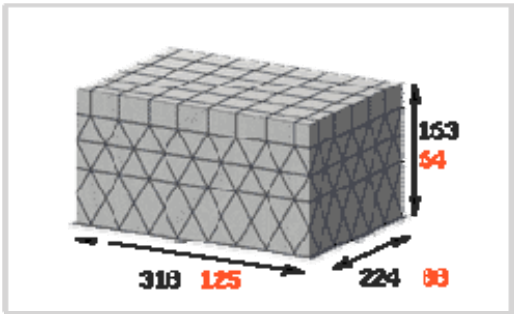
8) Trim Tank'a giden yakıtın indeks değeri tablosu

9) Ana yakıt tankına giden yakıtın indeks etkisi

10) Bölgelere göre ağırlık-indeks tablosu

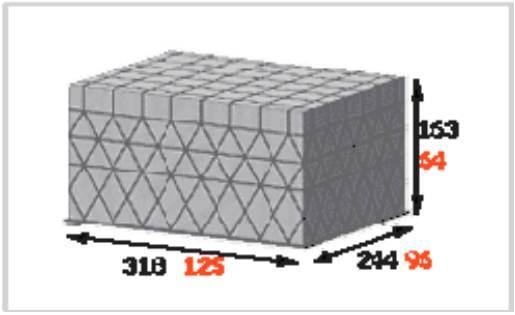
2.2.4 Kargo Konteyner Tipleri

- **PAG**: 88"x125"x64" / 224x318x163 cm boyutlarında, 380 ft³/10.6 m³ hacimli konteyner



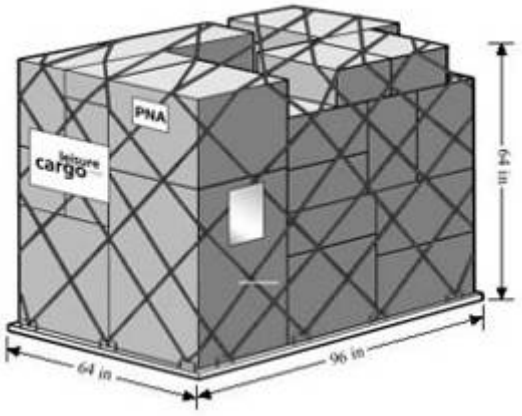
Şekil 2.4 PAG tipi konteyner.

- **PMC:** 96"x125"x64" / 244x318x163 cm boyutlarında, 415 ft³/11.6 m³ hacimli konteyner



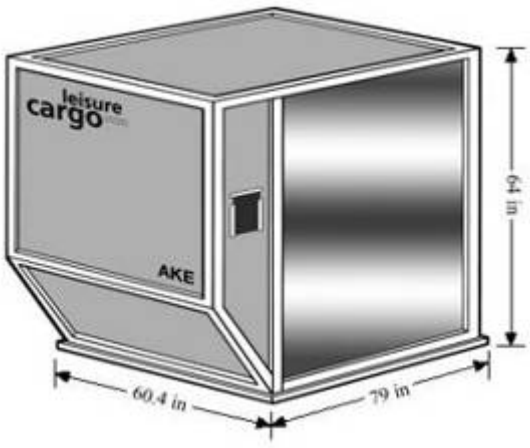
Şekil 2.5 PMC tipi konteyner.

- **PLA:** 60"x125"x64" / 153x318x163 cm boyutlarında, 253 ft³/7.2 m³ hacimli konteyner



Şekil 2.6 PLA tipi konteyner.

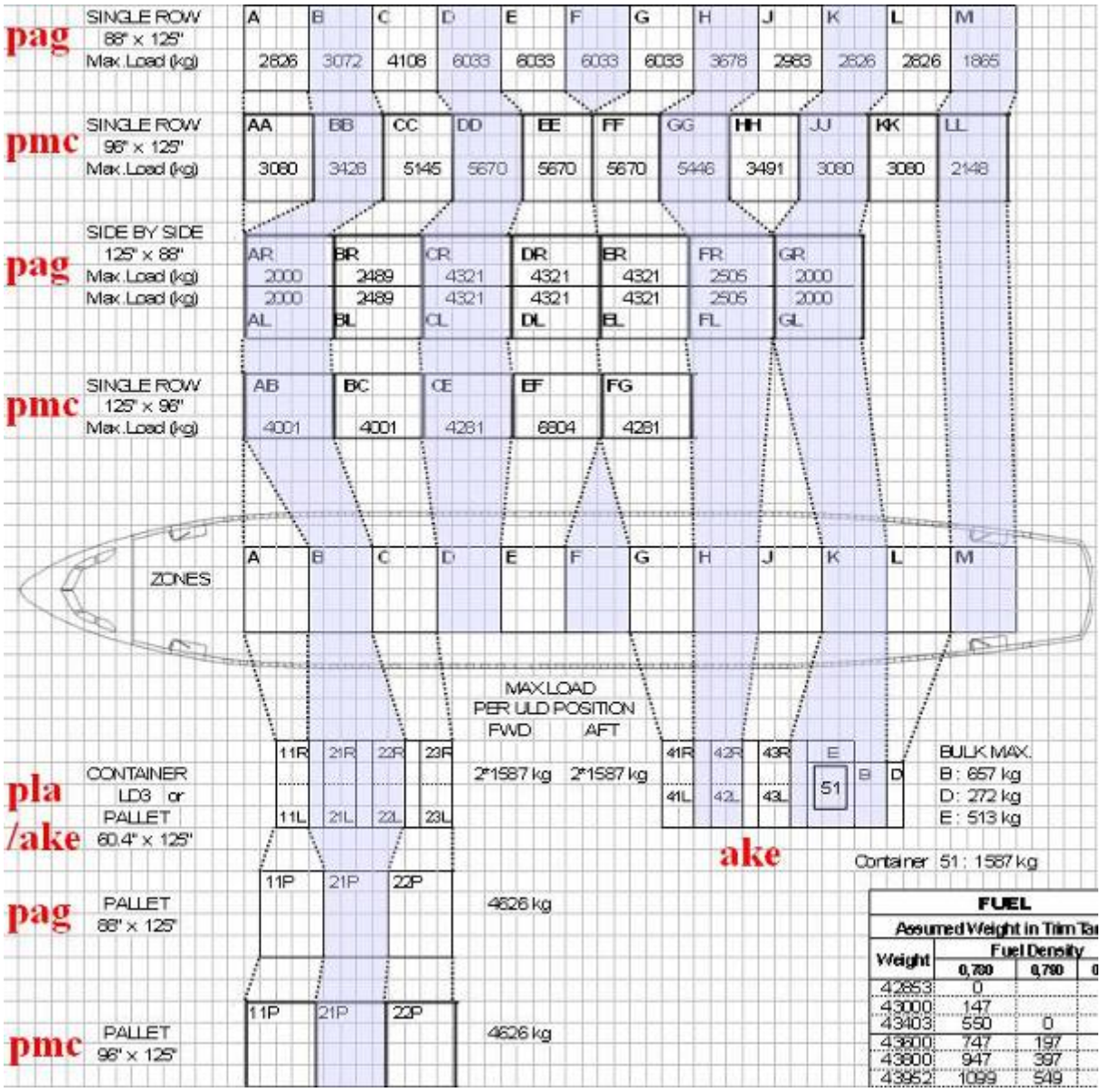
- **AKE:** 64.5"x60.4"x64" / 164x153x163 cm boyutlarında, 153 ft³/4.3 m³ hacimli konteyner



Şekil 2.7 AKE tipi konteyner.

2.3 Kargo Kompartımanları ve Operasyon Teknikleri

A310'da alt ve üst kargo kompartımanları bulunur. Alt kat, merkez boş kalacak şekilde ön ve arkada birbirinden bağımsız iki bölüm halindedir. Kapasitesi üst kata göre daha azdır. Üst kat ise önden arkaya kadar tek bir kompartıman halindedir. Kompartımanların daha detaylı şeması Şekil 2.8'de gösterilmiştir. Uçak gövdesi şemasında görüldüğü gibi, hesaplamaları kolaylaştırmak üzere, gövde uzunlamasına ekseni üzerinde A'dan M'ye kadar hayali bölgeler tanımlanmıştır. Böylece klasik ağırlık x moment kolu hesabı yerine, Şekil 2.3 10 numaralı



Şekil 2.8 A310-304F kargo kompartımanları ve konteyner tipleri.

tablodan ilgili bölgeye denk gelen toplam ağırlığın moment etkisi alınarak, hesaplama basitleştirilmiştir.

Şekil 2.8'de hangi kompartımana hangi kargo tipleri yüklenebileceği gösterilmiştir. Üst kata iki şekilde PAG, iki şekilde de PMC olacak şekilde dört ayrı şekilde yükleme yapılabilir. En üstte görülen şekilde PAG'lar dikey yerleştirilirse 12 tane, üçüncü sırada görülen şekilde ise iki sıra yatay halde 14 tane konulabilir. PMC'ler ise ikinci sırada görüldüğü gibi dikey olarak 11 tane, dördüncü sırada görüldüğü gibi ise yatay ve 5 tane konulabilir. Ayrıca bunların değişik kombinasyonları da ağırlık ve geometrinin izin verdiği ölçüde yapılabilir. Alt katı ise ön ve arka olarak değerlendirmek gerekir. Alt ön kısma 3 PAG / 3 PMC / 4 PLA / 8 AKE veya bunların kombinasyonları yüklenebilmektedir.

Alt arka kısma 3 PLA + 1 AKE + Bulk kargo / 7 AKE + Bulk kargo yüklenebilmektedir.

Bu çalışmada PMC ve PAG tipleri esas alınarak çözüm 2 çeşit konteyner için yapılmıştır. Buna göre alt ön kısma PMC ya da PAG yüklendiği , arka tarafta ise elle yerleştirilmiş AKE tipi konteynerler olduğu varsayılmıştır.

A, AA, AR, 11P gibi kodlar yüklenen konteynerin yer bilgisini belirtmek için kullanılır. Örnek olarak BB için, yüklenen konteyner PMC tipidir ve dik olarak ikinci sıradadır.

Ağırlık ile ilgili olarak ise şekildeki her kutucuğun içinde yazan değer, ilgili pozisyona yüklenebilecek azami ağırlık değerini göstermektedir. Kümülatif ağırlık kısıtları ise daha önce belirtildiği gibi Şekil 2.2'de 3 numaralı bölümden kontrol edilmektedir.

Şekil 2.8'deki aşağı ve yukarı doğru uzanan aynı renkli sütunlar ise, yüklenen konteynerin ağırlığının etkiyeceği bölgeyi göstermektedir.

2.3.1 Yükleme Metodu

Yüklemeciler, uçağı yüklemeden önce ellerine kargo konteynerlerinin listesi gelir. Bu listede ağırlık, tip ve yükleme önceliği bilgileri yer alır. Yüklemeci bu bilgiler ışığında, ilk olarak önceliği olan konteynerleri seçer. Daha sonra yapması gereken tecrübeleri doğrultusunda, aynı tipte kargoları gruplayarak geometrik yer kaybını azaltacak şekilde sıralama yapmak ve tahmini olarak CG'yi iyileştirecek yerleştirmeler yapmaktır. Yükleme formu üzerinde ilgili konteynerleri doldurarak, bölgesel ve kümülatif ağırlık kısıtlarını kontrol etmelidir. Uyumsuz bir durum varsa değişiklikler yapmalı ve kısıt hedefini tutturmalıdır.

2.3.2 Mevcut Metodun Sakıncaları

Yüklemecinin kağıt üzerinde yaptığı klasik yükleme metodunun sakıncalarına değinmek gerekirse;

- Yüklemeye başlarken yeterince iyi olmayan yüklemelerle başlamak (daha önce de bahsedildiği gibi çok çeşit ve kombinasyon olabilir), yükleme ilerledikçe kalan konteynerleri sığdırmak için CG'nin ideal olmasından vazgeçilmesine sebep olabilir.
- Kötü sonuçlar elde edilmesi durumunda işlemin birkaç defa tekrar edilmesi gerekebilir.
- Çoktan seçimli kargolar varsa, hangilerini yerde bırakmak gerektiği yanlış değerlendirilebilir, ağırlık maksimum olmayabilir.

- Yüklemeci uçakla seyahat ederek varış meydanındaki yüklemeyi de yapmalıdır. Dolayısıyla uçan her uçak için bir yüklemeci bulunmalıdır. Program kullanılsaydı yüklemeci uçakla gitmeyebilir, oradaki yükleme uçak personeli tarafından bilgisayara yaptırılabilirdi.
- Genel olarak iyi bir sonuç elde edilse bile bunun en olası en iyi sonuç olduğu garanti değildir.
- İnsan faktörü dolayısıyla hata olasılığı her zaman vardır.

3. YÜKLEME OPTİMİZASYONU

3.1 Buluşsal Çözüm Özellikleri

Daha öncede bahsedildiği gibi optimizasyonla ilgili çalışmalarda buluşsal yaklaşım çok yaygın olarak kullanılmıştır. Kombinasyonel optimizasyon problemlerinde buluşsal yaklaşım kullanımının değişik nedenleri vardır. Hill (1998), çalışmasında bu nedenlerden bahsetmiştir. Özellikle uçak kargo yükleme problemleriyle ilgili yapılan önceki çalışmalarda, temel olarak kargo ağırlıkları ve hacimlere dayalı optimizasyon yapılmıştır. Bu çözümler çoğu zaman üç boyutlu (CG'nin ve zemin dayanımlarının hesaba katıldığı) olmaktan uzak kalmışlardır. Çünkü aksi takdirde, hesaplanması gereken çok fazla ve hatta lineer olmayan değişken olacaktır. İşte bu aşamada buluşsal yaklaşım yaparak çözümü basitleştirmek mecburidir. Bu sayede optimum çözüme çok yakın çözümlere, çok kısa zamanda ulaşılabilir.

Bir diğer önemli nokta da en uygun çözümlerin gerçek dünyadaki sistemlere tam olarak adapte edilememesidir. Çözümde ufakta olsa bir kaç değişiklik yapılır ve dolayısıyla gerçek dünyada değiştirilen en uygun çözüm alt en uygun sonuçlar doğurur. Bunun nedeni problem modellenirken gözden kaçan ya da tam olarak tanımlanamayan, kimi zaman lineer olmayan özelliklerdir. Fakat buluşsal çözümler daha kolay bir şekilde gerçek dünya sistemlerine adapte edilebilir ve de en uygun sonuca göre buluşsal çözümde oluşan kayıp çoğu zaman az önce ifade edilen en uygun çözümlerin uygulanma esasında ortaya çıkan kayıplardan daha azdır. Çünkü buluşsal çözüm, doğası gereği kimi kabul ve toleranslara sahip olduğu için, gerçek hayatta daha kolay uygulanabilir.

3.2 Optimizasyon Metodu İçin Alternatif Yaklaşımlar, Algoritmaya Etkileri ve Uygulanabilirlikleri

3.2.1 Ön ve Arka Bölgeleri Eşleştirme

En genel anlamıyla dengenin ve uçaklar özelinde dengeli bir yüklemenin şartı momentler dengesini oluşturmaktan geçer. Yüklerin oluşturduğu toplam moment sıfırlanırsa, sonuç olarak yükleme de dengeli olmuş demektir. Buna göre ağır yükler daha az yapısal yük oluşturmak üzere uçağın orta kısmında olmalı ve daha hafif yükler öne ve arkaya dağıtılmalıdır. Yükleme probleminin çözümünde ilk olarak bu yaklaşım benimsenmiştir. Şekil 2.3 10 numaralı kısımda görülen indeks değerlerinin pozitif ve negatif olarak etki ettiği kısımlar tespit edilmiş, pozitif ve negatif bölgelere eşit bir dağılım yapılarak toplam

momentin sıfıra yaklaştırılması amaçlanmıştır. Her bir bölge için tablodaki değerler incelenmiş, tüm değerler için belli bir yaklaşıkla doğru olacak şekilde katsayılar tespit edilmiştir. Bu katsayılar ilgili bölgenin ağırlık-indeks katsayısı olmuştur ve o ağırlıkla çarpıldığında o bölgede oluşturduğu indeks değerini vermektedir. Böylece her defasında değişik bir ağırlık için tablodaki değerin tespit edilmesine gerek kalmamıştır. Bu katsayılar, daha sonraki çözüm yöntemlerinde de kullanılacaklardır.

Fakat gerçek yükleme örnekleri ile değerlendirildiğinde, bu tarz bir yüklemenin bu uçak için doğruluğu olmadığı anlaşılmış ve uygulamadan vazgeçilmiştir. Bu yaklaşımın geçerliliği olmamasının en önemli nedeni, uçağın tasarım özellikleridir. Birçok uçağın boş CG'si yine ağırlık-denge zarfı sınırları içerisindeyken, bu uçakta limit dışında olacak şekilde öndedir. Başka bir deyişle bu uçak boş haliyle uçamamaktadır. Mutlaka yakıt veya safra ağırlık alma zorunluluğu vardır. Durum böyle olduğu için, yüklenen kargoların toplam momentinin sıfırlanması CG'nin iyi bir değer almasını sağlamaz. Ancak yükleme sonucunda CG'yi ortaya alacak bir sapma oranı belirlemek de düşünülmüş, bunun da her yüklemede değişen sabit olmayan bir değer olduğu anlaşılarak bundan da vazgeçilmiştir.

3.2.2 Tüm Olasılıkların Hesaplanması

Buluşsal çözüm yerine optimum çözüm yaklaşımı da değerlendirilmiştir. Buna göre Şekil 2.8'de en üst sırada görülen 12 adet PAG tipi konteyner yerleştirilmesi denenmiştir. Bu yöntemle olası en iyi sonucun alınacağı kesindir, fakat optimum sonucu bulmak için bilgisayarın 12!, yani 479.001.600 kombinasyonu gözden geçirmesi gerekmektedir. Alt kat ön kompartımanı da hesaba katarsak bu sayı 15!, yani 1.307.674.368.000 olacaktır. Bilgisayarın böylesi çok işlemi, makul bir sürede bitirebilmesi olası olmadığı için optimum çözümden vazgeçilmek zorunda kalınmıştır.

3.2.3 Tüm Olasılıkların Hesaplanmasından Önce Kısıt Filtrelemesi

Tüm olasılıkların hesaplanmasında, bilgisayar tek tek bütün konteynerler için her bölgeyi kontrol etmelidir ve yukarıda bahsedildiği gibi ortaya çok büyük işlem sayısı çıkmaktadır. Ancak bu yaklaşımın iyileştirilebilmesi için, konteynerlerin kombinasyon hesabına girmeden önce bölgesel ve kümülatif kısıtların araştırıldığı bir filtreden geçirilmesi düşünülmüştür. Programa girilen konteynerler hem tek tek her bölgenin ağırlık kısıtları için araştırılmakta, hem de büyükten küçüğe doğru bir ağırlık sırasında, kendilerinden sonra gelen konteynerlerle kümülatif ağırlık kısıtları için araştırılmaktadır. Bu araştırma sonucunda her bir bölgeye

yerleşebilecek ağırlıkların vektörleri oluşturulmaktadır. Fakat filtreleme ile yapılan test hesaplamasında, bilgisayarın yine kabul edilebilir bir sürede işlemi bitiremeyeceği anlaşılmış ve optimum çözüm yaklaşımından vazgeçilmiştir. Gelecekte bilgisayar teknolojisi ilerleyip, çok daha hızlı bilgisayarlar piyasaya çıktıkça optimum çözüm yöntemi kullanılabilecektir.

3.2.4 Hafif Yüklerin Yerleştirilmesi Kabulü ve Geçerliliğinin İncelenmesi

Buraya kadar gelinen noktada, optimum çözüm bakımından değişken sayısının çokluğu, bunun neticesinde zaman sorunu ortaya çıkmıştır. Zaman sorununu ortadan kaldırmanın tek yolu değişken sayısını azaltmak olarak görülmektedir. Fakat yukarıda değerlendirilen metotlarla yinede makul bir süreye inilememiştir. Klasik toplam momenti sıfırlama yaklaşımının da uçağın tasarım özelliklerinden dolayı kullanışlı olmadığı anlaşılmıştır. Bu yüzden değişken sayısını azaltmak üzere hafif yüklerin öne yerleştirilmesi kabulü değerlendirmeye alınmıştır. Kombinasyonel optimizasyon programı çalışırken yapılan ölçümlerde, değişken sayısı dokuz olduğu takdirde kabul edilebilir bir sürede programın hesaplamayı tamamladığı görülmüştür.

3.2.4.1 Uygulamadaki Faydaları

İyi bir sonuç elde etmek için belirli ölçüde bilgisayarın kombinasyonel optimizasyon yapması zorunludur. Bunun aksi şekilde çalışıp iyi sonuç elde edecek bir algoritma geliştirilememiştir. Kombinasyonel optimizasyonun kullanılabilmesi için değişken sayısının mutlaka azaltılması gerekmektedir. Ancak bunun için de klasik yükleme yaklaşımının kullanılması mümkün değildir. Uçağın tasarım özellikleri de göz önünde bulundurularak, yükleme yapılacak kargolar arasından en hafif olan (toplam değişken sayısını dokuza düşürecek şekilde) konteyner seçilerek, arka iki konuma ve en ön pozisyonlara yerleştirilmesi düşünülmüştür. Böylece tasarım icabı zaten önde olan CG'nin daha da öne gelmesini ya da yeterince arkaya çekilememesini engellemek için, eldeki en az moment oluşturacak konteynerler arkaya ve öne yerleştirilmiş olur. Hem de kombinasyon sayısı oldukça azaltılarak, kalan konteynerler için bilgisayarın olası tüm yerleştirmeleri makul bir sürede gözden geçirmesi sağlanmış olur.

3.2.4.2 Örneklerle Geçerliliğin İncelenmesi

Bu kabulün geçerliliğini test etmek üzere daha önce yüklemeci tarafından yapılmış üç yükleme, bu prensibe göre çalışan mini bir programda denenmiş ve hepsinde de program daha iyi yüklemeler yapmıştır. Çizelge 3.1'de karşılaştırmalı olarak MACTOW sonuçları görülmektedir. Fark sütununda değerlendirilen yüzdelik değer ideal CG'ye yakınlıktır.

Örneklerden alınan sonuçlar, bu yaklaşımın geçerliliğini kanıtlamıştır. Hem uygulamadaki faydaları, hem de geçerliliği göz önünde bulundurularak bu yaklaşımın kullanılmasına karar verilmiştir.

Çizelge 3.1 Hafif yüklerin öne yüklenmesi test sonuçları.

Yüklemeci MACTOW	Bilgisayar MACTOW	I iyileşmesi (MAC)
23,0	26,7	+2,7
25,5	29,8	+3,3
25,8	26,5	+0,7

3.2.4.3 Marjinal Girdiler İçin Filtreleme İhtiyacı

Hafif yüklerin önde olması kabulünün geçerliliği yapılan yükleme testi ile anlaşılmıştır. Ancak bu durumda hafif olanlar dışında diğer konteynerler ilk etapta incelenmeyerek, hafif olanlar öne yerleştirildikten sonra geriye kalan tümünün açıkta kalan pozisyonlara yerleşebileceği varsayılmıştır. Ancak gözden kaçırılmaması gereken bir durum vardır ki, o da ağırlık kısıtları veya sayıca azlığı nedeni ile geometrik olarak mutlaka belirli bir pozisyona yüklenmesi gereken konteynerler olabileceğidir. Bunları marjinal girdiler olarak adlandırabiliriz. Bunlar incelenmeden hafifler seçilip yerleştirildiğinde, bunların olmazsa olmaz yerleşimleri bozulabilir.

Marjinal girdiler hafiflerin yerleştirilmesi nedeni ile kısıtlanmasa bile, ilk etapta nereye yerleşmesi gerektiği belli olan konteynerleri toplam değişken sayısından düşmek, böylece daha az hafif konteyneri öne yüklemek, bizi en iyi çözüme bir adım daha yaklaştırır.

Marjinal değerlerin tespit edilmesi için, hafif kargoların seçilmesinden önce devreye girerek tüm konteynerleri marjinal değer araştırmasından geçirecek bir algoritma hazırlanmıştır. Bu araştırmanın PAG ve PMC tipi konteynerler için yapılması yeterli olacaktır, çünkü ağır yük limitlerinin bulunduğu pozisyonlara bu tipler yüklenebilmektedir. İncelenmesi gereken iki değişken olduğuna için, önce biri sabit kabul edilip daha sonra diğeri buna bağlı olarak incelenmiştir. İlk adım olarak PMC tipi konteynerler incelenmiş, daha sonra PAG algoritmaya dahil edilmiştir.

3.2.4.4 Marjinal Girdi Filtreleme Algoritması

Marjinal girdi olarak adlandırılan konteynerler, aşağıdaki ağırlık aralıkları için şu şekilde listelenebilir.

PMC için;

- $5670 \leq x \leq 6804$ kg
- $5446 \leq y \leq 5670$ kg
- $5145 \leq z \leq 5446$ kg
- $4626 \leq p \leq 5145$ kg
- $4281 \leq r \leq 4626$ kg

PAG için;

- $4626 \leq x_1 \leq 6033$ kg
- $4321 \leq y_1 \leq 4626$ kg

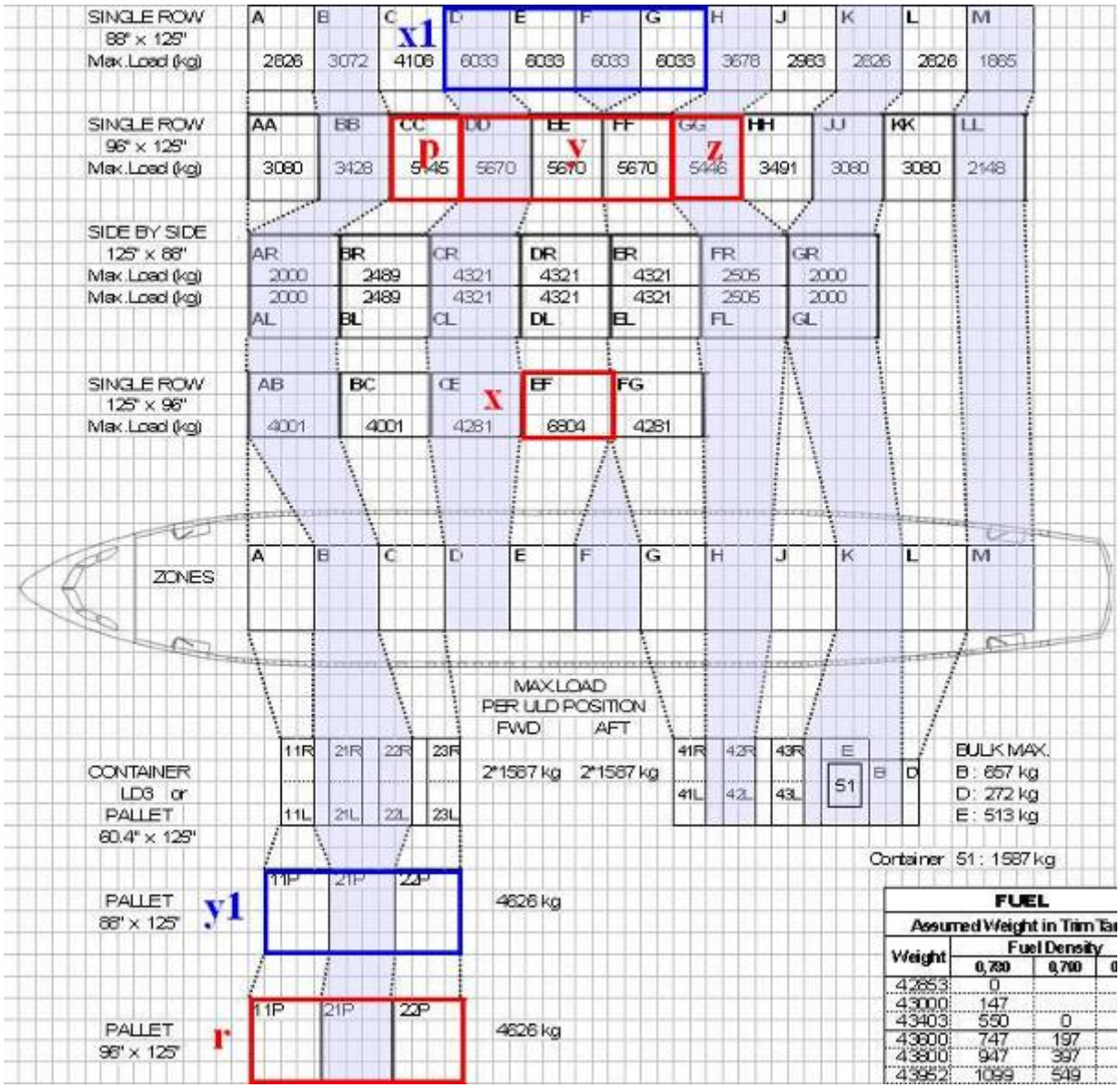
Bu ağırlıkların yerleşebileceği bölgeler Şekil 3.1'de gösterilmiştir.

Sembollerle tanımlanan bu pozisyonlar için azami sayı ise;

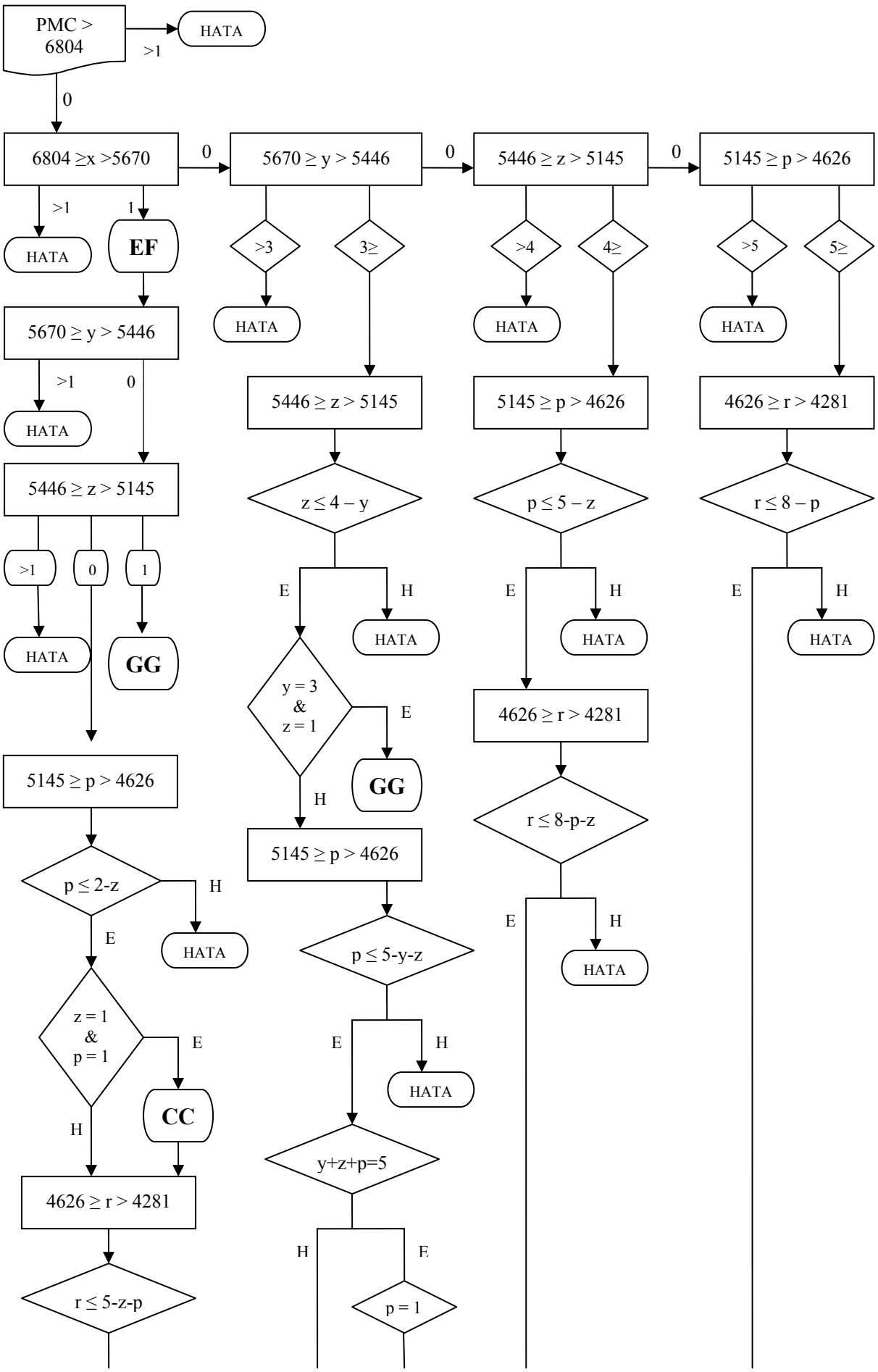
x = 1, y = 3, x_1 = 4'dür.

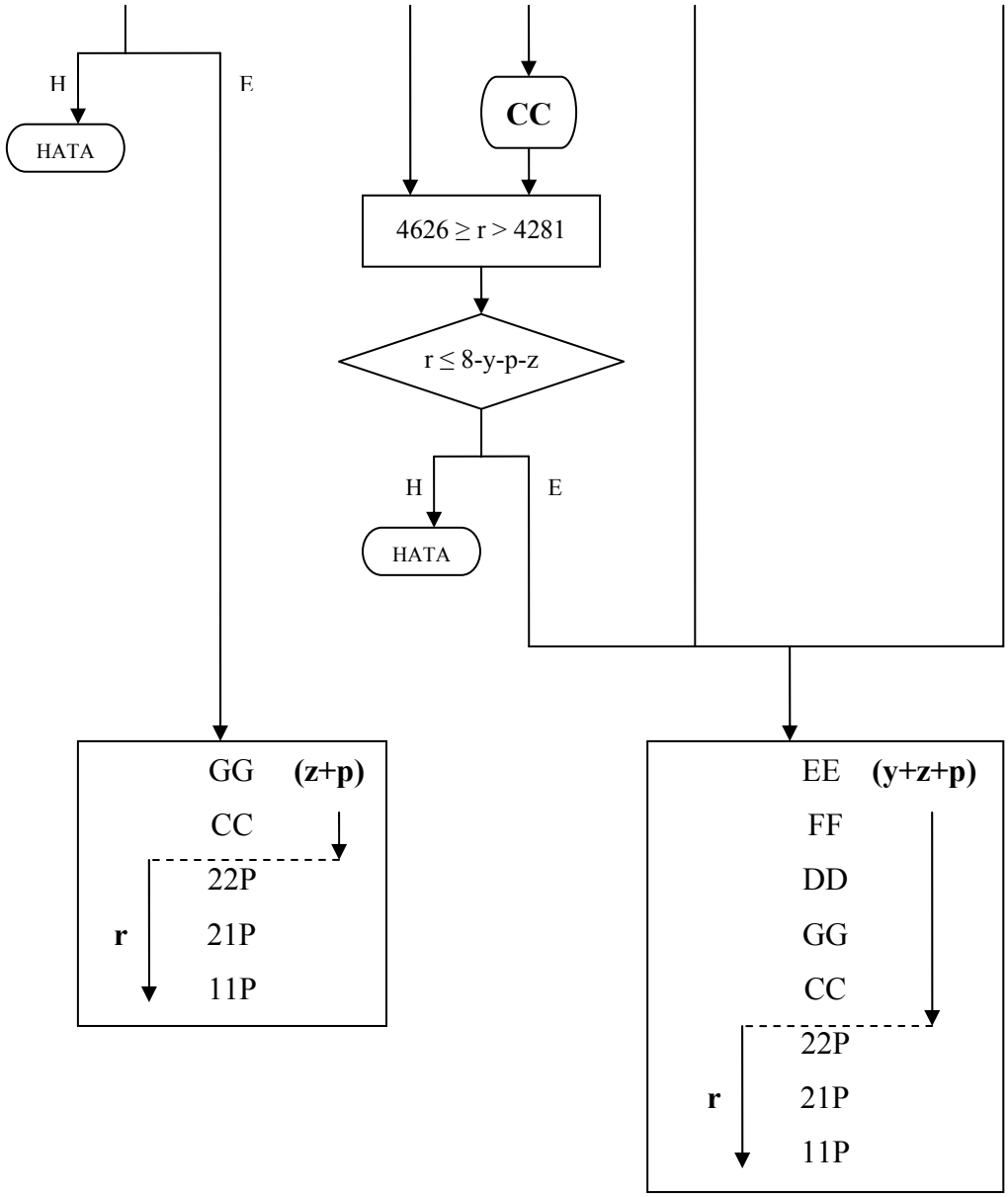
y_1, z, p ve r için ise bu sayı x, y ve x_1'in sayılarına göre değişmektedir. Bir üst ağırlık sınıfına nazaran hafif olan konteyner, ilgili ağır konteynerin pozisyonu boş ise oraya da yerleşebilir. Bu yüzden bunlar için sayı kısıtlaması algoritma içinde değişerek araştırılmıştır.

Marjinal Girdi Filtreleme Algoritması PMC bölümü , Şekil 3.2'de görülmektedir. Marjinal Girdi Filtreleme Algoritması PAG bölümü (PMC'ye bağlı) ise , Şekil 3.3'de görülmektedir. Algoritmaların sonuç bölümlerindeki kutucuklarda gösterilen konteyner konum bilgileri sıralaması, mutlak yerleştirme sırası olmayıp, muhtemel dolu konumları göstermek ve diğer konumların bundan nasıl etkilendiğini görebilmek içindir. Ancak tek bir konum bilgisinin çıkış olarak görüldüğü durumlarda, bu konuma ilgili marjinal değerin yerleştirileceği anlamına gelmektedir.

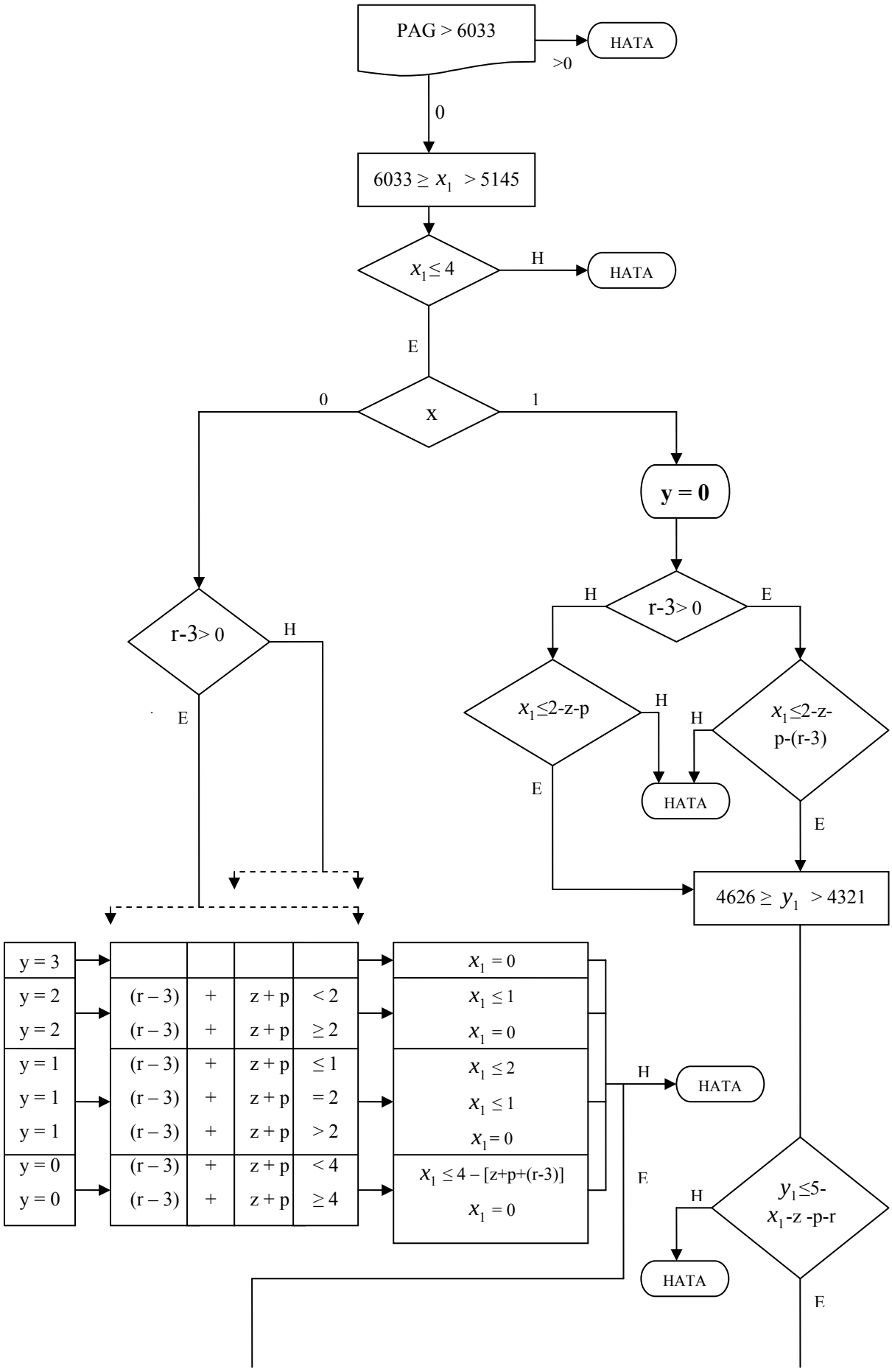


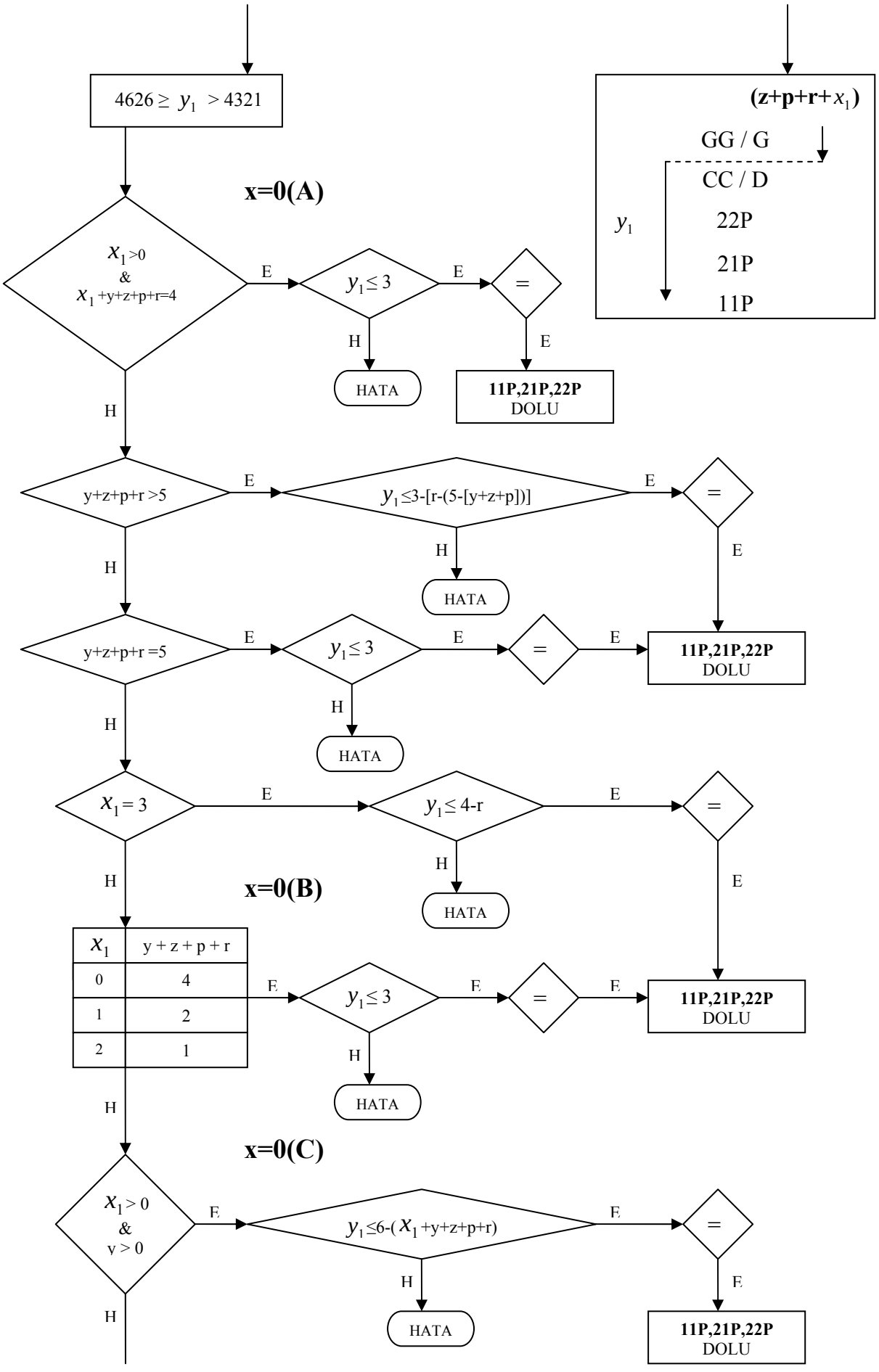
Şekil 3.1 Marjinal konteyner pozisyonları.

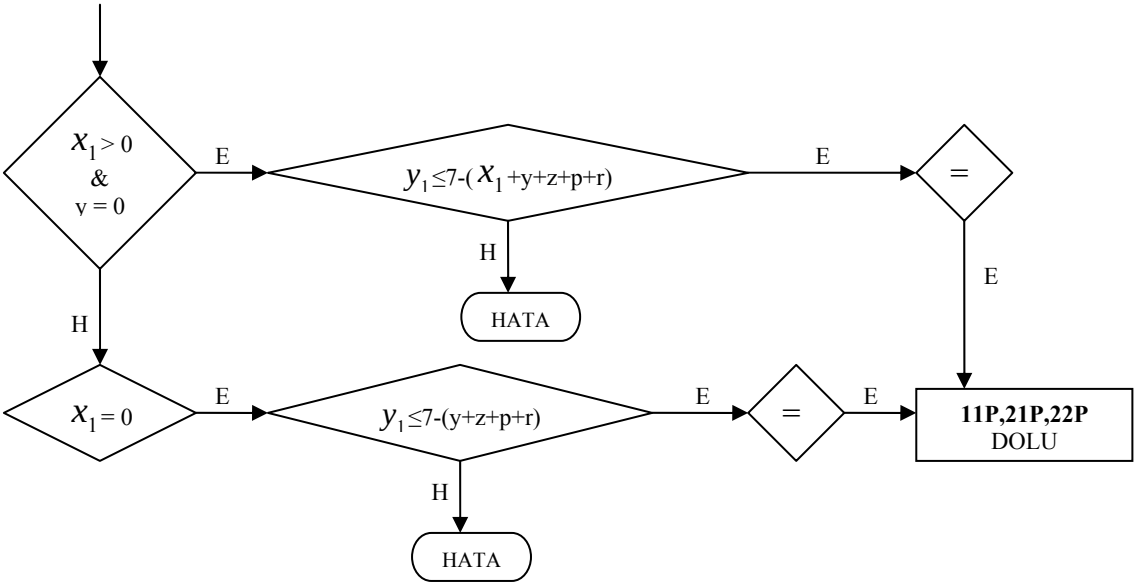

PMC > 6804
>1
HATA
0
6804 ≥x >5670
0
5670 ≥ y > 5446
0
5446 ≥ z > 5145
0
5145 ≥ p > 4626
>1
1
HATA
EF
5670 ≥ y > 5446
>1
0
HATA
5446 ≥ z > 5145
>1
0
1
HATA
GG
5145 ≥ p > 4626
p ≤ 2-z
H
E
HATA
z = 1 & p = 1
E
H
CC
4626 ≥ r > 4281
r ≤ 5-z-p
>3
3≥
HATA
5446 ≥ z > 5145
z ≤ 4 – y
E
H
HATA
y = 3 & z = 1
E
H
GG
5145 ≥ p > 4626
p ≤ 5-y-z
E
H
HATA
y+z+p=5
H
E
p = 1
>4
4≥
HATA
5145 ≥ p > 4626
p ≤ 5 – z
E
H
HATA
4626 ≥ r > 4281
r ≤ 8-p-z
E
H
HATA
>5
5≥
HATA
4626 ≥ r > 4281
r ≤ 8 – p
E
H
HATA



Şekil 3.2 Marjinal girdi filtreleme algoritması PMC bölümü.

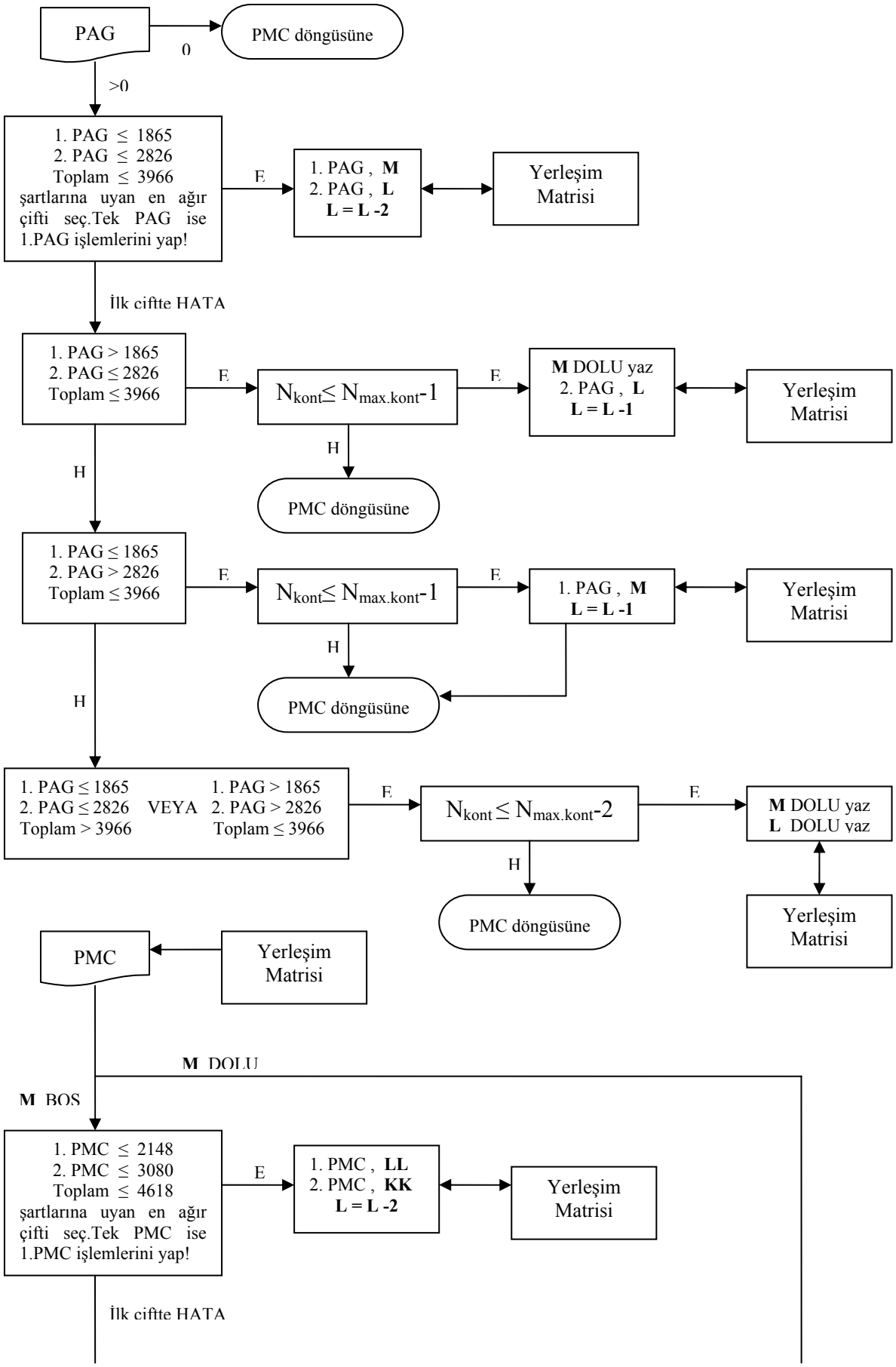

PAG > 6033
HATA
>0
0
6033 ≥ x1 > 5145
x1 ≤ 4
H
HATA
E
x
0
1
y = 0
r-3> 0
H
E
r-3> 0
H
E
x1≤2-z-p
H
HATA
E
x1≤2-z-p-(r-3)
H
E
4626 ≥ y1 > 4321
y = 3
y = 2
y = 2
y = 1
y = 1
y = 1
y = 0
y = 0
(r – 3) + z + p < 2
(r – 3) + z + p ≥ 2
(r – 3) + z + p ≤ 1
(r – 3) + z + p = 2
(r – 3) + z + p > 2
(r – 3) + z + p < 4
(r – 3) + z + p ≥ 4
x1 = 0
x1 ≤ 1
x1 = 0
x1 ≤ 2
x1 ≤ 1
x1 = 0
x1 ≤ 4 – [z+p+(r-3)]
x1 = 0
H
HATA
E
y1≤5-x1-z -p-r
H
HATA
E

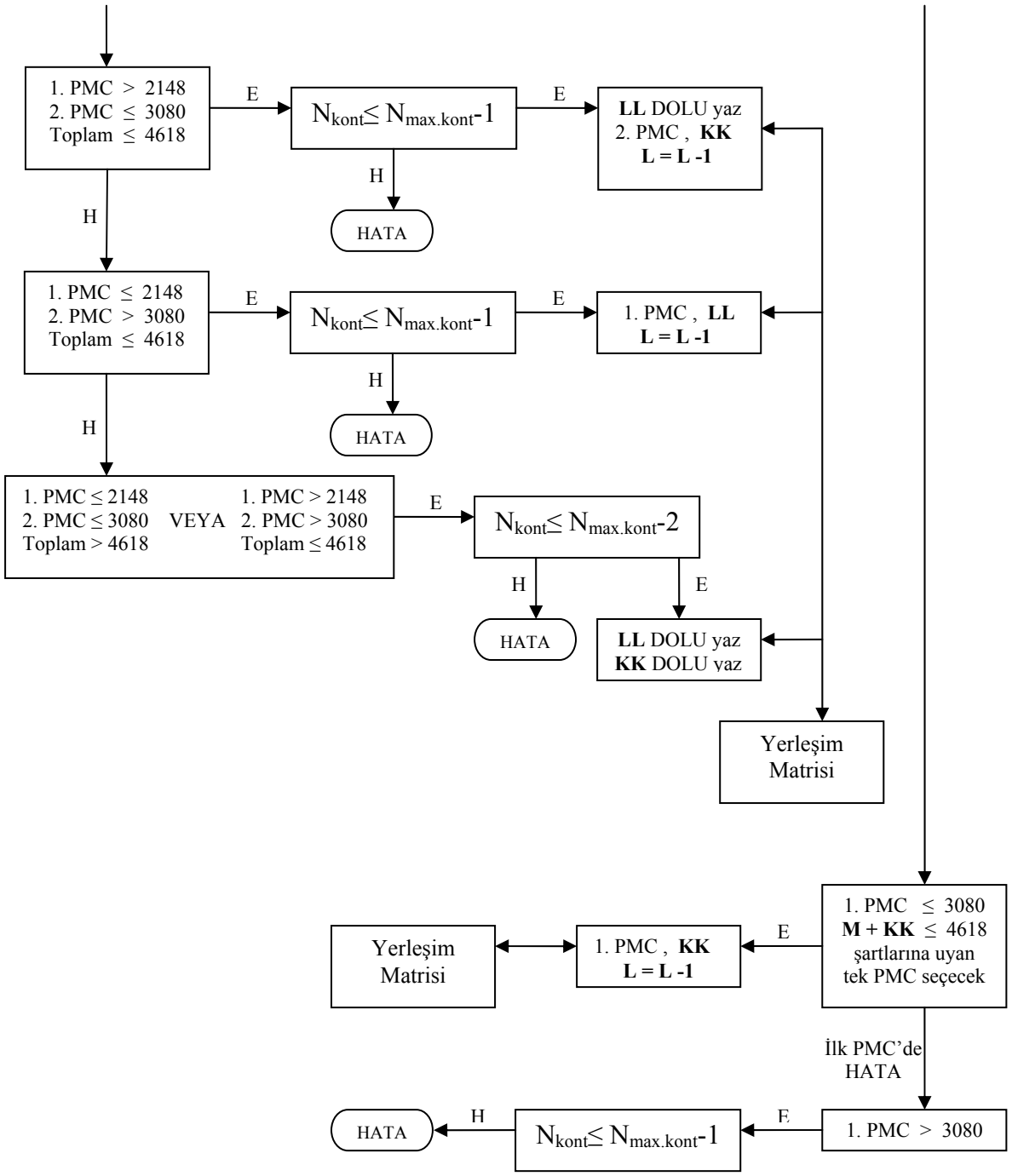

4626 ≥ y1 > 4321
x=0(A)
x1>0 & x1+y+z+p+r=4
E
y1≤3
E
=
H
HATA
E
11P,21P,22P DOLU
H
y+z+p+r >5
E
y1≤3-[r-(5-[y+z+p])]
E
=
H
HATA
E
H
y+z+p+r =5
E
y1≤3
E
=
E
11P,21P,22P DOLU
H
HATA
H
x1 = 3
E
y1≤4-r
E
=
H
HATA
E
H
x=0(B)
x1 | y + z + p + r
0 | 4
1 | 2
2 | 1
E
y1≤3
E
=
E
11P,21P,22P DOLU
H
HATA
H
x=0(C)
x1 > 0 & v > 0
E
y1≤6-(x1+y+z+p+r)
E
=
H
HATA
E
11P,21P,22P DOLU
H
(z+p+r+x1)
GG / G
CC / D
y1
22P
21P
11P



Şekil 3.3 Marjinal girdi filtreleme algoritması PAG bölümü.

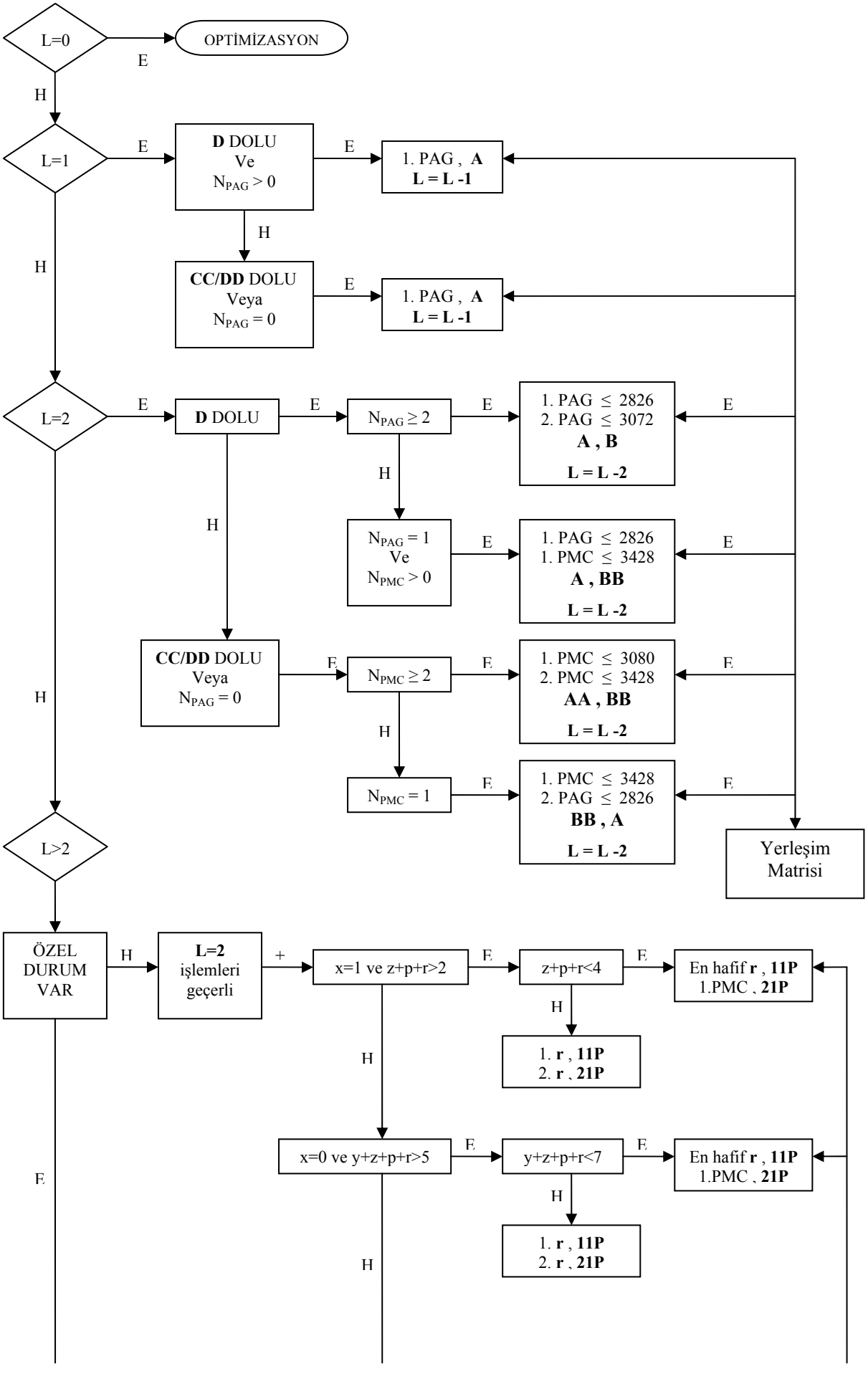
3.2.4.5 Hafif Yükleri Yerleştirme Algoritması

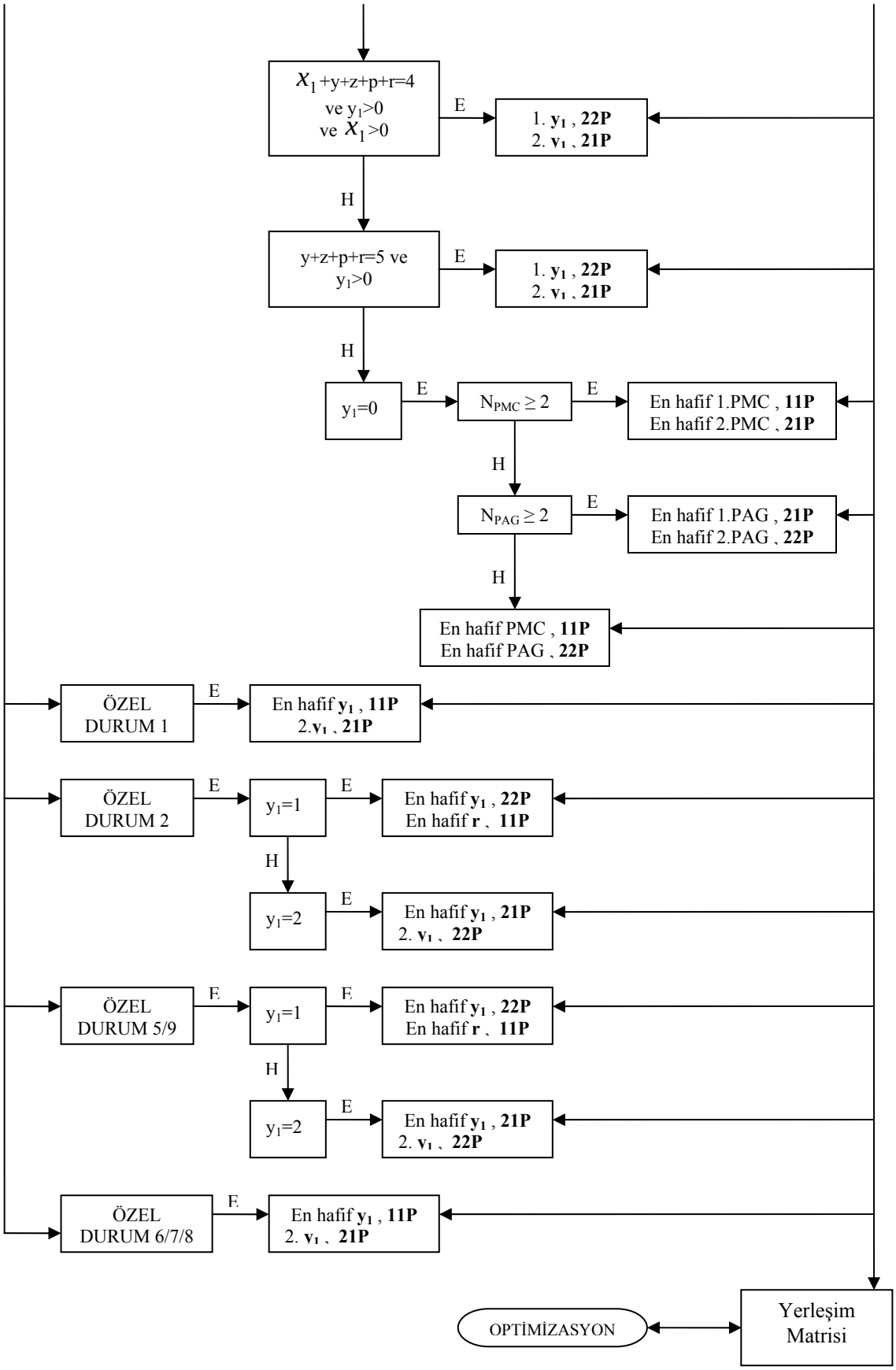




Şekil 3.4 Hafiflerin yerleştirilmesi algoritması (arka taraf).

$N_{max.kont}$ değeri değişik yükleme kombinasyonları için farklı değerler almaktadır. Program içinde o sıradaki yerleşime göre bu değerin tespit edilmesini sağlayan bir bölüm mevcuttur. Bu sayede ilgili yerleşim için boş konum kalıp kalmayacağı anlaşılmakta, gerektiği takdirde belli konumlar boş bırakılabilmektedir.


L=0
OPTİMİZASYON
E
H
L=1
E
D DOLU
Ve
N_PAG > 0
E
1. PAG , A
L = L -1
H
CC/DD DOLU
Veya
N_PAG = 0
E
1. PAG , A
L = L -1
H
L=2
E
D DOLU
E
N_PAG ≥ 2
E
1. PAG ≤ 2826
2. PAG ≤ 3072
A , B
L = L -2
E
H
N_PAG = 1
Ve
N_PMC > 0
E
1. PAG ≤ 2826
1. PMC ≤ 3428
A , BB
L = L -2
E
H
CC/DD DOLU
Veya
N_PAG = 0
E
N_PMC ≥ 2
E
1. PMC ≤ 3080
2. PMC ≤ 3428
AA , BB
L = L -2
E
H
N_PMC = 1
E
1. PMC ≤ 3428
2. PAG ≤ 2826
BB , A
L = L -2
E
H
L>2
Yerleşim
Matrisi
ÖZEL
DURUM
VAR
H
L=2
işlemleri
geçerli
+
x=1 ve z+p+r>2
E
z+p+r<4
E
En hafif r , 11P
1.PMC , 21P
H
1. r , 11P
2. r , 21P
H
x=0 ve y+z+p+r>5
E
y+z+p+r<7
E
En hafif r , 11P
1.PMC , 21P
H
1. r , 11P
2. r , 21P
H
E



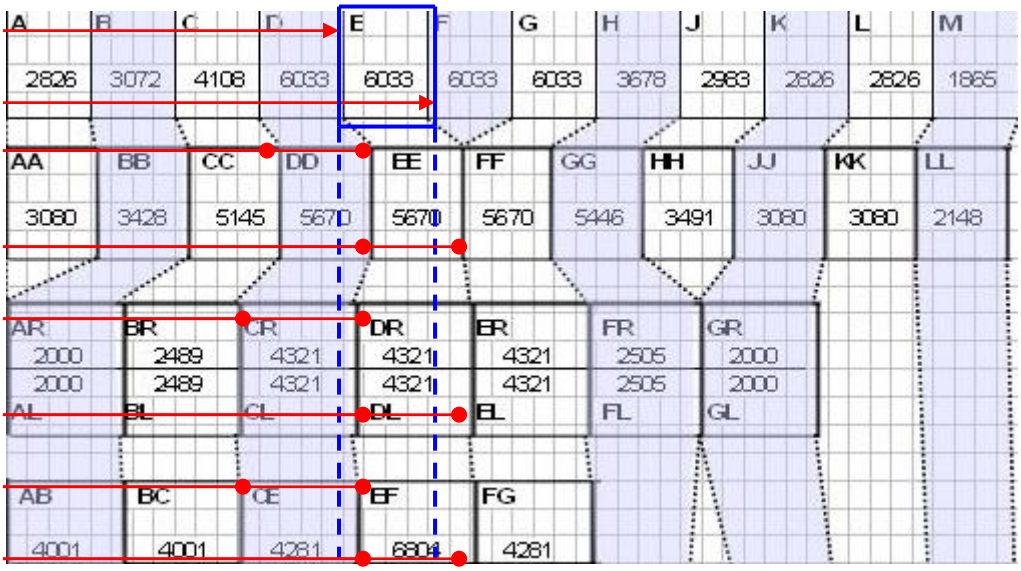
Şekil 3.5 Hafiflerin yerleştirilmesi algoritması (ön ve alt taraf).

3.3 Buluşsal Optimizasyon Algoritması

Programlamada kullanılan buluşsal çözüm algoritması genel olarak Şekil 3.7'de görülmektedir. Algoritma mantığını sırasıyla açıklamak gerekirse, ilk olarak yeri belli olan konteynerlerin konumları, ağırlıkları ve tipleri ile yerleştirmeye tabi olacak diğer konteynerlerin ağırlıkları ve tipleri girilir. Tüm kargo toplam ağırlık kontrolüne girer ve eğer izin verilen kargo yükleme ağırlığı aşılıyorsa, tercihli olmayan kargolar arasından en hafif olanı programdan çıkartılır. İzin verilen toplam kargo ağırlığı hedefi tutana kadar bu işlemi tekrarlar. Bundan sonra program yeri belli olmayan konteynerleri ağırlıklarına göre PAG ve PMC olacak şekilde iki ayrı matriste sıralar.

Sıralama işlemi sayesinde programda, iki ayrı tip için ağırlık sıralamasına göre matris ve bir de yerleşim matrisi bulunmaktadır. PAG ve PMC matrislerinde ağırlık bilgileri ve kargonun kullanılıp kullanılmadığını gösteren bir kod vardır. Bu sayede ilerleyen aşamalarda da başvurulabilecek dinamik bir havuz oluşturulmuş olur. Aynı şekilde yerleşim matrisinde de Şekil 2.8'de görülebilen değişik kargo yerleşim biçimlerinin hepsi matrisin satır ve sütunları şeklinde yer alır. İlgili konuma yükleme yapıldı ise, matris elemanı sıfır iken, yüklenen kargonun ağırlığı eleman olarak atanır. Geometrik olarak birbirini kapatan veya kısıtlayan konumlar ise programın içindeki bir döngü sayesinde herhangi bir yerleştirme yapıldığında, o yerleştirmeden etkilenen tüm konumlar hesaplanarak, matris elemanlarını bir yapmaktadır. Bu sayede ilgili konuma yerleştirme yapılamayacağı, aynı zamanda da gerçekte o konumda herhangi bir konteyner olmadığı anlaşılmaktadır. Örneğin Şekil 2.8'den de görüleceği gibi E konumuna yükleme yapıldığında DD,EE,CR,CL,DR,DL,EF ve CE konumlarına herhangi bir yükleme yapılamaz. Yükleme yapıldıysa matriste E'nin konumunda yüklenen ağırlık değeri, diğerlerinde ise bir olacaktır. Bu yüklemeden etkilenmeyen veya daha önce yükleme yapılmamış tüm elemanlar sıfır olacaktır.

Yapılan yüklemeden geometrik olarak etkilenen konumların hesaplanması ise şu prensibe göre yapılmıştır. Kompartıman baştan sona kadar bir skala gibi kabul edilmiş ve toplam birim uzunluk yataydaki konteyner sayısına bölünerek, her yükleme biçiminin birim katsayısı bulunmuştur. Herhangi bir yükleme yapıldığında ise, program konum bilgisini alır. Buna göre ilgili pozisyonun sırasını birimle çarparak üst sınır, bir eksiğini ise alt sınır olarak hesaplamış olur. Bu uzunluk değerleri, diğer yerleşim biçimlerinin birimlerine bölünerek, hangi aralıklara denk geldiği bulunur (Şekil 3.6). Asıl yerleşimin alt sınır değeri skalada değer olarak kabul edilip, diğer yerleşimlerin birim değerine bölünerek o bölümlerde nerelere denk geldiği tespit edilir. Aynı işlem üst sınır için de yapılarak kapanan tüm konumlar tespit edilir. İşlem her

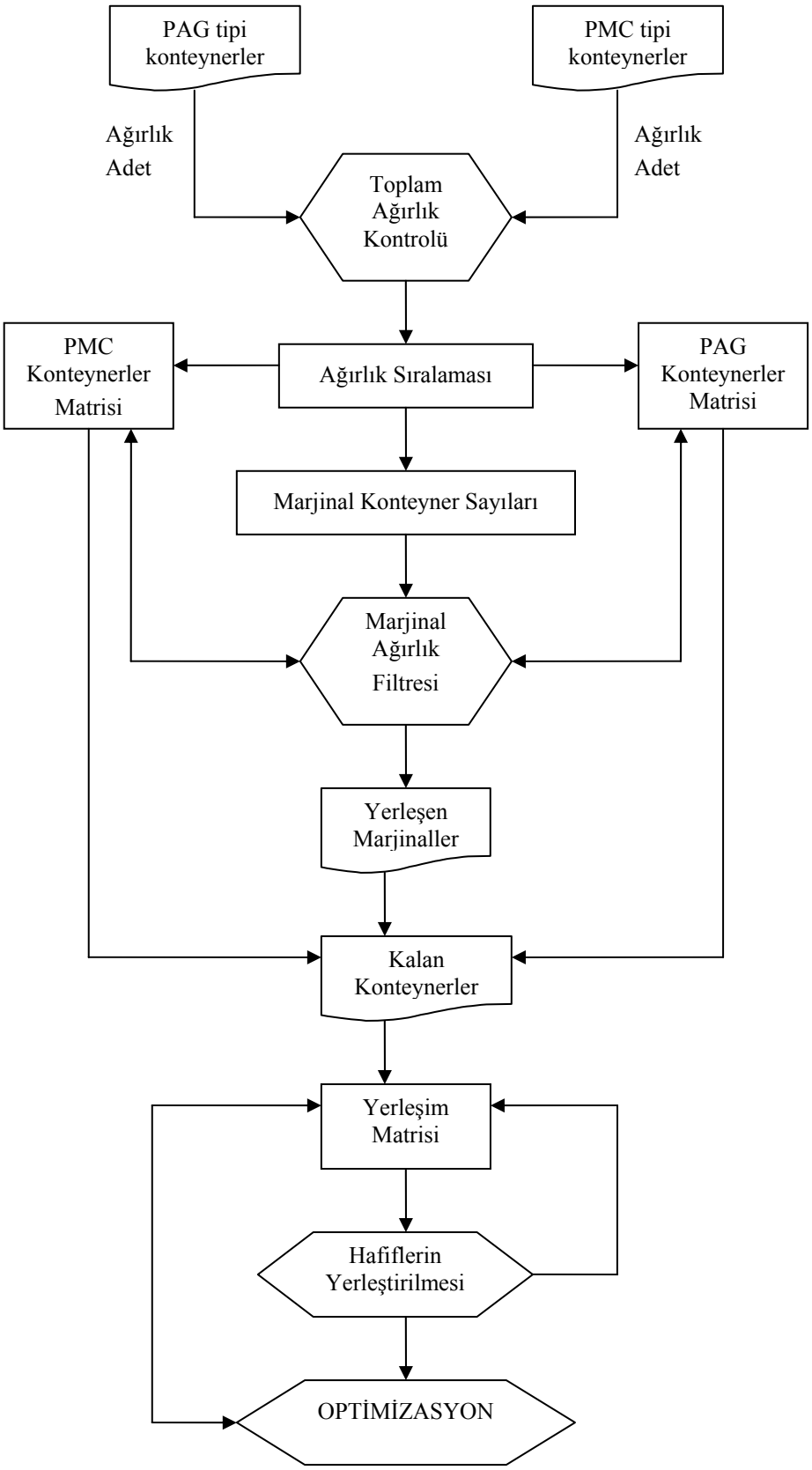


Şekil 3.6 Geometrik olarak kapalı konumların hesaplanması.

farklı yerleşim için tamamlandıktan sonra bulunan yerlere yerleşim matrisinde bir yazılır. Program bu bölüm, Geometrik İlişkilendirme Döngüsü olarak belirtilmiştir.

Sıralama tamamlandıktan sonra marjinal ağırlık filtresi devreye girer. Burada marjinal ağırlıkların sayıları tespit edilerek hata kontrolü yapılır. Yerleşimle ilgili herhangi bir kısıtlama veya zorunluluk olursa bu bilgi yerleşim matrisi, PMC ve PAG matrislerine gider.

Bu aşamada, azami değişken sayısı olan dokuz, yerleşen marjinal ve normal kargo sayılarıyla beraber toplam kargo sayısından çıkartılır. Bu sayede *L* olarak adlandırılan yerleştirilmesi gereken hafif konteyner sayısı hesaplanır. Eğer bu değer sıfır ise, program doğrudan optimizasyona geçer. Ancak sıfırdan büyükse, o zaman program hafiflerin yerleştirilmesi kısımlarından çalışmaya devam eder. Hafiflerin yerleştirilmesi tamamlandıktan sonra tercihli ve yeri belli olmuş konteynerler dışındaki değişkenler, kombinasyonel optimizasyon döngüsüne girer. Bu döngüden çıkan kombinasyonlarla toplam indeks değeri hesaplanır ve daha iyi bir değer hesaplanana kadar o ana kadar ki en iyi değer sonuç olarak kabul edilir. Tüm olasılıklar hesaplandıktan sonra eğer olası en iyi değer limitler içerisinde ise program indeks değerini ve bu değer için hangi konteynerin nereye yüklenmesi gerektiğini gösterir. Limit dışı olması durumunda hata mesajı verir ve yeni konteyner girdileri istemek üzere başa döner. Hafiflerin yerleştirilmesi algoritmasının detayları belki de tüm algoritmanın en kritik kısmıdır. Çünkü bu aşamada yapılacak yanlış bir değerlendirme iyi bir sonuç alınmasını doğrudan etkileyecektir. Algoritmanın bu parçasının detayları, Şekil 3.4 ve Şekil 3.5'de görülebilir.



Şekil 3.7 Buluşsal optimizasyon algoritması.

3.4 Buluşsal Optimizasyon C++ Program Kodu ve Açıklaması

```
int write_matrix(int, int, int);
int read_matrix(int, int);
int load_matrix[7][11];

double row_constant[5];
double up_limit;
double down_limit;
double max_size;
double min_size;
```

// Geometrik İlişkilendirme Döngüsü

```
int write_matrix(int x, int y, int val)
        {
        row_constant[0]=14.0;
        row_constant[1]=15.0;
        row_constant[2]=19.0;
        row_constant[3]=19.0;
        row_constant[4]=19.5;
```

} her yerleşimin skaladaki birim değeri

```
                int cnt;
                int i;

                min_size=y*row_constant[x];
                max_size=y*row_constant[x+1];

        for(cnt=0;cnt++;cnt<6)
                for(i=1;i++;i<12)
                        {
                        up_limit=i*row_constant[cnt];
                        down_limit=(i-1)*row_constant[cnt];
                                if (min_size>down_limit && min_size<up_limit)
                                        load_matrix[cnt][i]=1;
                                if (max_size>down_limit && max_size<up_limit)
                                        load_matrix[cnt][i]=1;
                                if (down_limit>min_size && max_size>up_limit)
                                        load_matrix[cnt][i]=1;
                        }
                load_matrix[x][y]=val;
                return 0;
        }

        int read_matrix(int x, int y)
        {
                int val;
                val=load_matrix[x][y];
                return val;
        }

int _tmain(int argc, _TCHAR* argv[])
{

        int f;
        int notpossible=0;
        int npx, npy;
        // Sıralama döngüsü için değişkenler
        int i, j, a;
        // Marjinal ağırlık filtresi için değişkenler
        int pmc_cond1, pmc_cond2, pmc_cond3, pmc_cond4, pmc_cond5;
        int pag_cond1, pag_cond2, pag_cond3, pag_cond4, pag_cond5;
        int total_containers, total_pmc, total_pag;
        char szInput [256];
        int I1, I2, I3, I4;
        int pag_list[15][1];
        int pmc_list[14][1];
```

```
        // Yerleşim kontrolü için değişkenler
        int light_check=0;
        int max_placeable=0;
        int remaining_containers;
        int suggestion1=0;
        int suggestion2=0;
        int suggestion3=0;
        int suggestion4=0;
        int suggestion5=0;
        int suggestion6=0;
        int suggestion7=0;
        int suggestion8=0;
        int suggestion9=0;
        int pag;
        int pmc;

        // Optimizasyon değişkenleri
        int f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15;
        int
d1[16],d2[16],d3[16],d4[16],d5[16],d6[16],d7[16],d8[16],d9[16],d10[16],d11[16],d12[
16],d13[16],d14[16],d15[16];
        float k[16];
        int mass[16][2];
        float Itotal=0;
        float hedef=0;
        float checkhedef=0;
        float oldhedef=90;

        // Konteyner ağırlıklarının girilmesi
  for(i=1;i<20;i++)
  {
    printf ("Enter mass: %d \n ", i);
        fgets ( szInput, 256, stdin );
        if (i<10)
        {
                mass[i][1] = atoi (szInput);
        }

        if (i>=10)
        {
                switch(i)
                {
                        case 10:
                                k[1]=atoi (szInput);
                                break;
                        case 11:
                                k[11]=atoi (szInput);
                                break;
                        case 12:
                                k[12]=atoi (szInput);
                                break;
                        case 13:
                                k[13]=atoi (szInput);
                                break;
                        case 14:
                                k[14]=atoi (szInput);
                                break;
                        case 15:
                                k[15]=atoi (szInput);
                                break;
                        case 16:
                                I1=atoi (szInput);
                                break;
                        case 17:
                                I2=atoi (szInput);
                                break;
                        case 18:
                                I3=atoi (szInput);
```

```
                                break;
                        case 19:
                                I4=atoi (szInput);
                                break;
                }
        }
  }
```

// Ağırlık sıralaması döngüsü

```
        for(j=2;j<=7;j++)
        {
                a=mass[j][1];
                i=j-1;
                while (i>0 && mass[i][1]>a)
                {
                        mass[i+1][1]=mass[i][1];
                        mass[i+1][2]=mass[i][2];
                        i--;
                }
        }
```

// Marjinal ağırlık sınıflandırması

```
        pmc_cond1=0;
        pmc_cond2=0;
        pmc_cond3=0;
        pmc_cond4=0;
        pmc_cond5=0;
        total_pmc=0;
        pag_cond1=0;
        pag_cond2=0;
        pag_cond3=0;
        pag_cond4=0;
        pag_cond5=0;
        total_pag=0;

        for (i=1;i<=total_containers;i++)
                if (mass[i][2]==1)
                {
                        total_pmc++;
                        if (mass[i][1]<=6804 && mass[i][1]>5670)
                                pmc_cond1++;
                        if (mass[i][1]<=5670 && mass[i][1]>5446)
                                pmc_cond2++;
                        if (mass[i][1]<=5446 && mass[i][1]>5145)
                                pmc_cond3++;
                        if (mass[i][1]<=5145 && mass[i][1]>4626)
                                pmc_cond4++;
                        if (mass[i][1]<=4626 && mass[i][1]>4281)
                                pmc_cond5++;
                        if (mass[i][1]>6804)
                                printf("The plane can not handle a container which
weights more than 6033!\n");
                        pmc_list[total_pmc][0]=mass[i][1];
                        pmc_list[total_pmc][1]=1;
                }
                else
                {
                        total_pag++;
                        if (mass[i][1]<=6033 && mass[i][1]>4626)
                                pag_cond1++;
                        if (mass[i][1]<=4626 && mass[i][1]>4321)
                                pag_cond2++;
                        if (mass[i][1]>6033)
                                printf("The plane can not handle a container which
weights more than 6033!\n");
                        pag_list[total_pag][0]=mass[i][1];
                        pag_list[total_pag][1]=1;

                }
```

// Marjinal Ağırlık Filtresi

```
        if (pmc_cond1>1)
                printf("The plane can not handle more than one container which
weight between 5670 and 6804Kg!\n");
        if (pmc_cond1=0 && pmc_cond2>3)
                printf("The plane can not handle more than three container which
weight between 5446 and 5670Kg!\n");
        if (pmc_cond1==0 && pmc_cond2==0 && pmc_cond3>4)
                printf("The plane can not handle more than four containers which
weight between 5145 and 5446Kg!\n");
        if (pmc_cond1==0 && pmc_cond2==0 && pmc_cond3==0 && pmc_cond4>5)
                printf("The plane can not handle more than five containers which
weight between 4626 and 5145Kg!\n");
        if (pmc_cond5>8-pmc_cond4)
                printf("This is an error!\n");

        if (pmc_cond1==1)
        {
                printf("EF\n");

                if (pmc_cond2>=1)
                        printf("This is an error!\n");

        // Gerçekten dolu ise "write_matrix" içine asıl ağırlığı yaz.Geometrik dolu
olan yerlere 1 yaz. Yerleşen konteynerin PAG ya da PMC matrisindeki ikinci sütun
elemanını 0 (kullanılmış) yap.

                if (pmc_cond3==1)
                        // printf("GG\n");
                                for (i=1;i<total_pmc+1;i++)
                                        if (pmc_list[i][0]<=5446 &&
pmc_list[i][0]>5145 && read_matrix(1,6)==0 && pmc_list[i][1]==1)
                                        {
                                                write_matrix(1, 6,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }

                if (pmc_cond3>1)
                        printf("This is an error!\n");

                if (pmc_cond4==2 && pmc_cond3!=0)
                        printf("This is an error!\n");

                if (pmc_cond4==1 && pmc_cond3==1)
                        // printf("CC "pmc_cond4" \n");
                        if (read_matrix(1,2)==0)
                                write_matrix(1, 2, 1);

                if (pmc_cond5>5-pmc_cond3-pmc_cond4)
                        printf("This is an error!\n");
                else
                        if (pmc_cond5>0)
                        {
                                if (pmc_cond3==0 && pmc_cond4==0 && pmc_cond5==1)
                                        // printf("GG Type4\n"); ******** write
mass values for filled spaces and the corresponding gaps will be filled with 1
                                        if (read_matrix(1,6)==0)
                                                write_matrix(1, 6, 1);
                                if (pmc_cond3==0 && pmc_cond4==0 && pmc_cond5>1)
                                        printf("Heaviest GG Type2\n");
                                if (pmc_cond3==1 && pmc_cond4==0 && pmc_cond5>0)
                                        printf("GG (pmc_cond3 to GG) Type2\n");
                                if (pmc_cond3==0 && pmc_cond4==1)
                                {
                                        if (pmc_cond5==4)
                                            printf("Light pmc_cond5 to 11P\n");
                                }
```

```
                                if (pmc_cond3==1 && pmc_cond4==1 && pmc_cond5>0)
                            printf("pmc_cond3 to GG, pmc_cond4 to CC, Type2\n");
                                if (pmc_cond3==0 && pmc_cond4==2 && pmc_cond5==3)
                                        printf("Light pmc_cond5 to 11P Type2\n");
                                if (pmc_cond3==0 && pmc_cond4==2 && pmc_cond5<3)
                                {
                                                if (mass[1][1]>mass[1][2])
                                        printf("Light pmc_cond5 to AA Type2\n");
                                                else
                                                printf("Light PAG to A Type1\n");
                                }
                                if (pmc_cond3==0 && pmc_cond4==0 && pmc_cond5==0)
                                        printf("Do nothing for now!\n");
                        }
        }
        if (pmc_cond1==0 && pmc_cond2<=3 && pmc_cond2>0)
        {
                printf("Condition EE");

                if (pmc_cond2==1)
                {
                        if (pmc_cond3>4-pmc_cond2)
                                printf("5145 ile 5446 arası PMC'ler ile 5446 ile
5670 arası PMC'lerin toplamı 4 olmalı\n");

                        if (pmc_cond3<=4-pmc_cond2)
                        {
                                if (pmc_cond3>0)
                                {
                                        if (pmc_cond2==3 && pmc_cond3==1)
                                                //printf("pmc_cond3 to GG\n");
                                                if (read_matrix(1,6)==0)
                                                        write_matrix(1, 6, 1);
                                }
                                if (pmc_cond4>0 && pmc_cond4<=5-pmc_cond2-
pmc_cond3)
                                {
                                        if (pmc_cond4>0 &&
pmc_cond2+pmc_cond3+pmc_cond4==5)
                                                if (read_matrix(1,2)==0)
                                                        write_matrix(1, 2, 1);

                                        if (pmc_cond5<=8-pmc_cond2-pmc_cond3-
pmc_cond4 && pmc_cond5>0)
                                        {
                                                if
(pmc_cond2+pmc_cond3+pmc_cond4==4)
                                                printf("pmc_cond5 to CC\n");
                                                if
(pmc_cond2+pmc_cond3+pmc_cond4>=5)
                                                        printf("Excess pmc_cond5
to 22P, 21P, 11P\n");
                                        }
                                }
                        }
                }
        }

        if (pmc_cond1==0 && pmc_cond2==0 && pmc_cond3>0 && pmc_cond3<=0)
        {
                if (pmc_cond4<=5-pmc_cond3 && pmc_cond5<=8-pmc_cond3-pmc_cond4)
                {
                        if (pmc_cond5<=8-pmc_cond2-pmc_cond3-pmc_cond4 &&
pmc_cond5>0)
                        {
                                if (pmc_cond2+pmc_cond3+pmc_cond4==4)
                                        printf("pmc_cond5 to CC\n");
                                if (pmc_cond2+pmc_cond3+pmc_cond4>=5)
                                printf("Excess pmc_cond5 to 22P, 21P, 11P\n");
```

```
                        }
                }
        }

        if (pmc_cond1==0 && pmc_cond2==0 && pmc_cond3==0 && pmc_cond4<=5 &&
pmc_cond4>0)
        {
                if (pmc_cond5<=8-pmc_cond4)
                {
                        if (pmc_cond5<=8-pmc_cond2-pmc_cond3-pmc_cond4 &&
pmc_cond5>0)
                        {
                                if (pmc_cond2+pmc_cond3+pmc_cond4==4)
                                        printf("pmc_cond5 to CC\n");
                                if (pmc_cond2+pmc_cond3+pmc_cond4>=5)
                                        printf("Excess pmc_cond5 to 22P, 21P,
11P\n");
                        }
                }
        }


        // x = 1
        if (pag_cond1<=4 && pmc_cond1==1 && pmc_cond2==0)
        {
                if (pmc_cond5>3)
                        if (2-(pmc_cond3+pmc_cond4+(pmc_cond5-3))>0 &&
pag_cond1<=2-(pmc_cond3+pmc_cond4+(pmc_cond5-3)))
                                printf("Check!\n");
                        else
                                printf("Excessive PAG - 5\n");
                else
                        if (2-(pmc_cond3+pmc_cond4)>0 && pag_cond1<=2-
(pmc_cond3+pmc_cond4))
                                printf("Check!\n");
                        else
                                printf("Excessive PAG - 5\n");

                if (pag_cond1==2)
                        printf("G and D are occupied!\n");

                if (pag_cond1==1)
                        if (pmc_cond4==0 && pmc_cond3==1)
                                printf("pag_cond1 to D and pmc_cond3 to GG\n");

                if (5-pmc_cond3-pmc_cond4-pmc_cond5-pag_cond1>=0)
                        if (pag_cond2<=5-pmc_cond3-pmc_cond4-pmc_cond5-pag_cond1)
                                printf("Check!\n");
                        else
                                printf("Excessive pag_cond2 in 5!\n");

                if (pag_cond2==5-pmc_cond3-pmc_cond4-pmc_cond5-pag_cond1)
                        printf("11p, 21P and 22P are occupied!\n");

                if (pag_cond2==1 && 5-pmc_cond3-pmc_cond4-pmc_cond5-pag_cond1==1)
                        printf("pag_cond2 to 22P!\n");
                if (pag_cond2==2 && 5-pmc_cond3-pmc_cond4-pmc_cond5-pag_cond1==2)
                        printf("Ligth pmc_cond5 to 11P!\n");

        }

        // x = 0
        if (pag_cond1<=4 && pmc_cond1==0)
        {
                if (pmc_cond4>3)
                {
                        if (pmc_cond2==3 && pag_cond1>0)
                                printf("This is an error!\n ");
                        if (pmc_cond2==2)
```

```
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)<2 &&
pag_cond1>1)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==2)
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)>=2 &&
pag_cond1!=0)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==1)
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)<=1 &&
pag_cond1>2)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==1)
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)==2 &&
pag_cond1>1)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==1)
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)>2 &&
pag_cond1!=0)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==0)
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)<4 &&
pag_cond1>(4-pmc_cond3+pmc_cond4+(pmc_cond5-3)))
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==0)
                                if(pmc_cond3+pmc_cond4+(pmc_cond5-3)>=4 &&
pag_cond1!=0)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                }
                else
                {
                        if (pmc_cond2==3 && pag_cond1>0)
                                printf("This is an error!\n");
                        if (pmc_cond2==2)
                                if(pmc_cond3+pmc_cond4<2 && pag_cond1>1)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==2)
                                if(pmc_cond3+pmc_cond4>=2 && pag_cond1!=0)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==1)
                                if(pmc_cond3+pmc_cond4<=1 && pag_cond1>2)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==1)
                                if(pmc_cond3+pmc_cond4==2 && pag_cond1>1)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==1)
                                if(pmc_cond3+pmc_cond4>2 && pag_cond1!=0)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==0)
                                if(pmc_cond3+pmc_cond4<4 && pag_cond1>(4-
pmc_cond3+pmc_cond4))
                                        printf("Excessive PAG count : y1
condition failed!\n");
                        if (pmc_cond2==0)
                                if(pmc_cond3+pmc_cond4>=4 && pag_cond1!=0)
                                        printf("Excessive PAG count : y1
condition failed!\n");
                }
```

```
        }

        if (pag_cond1<=4 && pmc_cond1==0)
        {
                // x = 0 (A)
                if (pag_cond1>0 &&
pmc_cond5+pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4==4 && pag_cond2<=3)
                        printf("Light pag_cond1 to D\n");
                if (pag_cond1>0 &&
pmc_cond5+pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4==4 && pag_cond2>3)
                        printf("The PAG number between the weights 4321 and 4626
(z1) is excessive (6-A-1)!\n");
                if (pag_cond1>0 &&
pmc_cond5+pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4==4 && pag_cond2==3)
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion1=1;
                }

                if (pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5>5 && pag_cond2<=3-
(pmc_cond5-(5-pmc_cond2-pmc_cond3-pmc_cond4)))
                        printf("Ckeck!\n");
                else
                        printf("Excessive PAG count! x=0(A)\n");

                if (pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==5 && pag_cond2<=3)
                        printf("Ckeck!\n");
                else
                        printf("Excessive PAG count! x=0(A)\n");

                if (pag_cond2==3 || pag_cond2==3-(pmc_cond5-(5-pmc_cond2-pmc_cond3-
pmc_cond4)))
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion2=1;
                }
                if (pag_cond2==1 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5>5 &&
pag_cond2==3-(pmc_cond5-(5-pmc_cond2-pmc_cond3-pmc_cond4)))
                {
                        printf("pag_cond2 to 22P\n");
                        suggestion3=1;
                }
                if (pag_cond2==2 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5>5 &&
pag_cond2==3-(pmc_cond5-(5-pmc_cond2-pmc_cond3-pmc_cond4)))
                {
                        printf("Light pmc_cond5 to 11P\n");
                        suggestion4=1;
                }

                // x = 0 (B)
                if (pag_cond1==3 && pag_cond2<=4-pmc_cond5)
                        printf("E is Full!\n");
                if (pag_cond1==2 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==1 &&
pag_cond2<=3)
                        printf("DD is Full!\n");
                if (pag_cond1==1 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==2 &&
pag_cond2<=3)
                        printf("DD is Full!\n");
                if (pag_cond1==0 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==4 &&
pag_cond2<=3)
                        printf("DD is Full!\n");

                if (pag_cond1==2 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==1 &&
pag_cond2==3)
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion5=1;
                }
```

```
                if (pag_cond1==1 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==2 &&
pag_cond2==3)
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion6=1;
                }
                if (pag_cond1==0 && pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==4 &&
pag_cond2==3)
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion7=1;
                }

                if (pag_cond1==3 && pag_cond2==4-pmc_cond5)
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion8=1;
                }

                // x = 0 (C)
                if (pag_cond1>0 && pmc_cond2>0 && pag_cond2<=6-
(pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5))
                        printf("Ckeck!\n");
                else
                        printf("Excessive PAG count! x=0(C)\n");
                if (pag_cond1>0 && pmc_cond2==0 && pag_cond2<=7-
(pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5))
                        printf("Ckeck!\n");
                else
                        printf("Excessive PAG count! x=0(C)\n");
                if (pag_cond1==0 && pag_cond2<=7-
(pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5))
                        printf("Ckeck!\n");
                else
                        printf("Excessive PAG count! x=0(C)\n");
                if (pag_cond2==7-(pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5) ||
pag_cond2==7-(pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5) || pag_cond2==6-
(pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5))
                {
                        printf("11P, 21P and 22P are occupied!\n");
                        suggestion9=1;
                }
        }

        // Kalan konteynerlerin hesaplanması
        for (i=0;i<16;i++)
        {
                if (pmc_list[i][1]==1)
                        pmc++;
                if (pag_list[i][1]==1)
                        pag++;
        }
        remaining_containers=pag+pmc;
                if (remaining_containers>9)
                        remaining_containers=remaining_containers-9;



        // N_max kont değerinin tespit edilmesi

        if (read_matrix(4,3)>1)
        {
                if (pmc_cond3+pmc_cond4+pmc_cond5==0 && total_pmc<=4)
                        max_placeable=14;
                if (pmc_cond3+pmc_cond4==0 && pmc_cond5+pag_cond2>3 && pag_cond2>=2
&& total_pmc<=4)
                        max_placeable=14;
                if (pmc_cond3==1 && pmc_cond4==0 && pmc_cond5<=3)
                        if (total_pmc<=(9-pag_cond2))
```

```
                        max_placeable=13;
        if (pmc_cond3==0 && pmc_cond4==1 && pmc_cond5<=3)
                if (total_pmc<=(2-pag_cond2))
                        max_placeable=13;
if (pmc_cond3==1 && pmc_cond4==1)
                max_placeable=12;
        if (pag_cond1==0 && pag_cond2<=3 && total_pag<=3)
                max_placeable=12;
}
else
        printf("Finding Max_Placable Failed for EF condition 1! \n");

if (read_matrix(4,3)<=1)
{
        if (pmc_cond3+pmc_cond4+pmc_cond2==0 && pmc_cond5<=3)
                max_placeable=15;
        if (pmc_cond3+pmc_cond4+pmc_cond2>0 || pmc_cond5>3)
                max_placeable=15;
        if (pag_cond2==3 && pmc_cond5>0)
                max_placeable=14;
}
else
        printf("Finding Max_Placable Failed for EF condition 0! \n");
```

// Hafiflerin Yerleştirilmesi
// Hafiflerin Yerleştirilmesi (arka)

```
light_check=0;
```

// PAG

```
if (total_pag>=2)
        for(i=1;i<total_pag;i++)
                if (pag_list[i][0]<=1865 && pag_list[i][1]==1)
                        if (pag_list[i+1][0]+pag_list[i][0]<=3966)

if (light_check>0)
{
        if (read_matrix(0,11)==0)
                {
                        write_matrix(0, 11, pag_list[light_check][0]);
                        pag_list[light_check][1]=0;
                }
        if (read_matrix(0,10)==0)
                {
                        write_matrix(0, 10, pag_list[light_check+1][0]);
                        pag_list[light_check+1][1]=0;
                }
        light_check=0;
        remaining_containers=remaining_containers-2;
}

if (pag_list[0][0]>1865 && pag_list[1][0]<=2826 && pag_list[1][0]>0)
        if ((max_placeable-1)>=(total_pag+total_pmc))
        {
                if (read_matrix(0,11)==0)
                        {
                                write_matrix(0, 11, 1);
                        }
                if (read_matrix(0,10)==0 && pag_list[1][1]==1)
                        {
                                write_matrix(0, 10, pag_list[1][0]);
                                pag_list[1][1]=0;

remaining_containers=remaining_containers-1;
                        }
        }

if (pag_list[0][0]<=1865 && pag_list[1][0]>2826 && pag_list[0][0]>0)
        {
                if (read_matrix(0,11)==0 && pag_list[0][1]==1)
                        {
```

```
                                        write_matrix(0, 11, pag_list[0][0]);
                                        pag_list[0][1]=0;

        remaining_containers=remaining_containers-1;
                                }
                }

        if (pag_list[0][0]>1865 && pag_list[1][0]>2826 ||
(pag_list[0][0]+pag_list[1][0])>3966)
                if ((max_placeable-2)>=(total_pag+total_pmc))
                        {
                                if (read_matrix(0,11)==0)
                                        write_matrix(0, 11, 1);

                                if (read_matrix(0,10)==0)
                                        write_matrix(0, 10, 1);
                        }

        if (total_pag==1)
        {
                if (pag_list[0][0]<=1865)
                {
                        if (read_matrix(0,11)==0 && pag_list[0][1]==1)
                                {
                                        write_matrix(0, 11, pag_list[0][0]);
                                        pag_list[0][1]=0;

        remaining_containers=remaining_containers-1;
                                }
                }
                else if ((max_placeable-1)>=(total_pag+total_pmc))
                        if (read_matrix(0,11)==0)
                                write_matrix(0, 11, 1);
        }

        // PMC
        light_check=0;
        if (total_pmc>=2 && read_matrix(0,11)==0)
                for(i=1;i<total_pmc;i++)
                        if (pmc_list[i][0]<=2148 && pmc_list[i][1]==1)
                                if (pmc_list[i+1][0]<=3080 && pmc_list[i+1][1]==1)
                                        if
(pmc_list[i+1][0]+pmc_list[i][0]<=4618)
                                                light_check=i;            // ****
Returns the suitable "i" value to the code below.

        if (light_check>0)
        {
                if (read_matrix(1,10)==0)
                        {
                                write_matrix(1, 10, pmc_list[light_check][0]);
                                pmc_list[light_check][1]=0;
                        }
                if (read_matrix(1,9)==0)
                        {
                                write_matrix(1, 9, pmc_list[light_check+1][0]);
                                pmc_list[light_check+1][1]=0;
                        }
                light_check=0;
                remaining_containers=remaining_containers-2;
        }


        if (pmc_list[0][0]>2148 && pmc_list[1][0]<=3080 && pmc_list[1][0]>0)
                if ((max_placeable-1)>=(total_pag+total_pmc))
                {
                        if (read_matrix(1,10)==0)
                                write_matrix(1, 10, 1);
```

```
                        if (read_matrix(1,9)==0 && pmc_list[1][1]==1)
                                {
                                        write_matrix(1, 9, pmc_list[1][0]);
                                        pmc_list[1][1]=0;

        remaining_containers=remaining_containers-1;
                                }
                }
                else
                        printf("No suitable light containers left for placement!
\n");


        if (pmc_list[0][0]<=2148 && pmc_list[1][0]>3080 && pmc_list[0][0]>0)
                if ((max_placeable-1)>=(total_pag+total_pmc))
                {
                        if (read_matrix(1,10)==0 && pmc_list[0][1]==1)
                                {
                                        write_matrix(1, 10, pmc_list[0][0]);
                                        pmc_list[0][1]=0;

        remaining_containers=remaining_containers-1;
                                }
                }
                else
                        printf("No suitable light containers left for placement!
\n");


        if (pmc_list[0][0]>2148 && pmc_list[1][0]>3080 ||
(pmc_list[0][0]+pmc_list[1][0])>4618)
                if ((max_placeable-2)>=(total_pag+total_pmc))
                        {
                                if (read_matrix(1,10)==0)
                                        write_matrix(1, 10, 1);

                                if (read_matrix(1,9)==0)
                                        write_matrix(1, 9, 1);
                        }
                        else
                                printf("No suitable light containers left for
placement! \n");

        if (read_matrix(0,11)>1)
        {
                light_check=0;
                for(i=0;i<total_pmc;i++)
                        if (pmc_list[i][0]<=3080 &&
read_matrix(0,11)+pmc_list[i][0]<=4618)
                                light_check=i;

                if (light_check>0)
                {
                        if (read_matrix(1,9)==0)
                                {
                                        write_matrix(1, 9,
pmc_list[light_check][0]);
                                        pmc_list[light_check][1]=0;

        remaining_containers=remaining_containers-1;
                                }
                        light_check=0;
                }
                else if ((max_placeable-1)<(total_pag+total_pmc))
                        printf("No suitable light containers left for placement!
\n");
        }

        if (total_pmc==1 && read_matrix(0,11)==0)
```

```
        {
                if (pmc_list[0][0]<=2148)
                {
                        if (read_matrix(1,10)==0 && pmc_list[0][1]==1)
                                {
                                        write_matrix(1, 10, pmc_list[0][0]);
                                        pag_list[0][1]=0;

        remaining_containers=remaining_containers-1;
                                }
                }
                else if ((max_placeable-1)<(total_pag+total_pmc))
                        printf("No suitable light containers left for placement!
\n");
        }
```

// Hafiflerin Yerleştirilmesi (ön)

```
        for (i=0;i<16;i++)
        {
                if (pmc_list[i][1]==1)
                        pmc++;
                if (pag_list[i][1]==1)
                        pag++;
        }

        if (remaining_containers==1)
        {
                if (read_matrix(0,3)>1 && pag>0)
                        for (i=0;i<16;i++)
                        {
                                if (pag_list[i][1]==1 && read_matrix(0,0)==0)
                                {
                                        write_matrix(0, 0,
pag_list[i][0]);
                                        pag_list[i][1]=0;
                                }
                        }
                if (read_matrix(1,2)>1 || read_matrix(1,3)>1 || pag==0)
                        for (i=0;i<16;i++)
                        {
                                if (pmc_list[i][1]==1 && read_matrix(1,0)==0)
                                {
                                        write_matrix(1, 0,
pmc_list[i][0]);
                                        pmc_list[i][1]=0;
                                }
                        }
        }

        if (remaining_containers==2)
        {
                if (read_matrix(0,3)>1)
                        if (pag>=2)
                                for (i=0;i<16;i++)
                                {
                                        if (pag_list[i][1]==1 &&
pag_list[i][0]<2826 && read_matrix(0, 0)==0)
                                        {
                                                write_matrix(0, 0,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                        if (pag_list[i][1]==1 &&
pag_list[i][0]<3072 && read_matrix(0, 1)==0)
                                        {
                                                write_matrix(0, 1,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
```

```
                                }

                        if (pag>=1 && pmc>0)
                                for (i=0;i<16;i++)
                                {
                                        if (pag_list[i][1]==1 &&
pag_list[i][0]<2826 && read_matrix(0, 0)==0)
                                        {
                                                write_matrix(0, 0,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                        if (pmc_list[i][1]==1 &&
pmc_list[i][0]<3428 && read_matrix(1, 1)==0)
                                        {
                                                write_matrix(1, 1,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                }

                if (read_matrix(1,2)>1 || read_matrix(1,3)>1 || pag==0)
                        if (pmc>=2)
                                for (i=0;i<16;i++)
                                {
                                        if (pmc_list[i][1]==1 &&
pmc_list[i][0]<3080 && read_matrix(1,0)==0)
                                        {
                                                write_matrix(1, 0,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (pmc_list[i][1]==1 &&
pmc_list[i][0]<3428 && read_matrix(1,1)==0)
                                        {
                                                write_matrix(1, 1,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                }
                        if (pmc==1)
                                if (pmc_list[i][1]==1 && pmc_list[i][0]<3428 &&
read_matrix(1,1)==0)
                                {
                                        write_matrix(1, 1, pmc_list[i][0]);
                                        pmc_list[i][1]=0;
                                }
                                if (pag_list[i][1]==1 && pag_list[i][0]<2826 &&
read_matrix(0,0)==0)
                                {
                                        write_matrix(0, 0, pag_list[i][0]);
                                        pag_list[i][1]=0;
                                }
        }
```

// Hafiflerin Yerleştirilmesi (alt)

```
if (remaining_containers>=3 && remaining_containers<=4 && suggestion1==0 &&
suggestion2==0 && suggestion3==0 && suggestion4==0 && suggestion5==0 &&
suggestion6==0 && suggestion7==0 && suggestion8==0 &&  suggestion9==0)
        {
                if (read_matrix(4,3)>1 && pmc_cond3+pmc_cond4+pmc_cond5>2)
                        for (i=0;i<16;i++)
                        {
                                if (remaining_containers==3 && pmc_list[i][1]==1
&& pmc_list[i][0]<4626 && pmc_list[i][0]>4281 && read_matrix(7,0)==0)
                                {
```

```
                                                write_matrix(7, 0,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pmc_list[i][1]==1
&& pmc_list[i][0]<4626 && pmc_list[i][0]>4281 && read_matrix(7,0)==0)
                                        {
                                                write_matrix(7, 0,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 &&
pmc_cond3+pmc_cond4+pmc_cond5<4 && pmc_list[i][1]==1 && pmc_list[i][0]<4626 &&
read_matrix(7,1)==0)
                                        {
                                                write_matrix(7, 1,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pmc_list[i][1]==1
&& pmc_list[i][0]<4626 && pmc_list[i][0]>4281 && read_matrix(7,1)==0)
                                        {
                                                write_matrix(7, 1,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                }
                        if (read_matrix(4,3)==0 &&
pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5>5)
                                for (i=0;i<16;i++)
                                {
                                        if (remaining_containers==3 && pmc_list[i][1]==1
&& pmc_list[i][0]<4626 && pmc_list[i][0]>4281 && read_matrix(7,0)==0)
                                        {
                                                write_matrix(7, 0,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pmc_list[i][1]==1
&& pmc_list[i][0]<4626 && pmc_list[i][0]>4281 && read_matrix(7,0)==0)
                                        {
                                                write_matrix(7, 0,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 &&
pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5<7 && pmc_list[i][1]==1 &&
pmc_list[i][0]<4626 && read_matrix(7,1)==0)
                                        {
                                                write_matrix(7, 1,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pmc_list[i][1]==1
&& pmc_list[i][0]<4626 && pmc_list[i][0]>4281 && read_matrix(7,1)==0)
                                        {
                                                write_matrix(7, 1,
pmc_list[i][0]);
                                                pmc_list[i][1]=0;
                                        }
                                }
                        if (pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==4 &&
pag_cond2>0 && pag_cond1>0)
                                for (i=0;i<16;i++)
                                {
                                        if (remaining_containers==3 && pag_list[i][1]==1
&& pag_list[i][0]<4626 && pag_list[i][0]>4321 && read_matrix(6,2)==0)
                                        {
```

```
                                                write_matrix(6, 2,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pag_list[i][1]==1
&& pag_list[i][0]<4626 && pag_list[i][0]>4321 && read_matrix(6,2)==0)
                                        {
                                                write_matrix(6, 2,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pag_list[i][1]==1
&& pag_list[i][0]<4626 && pag_list[i][0]>4321 && read_matrix(6,1)==0)
                                        {
                                                write_matrix(6, 1,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                }
                        if (pag_cond1+pmc_cond2+pmc_cond3+pmc_cond4+pmc_cond5==5 &&
pag_cond2>0)
                                for (i=0;i<16;i++)
                                {
                                        if (remaining_containers==3 && pag_list[i][1]==1
&& pag_list[i][0]<4626 && pag_list[i][0]>4321 && read_matrix(6,2)==0)
                                        {
                                                write_matrix(6, 2,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pag_list[i][1]==1
&& pag_list[i][0]<4626 && pag_list[i][0]>4321 && read_matrix(6,2)==0)
                                        {
                                                write_matrix(6, 2,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                        if (remaining_containers==4 && pag_list[i][1]==1
&& pag_list[i][0]<4626 && pag_list[i][0]>4321 && read_matrix(6,1)==0)
                                        {
                                                write_matrix(6, 1,
pag_list[i][0]);
                                                pag_list[i][1]=0;
                                        }
                                }

                        if (pag_cond2==0)
                }
```

// Optimizasyon

// Tüm kombinasyonları elde etmek için modüler aritmetik yoluyla matris elemanlarının kaydırılması.

```
        for(f=1;f<10;f++)
        {
                d1[f]=mass[f][1];
                d2[((f) % 9)+1]=mass[f][1];
                d3[((f+1) % 9)+1]=mass[f][1];
                d4[((f+2) % 9)+1]=mass[f][1];
                d5[((f+3) % 9)+1]=mass[f][1];
                d6[((f+4) % 9)+1]=mass[f][1];
                d7[((f+5) % 9)+1]=mass[f][1];
                d8[((f+6) % 9)+1]=mass[f][1];
                d9[((f+7) % 9)+1]=mass[f][1];
                d10[((f+8) % 9)+1]=mass[f][1];
                d11[((f+9) % 9)+1]=mass[f][1];
                d12[((f+10) % 9)+1]=mass[f][1];
                d13[((f+11) % 9)+1]=mass[f][1];
                d14[((f+12) % 9)+1]=mass[f][1];
                d15[((f+13) % 9)+1]=mass[f][1];
        }


        //  En düşük I değerini verecek şekilde sürekli çalışacak döngü
        for(f2=1;f2<16;f2++)
        {
                if ((d2[f2]<3072) && (d1[f1]+d2[f2]<12364))
                        k[2]=d2[f2];

        for(f3=1;f3<16;f3++)
        {
                if ((d3[f3]<4108) && (k[1]+k[2]+d3[f3]<17910))
                        k[3]=d3[f3];

        for(f4=1;f4<9;f4++)
        {
                if ((d4[f4]<6033) && (k[1]+k[2]+k[3]+d4[f4]<19606))
                        k[4]=d4[f4];

        for(f5=1;f5<9;f5++)
        {
                if (d5[f5]<6033)
                        k[5]=d5[f5];

        for(f6=1;f6<9;f6++)
        {
                if (d6[f6]<6033)
                        k[6]=d6[f6];

        for(f7=1;f7<9;f7++)
        {
                if (d7[f7]<6033)
                        k[7]=d7[f7];

        for(f8=1;f8<9;f8++)
        {
                if (d8[f8]<3678)
                        k[8]=d8[f8];

        for(f9=1;f9<9;f9++)
        {
                if (d9[f9]<2983)
                        k[9]=d9[f9];
```

```
        for(f10=1;f10<9;f10++)
        {
                if (d10[f10]<2826)
                        k[10]=d10[f10];

        for(f11=1;f11<9;f11++)
        {
                if (d11[f11]<2826)
                        k[11]=d11[f11];

        for(f12=1;f12<16;f12++)
        {
                if (d12[f12]<1865)
                        k[12]=d12[f12];

        for(f13=1;f13<16;f13++)
        {
                if (d13[f13]<4626)
                        k[13]=d13[f13];

        for(f14=1;f14<16;f14++)
        {
                if (d14[f14]<4626)
                        k[14]=d14[f14];

        for(f15=1;f15<16;f15++)
        {
                if (d15[f15]<4626)
                        k[15]=d15[f15];



notpossible=0;
        for (npx=1;npx<16;npx++)
                for (npy=1;npy<16;npy++)
                        if (npx != npy)
                                if (k[npx] == k[npy])
                                        notpossible=1;




                                                                        {
if ((k[1] != 0) && (k[2] != 0) && (k[3] != 0) && (k[4] != 0) && (k[5] != 0)
&& (k[6] != 0) && (k[7] != 0) && (k[8] != 0) && (k[9] != 0) && (k[10] != 0)
&& (k[11] != 0) && (k[12] != 0) && (k[13] != 0) && (k[14] != 0) && (k[15]
!= 0) && (notpossible == 0))

if ((k[12]+k[11]+k[10]+k[9]+k[8]+k[7]+k[6]<23781) &&
(k[12]+k[11]+k[10]+k[9]+k[8]+k[7]<21196) &&
(k[12]+k[11]+k[10]+k[9]+k[8]<16700) && (k[12]+k[11]+k[10]+k[9]<12041) &&
(k[12]+k[11]+k[10]<7306) && (k[12]+k[11]<3966) && (k[1]+k[13]<6732) &&
(k[1]+k[13]+k[2]+k[14]<12364) && (k[1]+k[13]+k[2]+k[14]+k[3]+k[15]<17910))

{
Itotal = (-5.42*(k[1]+k[13])-4.73*(k[2]+k[14])-3.34*(k[3]+k[15])-
1.9571*k[4]-
.5625*k[5]+.2*k[6]+1.1*k[7]+1.1*1110+2.44*k[8]+2.44*1080+3.47*k[9]+3.47*105
0+4.55*k[10]+4.55*981+5.8*k[11]+7*k[12])/1000;

hedef = fabs(40.0f-Itotal);
if (hedef < oldhedef)
```

```
{
oldhedef = hedef;
if (checkhedef != oldhedef)
{

printf("%f \n", Itotal);
printf("k[1] :  %f  k[2]  : %f k[3]  : %f \n", k[1] , k[2] , k[3]);
printf("k[4] :  %f  k[5]  : %f k[6]  : %f \n", k[4] , k[5] , k[6]);
printf("k[7] :  %f  k[8]  : %f k[9]  : %f \n", k[7] , k[8] , k[9]);
printf("k[10] : %f  k[11] : %f k[12] : %f \n", k[10], k[11], k[12]);
printf("k[13] : %f  k[14] : %f k[15] : %f \n", k[13], k[14], k[15]);
checkhedef = oldhedef;
                                                                        }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
}
}
return 0;
}
```

3.5 Program Çözüm Sınırları

Yapılan çözümlerde ortaya çıkan sonuçlara göre, kombinasyonel optimizasyona girecek konteynerlerin tipleri aynı değilse, daha önce öngörülen dokuz değişken sayısının kabul edilebilir bir sürede sonuç elde edilmesi için yeterli olmadığı anlaşılmıştır. Bunun sebebi ise marjinal ve tercihli konteynerler yerleştikten sonra geriye kalan konteynerler aynı tipte olduğunda, programın gözden geçirmesi gereken kombinasyon sayısı 9!, yani 362880 kombinasyon iken, değişken sayısı sabit kalsa bile bir tane bile farklı tipte konteyner bulunması durumunda hesaba katılması gereken kombinasyon sayısı 21'in 9'lu permutasyonu olmaktadır.

$$P(21,9) = 21! / (21-9)! = 1{,}06 \times 10^{11}$$

İşlem sayısındaki bu artışın sebebi değişik tip konteyner olduğu durumda, yerleşim matrisindeki tüm satırların geometrik ilişkiler bakımından kontrol edilme ihtiyacıdır. Bu durumda program hesaplamaları tamamlayamamaktadır. Bu miktarda hesaplamanın kabul edilebilir bir sürede bitirilebilmesi için ise çok düşük değişken sayısına inilmelidir. Fakat bu derece indirgeme için daha önce yapılan kabul sınırlarının genişletilmesi ihtiyacı ve daha çok ön yerleştirme gerekir. Bu durumda da optimum çözümden çok fazla uzaklaşılacağı ve hesaplamaların geçerliliğinin ispatlanamayacağı görülmüştür.

4. SAYISAL UYGULAMA

Sayısal uygulamalar daha önce bahsedilen program çözüm sınırları dikkate alınarak, programın çalışabileceği örnekler dikkate alınmıştır.

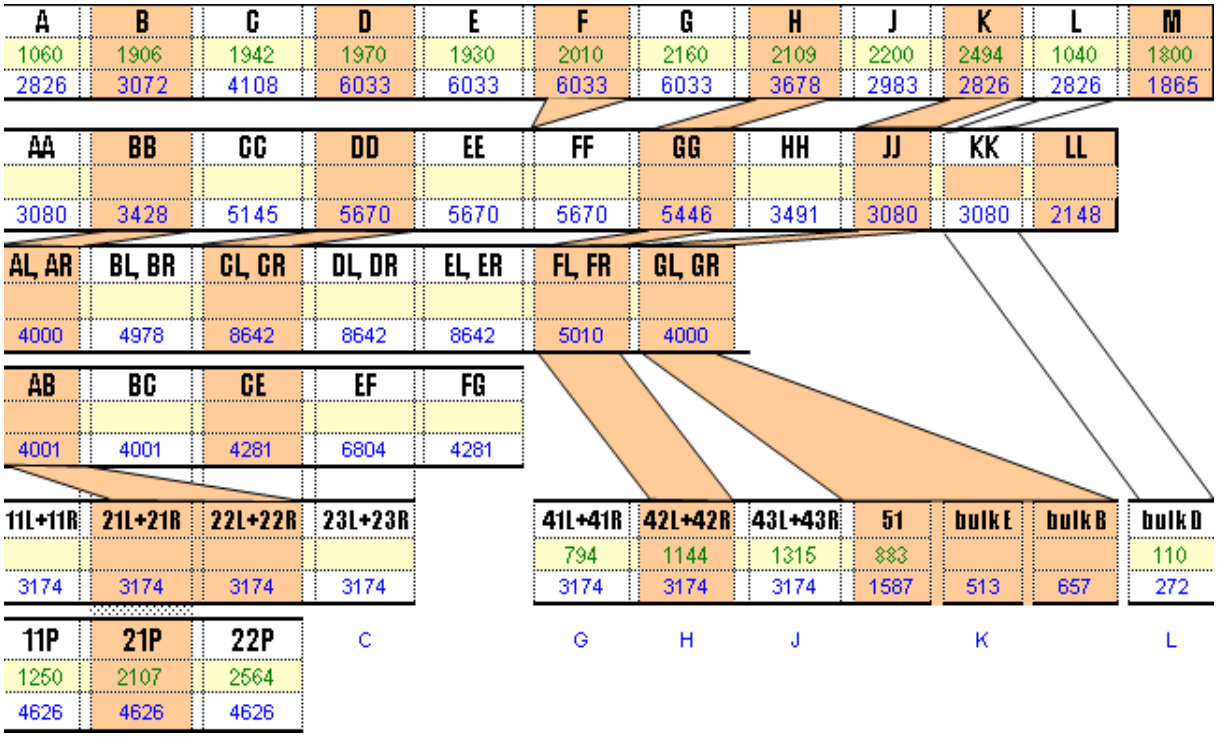
4.1 Yüklemeci Sonuçları

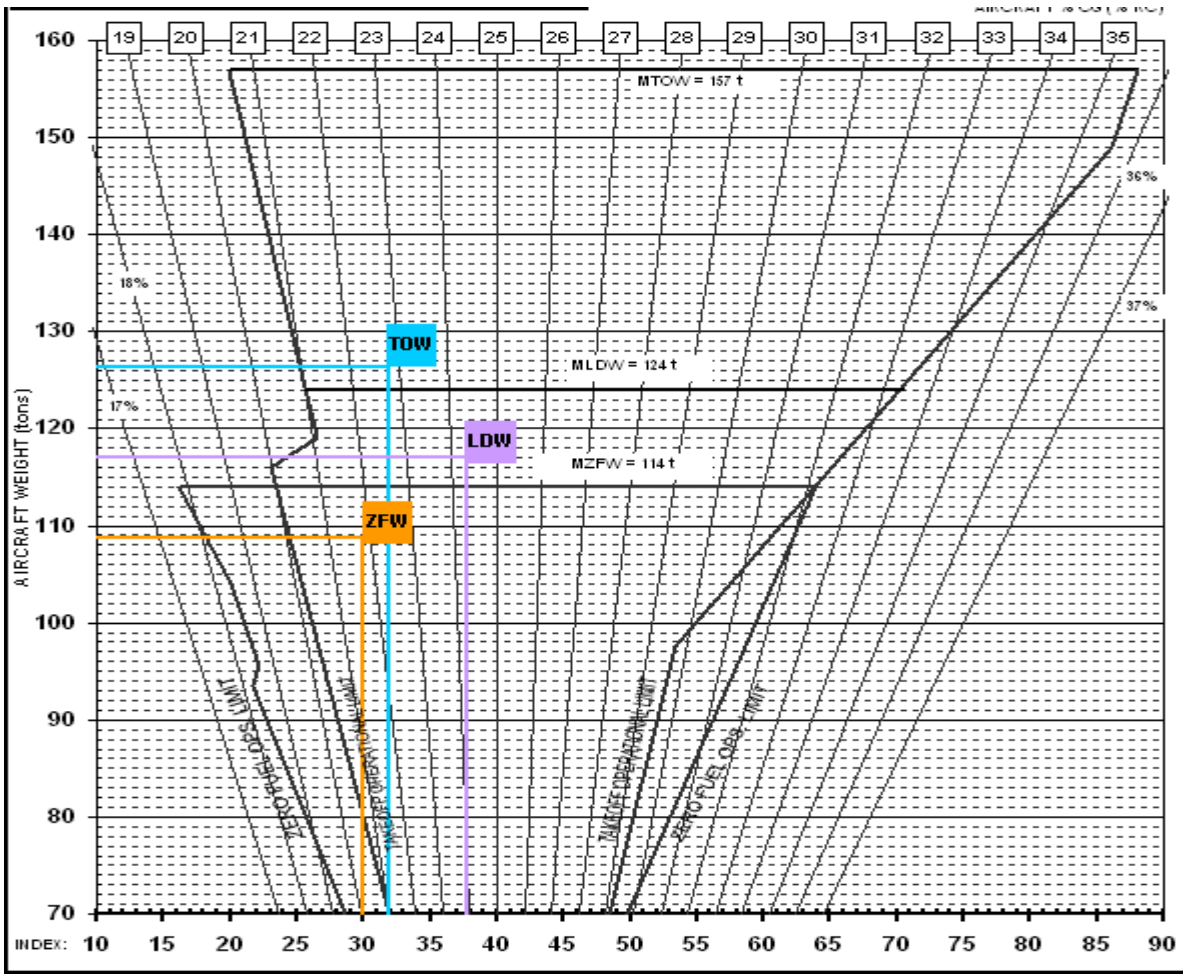
Daha önce yüklemeci tarafından yüklenmiş 6 örnek incelenecektir. Sırayla bu örneklerin yerleşim planları ve ağırlık-denge zarfında sonuçları aşağıda görülebilir. Örneklerde her pozisyonda mavi olarak görülen değerler daha önce Şekil 2.8'de belirtilen azami değerlerdir. İlgili pozisyonda yeşil değer olması ise oraya bir konteyner yüklendiğini ve onun ağırlık değerini göstermektedir.

Yerleşim planının altında ise ilgili yükleme değerine göre çıkan ağırlık-denge zarfı sonuçları görülmektedir.

Aynı örnekler için program tarafından da yükleme yapılacak ve daha sonra sonuçlar karşılaştırılacaktır. Örneklerde şekillerin üst kısmında yükleme pozisyonları ve ağırlıkları, alt kısımda ise TOW, ZFW ve LDW ağırlıklarına karşılık gelen CG değerleri grafiksel olarak zarf üzerinde görülmektedir.

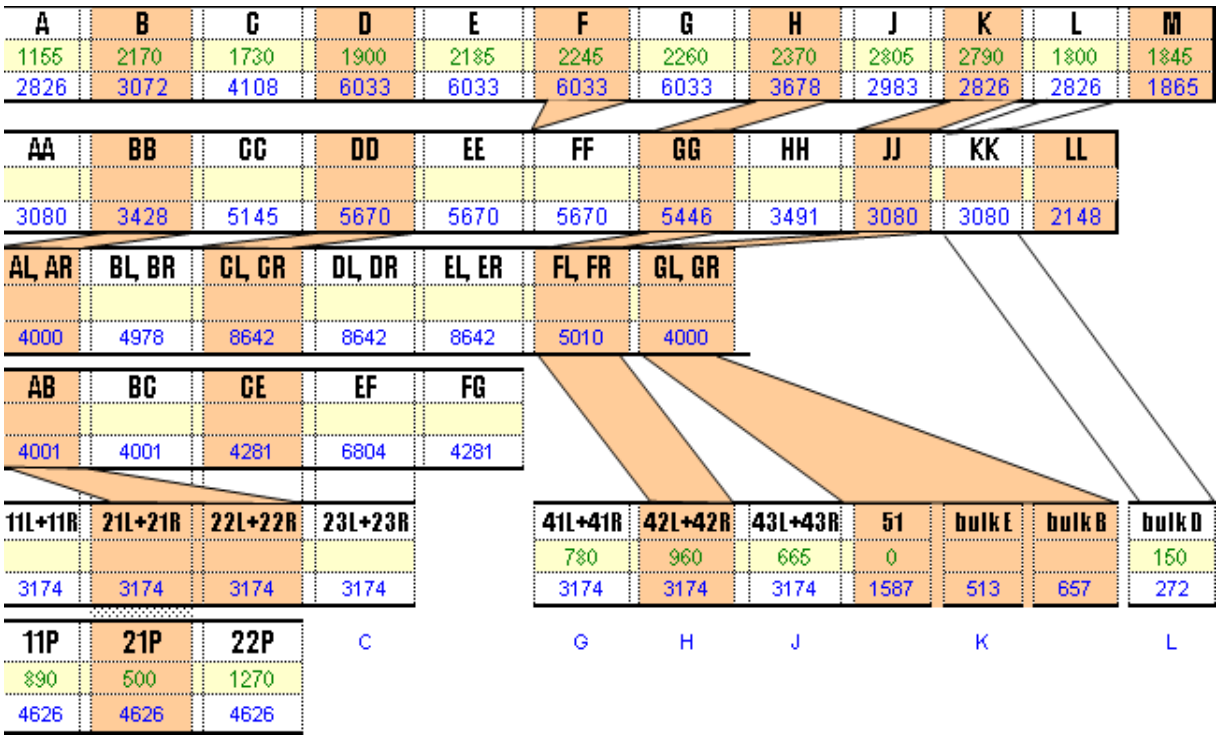
- **Örnek 1**

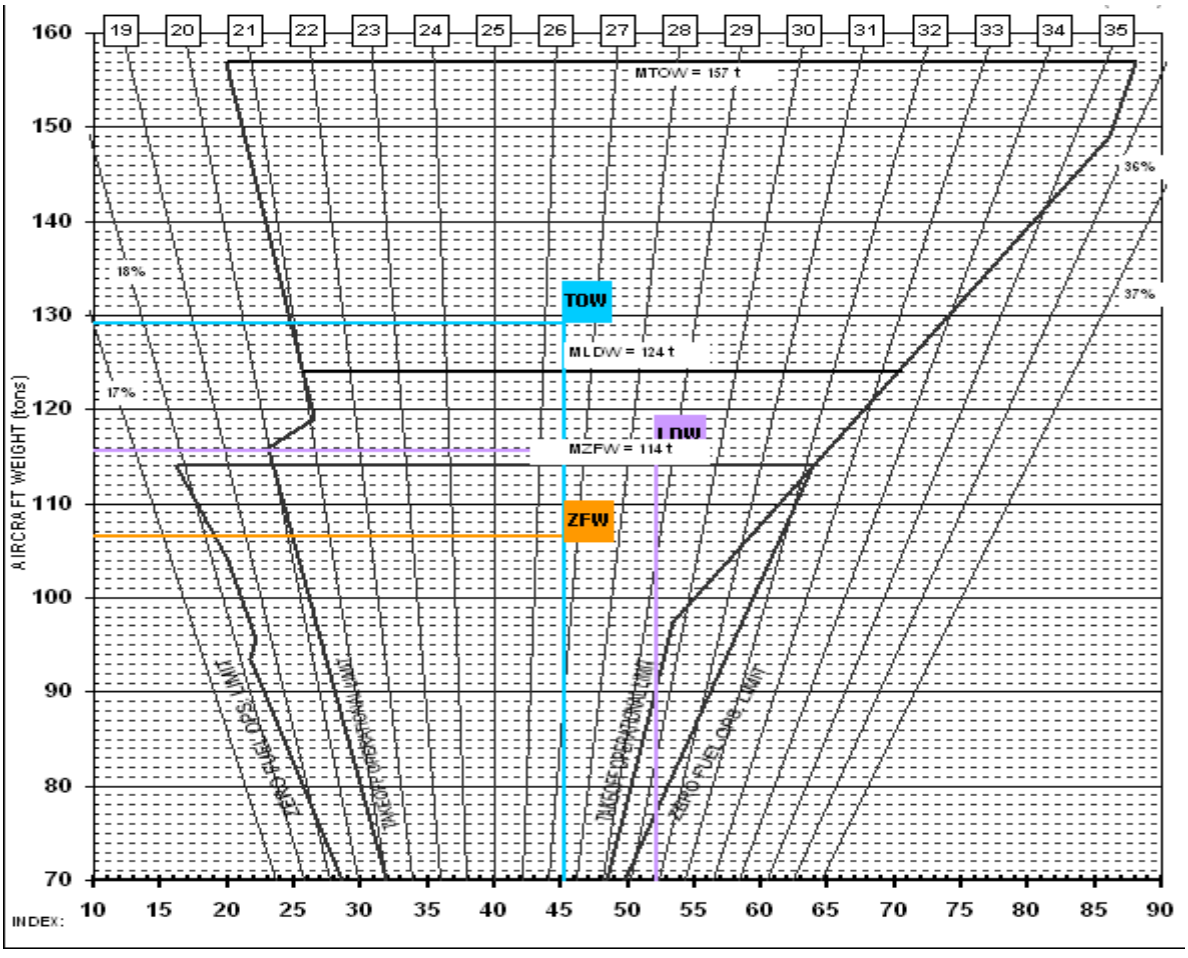




Şekil 4.1 Yüklemeci örnek 1 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 2**





Şekil 4.2 Yüklemeci örnek 2 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 3**

A	B	C	D	E	F	G	H	J	K	L	M
1575	1145	2595	1975	3045	1320	3125	3180	2690	2730	2095	1490
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

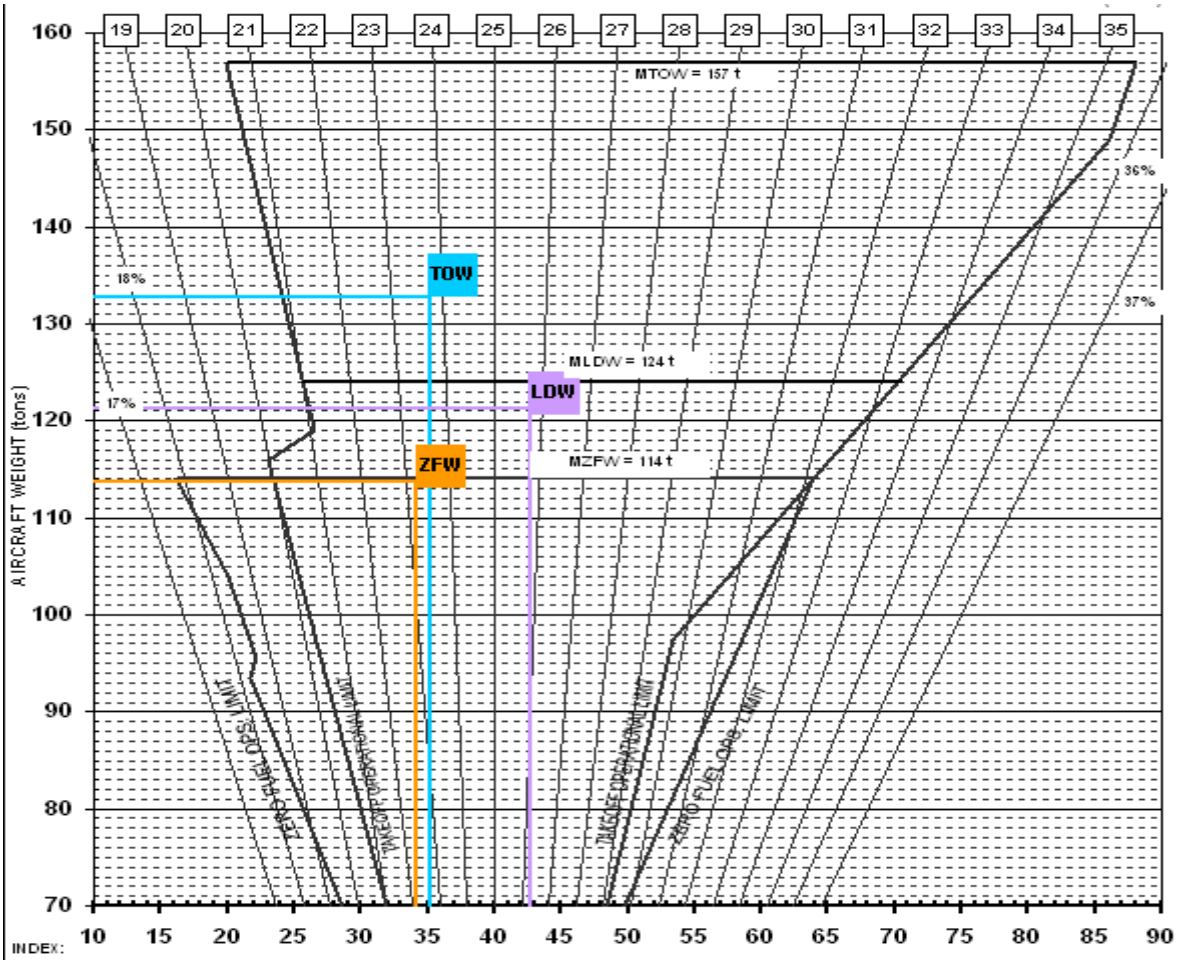
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
1691	1445	1215	0			150
3174	3174	3174	1587	513	657	272

11P	21P	22P
1300	2450	2480
4626	4626	4626

C G H J K L



Şekil 4.3 Yüklemeci örnek 3 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 4**

A	B	C	D	E	F	G	H	J	K	L	M
1406	1808	1252	1140	2972	2214	1342	1820	1628	1674	1560	1590
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

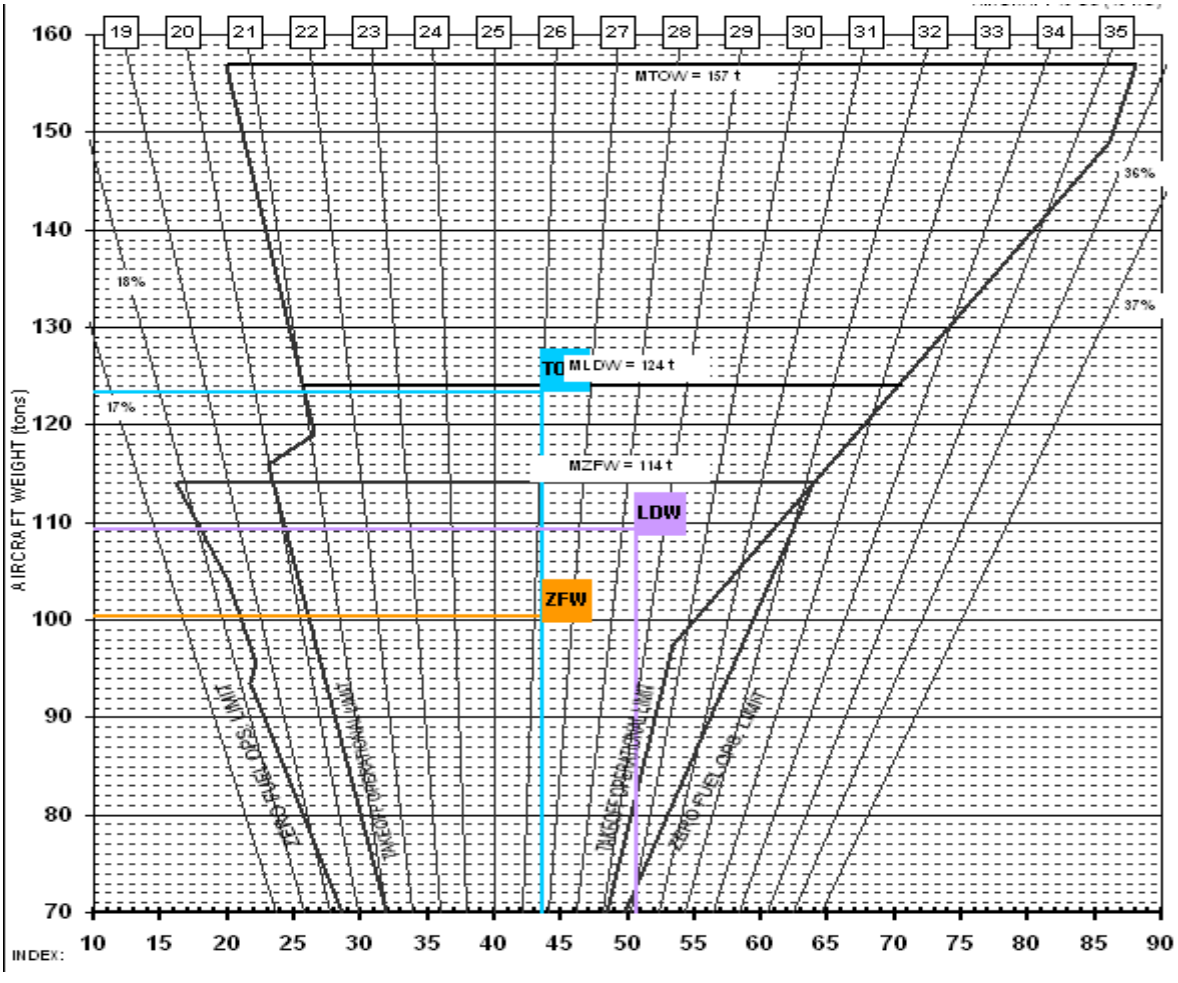
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
1306	1110	886	0			0
3174	3174	3174	1587	513	657	272

11P	21P	22P
0	0	652
4626	4626	4626

C G H J K L



Şekil 4.4 Yüklemeci örnek 4 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 5**

A	B	C	D	E	F	G	H	J	K	L	M
1375	1610	1930	1670	1855	2175	3085	3385	2300	2630	900	755
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

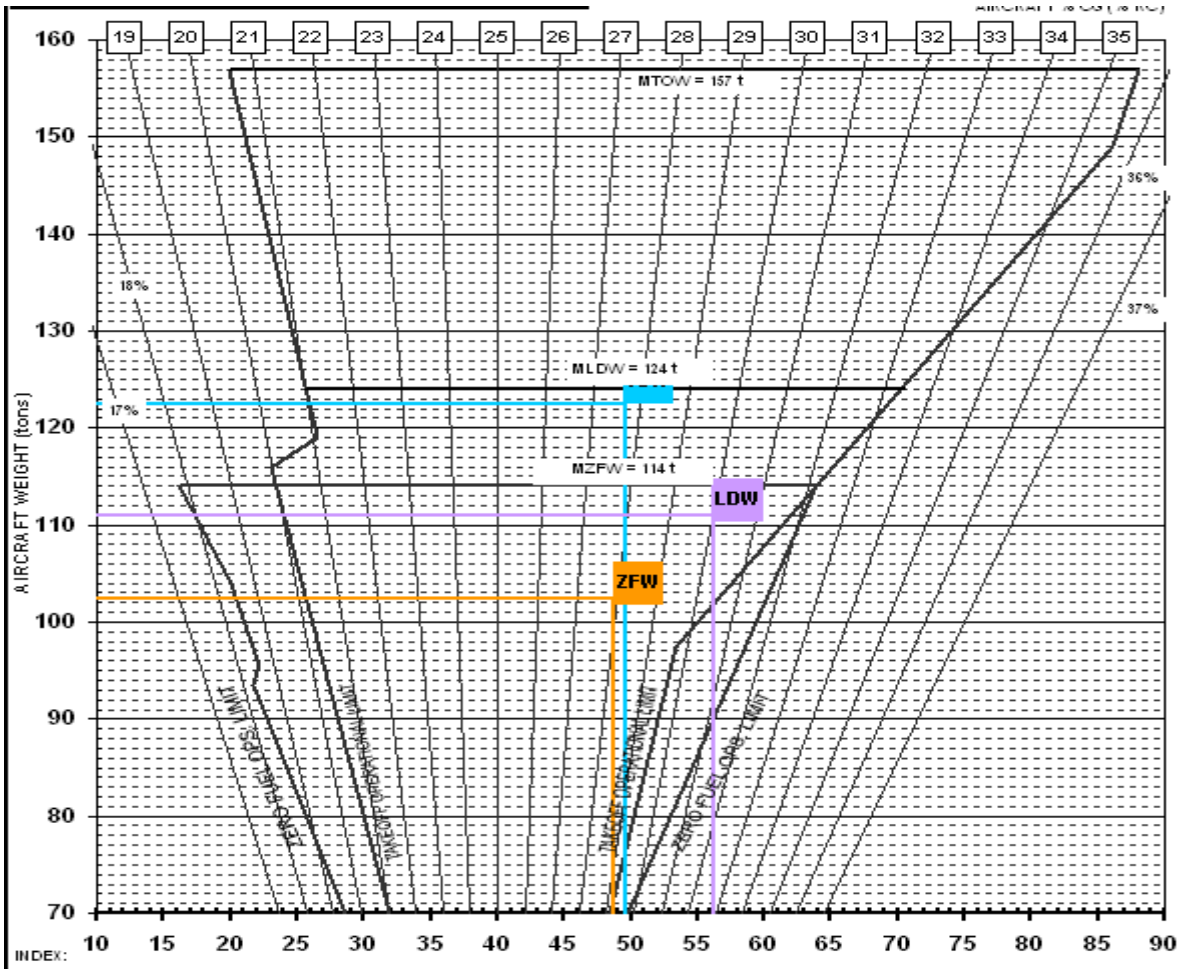
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
0	1523	865	128			279
3174	3174	3174	1587	513	657	272

11P	21P	22P
0	0	0
4626	4626	4626

C G H J K L



Şekil 4.5 Yüklemeci örnek 5 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 6**

A	B	C	D	E	F	G	H	J	K	L	M
1115	1140	2460	1140	4090	3405	1110	3310	2605	2300	1700	675
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

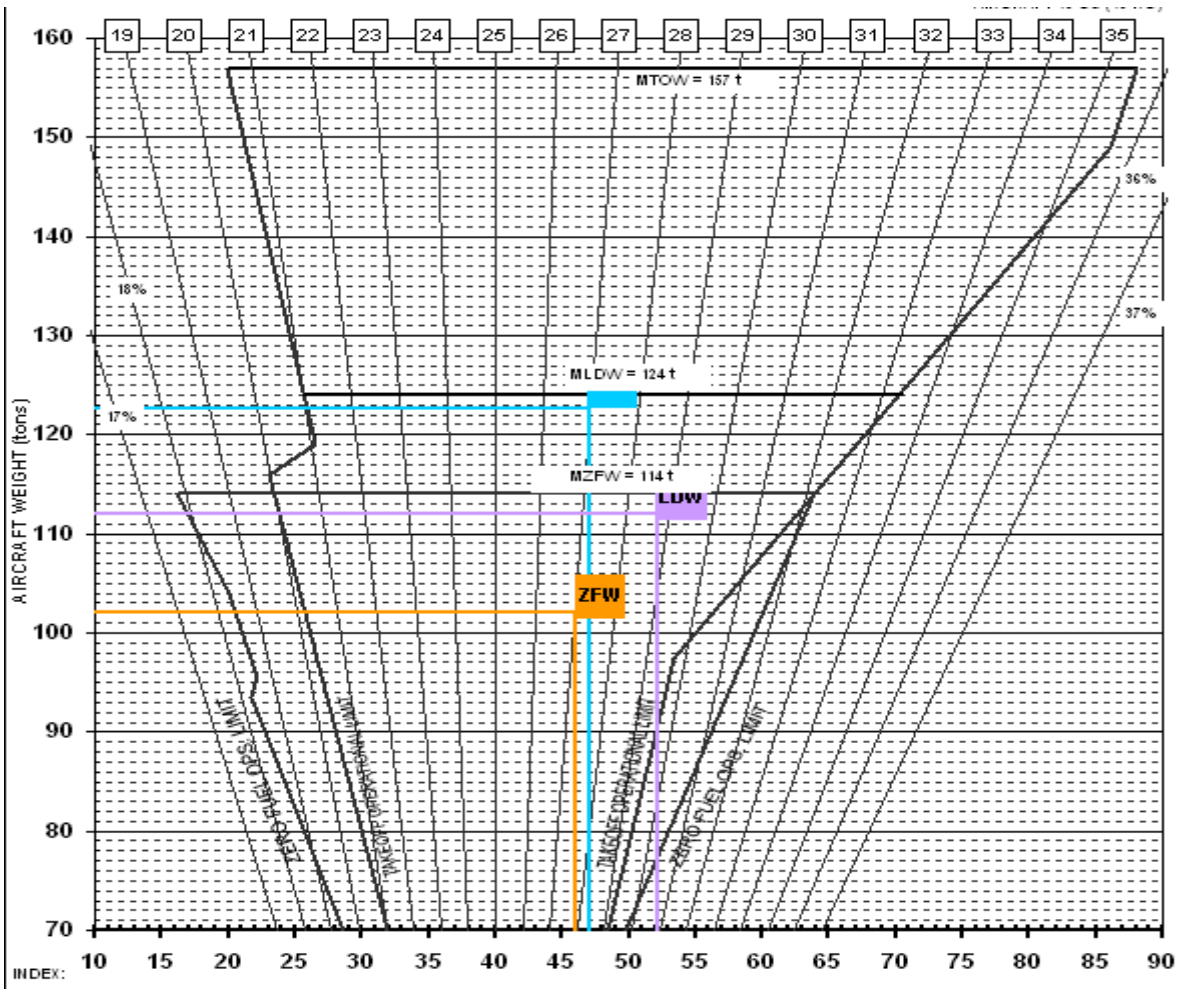
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
605	0	0	190			279
3174	3174	3174	1587	513	657	272

11P	21P	22P
0	0	0
4626	4626	4626

C G H J K L



Şekil 4.6 Yüklemeci örnek 6 yerleşim planı ve ağırlık-denge zarfı sonuçları.

4.2 Algoritma Sonuçları

- **Örnek 1**

A	B	C	D	E	F	G	H	J	K	L	M
1060	1930	1942	1970	2107	2109	2160	2200	2564	2494	2010	1800
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

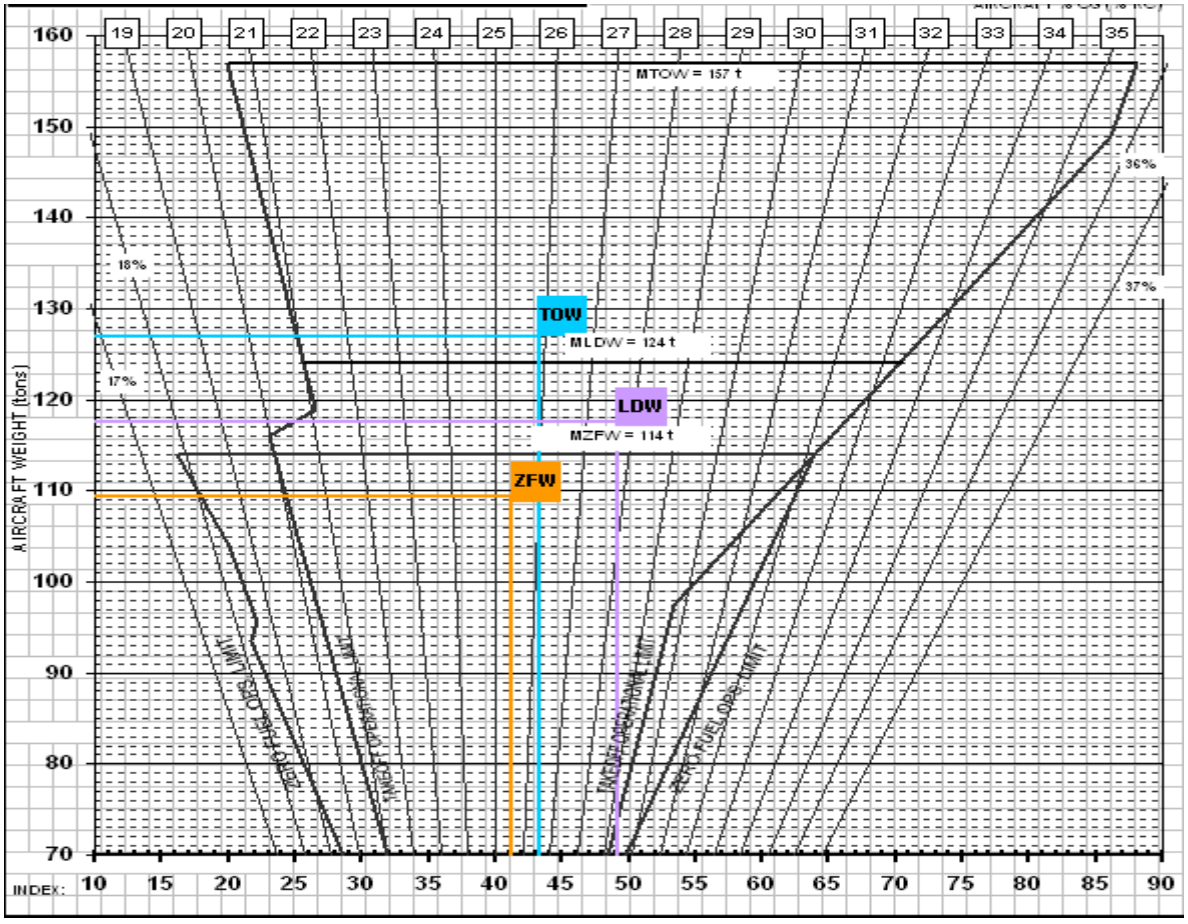
AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R	41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
				794	1144	1315	883			110
3174	3174	3174	3174	3174	3174	3174	1587	513	657	272

11P	21P	22P	C	G	H	J	K	L
1250	1640	1906						
4626	4626	4626						



Şekil 4.7 Algoritma örnek 1 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 2**

A	B	C	D	E	F	G	H	J	K	L	M
500	1730	1800	2170	2185	2245	2260	2370	2805	2790	1900	1845
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

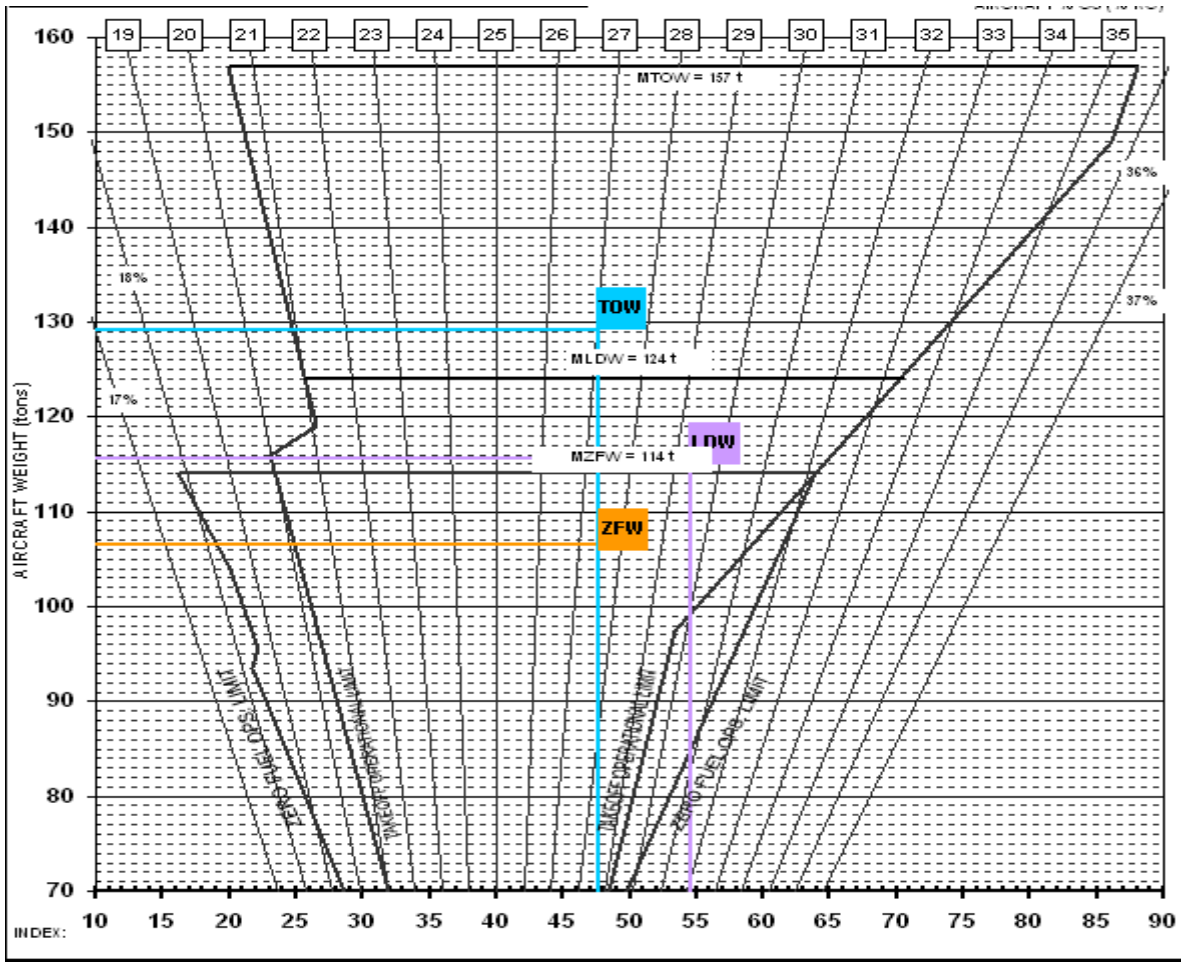
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
780	960	665	0			150
3174	3174	3174	1587	513	657	272

11P	21P	22P
890	1155	1270
4626	4626	4626

C G H J K L



Şekil 4.8 Algoritma örnek 2 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 3**

A	B	C	D	E	F	G	H	J	K	L	M
1145	2095	2450	2480	2595	3045	3125	3180	2690	2730	1975	1575
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

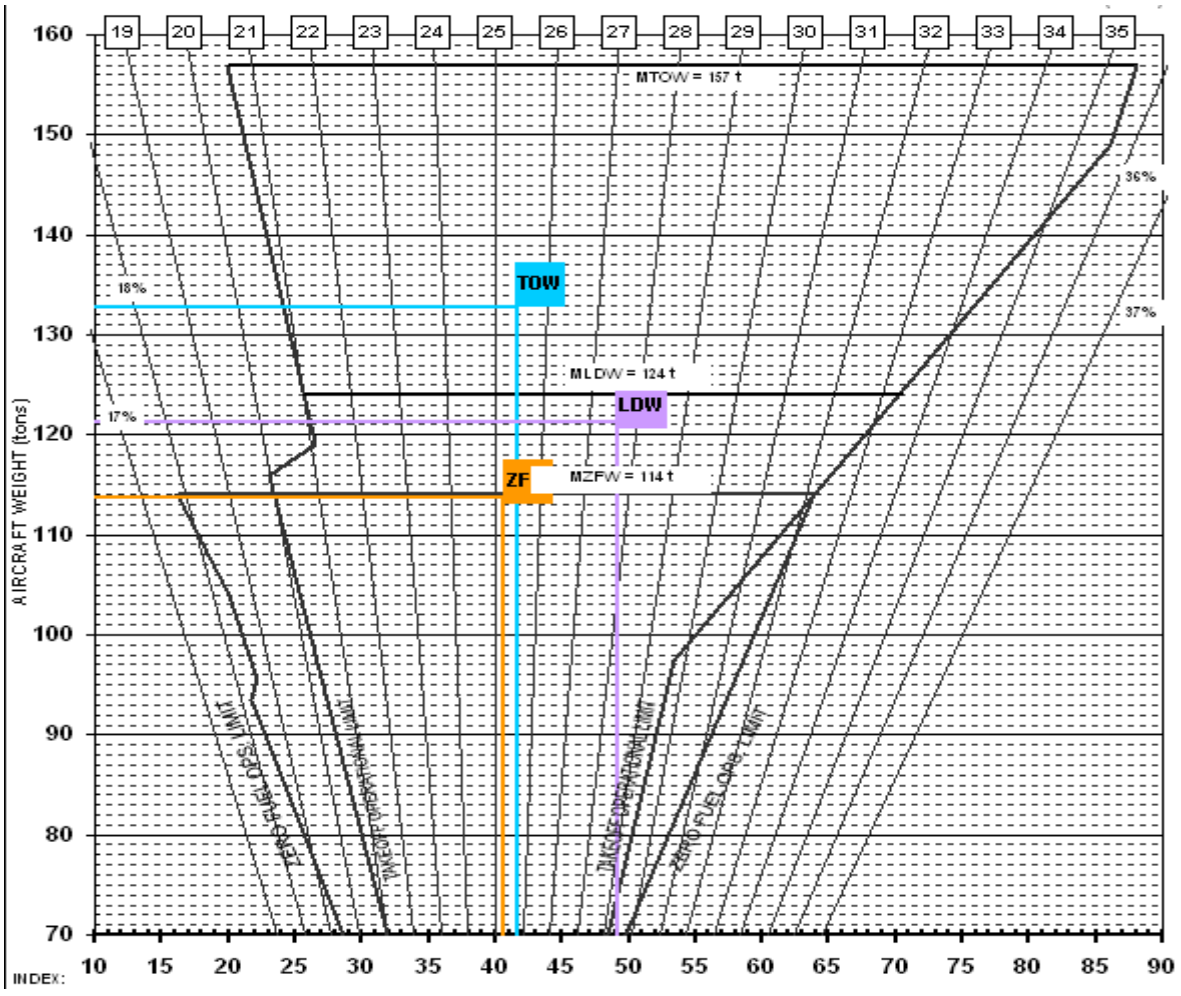
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
1691	1445	1215	0			150
3174	3174	3174	1587	513	657	272

11P	21P	22P
1300	1320	1490
4626	4626	4626

C G H J K L



Şekil 4.9 Algoritma örnek 3 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 4**

A	B	C	D	E	F	G	H	J	K	L	M
0	1252	1342	1406	1560	1590	2972	1674	1628	2214	1820	1808
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

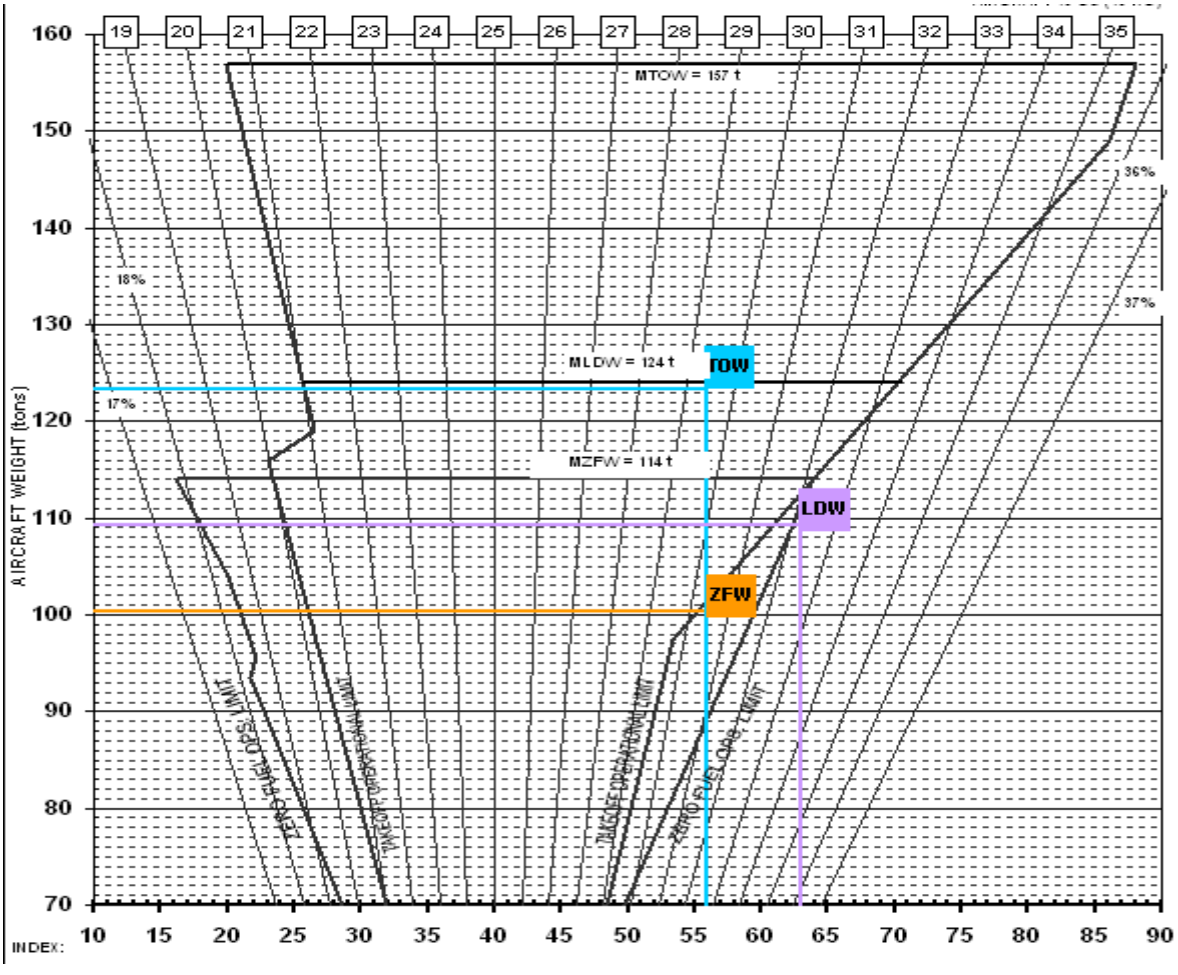
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
1306	1110	886	0			0
3174	3174	3174	1587	513	657	272

11P	21P	22P
0	652	1140
4626	4626	4626

C G H J K L



Şekil 4.10 Algoritma örnek 4 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 5**

A	B	C	D	E	F	G	H	J	K	L	M
0	900	1610	3385	3085	2300	2175	2630	1670	1930	1855	1375
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

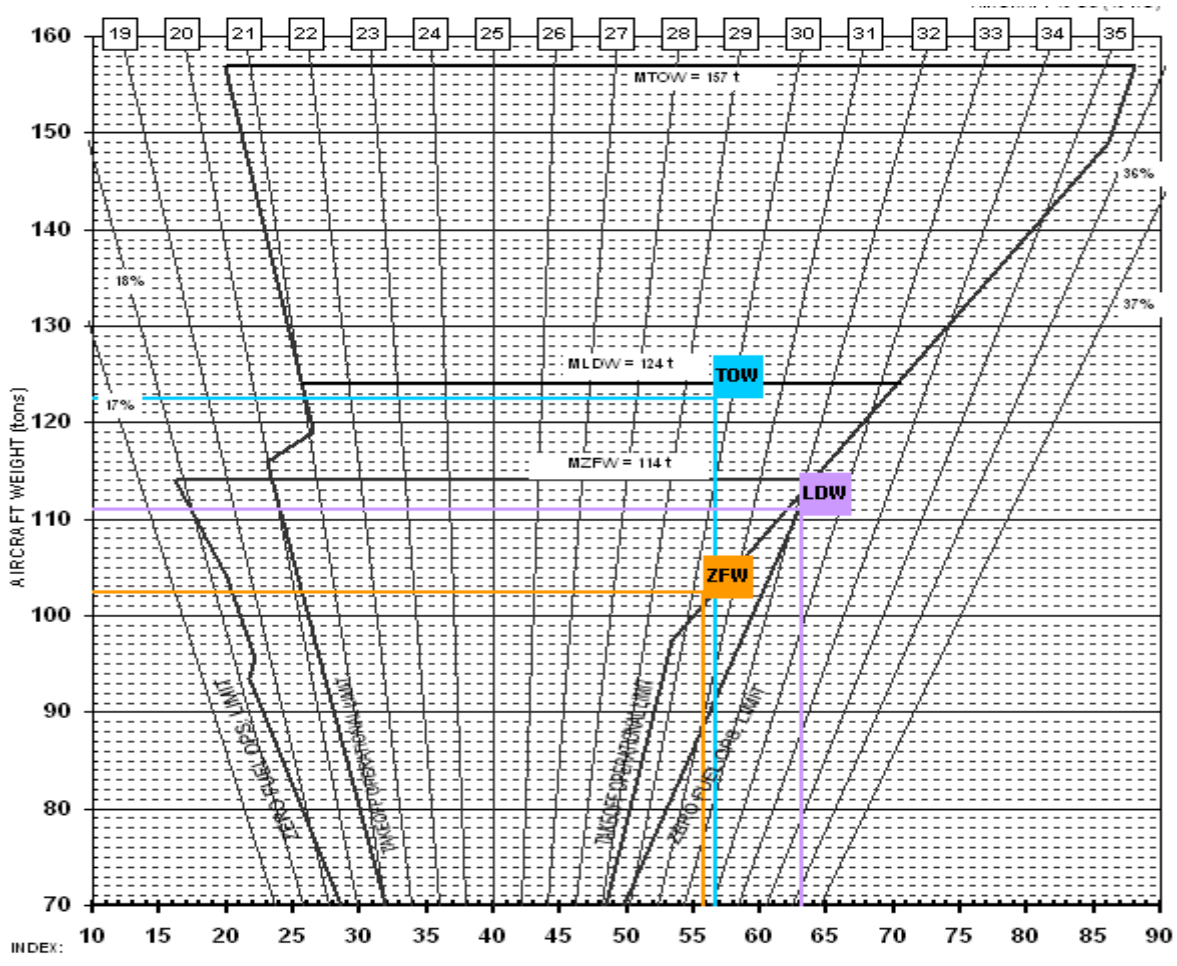
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
0	1523	865	128			279
3174	3174	3174	1587	513	657	272

11P	21P	22P
0	0	755
4626	4626	4626

C G H J K L



Şekil 4.11 Algoritma örnek 5 yerleşim planı ve ağırlık-denge zarfı sonuçları.

- **Örnek 6**

A	B	C	D	E	F	G	H	J	K	L	M
0	1110	1115	4090	1140	3310	3310	3405	2605	2460	1700	1141
2826	3072	4108	6033	6033	6033	6033	3678	2983	2826	2826	1865

AA	BB	CC	DD	EE	FF	GG	HH	JJ	KK	LL
3080	3428	5145	5670	5670	5670	5446	3491	3080	3080	2148

AL, AR	BL, BR	CL, CR	DL, DR	EL, ER	FL, FR	GL, GR
4000	4978	8642	8642	8642	5010	4000

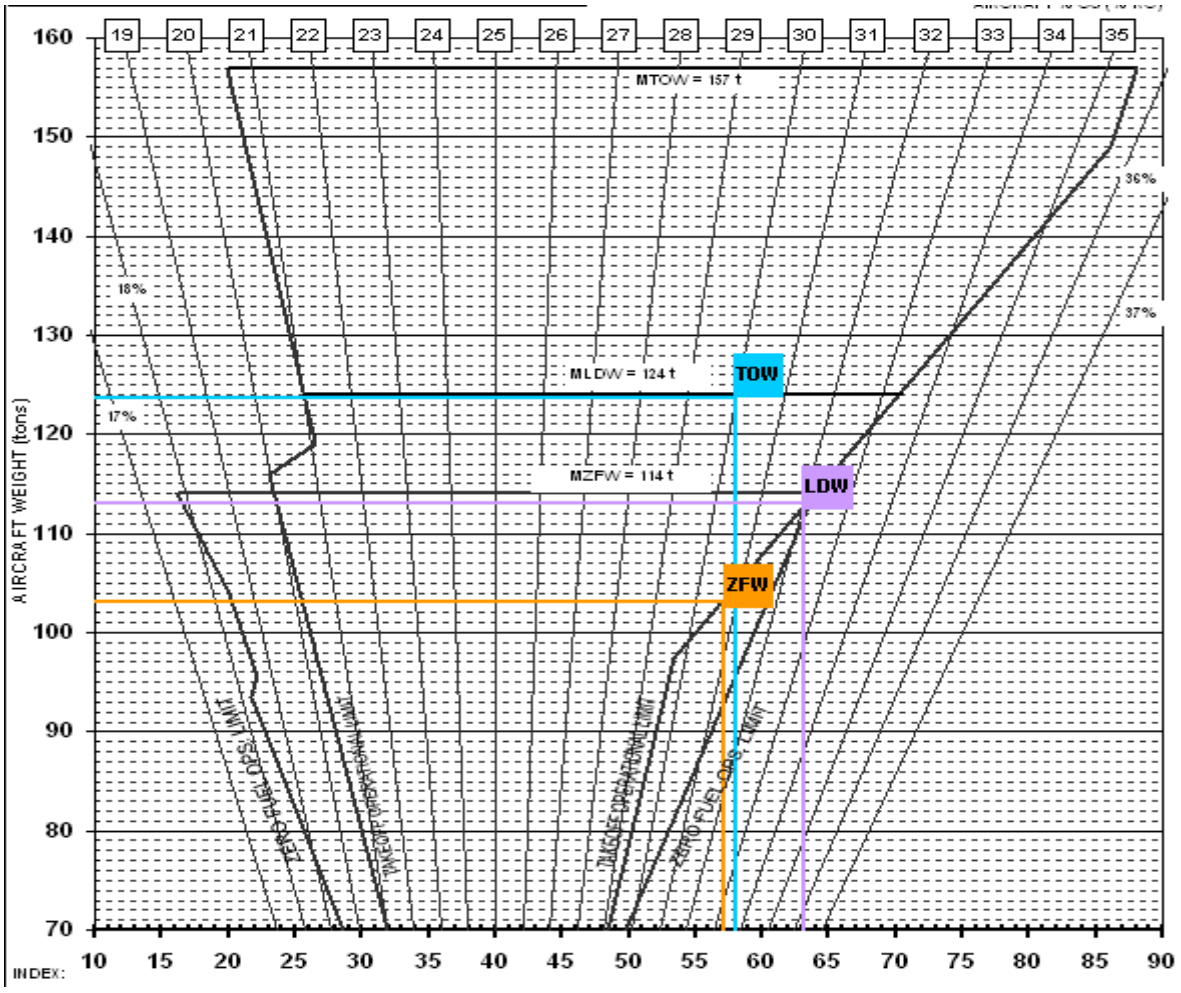
AB	BC	CE	EF	FG
4001	4001	4281	6804	4281

11L+11R	21L+21R	22L+22R	23L+23R
3174	3174	3174	3174

41L+41R	42L+42R	43L+43R	51	bulk E	bulk B	bulk D
605	0	0	190			279
3174	3174	3174	1587	513	657	272

11P	21P	22P
0	0	675
4626	4626	4626

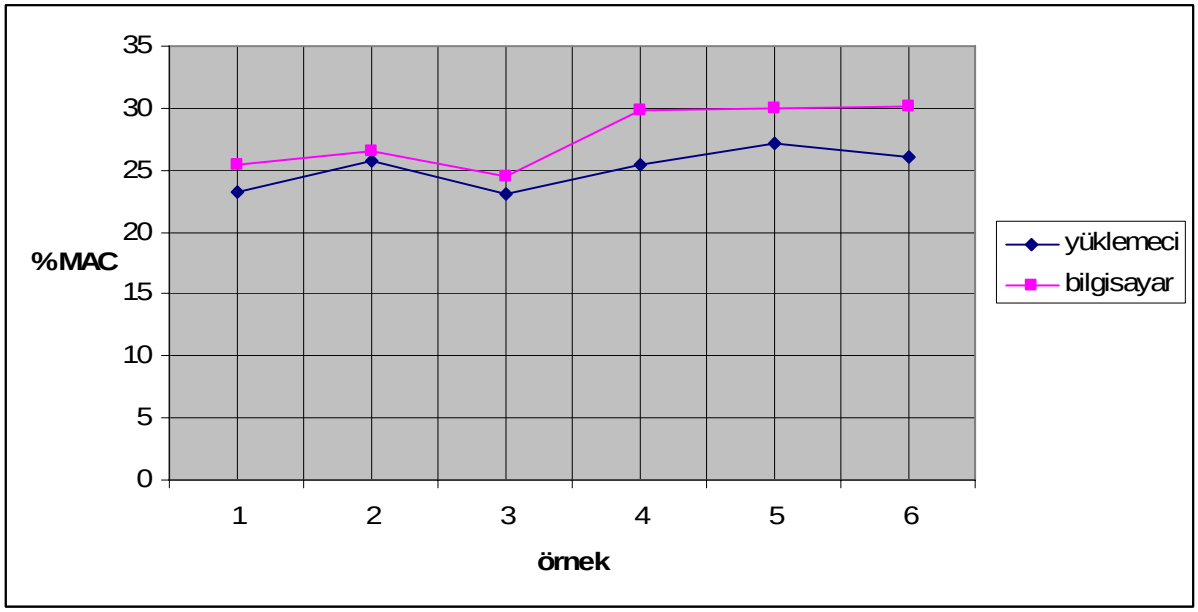
C G H J K L



Şekil 4.12 Algoritma örnek 6 yerleşim planı ve ağırlık-denge zarfı sonuçları.

4.3 Sayısal Sonuçların Karşılaştırılması

Yüklemeci tarafından daha önce yapılmış altı tane gerçek yükleme örneği, aynı zamanda bilgisayara da yaptırılmıştır. İlk etapta ağırlık-denge zarfı grafiklerinden de görüleceği gibi bilgisayarın yüklemesi ile ortaya çıkan sonuçlar, yüklemecinin sonuçlarına göre daha iyidir. Sonuçların karşılaştırmalı grafiği Şekil 4.13'de gösterilmiştir.



Şekil 4.13 Yüklemeci ve bilgisayar MACTOW sonuçları karşılaştırması.

Grafikten de anlaşılacağı gibi, örneklerin hepsinde bilgisayar ideal MAC değeri olan 29,3'e yüklemeciye göre daha yakın sonuçlar elde etmiştir. Örnekler arasında CG iyileştirmesi oransal olarak farklı olsa da, hepsinde daha iyileşmesi yüklemenin bilgisayar tarafından yapılmasının mutlaka iyi sonuç vereceğini göstermiştir.

5. SONUÇLAR VE DEĞERLENDİRME

Daha önce bahsedildiği gibi program belli durumlar için hesap yapamamaktadır. Ancak eldeki gerçek yükleme bilgileri gözden geçirildiğinde, büyük miktarda yüklemenin örneklerde olduğu gibi programın çalışabileceği sınırlar içinde olduğu görülmüştür. Bu sayede program yükleme açısından tüm sahaları kapsamasa da, kullanılabilir ve faydalı olduğuna kanaat getirilmiştir. Aynı zamanda çözüm yapamadığı sahaları anlamak ve ileriye dönük daha yeni programlar geliştirilmesi bakımından da yol gösterici olduğu düşünülmektedir.

Örneklerdeki yüklemenin bilgisayar tarafından yapılması sayesinde kazanılan avantajları düşünmek gerekirse, bunlardan ilki uçuş emniyeti ve insan hatasının engellenmesidir. Diğer bir avantaj ise CG'nin ideale yaklaştırılarak yakıt sarfiyatının azaltılmasıdır. Uçağın MAC uzunluğu 5,8287 m'dir. Bunu yüzde olarak düşünürsek her birime 5.8287 cm denk gelmektedir. Bu büyüklükteki bir uçak için yaklaşık olarak her on bin kilometrelik uçuş için CG'nin 1 cm yer değiştirmesi, 50 kg yakıt sarfiyatı anlamına gelmektedir. Bu değere göre muhtemel bilgisayar yüklemelerinin yakıt tasarrufuna nasıl bir katkı yapabileceğini hesaplamak istersek ;

- Bu altı örnekte yaklaşık olarak toplam 20000 km uçuş yapılmıştır.
- Toplam MAC iyileştirmesi yaklaşık %15, yani 87,43 cm'dir.

Dolayısıyla her on bin kilometrede 1 cm için 50 kg yakıt tasarrufu ilkesine göre;

$$2 \times 87{,}43 \times 50 = 8743 \text{ kg}$$

yakıt tasarrufu sağlanabilir.

Yıllık olarak gerçekçi bir değer elde etmek bakımından T.H.Y. Bilgi Yönetim ve Kontrol Dairesi'ne başvurulmuş ve TC-JCT uçağının son bir yıl içinde yaptığı toplam uçuş mesafesi istatistik bilgisi alınmıştır. İstatistiklere göre uçak bir yıl içinde yaklaşık 2600000 km mesafe katetmiştir.

Buna göre, yıllık tasarruf;

$$8743 \times 2600000 / (20000 \times 1000) \approx 1100 \text{ ton}$$

olur.

Bu miktar yadsınamayacak kadar büyüktür ve yükleme optimizasyonunun önemini vurgulamaktadır. Bu sayede hem parasal kayıp azaltılacak, hem de günümüzde güncel olarak tartışılan küresel ısınma sorununda en çok payı bulunan sektörlerden biri olan hava taşımacılığının iyileştirilmesi söz konusu olacaktır.

Her ne kadar insan doğasında bulunan beklenmeyen sorunları çözebilme yetisi bilgisayarlarda bulunmasa da, gelecekte bilgisayarların gelişmesiyle birlikte kimi zaman yapılmak zorunda kalınan kabuller ve buluşsal çözüm yerine en iyiyi bulan programlar geliştirilebilir. O zaman bu çalışmada tamamen bir kombinasyonel optimizasyon algoritması ile yenilenebilir.

KAYNAKLAR

Kevin, K.Y.NG, (1992), "A Multicriteria Optimization Approach to Aircraft Loading", Operations Research, 40, 6: 1200-1205.

Mathur, K., (1998), "An Integer-Programming-Based Heuristic For The Balanced Loading Problem ", Operations Research Letters, 22: 19-25.

Hill, R.R., (1998), "An Analytical Comparison of Optimization Problem Generation Methodologies", Proceedings of the 1998 Winter Simulation Conference, 609-615.

Thomas, C., Campbell, K., Hines, G. Ve Racer, M. (1998), "Airbus Packing at Federal Express", Interfaces, 28, 4: 21-30.

Airbus S.A.S. (2002), A310 Weight and Balance Manual, Airbus, Blagnac Cedex.

Amiouny S.V., Bartholdi, J.J., Vande Vate, J.H., ve Zhang, J. (1992) "Balanced Loading", Operations Research, 40, 2: 239-246.

AAR Cargo Systems, (2003), Cargo Loading Manual, AAR Cargo Systems, Michigan

JAA, (2001), Mass and Balance Theoritical Knowledge Manual, Jeppesen, Frankfurt

Mongeau, M., ve Bes, C. (2001), "Optimization of Aircraft Container Loading", IEEE Transactions On Aerospace and Electronic Systems, VOL 39, No.1-808645 January (2003).

ÖZGEÇMİŞ

Doğum tarihi	18.03.1981	
Doğum yeri	İstanbul	
Lise	1992-1999	Beşiktaş Atatürk Anadolu Lisesi
Lisans	1999-2003	Yıldız Teknik Üniversitesi Mühendislik Fakültesi Makine Mühendisliği Bölümü

Çalıştığı kurum

2004-Devam ediyor	T.H.Y. A.O. İkinci Pilot